International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

Intrusion Mitigation-Based Adaptive Resource Allocation Using LSTM-RBF-FFO in Cloud Environments

[1]Mudavath Yuvaraj Naik, [2]C. Sivakumar

[1]Research Scholar, School of Computing, Department of CSE, Mohan Babu University, Tirupati, India.
[2]Associate Professor, School of Computing, Department of CSE, Mohan Babu University, Tirupati, India.

[1]yuvarajnaik538@gmail.com, ORCID iD: 0009-0005-3314-4259,
[2]svkumar650@gmail.com, ORCID iD: 0009-0007-7630-7315,

**Corresponding author: Mudavath Yuvaraj Naik*

Abstract: Cloud computing systems are becoming increasingly dynamic, requiring effective security mechanisms and efficient resource allocation strategies. As cyber threats become more sophisticated, cloud systems must not only detect and mitigate intrusions but also adaptively allocate resources to maintain system performance. Existing optimization-based approaches, such as the Radial Basis Function Firefly (RBF-FF) algorithm, have shown promise in balancing resource utilization and security in cloud environments. However, static RBF models do not adapt effectively to evolving intrusion patterns and workload variations in real time, resulting in delayed threat response and suboptimal resource utilization. Existing approaches generally address intrusion detection and resource allocation independently, limiting their ability to respond effectively to dynamic cloud environments. This paper proposes a Hybrid LSTM-RBF Firefly Optimization (HLR-FFO) framework that integrates Long Short-Term Memory (LSTM) networks, Radial Basis Function (RBF) neural networks, and an adaptive Firefly Optimization algorithm. The LSTM component learns temporal patterns from system logs and network traffic to improve intrusion prediction, while the RBF layer performs rapid and accurate classification. The Firefly Optimization algorithm employs an adaptive brightness decay factor to dynamically allocate cloud resources based on real-time user demand, VM capacity, and detected risk levels. Simulation results obtained over 1000 optimization iterations indicate that HLR-FFO achieves a detection accuracy of 99.1%, a false positive rate of 0.09%, and a false negative rate of 0.17%, demonstrating improved performance compared with GRU-PSO, CNN-LSTM-GA, and RBF-FF under the considered simulation settings. The proposed framework achieves an internal processing latency of 18.7 ms while reducing end-to-end response latency to 103 ms, demonstrating its ability to provide faster response while maintaining effective security and resource utilization.

Keywords: Cloud security, LSTM-RBF hybrid model, Firefly optimization, Intrusion detection, Adaptive resource allocation.

1 INTRODUCTION

Cloud computing has significantly transformed the way computing resources are accessed and utilized. Its rapid growth has enabled a wide range of applications, including edge computing, Internet of Things (IoT) platforms, and enterprise workloads [1]. These applications leverage cloud resources to obtain scalable computing and storage services on demand. Cloud infrastructures support virtualized services, enable efficient resource distribution, and provide flexible access to computing and storage resources [2], [3]. As cloud systems become more widely adopted, ensuring data privacy, trust, and secure resource management has become increasingly important [3].

These threats include unauthorized access, malicious activities, and misuse of cloud resources. Conventional security approaches often struggle to address evolving threats in multi-tenant and virtualized cloud environments, highlighting the need for adaptive security and resource management mechanisms. Although recent studies have improved anomaly detection, accurately identifying intrusions while simultaneously managing cloud resources in a risk-aware manner remains a challenging problem. Accurate anomaly detection is difficult because cloud environments continuously generate large volumes of dynamic real-time data. The high volume and velocity of system logs and network traffic further increase the complexity of real-time intrusion detection [4]. In addition, existing detection methods often produce false positives and missed attacks when operating under diverse and dynamic cloud conditions. This limitation becomes more pronounced when detection algorithms are required to handle diverse and evolving attack vectors [5].

Furthermore, continuously changing workloads and threat levels reduce the effectiveness of static resource allocation strategies used in cloud orchestration. As a result, cloud systems may experience reduced service quality, increased response time, and higher operational risk [6]. To address these challenges, this work aims to develop a unified framework capable of accurately detecting intrusions while adaptively allocating cloud resources according to the identified threat levels.

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

Machine learning approaches such as Support Vector Machines (SVMs), Decision Trees (DTs), Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM) networks have demonstrated promising intrusion detection capabilities. However, many of these approaches primarily focus on detection and provide limited support for adaptive resource management under dynamic cloud conditions [7]. Hybrid models such as CNN-LSTM-GA and GRU-PSO improve prediction performance; however, their computational complexity and limited adaptation to dynamic cloud traffic remain important challenges [8], [9]. Similarly, optimization techniques such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) may not consistently achieve effective resource allocation under varying attack conditions [10]. One limitation is that these optimization strategies generally do not adapt their allocation decisions according to dynamically changing risk levels. These limitations motivate the development of an integrated framework capable of supporting both intrusion detection and adaptive resource allocation with low computational overhead.

The primary objective of this work is to develop a framework that integrates deep learning, neural network-based classification, and bio-inspired optimization to support intrusion detection and adaptive resource allocation in cloud environments. The proposed framework combines Long Short-Term Memory (LSTM) and Radial Basis Function (RBF) models for intrusion detection. The LSTM network captures temporal dependencies in system logs and network traffic, while the RBF layer performs nonlinear classification based on the extracted temporal features. To support adaptive resource allocation, the framework incorporates an Adaptive Firefly Optimization (AFO). Using the risk estimates derived from the prediction scores, the AFO dynamically adjusts resource allocation according to the current threat level. Unlike existing methods that perform intrusion detection and resource allocation independently, the proposed HLR-FFO jointly integrates temporal intrusion prediction, nonlinear classification, adaptive Firefly optimization, risk-aware resource allocation, and continuous online retraining within a unified cloud management framework.

2 Related Work

Security-aware resource management and efficient resource allocation remain two major challenges in modern cloud and edge computing environments. To address these challenges, researchers have proposed hybrid frameworks that combine deep learning with optimization techniques for intelligent resource management. Such frameworks aim to improve the responsiveness, scalability, and security of cloud computing environments. Integrating optimization algorithms with deep learning models has become a common approach for improving resource allocation and task scheduling in cloud environments. For example, the RATS-HM framework [11] combines ICS-TS for task scheduling with GO-DNN for resource allocation, enabling simultaneous optimization of both processes. The proposed RATS-HM framework reduces makespan time while improving resource utilization. Similarly, the Hybrid Lyrebird Falcon Optimization (HLFO) model employs intelligent optimization techniques for dynamic virtual machine management.

