Tabriz Journal of Electrical Engineering (TJEE), vol. 56, no. 1, 2026 Serial no. 115
DOI: 10.22034/tjee.2026.71004.5113

Improving the Stability of LSTM-Based Intrusion Detection for IoMT Networks via Harris Hawks Optimization

H. Zangiabadi Zadeh, H. Mohammadi*
Department of Computer Engineering, Payame Noor University (PNU), Tehran, Iran.
E-mail: hamid.zangiabadi@gmail.com, h.mohammadi@pnu.ac.ir
*Corresponding author

Received: 04/01/2026, Revised: 30/03/2026, Accepted: 06/05/2026.

Abstract
Intrusion detection in the Internet of Medical Things (IoMT) is a safety-critical task, as healthcare infrastructures depend on highly sensitive data and real-time medical devices. While Long Short-Term Memory (LSTM) networks offer strong capability in detecting complex network attacks, their performance is highly influenced by appropriate hyperparameter selection. To address this challenge, this study introduces a Modified Harris Hawks Optimization (MHHO) algorithm that enhances the standard HHO by integrating dynamic inertia weighting and Lévy-flight-based exploration. These mechanisms improve the balance between exploration and exploitation, effectively reducing premature convergence and strengthening global search behavior.
The proposed MHHO is employed to adaptively optimize key hyperparameters of an LSTM-based intrusion detection model, eliminating the need for manual tuning or conventional methods such as grid search. Furthermore, a robust, recall-focused fitness evaluation strategy is used to obtain hyperparameter configurations that improve model reliability and decrease sensitivity to class imbalance.
Experimental results on the CICIoMT2024 dataset across five independent runs show that both HHO-LSTM and MHHO-LSTM outperform the base LSTM in terms of average F1-score, with the MHHO-LSTM model demonstrating significantly lower performance variability. Since the optimization is performed only once during offline training, the final deployed system maintains the standard LSTM inference complexity. These findings highlight that the proposed MHHO-LSTM framework delivers enhanced stability, dependable performance, and practical suitability for safety-critical IoMT environments.

Keywords
Internet of Medical Things (IoMT), Deep Learning, Long Short-Term Memory (LSTM), Harris Hawks Optimization (HHO), Hyperparameter Optimization

1. Introduction

Nowadays, the healthcare industry is one of the largest contributors to global employment and revenue worldwide, and it is continuously expanding. However, access to timely and accurate medical services remains challenging, particularly in rural and remote regions where physical attendance is often required for diagnosis and treatment. Advances in portable and intelligent medical devices, together with the Internet of Things (IoT), have revolutionized healthcare delivery by enabling continuous monitoring, remote diagnosis, and personalized treatment [1].
IoT refers to a network of interconnected physical devices that collect, exchange, and process data via sensors and communication protocols. Its integration into the medical domain has given rise to the IoMT, a specialized ecosystem of intelligent medical devices, wearable systems, and diagnostic platforms connected through secure communication networks. IoMT offers significant benefits such as real-time patient monitoring, early detection of abnormal health conditions, and reduced dependence on in-person visits. Nevertheless, due to the critical nature of medical data, IoMT infrastructures face substantial security challenges, including vulnerabilities in device communication layers and susceptibility to cyberattacks.
Intrusion Detection Systems (IDS) play a vital role in safeguarding IoMT networks by identifying malicious activities in real time. Deep learning, particularly Long Short-Term Memory (LSTM) models, has shown great promise in detecting sophisticated attack patterns within data streams [2]. However, the performance and reliability of LSTM models heavily depend on properly tuned hyperparameters. To enhance stability and reduce variance across multiple runs, this study applies a Harris Hawks (HHO) and Modified Harris Hawks Optimization (MHHO) algorithm with a self-adjusting fitness function [3].
Despite their powerful representational capabilities, deep learning–based IDS models in IoMT environments must meet requirements beyond raw detection accuracy. Medical cyber-physical systems are inherently safety-critical, resource-constrained, and heterogeneous, where unstable or inconsistent detection behaviour can result in false alarms, missed attacks, or even disruption of clinical workflows. In this context, model stability and

robustness across multiple training runs are as important as achieving high performance metrics. Traditional hyperparameter optimization approaches typically aim to maximize a single metric, such as accuracy or F1-score, often overlooking the variance and reliability of the solutions. This limitation is particularly critical in IoMT networks, where fluctuating traffic patterns, limited labelled data, and noisy sensor streams can adversely affect model convergence and generalization. To address these challenges, metaheuristic optimization algorithms have emerged as effective tools for navigating complex, non-convex hyperparameter search spaces in deep learning models, enabling both high performance and consistent, reliable behaviour [4].

Among them, the HHO algorithm has attracted considerable attention due to its dynamic exploration and exploitation balance inspired by cooperative hunting strategies. [5] In this study, we use HHO algorithm and a modified version to optimize LSTM hyperparameters.

The main contributions and novelties of this work are summarized as follows:

- MHHO algorithm is proposed by integrating dynamic inertia weighting and Lévy-flight-based exploration. These mechanisms enhance the balance between exploration and exploitation, improve global search capability, and reduce the risk of premature convergence compared with the standard HHO.
- The proposed MHHO algorithm is employed for adaptive hyperparameter optimization of an LSTM-based intrusion detection model for IoMT environments, eliminating the need for manual tuning or traditional search strategies such as grid search.
- A recall-oriented and robust fitness evaluation strategy is designed to address class imbalance in intrusion detection. By considering multiple evaluation metrics and different decision thresholds, the optimization process produces more reliable and stable hyperparameter configurations.
- The proposed MHHO-LSTM framework improves detection stability across multiple experimental runs while maintaining the same inference complexity as a standard LSTM model, since the optimization process is performed only once during the offline training stage.

The rest of this paper is organized as follows: Section 2. reviews related work on IoMT security and intrusion detection systems. Section 3. describes the proposed MHHO-LSTM framework and the stability-aware fitness design. Section 4. presents the experimental setup, datasets, preprocessing process, and evaluation metrics. Section 5. discusses the results and compares the proposed method with baseline approaches. Finally, Section 6. concludes the paper and outlines future research directions.

The implementation of the proposed framework and the result of the project is publicly available at https://github.com/hamidzangiabadi/LSTM-HHO-for-IoMT-Dataset

2. Related Works

Several studies have analysed cyberattacks on IoMT systems and their defence mechanisms. Research in [6] examines privacy and security in IoMT, classifying attacks and outlining countermeasures. Studies [7], [8] review Machine Learning and Deep Learning methods for intrusion detection in medical devices, though most don't address practical deployment challenges in IoMT environments with their unique constraints and critical data requirements.

IDS for IoMT consist of monitoring, detection, and response components. They use signature-based, anomaly-based, or specification-based mechanisms and are categorized as network-based (NIDS), host-based (HIDS), or hybrid systems. Reference [9] provides a detailed IDS taxonomy for developing robust IoMT detection models.