The model dynamically classifies virtual machines (VMs) according to their workload conditions. It subsequently performs multi-objective optimization while considering quality-of-service constraints such as task priority and energy consumption. Similarly, D. Alqahtani et al. [12] proposed a hierarchical two-stage workload prediction framework that combines signal decomposition techniques with CNN and Bi-LSTM networks to capture complex temporal dependencies in cloud workloads. By integrating CEEMDAN and Variational Mode Decomposition (VMD), the framework effectively reduces noise and improves workload prediction accuracy across dynamic cloud environments.

Deep learning models have been widely investigated for predicting future resource demand in cloud computing environments. S. Simaiya et al. [13] proposed a hybrid deep learning framework that combines Particle Swarm Optimization and Genetic Algorithm (PSO-GA) with CNN-LSTM for dynamic cloud workload provisioning. The hybrid PSO-GA approach is used to optimize model hyperparameters, while the CNN-LSTM network predicts virtual machine workloads to improve resource utilization, load balancing, and service-level agreement (SLA) compliance in cloud environments. Similarly, the study in [14] proposed an IoT–fog–cloud resource allocation framework that integrates an Enhanced Support Vector Machine (ESVM) with the Communication and Energy Integration for Latency Improvement (CAELI) algorithm to improve link reliability, energy efficiency, and Quality of Service (QoS) through adaptive multi-objective optimization. The study in [15] proposed a security-aware deep reinforcement learning (DRL) framework that jointly optimizes cloud resource allocation and security monitoring through an adaptive reward function, improving resource utilization while maintaining high threat detection performance under dynamic attack scenarios.

Wang et al. [16] proposed a differential privacy-based joint task offloading and resource allocation strategy for vehicular edge computing, demonstrating that secure resource allocation can be achieved without compromising user privacy. C. Dong et al. [17] proposed a self-adaptive Deep Neural Network (DNN) partitioning framework with cost-effective resource allocation for collaborative computation between IoT devices and edge servers, improving task execution efficiency while reducing resource cost and inference latency. The study in [18] proposed a machine learning-based framework for cloud resource allocation and task scheduling that predicts future workload variations using historical system performance data. The framework dynamically adjusts resource allocation to improve load balancing, reduce latency, enhance resource utilization, and support scalable cloud computing environments.

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

S. Ahmadi [19] presented a comprehensive review of machine learning applications in data warehousing, highlighting its role in workload management, query optimization, automated data management, and adaptive resource allocation. The study emphasizes that machine learning can improve operational efficiency while supporting predictive analytics and intelligent resource management, although challenges related to data privacy, security, and resource constraints remain. K. B. Rajesh Naidu and J. Kumaran Kumar [20] presented a comprehensive survey of artificial intelligence-based approaches for predictive resource scaling in cloud environments, highlighting the use of machine learning, deep learning, ensemble models, and metaheuristic optimization to improve resource allocation accuracy, energy efficiency, and service-level agreement (SLA) compliance. The survey also identifies key challenges and future research directions for intelligent and adaptive cloud resource management.

S. Priyadarshini et al. [21] investigated secure and scalable resource allocation for AI/ML workloads in cloud environments by integrating advanced workload scheduling, parallel computing, and security mechanisms. The proposed framework combines adaptive batch stream scheduling, hybrid optimization algorithms, and secure computation techniques to improve resource utilization, scalability, and workload performance while maintaining data confidentiality. Experimental results demonstrated improvements in both accuracy and throughput for cloud-based AI/ML applications. Building upon these studies, recent research has increasingly focused on developing intelligent, adaptive, and multi-layered frameworks for cloud and edge computing environments [22], [18], [23]. These studies include security-aware DRL frameworks, privacy-preserving cloud and IoT architectures, and hybrid scheduling models that combine deep learning with metaheuristic optimization techniques [17], [18], [23], [19].

Collectively, these studies demonstrate the growing importance of integrating intelligent prediction, optimization, and security mechanisms for efficient cloud resource management. Despite these advances, most existing studies address workload prediction, intrusion detection, and resource allocation as separate problems or optimize only one aspect of cloud management [15]-[23]. Limited attention has been given to integrating temporal intrusion detection with risk-aware adaptive resource allocation within a unified framework that can respond dynamically to both workload variations and security threats. This limitation motivates the development of the proposed HLR-FFO framework.

3 Proposed Method

The proposed HLR-FFO framework integrates intrusion detection and adaptive resource allocation within a unified cloud management workflow. The framework first employs a hybrid LSTM-RBF neural network to detect intrusion patterns from temporal cloud monitoring data. The predicted risk level, together with service demand, is subsequently used by the AFO to determine appropriate resource allocation. Each firefly represents a candidate resource allocation configuration, and its brightness is determined by system load, predicted risk score, and energy efficiency. An adaptive brightness decay mechanism improves convergence while allowing the optimization process to respond to changing workload conditions. The framework operates continuously by updating the detection and resource allocation modules using streaming cloud data, thereby maintaining detection performance and resource management effectiveness over time. The overall workflow of the proposed HLR-FFO framework is illustrated in Fig. 1.

3.1. Initialization

The proposed HLR-FFO framework begins by initializing three major components: the cloud system configuration, the LSTM-RBF neural network parameters, and the AFO settings. These initialization settings establish the computational environment and learning parameters required for the framework to operate effectively in a real-time cloud environment. Proper initialization ensures consistent model training and stable optimization during cloud resource management.

3.2. System Configuration Setup

Cloud computing environments are inherently heterogeneous, consisting of multiple virtual machines (VMs), diverse services, and dynamic user workloads. Table 1 summarizes the configuration of the simulated cloud environment used for experimental evaluation. The system starts by loading these configurations into the management module, which keeps track of VM status (busy, idle, compromised), service request volumes, and the network throughput.