Despite their powerful representational capabilities, deep learning–based IDS models in IoMT environments must meet requirements beyond raw detection accuracy. Medical cyber-physical systems are inherently safety-critical, resource-constrained, and heterogeneous, where unstable or inconsistent detection behaviour can result in false alarms, missed attacks, or even disruption of clinical workflows. In this context, model stability and robustness across multiple training runs are as important as achieving high performance metrics. Traditional hyperparameter optimization approaches typically aim to maximize a single metric, such as accuracy or F1-score, often overlooking the variance and reliability of the solutions. This limitation is particularly critical in IoMT networks, where fluctuating traffic patterns, limited labelled data, and noisy sensor streams can adversely affect model convergence and generalization. To address these challenges, metaheuristic optimization algorithms have emerged as effective tools for navigating complex, non-convex hyperparameter search spaces in deep learning models, enabling both high performance and consistent, reliable behaviour.

2.1. Internet of Medical Things (IoMT)

IoMT originated from developments in communication technologies, wireless networks, and connectable medical equipment beginning in the early 1990s with initial telemedicine advances. Initially, medical devices could only transmit simple data like blood pressure or heart rate to medical centres asynchronously over telephone networks, with focus on accessibility rather than security or privacy.

With Internet expansion and standardized protocols like Wi-Fi and Bluetooth, the second IoMT generation emerged in the 2000s, including advanced medical equipment such as cardiac monitors, insulin pumps, and networked medical imaging systems. Cloud platforms and mobile applications enabled near-real-time patient data storage, analysis, and display. Cyberattacks on healthcare began during this period, gradually raising awareness of security and intrusion detection needs [8].

From approximately 2010, as IoMT data volume and variety increased, Machine Learning approaches entered this domain for behavioural pattern analysis and anomaly detection. Algorithms like SVM, Decision Trees, and

KNN were applied to detect attack patterns and abnormal behaviours [10], though requiring manual feature extraction and high dependency on pre-processed data quality. ML solutions provided higher accuracy than traditional rule-based methods for threat detection, but limitations in detecting emerging attacks.

From the late 2010s, Deep Learning with complex neural networks like CNN, RNN, and LSTM entered IoMT. Their ability to automatically extract features and model network processes and spatiotemporal dependencies in medical data significantly improved IDS performance in IoMT environments. Hybrid structures combining meta-heuristic algorithms (e.g., PSO-LSTM) for real-time attack detection represented a new generation of intelligent IoMT security, not only increasing detection accuracy but also improving unknown attack identification capability, providing crucial steps toward complete patient privacy protection [11], [12].

Recent work by Doménech et al. (2025) investigated the effectiveness of machine learning–based intrusion detection systems for IoMT environments using the CICIoT2023 and CICIoMT2024 datasets. Their study highlighted the importance of domain-specific datasets, showing that models trained on one dataset may experience significant performance degradation when evaluated on another. In addition, the authors proposed several dataset optimization strategies, including uniform windowing, improved dataset balancing, and proper train–validation–test splits, demonstrating substantial improvements in IDS performance [13].

2.2. LSTM-based deep learning

Deep recurrent neural networks (RNNs) extend the basic recurrent structure by stacking multiple recurrent layers, enabling hierarchical temporal feature extraction. In such architectures, lower layers typically capture short-term, fine-grained temporal patterns, while higher layers encode more abstract, long-term dependencies. This hierarchical representation allows deep RNNs to model complex sequential relationships that shallow networks or single-layer RNNs may fail to capture. However, stacking multiple recurrent layers exacerbates the vanishing and exploding gradient problems inherent in classical RNNs, making training unstable and slow. LSTM and its variants, such as stacked LSTM and bidirectional LSTM, are specifically designed to mitigate these issues through gated memory units and controlled information flow across layers. By leveraging multiple layers, deep LSTM networks can learn richer temporal hierarchies, improving performance on tasks requiring both local and global sequence understanding, such as natural language processing, time-series forecasting, and anomaly detection in cyber-physical systems. Nevertheless, the increased representational power comes at the cost of a larger hyperparameter search space, making automated optimization strategies, such as HHO, crucial for identifying effective configurations while maintaining training stability and computational efficiency.

2.3. Metaheuristics and HHO

Metaheuristic algorithms emerged in the 1960s-1970s as approximate search techniques for complex nonlinear optimization problems, inspired by natural processes, evolutionary rules, or physical phenomena. Early methods like Simulated Annealing (SA, 1983) and Genetic Algorithms (GA, 1970s) expanded classical optimization capabilities [14].

These pioneered solving NP-hard problems and multi-objective optimization tasks. Common characteristics include independence from precise mathematical problem structure and ability to work with continuous, discrete, or multimodal objective functions. By balancing exploration and exploitation, they efficiently search vast spaces to reach near-optimal solutions within acceptable time.

The 1990s-early 2000s witnessed emergence of Swarm Intelligence algorithms like Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO), directly modelling social behaviour of birds and ants. Additional families based on hunting behaviour, migration patterns, chemical reactions establishing metaheuristics as a comprehensive framework for modeling complex search processes [4], [15], [16].

Notable algorithms include Genetic Algorithms based on natural selection principles, PSO inspired by bird flocking behaviour, or ACO modelling ant pathfinding behaviour, Grey Wolf Optimizer simulating wolf pack hunting hierarchy [17].

2.4. Research gap and positioning

HHO was introduced in 2019 by Heidari et al. modelling cooperative hunting behaviour of Harris hawks. It incorporates exploration and exploitation phases through rapid dive attacks, multi-stage encirclement, and adaptive prey escape energy. Dynamic transition between exploration and exploitation makes HHO effective in navigating complex multimodal search spaces, with multiple update strategies including random jumps, population-driven movement, soft/hard besiege, and rapid-dive variations [5].

HHO has been widely applied in signal processing, feature selection, engineering design, and other machine learning optimization [18]. However, existing studies rarely examine stability of optimization-based hyperparameter tuning in intrusion detection systems, especially within IoMT environments where consistent and reliable performance is essential due to critical medical data nature.

3. Proposed Method

To address the identified research gap and enhance the stability and reliability of intrusion detection in IoMT networks, this study proposes a hybrid optimization deep learning framework based on MHHO, tuned LSTM model. Deep learning has been effectively applied to various cybersecurity challenges such as DDoS attack detection using multi-layer perceptron (MLP) models [19], which demonstrates the potential of these methods for learning complex network traffic patterns. The core objective of the proposed method is to improve detection accuracy, convergence stability, and robustness of deep learning based IDS when operating on highly sensitive and dynamic medical data streams.

3.1. LSTM Networks

Long Short-Term Memory networks represent a specialized class of recurrent neural architectures designed to capture long-term dependencies in sequential data while mitigating the vanishing gradient problem inherent in traditional RNNs. The LSTM cell maintains an internal memory state through a gating mechanism that regulates information flow across temporal sequences [20].

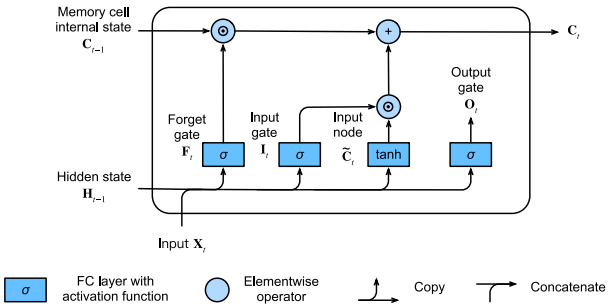

Fig. 1. LSTM Gates Visualization

Given input, previous hidden state, and previous cell state, the LSTM gates are computed as

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (1)$$

In this gate $\mathbf{x}_t$ is the input vector at time step t and $\mathbf{h}_{t-1}$ denotes the previous hidden state.

The input gate controls the incorporation of new information into the cell state:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (2)$$

Eq. (2) Computes the input gate, determining how much new information is allowed into the cell state. Where W_i and b_i are the weight matrix and bias vector.

The output gate regulates the information transmitted to the next hidden state:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (3)$$

Eq. (3) Computes the output gate, controlling how much cell state information is exposed to the hidden state. O_t controls the exposure of the cell state to the hidden state. The candidate cell state represents the potential new information to be added:

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (4)$$

Eq. (4): Generates the candidate cell state representing potential new information to be added. The cell state update combines the forgotten information with the new candidate values:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (5)$$

Eq. (5) Updates the cell state by combining retained past information with the new candidate content. The hidden state is computed as a filtered version of the current cell state:

$$h_t = o_t \odot \tanh(c_t) \quad (6)$$

Eq. (6) Produces the hidden state as a gated and activated version of the current cell state, where $\sigma(\cdot)$ denotes the sigmoid activation function, $\tanh(\cdot)$ is the hyperbolic tangent function, W and b are trainable weight matrices and bias vectors, and $\odot$ denotes element-wise multiplication [20].

By this mechanism, the LSTM can preserve long-term dependencies through the cell state while adapting to new inputs at each step. The combination of the input, forget, and output gates, along with the candidate cell state, provides a flexible and stable way to handle sequences of varying lengths and dependencies, mitigating the vanishing and exploding gradient problems common in classical RNNs.

3.2. Harris Hawks Optimization (HHO)

In HHO, each hawk represents a candidate solution in the search space, while the prey (rabbit) corresponds to the current best solution. The optimization process is guided by the prey's escaping energy $E(t)$, which dynamically controls the balance between exploration and exploitation throughout the iterations. This energy decreases over time according to the maximum number of iterations, enabling a gradual shift from global exploration to local exploitation [5]. The escaping energy is defined as

$$E(t) = 2E_0\left(1 - \frac{t}{T}\right), \quad (7)$$

Where $E_0 \in [-1,1]$, t is the current iteration, and T is the maximum number of iterations.

During the exploration phase $|E(t)| \geq 1$ the hawk position is updated with two strategies.

First strategy is perching based on random tall trees selected by probability q. Where $q >= 0.5$:

$$X_i(t+1) = X_{\text{rand}}(t) - r_1 |X_{\text{rand}}(t) - 2r_2 X_i(t)| \quad (8)$$

Where X_{rand} is a randomly selected hawk and $r_1, r_2 \in (0,1)$.

Another exploration strategy, where $q < 0.5$ based on the average position of the population is given by

$$X_i(t+1) = (X_{\text{rabbit}}(t) - X_m(t)) - r_3(LB + r_4(UB - LB)) \quad (9)$$

Where $X_m(t)$ denotes the mean position of all hawks, and *LB* and *UB* represent the lower and upper bounds of the search space, respectively.

During the exploitation phase $|E(t)| < 1$, the hawk position is updated using the soft besiege strategy as

$$X_i(t+1) = X_{\text{rabbit}}(t) - E(t)|JX_{\text{rabbit}}(t) - X_i(t)|, \quad (10)$$

Where $J = 2(1 - r_5)$, r_3, r_4, $r_5 \in (0, 1)$, and $E(t)$ denotes the escaping energy of the prey [21].

In the exploitation phase, the algorithm follows the original HHO framework. For conciseness, only the soft besiege strategy is mathematically presented, while the remaining exploitation behaviours are implemented implicitly. The exploration phase follows two perching strategies as defined in the original HHO framework,

while the exploitation phase is governed by the prey's escaping energy.

3.3. Modified Harris Hawks Optimization (MHHO)

The proposed MHHO algorithm introduces two key modifications to enhance convergence speed and solution quality: 1. Dynamic inertia weighting and 2. Levy flight integration.

Dynamic Inertia Weight:

A dynamic inertia weight is introduced to balance exploration and exploitation:

$$\omega(t) = \omega_{\max} - (\omega_{\max} - \omega_{\min})\frac{t}{T}. \tag{11}$$

The position update is modified as a weighted combination:

$$X_i^{t+1} = \omega_t X_i^t + (1-\omega_t)X_{\text{HHO}} \tag{12}$$

The proposed framework enhances the original HHO algorithm by strengthening both global exploration and local refinement. The dynamic inertia weight introduced in Eq. (11) plays a critical role during the early and intermediate stages of optimization. Initially, a larger inertia weight promotes extensive exploration of the hyperparameter space, allowing the hawks to investigate diverse LSTM configurations and reduce the risk of premature convergence. As iterations progress, the inertia weight gradually decreases, guiding the search toward exploitation and enabling finer adjustments around promising solutions. This adaptive transition improves convergence stability and facilitates the identification of high-quality hyperparameter configurations.

To enhance local exploitation, Levy flight is incorporated during the soft besiege phase as

$$X_i(t+1) = X_i^{\text{new}}(t+1) + \alpha \oplus \text{Levy}(\beta), \tag{13}$$

Where α is the Levy scaling factor, $\beta \in (1,2]$. To To further strengthen local exploitation, a Lévy-flight perturbation mechanism is integrated during the soft besiege phase, as described in Eq. (13). Lévy flight introduces controlled stochastic jumps around promising regions of the search space, enabling the algorithm to escape local optima and explore nearby candidate solutions more effectively. This mechanism is particularly advantageous in high-dimensional and non-convex hyperparameter landscapes. Together, these modifications enable MHHO to achieve a more effective balance between global exploration and local exploitation during the optimization process.

3.4. Hyperparameters Decoding

In the proposed LSTM framework, each candidate solution generated by the optimization algorithm is represented as a three-dimensional continuous position vector:

$$X_i = [x_1, x_2, x_3] \tag{14}$$

where x_1 corresponds to the number of LSTM hidden units, x_2 represents the dropout rate, and x_3 denotes the learning rate. Since LSTM hyperparameters must satisfy specific domain constraints and data-type requirements, a decoding mechanism is applied to transform each position vector into a valid, executable LSTM configuration. Specifically, the number of hidden units is rounded to the nearest integer and clipped to the interval [16, 256], the dropout rate is constrained to [0.1, 0.5], and the learning rate is restricted to the range 10^{-5} to 10^{-2} Batch size is fixed at 128 for all experiments and is therefore excluded from the optimization process to reduce computational overhead and ensure fair comparison across different hyperparameter configurations. As illustrated in Fig. 2, this decoding strategy ensures that each hawk's position corresponds to a feasible LSTM hyperparameter set, maintaining compatibility with practical implementation settings and avoiding invalid or unstable model configurations.