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

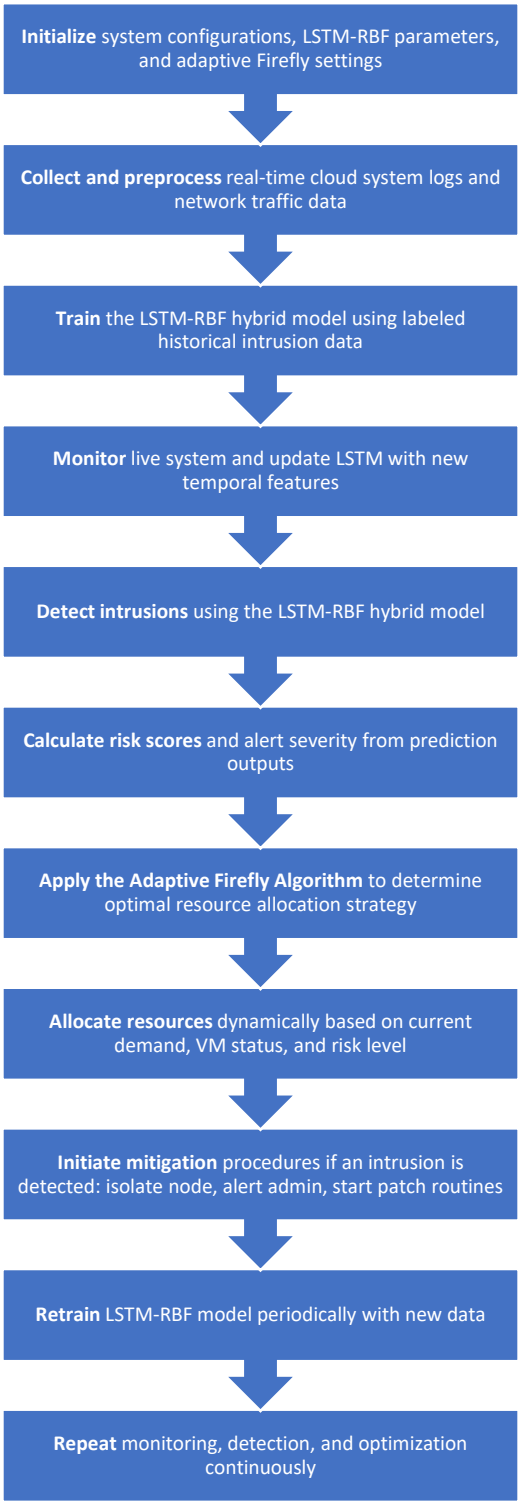

Fig. 1. Overall workflow of the proposed HLR-FFO framework

Table 1. Cloud Environment Configuration Parameters

Parameter	Value
Number of Virtual Machines (VMs)	150
User Request Rate (req/sec)	250 to 1500
VM Memory Range	2 GB – 64 GB
CPU Cores per Virtual Machine	2 – 8 cores
Network Bandwidth	1 Gbps
Monitoring Interval	5 seconds

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

Let the LSTM cell equations be defined as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (3)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (4)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t * tanh(C_t) \quad (6)$$

where x_t denotes the input vector at time t, h_t represents the hidden state, C_t is the cell state, f_t, and o_t denote the forget, input, and output gates, respectively, W and b represent trainable weights and bias vectors, while $\sigma(.)$ and $tanh(.)$ denote the sigmoid and hyperbolic tangent activation functions. The hidden state ht produced by the LSTM is used as the input feature vector for the RBF classifier (i.e., $x = h_t$). The output of the RBF network is therefore defined as:

$$y(x) = \sum_{i=1}^{N} w_i \cdot \phi(\|x - c_i\|), \quad \phi(r) = e^{-\beta r^2} \quad (7)$$

Where c_i is the center of the i-th radial basis function, β is the spread factor $\left(\text{usually } \beta = \frac{1}{2\sigma^2}\right)$, w_i are trainable weights, $\phi(\cdot)$ is the Gaussian activation function. Table 2 presents initialized values for LSTM-RBF hyperparameters.

Table 2. LSTM-RBF Neural Network Initialization Parameters

Parameter	Value
LSTM Hidden Units	128
Number of Time Steps	10
RBF Kernel Type	Gaussian
RBF Centers (N)	50
Spread (σ)	1.0
Learning Rate	0.001
Loss Function	Categorical Cross-Entropy

3.4. Adaptive Firefly Parameter Initialization

The proposed framework employs an AFO, in which the brightness decay factor, γ(t), is dynamically adjusted during optimization to improve resource allocation under changing cloud workloads. The firefly movement rule is given by:

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma(t) r_{ij}^2}(x_j^t - x_i^t) + \alpha(rand - 0.5) \quad (8)$$

where r_{ij} denotes the Euclidean distance between fireflies i and j, x_i and x_j are firefly positions (resource configurations), β_0 is initial attractiveness, $\gamma(t) = \gamma_0/(1 + \lambda t)$ is the adaptive brightness decay, α is a randomization parameter. This formulation promotes exploration during early iterations and gradually shifts toward exploitation. Table 3 lists the FFA parameters initialized for optimization.

Table 3. Adaptive Firefly Optimization Parameters

Parameter	Value
Population Size	100 Fireflies
Initial Attractiveness (β₀)	1.0
Brightness Decay (γ₀)	0.1
Adaptation Rate (λ)	0.01
Randomization Coefficient (α)	0.2
Optimization Iterations	1000

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

The adaptive decay factor balances global exploration during early iterations with local exploitation as the optimization converges.

4 PREPROCESSING

The performance of the LSTM-RBF model largely depends on the quality of the collected cloud monitoring data.

4.1. Data Sources and Features

Logs and network traffic are collected from user sessions, VM status, API calls, and network packets. Key features extracted for model training are listed in Table 4.

Table 4. Extracted Features from Real-Time Cloud Logs

Feature Name	Description
CPU Utilization	CPU usage per VM (%)
Memory Utilization	Memory usage per VM (%)
Pkt_In_Rate	Packets received/sec
Pkt_Out_Rate	Packets sent/sec
Failed Logins	Count of failed user login attempts
API Error Count	Number of failed API requests
Session Duration	Duration of user sessions (in seconds)
Intrusion Label	1 for intrusion, 0 for normal

4.2. Data Preprocessing Pipeline

Collected data is passed through several preprocessing steps:

- **Normalization**: Feature scaling using Min-Max normalization:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{9}$$

- **Encoding**: Categorical fields like user roles or attack types are one-hot encoded.
- **Temporal Structuring**: Data is framed in sequences of 10 time steps for LSTM ingestion.
- **Label Alignment**: Intrusion labels are matched against time windows for supervised learning.

The preprocessed data forms the input X_t for time-series classification, enabling the LSTM-RBF model to learn contextual dependencies and anomaly patterns.

4.3. Dataset and Labeling

Historical cloud system data is collected from security logs, NetFlow traffic, and the VM resource records. Each entry in the dataset includes features like CPU usage, failed logins, network packet rates, and the memory consumption, labeled as either "intrusion" or "normal." As seen in Table 5, each sequence of time steps is used to construct temporal windows (sliding window approach) for input to the LSTM network.

Table 5. Labelled Training Data for Intrusion Patterns

Time Step	CPU Utilization	Memory Utilization	Failed Logins	Pkt_In_Rate	Label
t-3	62%	70%	0	320	0 (Normal)
t-2	80%	82%	1	450	1 (Intrusion)
t-1	78%	85%	3	470	1 (Intrusion)
t	65%	72%	0	280	0 (Normal)

5 PROPOSED HLR-FFO FRAMEWORK

5.1. Model Architecture and Training

The proposed LSTM-RBF architecture consists of:

- An LSTM layer with 128 hidden units for temporal feature extraction.
- A Radial Basis Function (RBF) layer that performs the final classification.

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

Each input sequence, $X = [x_{t-n}, x_{t-n+1}, \ldots, x_t]$, is processed by the LSTM, which updates the memory cell C_t and hidden state h_t at each time step. The LSTM updates its internal states using the standard gating equations introduced in Section 3.3. The final hidden state h_t captures the temporal characteristics of cloud behavior and is subsequently provided to the RBF classifier. The RBF classifier maps h_t to a decision surface using:

$$\phi(x) = e^{-\beta \|x - c_i\|^2} \tag{10}$$

$$y = \sum_{i=1}^{N} w_i \cdot \phi(\|h_t - c_i\|^2) \tag{11}$$

where ϕ is the Gaussian function, c_i are centers of radial basis functions, β is the spread factor, w_i are output weights. The objective is to minimize classification loss L, typically using categorical cross-entropy:

$$L = -\sum_{k=1}^{K} y_k \cdot log(\hat{y}_k) \tag{12}$$

The LSTM-RBF hybrid network is trained using labeled historical cloud data containing both normal and intrusion patterns. This supervised training process enables the model to learn temporal dependencies and discriminative feature representations for accurate intrusion classification. The training process is configured using the following settings to ensure stable learning and effective model generalization:

- The labeled dataset is divided into 80% training data and 20% validation data.
- The network weights, biases, and RBF centers are optimized using backpropagation through time (BPTT) and gradient-based optimization.
- Early stopping and dropout regularization are employed to reduce overfitting and improve model generalization.

5.2. Online Monitoring and Real-Time Data Acquisition

Once the model is trained, it is deployed for online monitoring, where it continuously analyzes real-time cloud telemetry and responds to evolving threat patterns. The framework continuously receives logs, API requests, packet flow data, and VM performance metrics through cloud telemetry platforms such as Ceilometer and Prometheus. A time-series buffer stores the most recent T observations, where T denotes the sequence length used as input to the LSTM module.

Table 6. Live Telemetry Buffer Snapshot

Time	CPU Utilization	Memory Utilization	Failed Logins	Pkt_In_Rate
t-2	58%	65%	0	310
t-1	61%	66%	0	315
t	64%	67%	1	318

The buffered sequence (Table 6) is periodically provided to the trained LSTM, enabling temporal feature extraction for subsequent intrusion detection.

5.3. Incremental Model Update

For each incoming sequence in the live buffer, the LSTM updates its internal state and generates the hidden representation h_t , which captures the evolving behavior of the cloud environment. Newly predicted samples with confidence greater than 0.95 are treated as pseudo-labeled data. To maintain detection performance under evolving workload and attack patterns, the framework periodically performs semi-supervised model updates through the following mechanisms:

- Confidence-based pseudo-labeling is employed to assign labels to newly detected patterns with high prediction confidence.
- After a predefined interval (e.g., every 1000 monitoring steps), the LSTM-RBF model is incrementally retrained using both historical and pseudo-labeled data.

This strategy enables the framework to adapt to evolving cloud behavior while reducing the need for continuous manual annotation.

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

5.4. Detect Intrusions Using LSTM-RBF Hybrid Model

Once the updated sequence is processed by the LSTM, the resulting hidden representation h_t is passed to the RBF layer, which generates a probability distribution over the two classes: {Normal, Intrusion}. Let:

$$\hat{y} = \text{RBF}(h_t) = [P(\text{Normal}), P(\text{Intrusion})] \tag{13}$$

A softmax activation ensures probability normalization:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{14}$$

The Softmax function converts the classifier outputs into normalized posterior probabilities for each class. If the predicted intrusion probability exceeds a predefined threshold θ (set to 0.6 in the present implementation), the corresponding sequence is classified as an intrusion. The threshold was selected empirically to balance detection sensitivity and false alarms.

Table 7. Real-Time Intrusion Detection Output

Time	Prediction	Confidence Score	Action Taken
t-3 to t	Intrusion	0.87	VM Isolated
t-2 to t	Normal	0.93	Continue
t-1 to t	Intrusion	0.78	Alert Admin

As shown in Table 7, the LSTM-RBF model produces high-confidence intrusion predictions. The predicted class labels and confidence scores are subsequently used for risk assessment and adaptive response in the following stages of the HLR-FFO framework.

5.5. Risk Assessment

After the LSTM-RBF model classifies each input sequence, the prediction output is converted into a risk score and mapped to an alert severity level to support subsequent resource allocation decisions. The output of the RBF layer is a softmax probability vector:

$$\hat{y} = [P(\text{Normal}), P(\text{Intrusion})] \tag{15}$$

The risk score R_t is calculated as a weighted function of:

- Intrusion probability P_I
- Number of failed logins F
- API error rate E
- Packet rate anomaly A

$$R_t = \alpha_1 P_I + \alpha_2 \cdot \text{Norm}(F) + \alpha_3 \cdot \text{Norm}(E) + \alpha_4 \cdot \text{Norm}(A) \tag{16}$$

where $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1$, Norm(·) indicates normalized feature value in [0,1]. The weighted formulation combines prediction confidence with operational indicators to provide a comprehensive estimate of system risk. The weights were empirically selected based on preliminary experiments and security expert heuristics. Table 8 presents representative risk score calculations obtained using the proposed risk assessment model under different cloud operating conditions.