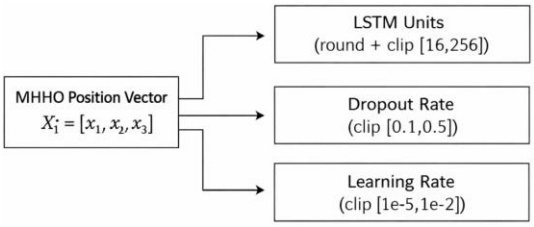

Fig. 2. Hyperparameter Decoding

3.5. Fitness Function

To guide the metaheuristic optimization process, a custom fitness function is designed to explicitly address the severe class imbalance inherent in the attack detection task. Each candidate solution θ, representing a specific set of hyperparameters, is evaluated by training an LSTM model and assessing its predictive performance on a validation set. The model outputs a probability $pi \in [0,1]$ for each validation sample, which is converted into a binary label using a decision threshold. Based on the predicted and true labels, recall and precision are computed as

$$\text{Recall}(\theta) = \frac{TP}{TP+FN} \tag{15}$$

$$\text{Precision}(\theta) = \frac{TP}{TP+FP} \tag{16}$$

Where *TP*, *FN*, and *FP* denote true positives, false negatives, and false positives, respectively.

This weighting scheme reflects the safety-critical nature of the problem, where false negatives are more costly than false positives, while still maintaining awareness of prediction quality through precision.

To further improve training stability and robustness during optimization, cost-sensitive learning is incorporated by applying class weights during model training, thus compensating for the dominance of the majority class. Early stopping based on validation loss is employed to mitigate overfitting and reduce unnecessary computation. In addition, fixed random seeds are used to control stochastic effects originating from weight initialization and optimization dynamics, ensuring more consistent fitness evaluations across iterations.

$$F(\theta) = 0.7 \cdot \text{Recall}(\theta) + 0.3 \cdot \text{Precision}(\theta) \quad (17)$$

Finally, a caching mechanism is used to enhance computational efficiency. Previously evaluated hyperparameter configurations are stored and reused, preventing redundant training and reducing noise in the optimization process. Collectively, these design choices ensure that the fitness function provides a reliable and imbalance-aware objective for guiding both HHO and MHHO toward stable and robust LSTM hyperparameter configurations.

3.6. Time Complexity

To The computational cost of the proposed framework consists of two major components: 1. the cost of training the LSTM model, and 2. the additional overhead introduced by the metaheuristic optimizer.

The baseline LSTM model is trained only once, and its computational cost is fully determined by the training complexity of the network, denoted as C_{train}.

In the case of HHO-based hyperparameter optimization, multiple candidate solutions are evaluated across several iterations. If T represents the number of iterations and N the population size, the overall time complexity of HHO-LSTM becomes:

$$O(T \times N \times C_{train}) \quad (18)$$

The proposed MHHO algorithm introduces two additional mechanisms—Lévy flight exploration and dynamic inertia weighting. These operations require only lightweight vector computations and add a small overhead of:

$$O(T \times N) \quad (19)$$

which is negligible compared to the cost of training the LSTM model. Therefore, the overall complexity of MHHO-LSTM remains dominated by LSTM training:

$$O(T \times N \times C_{train}) + O(T \times N) \approx O(T \times N \times C_{train}) \quad (20)$$

Although MHHO introduces extra exploratory mechanisms, their computational load is minimal. Consequently, the overall complexity of the MHHO-LSTM framework is effectively governed by the cost of training the LSTM model itself.

4. Experimental Setup

4.1. Dataset description and preprocessing

The experiments were conducted on a large-scale network intrusion detection dataset (CICIoMT2024) comprising 7,160,831 samples with 45 numerical features extracted from network traffic patterns. The dataset covers 51 distinct attack categories representing various network intrusion types, including Distributed Denial of Service (DDoS) attacks, Denial of Service (DoS) attacks, MQTT-based attacks, and ARP spoofing, alongside benign network traffic samples [22].

The original dataset exhibits a highly imbalanced multi-class structure with 51 categorical labels. To formulate the problem as a binary classification task distinguishing malicious traffic from benign activity, a binary label transformation was applied. Samples labelled as 'Benign train' were assigned class 0, while all attack categories were collectively assigned class 1. The resulting binary distribution reveals a severe class imbalance, with malicious traffic constituting 97.31% of samples and benign traffic representing only 2.69% of the dataset. This imbalance ratio of approximately 36:1 reflects realistic network traffic scenarios where normal operations substantially outnumber attack instances, thereby presenting significant challenges for classification model training.

Then, to ensure numerical stability and accelerate neural network convergence, all feature values were standardized using z-score normalization.

To reduce computational overhead while preserving the statistical characteristics of the original dataset, representative subsets were constructed from the training and validation sets. Specifically, a stratified random sampling strategy was employed to extract a subset of 50,000 samples from the training data and 50,000 samples from the validation data. Stratification was applied based on the binary class labels to ensure that the class distribution within each subset accurately reflects the original malicious-to-benign ratio of 97.31% to 2.69%.

The use of these subsets enables efficient hyperparameter optimization and repeated model evaluations without sacrificing data representativeness. Given the large-scale nature of the original dataset, training and validating the LSTM model on the full data during metaheuristic optimization would be computationally prohibitive. By operating on stratified subsets, the optimization process remains tractable while preserving the inherent class imbalance and attack characteristics of the IoMT traffic. The validity of the subset selection was verified by comparing class ratios between the subsets and their corresponding full datasets, confirming negligible deviation and ensuring unbiased performance estimation during model tuning.

4.2. Experimental environment

All experiments were carried out on a high-performance workstation equipped with an Intel Core i7-13620H processor, 32 GB of RAM, and an NVIDIA GeForce RTX 4050 GPU with dedicated video memory. The system operated under the Ubuntu Linux environment, providing a stable and efficient platform for deep learning workflows.

The proposed method was implemented in Python. TensorFlow was used to construct and train the LSTM model, with GPU acceleration enabled through the CUDA library to ensure efficient execution of matrix operations and backpropagation. The scikit-learn library supported key data preprocessing steps, including stratified data splitting, feature standardization, and performance metric computation. NumPy was employed extensively for array manipulation and numerical operations throughout the optimization and training pipeline.

To ensure full experimental reproducibility, all random number generators were initialized using a fixed seed value (seed = 42) across TensorFlow, NumPy, and Python's built-in random module. This deterministic configuration guarantees that all experiments can be

replicated with identical outcomes, enabling consistent comparison of algorithmic variants and hyperparameter settings.

5. Results and Discussion

The experiments were conducted on a large-scale network intrusion detection dataset (CICIoMT2024) comprising 7,160,831 samples with 45 features, as summarized in Table I. The dataset exhibits significant class imbalance, with 6,968,099 attack samples (97.3%) and 192,732 benign samples (2.7%).

Table I summarizes the main characteristics of the dataset used in this study. As can be seen, significant class imbalance is observed in the dataset. Specifically, the benign class contains 192,732 samples, while the attack class comprises 6,968,099 instances, accounting for more than 97% of the total samples.

This inverted distribution compared to typical real-world networks (where benign traffic usually exceeds 95%) indicates that CICIoMT2024 is likely a curated dataset, intentionally attack-focused for algorithm performance evaluation.

Additional methodological details were included to further enhance transparency and reproducibility. The dataset was partitioned into training (70%), validation (15%), and testing (15%) subsets prior to stratified sampling, ensuring consistent evaluation across all experimental conditions. Each experiment was repeated five times using a fixed random seed (42) applied uniformly to Python, NumPy, and TensorFlow to eliminate unintended stochastic variation. The stratified subsets used during hyperparameter optimization were constructed to preserve the exact benign-to-attack ratio observed in the full dataset, thereby preventing sampling bias.

Given the class imbalance present in the dataset (as detailed in Table I.), First, a stratified subsampling strategy was applied to the training set, ensuring that the minority and majority classes were represented in the same proportions as in the original data. This approach also substantially reduced computational costs during the optimization process while preventing the distortion of class distributions that could otherwise bias model learning. Then, to directly address the learning bias toward the majority class, a cost-sensitive learning mechanism was incorporated into the fitness function through the use of class weights. By assigning higher misclassification penalties to the minority class, the optimization process explicitly encouraged solutions that improved sensitivity to rare events rather than merely maximizing overall accuracy. This is why, these imbalance-aware strategies enabled the model to effectively learn patterns for both classes. As confirmed by the experimental results, the proposed framework significantly outperformed the baseline LSTM model trained without imbalance handling, yielding higher recall and F1-score. These findings highlight that explicitly addressing class imbalance is not optional but essential for achieving reliable and meaningful performance on this large-scale dataset.

Table I. Dataset Description

Attribute	Description
Number of Samples	7,160,831
Number of Features	45
Number of Classes	2
Benign Class Count	192,732
Attack Class Count	6,968,099

Table II. summarizes the hyperparameter search space used for the models. The number of LSTM units was varied within the range [16,256] to balance model capacity and computational complexity. The learning rate was explored within a continuous range of ($[10^{-5}, 10^{-2}]$) to ensure stable and efficient convergence during training. To reduce overfitting and improve generalization, a dropout rate between 0.1 and 0.5 was applied. Other training parameters, including the batch size, were kept fixed and excluded from optimization because preliminary analysis showed minimal performance sensitivity to batch size compared to its significant impact on computational cost and training stability. Moreover, the batch size was kept fixed due to its relatively low sensitivity on model performance compared to its substantial impact on computational cost and training stability, ensuring a consistent and efficient hyperparameter evaluation process.

Table II. Hyperparameters Search Space

Hyperparameter	Range / Candidate Values
Number of LSTM units	[16,256]
Learning rate	$[10^{-5}, 10^{-2}]$
Dropout rate	[0.1,0.5]

As can be seen in Table III, to ensure reproducibility and performance evaluation, all experiments were conducted over five independent runs with different random seeds. The optimization problem was defined in a three-dimensional search space and solved using a small population of hawks, due to low-dimensional search space, over a maximum of 30 iterations. Performance metrics and execution times were recorded for the baseline, HHO, and MHHO methods.

Table III. Experimental Run Configuration Parameters

Parameter	value
N-Runs	5
dim	3
N-Hawks	8
Max-Iter	30

Fig. 3. compares the mean F1-score and corresponding standard deviation of the Base-LSTM, HHO-LSTM, and MHHO-LSTM models across multiple runs. The Base-LSTM achieves a mean F1-score of approximately 0.9675, with noticeably higher variability, indicating less stable performance across different initializations.

Both metaheuristic-optimized models substantially outperform the baseline. The HHO-LSTM improves the mean F1-score to 0.9785 while significantly reducing performance variability. The proposed MHHO-LSTM achieves the highest mean F1-score ($\approx$0.9785) and the lowest standard deviation, demonstrating superior

stability and consistent convergence across runs. The narrower error bars observed for MHHO-LSTM indicate enhanced robustness compared to both the baseline and the standard HHO-based approach.

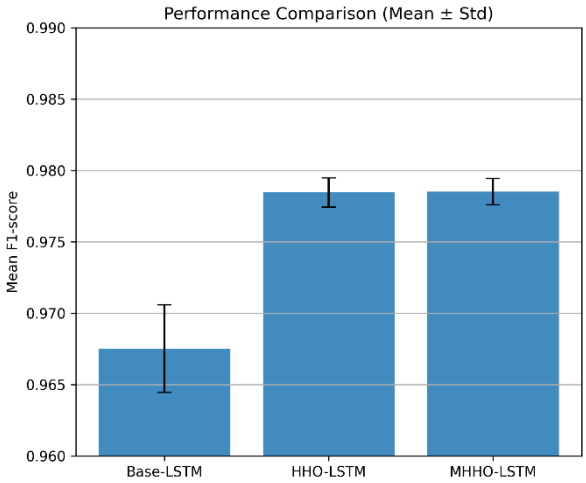

Fig. 3. Performance comparison of the models

These results suggest that the MHHO modification enhances the original HHO algorithm's exploration and exploitation balance, allowing it to locate superior hyperparameter configurations that generalize well across repeated runs. The improved stability is particularly important in imbalanced or noisy datasets, where even small deviations in hyperparameters can lead to large variations in model performance.

Overall, Fig. 3 provides clear evidence that the proposed MHHO-LSTM not only maximizes average performance but also ensures repeatable and dependable results, making it a more trustworthy optimization strategy for deep sequence modeling tasks.

Table IV. shows the visual comparison presented in Fig. 3. by providing a detailed statistical summary of model performance across multiple runs. As shown, the Base-LSTM attains a mean F1-score of 0.9675 with comparatively high variability, indicating sensitivity to initialization and training stochasticity.

Table IV. Performance comparison of Base-LSTM, HHO-LSTM, and MHHO-LSTM across multiple runs

Model	Mean F1-Score	Std F1-Score	Min F1	Max F1
Base-LSTM	0.9675	0.00306	0.9635	0.9709
HHO-LSTM	0.9785	0.00103	0.9769	0.9798
MHHO-LSTM	0.9785	0.00092	0.9772	0.9798

In contrast, both metaheuristic-optimized models demonstrate a substantial improvement in mean F1-score while significantly reducing performance variance. The HHO-LSTM increases the mean F1-score to 0.9785 with a markedly lower standard deviation, whereas the proposed MHHO-LSTM achieves the highest mean F1-score alongside the lowest variability.

To further evaluate the effectiveness of the proposed approach, a comparative analysis was conducted against several machine learning baselines reported in the recent study by Doménech et al. (2025) [13], which investigates intrusion detection performance using the CICIoMT2024 dataset. As summarized in Table V., classical machine learning models reported in that study achieve F1-scores between 0.8595 and 0.9419. In particular, the feed-forward neural network (FNN) achieves an F1-score of 0.8595, while tree-based models such as Decision Tree (DT) and XGBoost (XGB) obtain higher performance with F1-scores of 0.9344 and 0.9419, respectively. In comparison, the deep learning baseline adopted in this study, namely the LSTM model, achieves a higher detection performance with an F1-score of 0.9675, indicating the advantage of sequence-based architectures for modelling network traffic patterns in IoMT environments.