Table 8. Risk Score Computation

Time	P (Intrusion)	Failed Logins	API Errors	Anomaly (Pkt_Rate)	Risk Score Rt
t1	0.92	4	3	High	0.86
t2	0.31	1	1	Low	0.29

Based on the risk score R_t, alert severity is assigned as:

$$\text{Severity} = \begin{cases} \text{High}, & R_t \geq 0.75 \\ \text{Medium}, & 0.4 \leq R_t < 0.75 \\ \text{Low}, & R_t < 0.4 \end{cases} \tag{17}$$

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

The resulting alert severity levels corresponding to different risk scores are summarized in Table 9.

Table 9. Alert Severity Levels from the Risk Scores

Risk Score Rt	Alert Severity
0.86	High
0.29	Low
0.52	Medium

These severity levels are subsequently used to prioritize VM isolation, resource throttling, and bandwidth reallocation during adaptive resource management.

5.6. Adaptive Firefly Optimization

Once the risk scores have been computed, the AFO is invoked to optimize resource allocation across the cloud environment. Each firefly x_i represents a candidate resource allocation vector:

$$x_i = [\mathrm{CPU}_1, \mathrm{CPU}_2, \ldots, \mathrm{RAM}_1, \mathrm{RAM}_2, \ldots, \mathrm{BW}_1, \mathrm{BW}_2, \ldots] \quad (18)$$

where CPU, RAM, and the Bandwidth (BW) allocations are defined per VM or service unit. Each dimension corresponds to the resource allocation of one VM or service instance. The fitness function $f(x)$ evaluates each firefly based on:

$$f(x) = \omega_1 \cdot U(x) + \omega_2 \cdot S(x) - \omega_3 \cdot C(x) \quad (19)$$

Here $U(x)$ is the average normalized CPU, RAM, and bandwidth utilization, $S(x)$ is the normalized security score derived from the risk model, and $C(x)$ represents normalized allocation cost.

$$\omega_1 + \omega_2 + \omega_3 = 1 \quad (20)$$

The movement equation of fireflies is:

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma(t) r_{ij}^2} (x_j^t - x_i^t) + \alpha \cdot (\mathrm{rand} - 0.5) \quad (21)$$

Here $\gamma(t) = \gamma_0/(1 + \lambda t)$ is the adaptive brightness decay, r_{ij} is the distance between fireflies x_i and x_j.

Table 10: Firefly Fitness Comparison (Top 3 Candidates)

Firefly ID	CPU Allocation (%)	RAM Allocation (%)	BW Allocation (%)	Risk-Aware Score	Fitness Value
FF-07	[70, 60, 50]	[80, 60, 60]	[90, 85, 75]	0.92	0.842
FF-13	[75, 55, 60]	[85, 65, 60]	[80, 90, 80]	0.88	0.825
FF-22	[60, 60, 60]	[75, 75, 75]	[70, 70, 70]	0.79	0.794

Fireflies with higher fitness values represent better allocation plans and are used to guide the swarm's movement. Once the optimal resource vector is chosen by the AFO, it is applied to the cloud environment in real time. This step ensures that:

- High-risk VMs are isolated or throttled.
- Trusted VMs receive additional CPU, RAM, and bandwidth resources when operating under heavy workloads.
- Overall system cost is kept minimal.

Allocation is governed by the following rule set:

$$\mathrm{Allocated\_Resource}_i = \begin{cases} \text{Reduce by } \delta, & \text{if } \mathrm{R}_i \geq 0.75 \\ \text{Maintain}, & 0.4 \leq R_i < 0.75 \\ \text{Increase by } \delta, & R_i < 0.4 \text{ and Utilization} > \theta \end{cases} \quad (22)$$

here δ is the adjustment factor (e.g., 10%), and the θ is the utilization threshold (e.g., 80%). Based on the resource allocation rule, representative resource adjustment decisions are presented in Table 11.

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

Table 11. Resource Adjustment Based on Risk Level

VM ID	Risk Score Ri	Current CPU (%)	Action	New CPU (%)
VM-12	0.82	75	Reduce by 10%	67.5
VM-19	0.31	85	Increase by 10%	93.5
VM-23	0.55	65	Maintain	65

This dynamic allocation strategy ensures that resource allocation follows the computed risk levels, preventing service degradation due to both underutilization and oversaturation during intrusion responses. Following resource allocation, the framework initiates mitigation actions for confirmed intrusions based on the computed risk level. These actions follow a severity-based decision framework as shown in Table 12.

Table 12. Mitigation Strategy Based on Risk Score

Risk Score Rt	Action	Sample
Rt ≥ 0.75	Isolate VM, Alert Admin, Patch	Detected Malware Bot
0.4 ≤ Rt < 0.75	Alert Admin, Throttle VM	Brute Force Attempt
Rt < 0.4	Log Event, Monitor Continuously	Normal or Benign Spike

For high-severity threats, the system isolates the affected VM by:

- Disabling external IP access
- Restricting internal VM-to-VM communication
- Blocking ports via SDN controllers or firewall policies

The isolation decision matrix is governed by:

$$\text{Isolation}_i = \begin{cases} 1, & \text{if } R_i \geq 0.75 \text{ and } P_{\text{Intrusion}} \geq 0.8 \\ 0, & \text{otherwise} \end{cases} \quad (23)$$

Simultaneously, a high-priority alert is sent to administrators via email or webhook (Slack, PagerDuty). If the signature of the intrusion matches a known vulnerability (e.g., from CVE database), the system triggers automated patch scripts or applies container image updates. Examples of mitigation actions generated by the proposed framework are presented in Table 13.

Table 13. Real-Time Mitigation Log

VM ID	Intrusion Type	Risk Score	Action Taken	Time
VM-103	Ransomware	0.93	Isolated, Alert Sent	13:22:10
VM-024	Port Scan	0.68	Throttled, Monitored	13:25:05
VM-075	Normal Spike	0.29	Logged, No Action	13:30:42

Mitigation reduces blast radius and enhances the overall resilience of the cloud system.