By applying the HHO algorithm for hyperparameter tuning, the HHO-LSTM model further improves the detection capability and achieves an F1-score of 0.9785. The proposed MHHO-LSTM model reaches the same mean F1-score while demonstrating improved stability across multiple experimental runs, reducing the standard deviation from 0.00103 to 0.00092.

Table V. Comprehensive Comparison with Existing Methods on the CICIoMT Dataset

Category	Model	F1-Score	Source
Classical ML	DT	0.9344	Doménech et al.
Classical ML	XGB	0.9419	Doménech et al.
Neural Network	FNN	0.8595	Doménech et al.
Deep Learning	LSTM	0.9675	This study
Optimized DL	HHO-LSTM	0.9785	This study
Proposed	MHHO-LSTM	0.9785	This study

This reduction in performance variability indicates that the proposed hybrid optimization strategy enhances the robustness and reliability of the IDS while maintaining state-of-the-art detection performance for IoMT environments.

As can be seen, Fig. 4. illustrates the distribution of final F1-scores obtained by HHO-LSTM and MHHO-LSTM across multiple independent runs. While both methods achieve high peak performance, the MHHO-LSTM demonstrates a noticeably tighter distribution, reflecting reduced variability and greater consistency in repeated experiments. This compact interquartile range indicates that MHHO-LSTM is less sensitive to stochastic factors, which often affect standard HHO-LSTM and baseline LSTM models. The narrower spread of scores suggests that MHHO-LSTM consistently converges to high-quality hyperparameter configurations, enhancing reliability in practical applications. Overall, Fig. 4 provides clear evidence that the proposed MHHO-LSTM not only maximizes average performance but also

ensures repeatable, dependable results across independent trials. Such stability is particularly critical in large-scale and safety-critical detection systems, where performance fluctuations can significantly impact deployment confidence and operational robustness.

As summarized in Table IV, both HHO-LSTM and MHHO-LSTM achieve comparable mean F1-scores, demonstrating that metaheuristic optimization substantially improves predictive performance over the baseline LSTM. However, Fig. 4 highlights a more detailed perspective by illustrating the distribution of F1-scores across multiple independent runs. The MHHO-LSTM consistently exhibits reduced variability and a more compact interquartile range compared to HHO-LSTM.

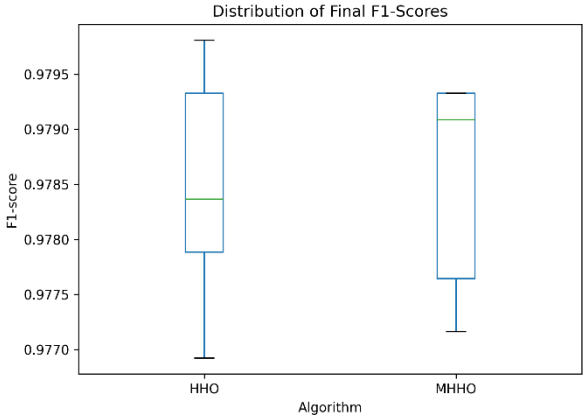

Fig. 4. Distribution of Final F1-Scores

To further quantify robustness, Table VI. presents a run-to-run analysis of the final F1-scores for the baseline LSTM, HHO-LSTM, and MHHO-LSTM models. In addition to reporting the standard deviation, the table includes the worst- and best-case performances observed across runs, providing a comprehensive assessment of each model's reliability.

Table VI. Run-to-run Robustness Analysis

Model	Std_F1	Worst Run F1	Best Run F1
Base	0.0030	0. 9634	0. 97091
HHO	0.0010	0. 9769	0. 9798
MHHO	0.0009	0. 9771	0. 9793

The results reveal that while HHO-LSTM occasionally reaches high peak performance, its variability remains higher than MHHO-LSTM. In contrast, MHHO-LSTM not only maintains competitive peak performance but also ensures tighter score distributions across trials, confirming that the proposed modifications improve both performance and repeatability.

Fig. 5. Represents mean convergence curves of the HHO-LSTM and MHHO-LSTM models in terms of F1-score over 30 optimization iterations, averaged across multiple independent runs. The shaded regions represent the standard deviation, indicating run-to-run variability. HHO-LSTM exhibits a rapid improvement during the initial iterations, followed by an early stabilization, suggesting fast exploitation but limited late-stage improvement. In contrast, MHHO-LSTM demonstrates a more gradual and consistent performance increase with reduced fluctuation in later iterations, indicating improved balance between exploration and exploitation and enhanced convergence stability. The smoother trajectory and tighter variability observed for MHHO-LSTM corroborate its superior robustness as reported in Table VI., which can be attributed to the incorporation of dynamic inertia weights that adaptively regulate the search dynamics throughout the optimization process.

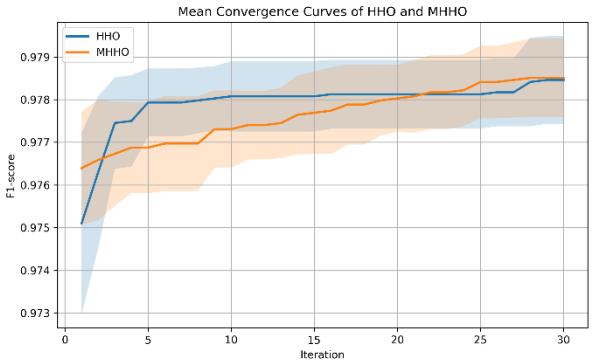

Fig. 5. Mean Convergence Curves

The convergence behaviour observed in Fig. 6 highlights the differing search dynamics of the HHO-LSTM and MHHO-LSTM models. HHO-LSTM demonstrates rapid early-stage improvements, reaching a high F1-score quickly within the first few iterations. This behaviour indicates strong exploitation capability, allowing the algorithm to quickly converge toward promising regions of the search space.

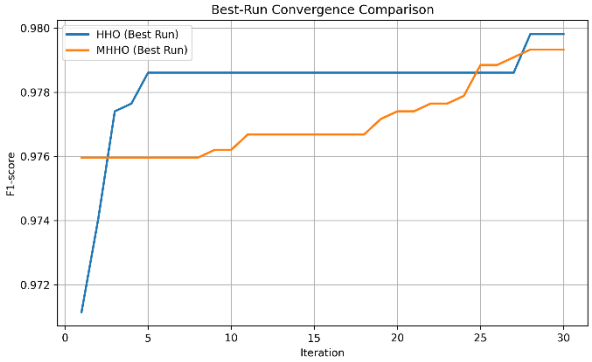

Fig. 6. Best-Run Convergence Comparison

However, after iteration 5, further improvements are minimal, suggesting that HHO may prematurely converge to a local optimum, limiting its ability to explore alternative configurations. In contrast, MHHO-LSTM exhibits a more gradual but sustained improvement throughout the entire optimization process. Its incremental gains in later iterations indicate superior exploration capability, allowing the algorithm to escape potential local optima and refine the hyperparameter configuration more consistently.