5.7. Retraining LSTM-RBF

Given the evolving nature of cloud workloads and intrusion patterns, the LSTM-RBF model is periodically retrained to maintain its detection accuracy and adaptability. The framework maintains a continuously updated buffer of recently observed sequences consisting of:

- Confirmed intrusions (true positives)
- Mistaken classifications (false positives and false negatives)
- Evolving workload patterns under normal operation

Table 14. Data for Retraining Buffer

Time Window	Feature Vector Snapshot	Label (Confirmed)
t-5 to t	[78, 80, 1, 420, 0.9]	Intrusion
t-4 to t	[55, 60, 0, 300, 0.1]	Normal
t-3 to t	[68, 75, 2, 510, 0.85]	Intrusion

This data is used to fine-tune the model using online or incremental learning techniques. For the LSTM, weights are adjusted using backpropagation through time (BPTT) on newly labeled sequences. For the RBF layer, centroid updates are performed:

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

$$c_i^{\text{new}} = c_i^{\text{old}} + \eta\left(x_{\text{new}} - c_i^{\text{old}}\right) \quad (24)$$

where η is the learning rate and c_i is the center of the i-th radial basis neuron. Retraining is triggered every T minutes (e.g., 60 minutes) or after accumulating N new labeled samples (e.g., 1000). A validation set (from previous clean data) is used to avoid catastrophic forgetting, ensuring the model doesn't lose previously learned patterns during updates. The HLR-FFO framework operates as a continuous feedback loop, enabling uninterrupted monitoring, intrusion detection, adaptive resource allocation, and periodic model retraining. This closed-loop design enhances adaptability while maintaining low operational latency with minimal manual intervention. Overall detection-to-response latency L_{total} is composed of:

$$L_{\text{total}} = L_{\text{monitoring}} + L_{\text{inference}} + L_{\text{optimization}} + L_{\text{execution}} \quad (25)$$

The overall detection-to-response latency is obtained by summing the monitoring, inference, optimization, and mitigation execution times. The proposed HLR-FFO framework maintains low processing latency through continuous monitoring and periodic model retraining. This feedback mechanism enables the framework to adapt to evolving workloads while sustaining reliable intrusion detection and adaptive resource allocation.

6 Results and Discussion

The proposed HLR-FFO framework is compared with three representative baseline methods, namely GRU-PSO, CNN-LSTM-GA, and RBF-FF, under identical simulation conditions. To evaluate the effectiveness of the proposed HLR-FFO framework, a comprehensive simulation environment was developed using CloudSim Plus, a widely used cloud computing simulation platform. The simulation environment models a multi-data-center cloud infrastructure in which dynamic resource allocation, intrusion detection, and intrusion mitigation can be evaluated under realistic workload conditions. The hardware specifications and simulation parameters used in the experimental evaluation are summarized in Tables 15 and 16, respectively. These baseline methods were selected because of their relevance to temporal modeling and resource optimization in cloud computing environments. To ensure a fair comparison, all methods were evaluated under identical workload conditions.

Table 15. Hardware Specifications

Component	Specification
Processor	Intel Core i7 @ 2.8 GHz
Operating System	Windows 10 Pro 64-bit
RAM	8 GB DDR4
Hard Disk	1 TB HDD
Simulation Tool	CloudSim Plus (v4.0)
Java version	Java 11

Table 16. Simulation Parameters

Component	Parameter	Value/Range
Number of Cloudlets	Task Length	1600 – 3400
	Total Tasks	30 – 300
Virtual Machines	Hosts per Data Center	8
Physical Machine	Memory (GB)	64
	Bandwidth (Gbps)	1
	Storage	500 GB
	Number of Iterations	1000
Data Center Setup	Risk-Based Isolation	Enabled
Attack Types Simulated	DoS, R2L, U2R, Probe	4 types
Dataset Used	NSL-KDD + Real Logs	Combined

The virtual machines described in Section 3 were distributed across the available physical hosts during simulation. Real operational logs from the cloud environment were combined with the NSL-KDD dataset. The bandwidth parameter (1 Gbps) represents the configured host communication capacity within the CloudSim Plus simulation environment. Cloud operational logs were generated during CloudSim Plus simulation and combined with the NSL-KDD dataset to emulate realistic cloud operating conditions. Security events were synthetically injected into the CloudSim-generated workload following NSL-KDD attack distributions.

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

6.1. Performance Metrics

The performance of the proposed HLR-FFO framework is evaluated using standard classification and system performance metrics that quantify detection capability, reliability, and computational efficiency.

1. *Intrusion Detection Rate (IDR)*: Intrusion Detection Rate (IDR) measures the proportion of actual intrusion instances that are correctly identified by the proposed framework. Higher IDR values indicate better intrusion detection capability and are computed as:

$$\text{IDR} = \left(\frac{TP}{TP + FN}\right) \times 100 \tag{26}$$

where TP is True Positives and FN is False Negatives.

2. *False Positive Rate (FPR)*: FPR measures the proportion of normal traffic incorrectly classified as an intrusion. A lower FPR is desirable as it reflects reliability by minimizing false alarms:

$$\text{FPR} = \left(\frac{FP}{FP + TN}\right) \times 100 \tag{27}$$

where FP is False Positives and TN is True Negatives.

3. *False Negative Rate (FNR)*: FNR is the proportion of actual intrusions that were not detected by the model. A lower FNR is important for robust security in the cloud environments:

$$\text{FNR} = \left(\frac{FN}{TP + FN}\right) \times 100 \tag{28}$$

4. *Latency (ms)*: Latency denotes the end-to-end response time between the arrival of a cloud event and the execution of the corresponding detection, resource allocation, or mitigation action. Lower latency indicates faster system responsiveness.
5. Accuracy (ACC): Accuracy measures the proportion of correctly classified instances among all evaluated samples and is computed as:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \tag{29}$$

Accuracy measures overall classification correctness while IDR measures the proportion of actual intrusions correctly detected.

Table 17. Performance Before and After Retraining

Metric	Before Retrain	After Retrain
Accuracy (%)	96.8	99.1
False Positives (%)	0.23	0.09
False Negatives (%)	0.62	0.17

Table 17 compares the classification performance before and after periodic retraining of the LSTM-RBF model. The results indicate that incremental retraining improves the overall detection performance by enabling the model to adapt to evolving workload characteristics and newly observed intrusion patterns. Consequently, classification accuracy increases, while both false positive and false negative rates are reduced. To evaluate the computational efficiency of the proposed framework, the execution time of each internal processing stage was measured. The resulting component-wise processing latency is summarized in Table 18.

Table 18. Component-wise Internal Processing Latency

Component	Average Latency (ms)
Monitoring	4.2
LSTM-RBF Inference	3.8
Firefly Optimization	5.6
Mitigation Execution	5.1
Total	**18.7 ms**

Table 18 summarizes the average execution time of each processing stage within the proposed HLR-FFO framework. The results indicate that the proposed HLR-FFO framework achieves an average end-to-end response latency of 18.7 ms, demonstrating its suitability for real-time cloud intrusion detection and adaptive resource management.

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

The comparative performance of the proposed HLR-FFO framework is evaluated against existing methods under the same experimental conditions. Fig. 2 to 5 illustrate the behavior of the proposed framework with respect to intrusion detection capability, false alarm rate, missed intrusion rate, and response latency. The RBF-FF baseline was implemented by combining an RBF neural network-based intrusion detection model [24] with a Firefly optimization strategy [25] reported in the literature to provide a representative optimization-assisted RBF classifier for comparison.

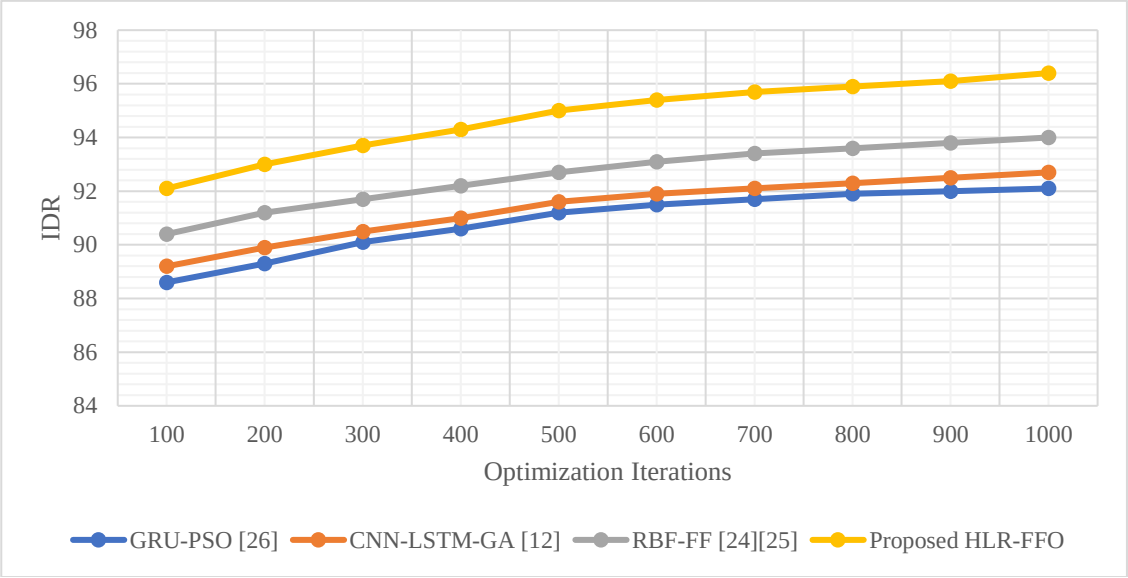

Fig. 2. Intrusion Detection Rate (%)

Fig. 2 compares the intrusion detection rate achieved by the proposed HLR-FFO framework with the selected baseline methods over different numbers of optimization iterations. The proposed method consistently records a higher detection rate throughout the evaluation and reaches 96.4% at 1000 iterations. This improvement can be attributed to the combined learning capability of the LSTM-RBF model together with the adaptive resource optimization strategy, which enables more reliable identification of intrusion patterns under varying cloud workloads.

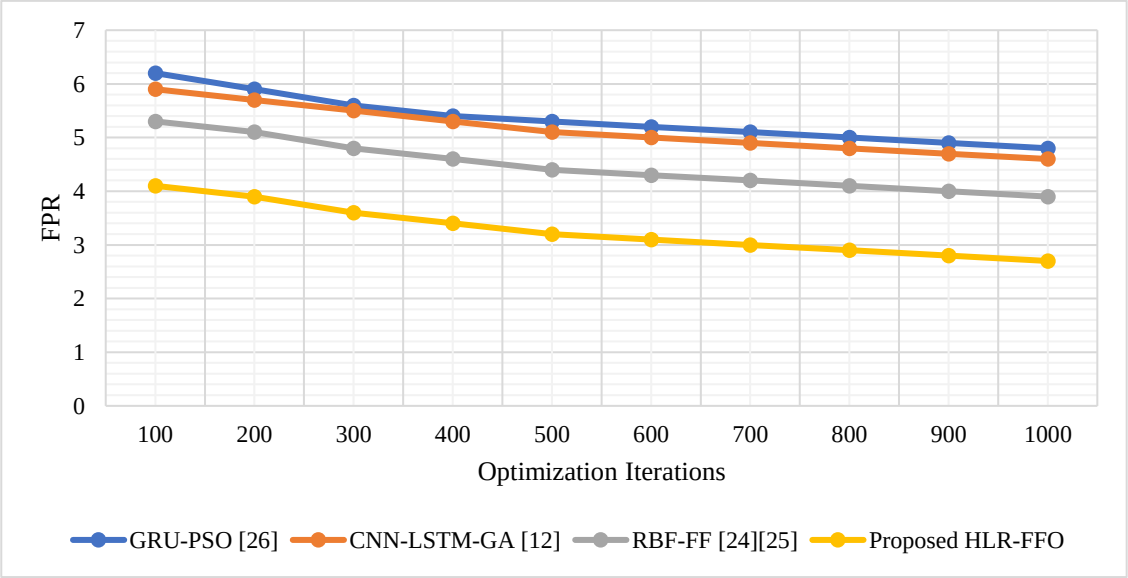

Fig. 3. False Positive Rate (%)

The false positive rates obtained by different methods are presented in Fig. 3. The proposed framework maintains a lower false positive rate across all iterations and achieves the minimum value of 2.7% at the final iteration. A lower false positive rate reduces unnecessary alerts and avoids needless resource reallocation, thereby improving the stability of cloud operations.