Table VII. presents the improvement dynamics of the optimized models with respect to the baseline LSTM classifier. Although the HHO-based model achieves a

higher average improvement (0.3457%), it exhibits notably larger variability, as indicated by its higher standard deviation (0.2747%). In contrast, the proposed MHHO approach demonstrates a lower mean improvement (0.2168%) but with substantially reduced variability (0.1075%), reflecting more stable and consistent performance across runs. The negative gain over HHO (−0.129%) for MHHO suggests a trade-off between peak improvement and robustness, where MHHO prioritizes reliability and convergence stability rather than aggressive short-term gains.

Table VII. Relative improvement statistics of HHO-LSTM and MHHO-LSTM during optimization

Model	Mean Improvement %	Std Improvement %	Gain over HHO %
HHO	0.3457	0.2747	0.0
MHHO	0.2168	0.1075 ↓	-0.128

These results indicate that while HHO tends to achieve higher peak improvements, MHHO offers superior robustness by significantly reducing performance fluctuations. This stability becomes particularly important in imbalanced attack detection scenarios, where consistent recall-oriented behaviour is more critical than occasional maximal accuracy gains.

The convergence analyses further reveal that HHO exhibits faster early exploitation, whereas MHHO follows a smoother and more stable optimization trajectory with sustained late-stage refinement. These findings, supported by convergence plots and relative improvement statistics, confirm that MHHO-LSTM offers a better balance between performance and stability, making it a reliable optimization strategy for imbalanced and stochastic learning scenarios.

To complement the theoretical analysis, Table VIII. summarizes the empirical runtime obtained from five independent experimental runs.

The MHHO method incurs an 11.8% runtime overhead compared to HHO, which is expected due to the added exploration mechanisms.

Although the metaheuristic optimization requires a noticeably higher runtime compared to the baseline LSTM model (1524.50 s for HHO-LSTM and 1703.99 s for MHHO-LSTM), this computational cost is incurred only once during the offline training phase. Such values are fully consistent with hybrid approaches that combine deep learning with population-based optimization, where multiple neural network trainings are performed to evaluate different candidate solutions.

Importantly, the final deployed IDS do not use the optimizer during operation. After the best hyperparameters are obtained, inference is performed using a single LSTM model, resulting in an execution time identical to the baseline model (9.67 s for full training, and sub-second inference time per batch). Therefore, while the optimization phase is computationally intensive, it does not affect real-time performance, deployment costs, or operational latency of the intrusion detection system. This training-only overhead is typical and acceptable in metaheuristic-assisted deep learning frameworks that target improved accuracy rather than faster training.

Table VIII. The empirical runtime obtained from five independent experimental runs

Model	Mean Training Time (s)	Time per Iteration (s)
Base-LSTM	9.67	-
HHO-LSTM	1524.50	101.63
MHHO-LSTM	1703.99	113.60

The scalability of the proposed MHHO-LSTM framework was analyzed in terms of dataset size and search space dimensionality. The complete pipeline consists of data preprocessing, LSTM-based classification, and hyperparameter optimization using HHO and the proposed MHHO variant. Each component exhibits well-defined computational characteristics that collectively determine the overall scalability of the system. From the modeling perspective, the computational complexity of LSTM training is directly influenced by the number of samples N , the number of features d , and the number of hidden units h . The per-epoch training complexity of an LSTM network can be approximated as:

$$O(N(dh + h^2)) \tag{21}$$

Where the term h^2 dominates for moderate and large hidden dimensions. Since the sequence length used in our model is fixed ($T = 1$), the training cost grows linearly with the dataset size. This linear behavior was also confirmed empirically: as the number of training samples increased, the training time increased proportionally without introducing instability or sudden performance degradation. Such a property indicates that the LSTM component of the proposed framework is well-suited for real-time and large-volume network traffic environments.

Regarding the optimization process, both HHO and MHHO operate by evaluating a population of candidate hyperparameter configurations (hawks) over a fixed number of iterations. The computational cost is therefore expressed as:

$$O(N_{hawks} \times Iter \times \mathrm{Cost}_{fitness}) \tag{22}$$

Where each fitness evaluation corresponds to one LSTM training cycle. Importantly, the meta-heuristic enhancements in MHHO—such as dynamic inertia weighting, soft-besiege Lévy perturbations, and improved exploration–exploitation dynamics—introduce only a lightweight overhead on the order of $O(\dim)$. Given that the search space dimensionality in this work is small (i.e., four hyperparameters), these additional mechanisms do not alter the overall computational order of the optimizer. Consequently, MHHO maintains the same scalability characteristics as the classical HHO, while converging more rapidly to high-quality solutions.

Furthermore, the entire optimization stage scales linearly with both the number of hawks and the number of

iterations. This makes the algorithm highly adaptable: increasing population size or search depth can improve accuracy while maintaining predictable and manageable computational growth. Such linear scalability is particularly valuable when deploying the system in large-scale cybersecurity infrastructures where hyperparameter tuning must be computationally feasible.
Fig. 7. Represents, scalability projection of the Baseline LSTM, HHO-LSTM, and MHHO-LSTM models. The execution times were measured on the original training dataset containing 50,000 samples. Since re-running the full hyperparameter optimization on multiple dataset sizes is computationally prohibitive, the scalability trend is estimated analytically by assuming a linear relationship between training time and dataset size, which is consistent with fixed architecture, batch size, and epoch count. The projected results show that all three models scale approximately linearly with the number of samples, while MHHO-LSTM maintains competitive scalability despite integrating additional exploration mechanisms.

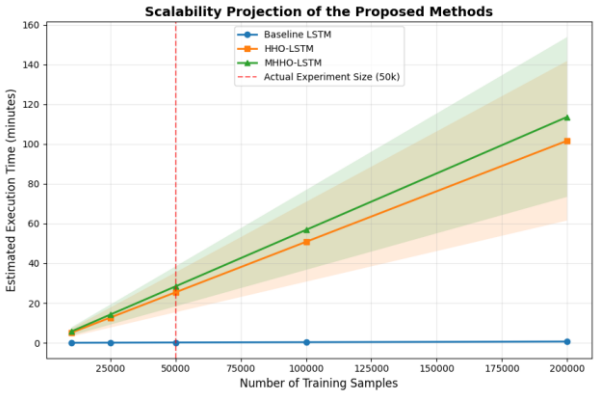

Fig. 7. Scalability Projection of the Proposed Methods

In summary, the comparative analysis shows that the proposed MHHO-LSTM framework provides a more stable and reliable optimization strategy for IoMT intrusion detection. While HHO-LSTM achieves slightly higher peak improvements, its larger variability reflects less consistent behaviour across runs. MHHO-LSTM effectively reduces these fluctuations while maintaining competitive performance, indicating a better balance between exploration and exploitation—an essential property in imbalanced attack-detection scenarios.
The convergence analysis further demonstrates that MHHO yields smoother and more controlled optimization dynamics, enabling sustained refinement in later iterations rather than unstable early-stage jumps. Additionally, scalability results confirm that MHHO-LSTM maintains near-linear computational growth with respect to dataset size and search parameters, making it practical for large-scale and real-time cybersecurity applications. Overall, the findings highlight MHHO-LSTM as a robust and scalable optimization approach for deep learning–based IDSs in dynamic and data-sensitive IoMT environments.

6. Conclusion and Future Work

This paper presented an optimized deep learning framework for large-scale and highly imbalanced intrusion detection by integrating LSTM networks with the HHO algorithm and its enhanced variant, Modified HHO (MHHO), evaluated on the CICIoMT2024 dataset. The proposed approach aims to improve not only detection accuracy but also training stability and optimization reliability in stochastic, high-dimensional search spaces.
Experimental results show that both HHO-LSTM and MHHO-LSTM significantly outperform the baseline LSTM model, achieving mean F1-scores of approximately 0.9785 on the imbalanced IoMT dataset. While both metaheuristic-optimized models reach strong peak performance, MHHO-LSTM demonstrates consistently lower variance across five independent runs, indicating improved stability and robustness.
Further analysis using boxplots and convergence curves confirms that MHHO provides smoother and more controlled optimization dynamics, reducing stochastic fluctuations observed in the standard HHO. These characteristics make MHHO-LSTM particularly suitable for safety-critical and large-scale IoMT environments, where consistent and reliable detection performance is more important than marginal improvements in peak accuracy.
Future work may extend the framework to multi-class attack detection for more detailed threat characterization. In addition, adaptive fitness functions could further enhance optimization under extreme class imbalance. Parallel or distributed implementations of MHHO may also reduce optimization time for very large datasets, while evaluations in real-time or cross-domain IoT/IoMT environments would further validate the generalizability and deployment potential of the proposed approach.

7. References

[1] R. M. Rajab, M. Abuhmida, I. Wilson, and R. P. Ward, “A Review of IoMT Security and Privacy related Frameworks,” *European Conference on Cyber Warfare and Security*, vol. 23, no. 1, pp. 733–743, Jun. 2024, doi: 10.34190/ECCWS.23.1.2239.

[2] G. Akar, S. Sahmoud, M. Onat, U. Cavusoglu, and E. Malondo, “L2D2: A Novel LSTM Model for Multi-Class Intrusion Detection Systems in the Era of IoMT,” *IEEE Access*, vol. 13, pp. 7002–7013, 2025, doi: 10.1109/ACCESS.2025.3526883.

[3] M. Esbati, M. Ahmadieh Khanesar, and A. Shahzadi, “A comparison between optimization algorithms for the tuning of fuzzy based controlled communication for networked controlled systems,” *Tabriz Journal of Electrical Engineering*, vol. 48, no. 4, pp. 1425–1436, Feb. 2019, Accessed: Dec. 25, 2025. [Online]. Available: https://tjee.tabrizu.ac.ir/article_8486_en.html

[4] L. Velasco, H. Guerrero, and A. Hospitaler, “A Literature Review and Critical Analysis of Metaheuristics Recently Developed,” *Archives of Computational Methods in Engineering*, vol. 31, no. 1, pp. 125–146, Jan. 2024, doi: 10.1007/S11831-023-09975-0/TABLES/9.

[5] 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, Aug. 2019, doi: 10.1016/J.FUTURE.2019.02.028.

[6] H. Liu and B. Lang, "Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey," *Applied Sciences 2019, Vol. 9, Page 4396*, vol. 9, no. 20, p. 4396, Oct. 2019, doi: 10.3390/APP9204396.

[7] Y. Rbah *et al.*, "A Comprehensive Survey on Deep Learning Approaches for Safeguarding the Internet of Medical Things from Malicious Intrusions," *2024 International Conference on Circuit, Systems and Communication, ICCSC 2024*, 2024, doi: 10.1109/ICCSC62074.2024.10616971.

[8] M. Shafiq, J. G. Choi, O. Cheikhrouhou, and H. Hamam, "Advances in IoMT for Healthcare Systems," *Sensors 2024, Vol. 24, Page 10*, vol. 24, no. 1, p. 10, Dec. 2023, doi: 10.3390/S24010010.

[9] M. A. Uddin, N. H. Chu, and R. Rafeh, "A Hierarchical IDS for Zero-Day Attack Detection in Internet of Medical Things Networks," Aug. 2025, Accessed: Sep. 30, 2025. [Online]. Available: https://arxiv.org/pdf/2508.10346v1

[10] S. Zhang, "Challenges in KNN Classification," *IEEE Trans Knowl Data Eng*, vol. 34, no. 10, pp. 4663–4675, Oct. 2022, doi: 10.1109/TKDE.2021.3049250.

[11] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent Advances in Recurrent Neural Networks," Dec. 2017, Accessed: Oct. 20, 2025. [Online]. Available: https://arxiv.org/pdf/1801.01078

[12] M. Kaur and A. Mohta, "A Review of Deep Learning with Recurrent Neural Network," *Proceedings of the 2nd International Conference on Smart Systems and Inventive Technology, ICSSIT 2019*, pp. 460–465, Nov. 2019, doi: 10.1109/ICSSIT46314.2019.8987837.

[13] J. Doménech, O. León, M. S. Siddiqui, and J. Pegueroles, "Evaluating and enhancing intrusion detection systems in IoMT: The importance of domain-specific datasets," Internet of Things, vol. 32, p. 101631, 2025. https://doi.org/10.1016/j.iot.2025.101631

[14] Y. Rahmat-Samii, "Genetic algorithm (GA) and particle swarm optimization (PSO) in engineering electromagnetics," *Conference Proceedings - ICECom 2003: 17th International Conference on Applied Electromagnetics and Communications*, pp. 1–5, 2003, doi: 10.1109/ICECOM.2003.1290941.

[15] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput Intell Mag*, vol. 1, no. 4, pp. 28–39, Nov. 2006, doi: 10.1109/MCI.2006.329691.

[16] M. Juneja and S. K. Nagar, "Particle swarm optimization algorithm and its parameters: A review," *ICCCCM 2016 - 2nd IEEE International Conference on Control Computing Communication and Materials*, May 2017, doi: 10.1109/ICCCCM.2016.7918233.

[17] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, Mar. 2014, doi: 10.1016/J.ADVENGSOFT.2013.12.007.

[18] 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, Aug. 2019, doi: 10.1016/J.FUTURE.2019.02.028.

[19] M. Vasoujouybari, E. Ataie, and M. Bastam, "An MLP-based Deep Learning Approach for Detecting DDoS Attacks," Tabriz Journal of Electrical Engineering, Original Article, doi: 10.22034/tjee.2022.15567.

[20] Y. Yu, X. Si, C. Hu, and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *Neural Comput*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019, doi: 10.1162/NECO_A_01199.

[21] M. Shehab *et al.*, "Harris Hawks Optimization Algorithm: Variants and Applications," *Archives of Computational Methods in Engineering 2022 29:7*, vol. 29, no. 7, pp. 5579–5603, Jul. 2022, doi: 10.1007/S11831-022-09780-1.

[22] S. Dadkhah, E. C. P. Neto, R. Ferreira, R. C. Molokwu, S. Sadeghi, and A. A. Ghorbani, "CICIoMT2024: A benchmark dataset for multi-protocol security assessment in IoMT," *Internet of Things*, vol. 28, p. 101351, Dec. 2024, doi: 10.1016/J.IOT.2024.101351.