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

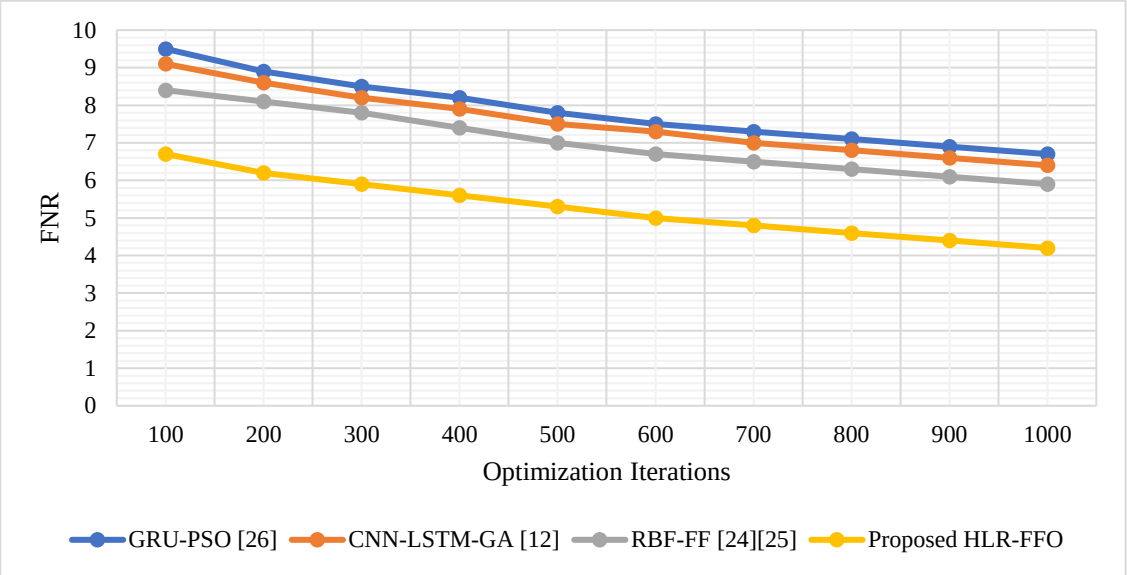

Fig. 4. False Negative Rate (%)

Fig. 4 illustrates the false negative rate of the compared methods. The proposed HLR-FFO framework produces the lowest false negative rate, indicating that fewer intrusion attempts remain undetected. This behaviour suggests that the hybrid LSTM-RBF model is able to preserve detection accuracy while adapting to changing traffic patterns in the cloud environment.

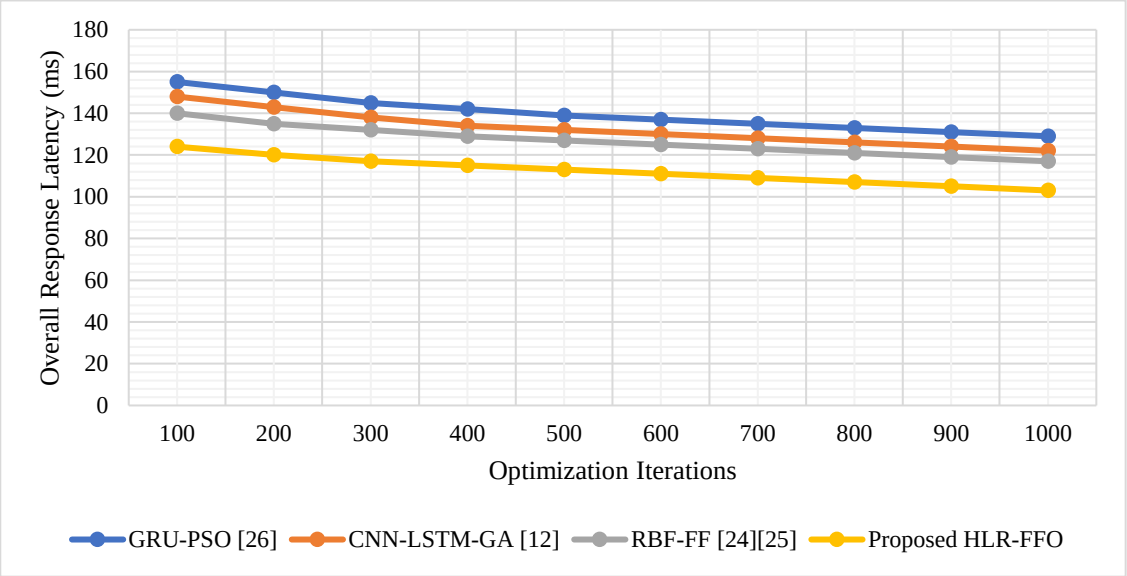

Fig. 5. Overall Response Latency

The overall response latency of the proposed framework and the existing methods is compared in Fig. 5. The proposed HLR-FFO framework consistently requires less response time and records a latency of 103 ms at 1000 iterations. The reduced latency indicates that the proposed detection and resource allocation strategy can respond more efficiently under dynamic cloud workloads without introducing additional processing overhead. Table 18 reports internal processing latency, whereas Fig. 5 reports end-to-end operational latency including monitoring and cloud execution delays.

Overall, the experimental results demonstrate that the proposed HLR-FFO framework consistently achieves improved intrusion detection performance while maintaining lower false alarm rates and reduced response latency. These findings indicate that the integration of the LSTM-RBF model with AFO provides an effective and practical solution for secure and efficient cloud resource management under dynamic operating conditions. The current study is limited to simulation-based evaluation and does not include deployment in a production-scale cloud infrastructure.

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

7 CONCLUSION

The proposed HLR-FFO framework addresses the challenges of real-time intrusion detection and adaptive resource allocation in cloud computing environments. The framework combines the temporal learning capability of the Long Short-Term Memory (LSTM) network with the nonlinear classification capability of the Radial Basis Function (RBF) network to effectively identify intrusion patterns from cloud monitoring data. Based on the predicted intrusion probability and associated risk score, the Adaptive Firefly Optimization dynamically optimizes resource allocation to improve system efficiency while maintaining security. The experimental evaluation indicates that the proposed framework improves intrusion detection performance while reducing false positive rate, false negative rate, and response latency compared with the selected baseline methods. In addition, the continuous feedback and periodic retraining mechanism enables the framework to adapt to evolving workload characteristics and emerging intrusion patterns, thereby supporting reliable operation in dynamic cloud environments. Future work will focus on validating the proposed framework using larger real-world cloud datasets, investigating its scalability in large-scale cloud infrastructures, and extending its applicability to heterogeneous cloud-edge computing environments.

FUNDING INFORMATION

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

ETHICS STATEMENT

This study did not involve human or animal subjects and, therefore, did not require ethical approval.

STATEMENT OF CONFLICT OF INTERESTS

The authors declare no conflicts of interest related to this study.

REFERENCES

[1] F. C. Andriulo, M. Fiore, M. Mongiello, E. Traversa, and V. Zizzo, “Edge Computing and Cloud Computing for Internet of Things: a review,” *Informatics*, vol. 11, no. 4, p. 71, Sep. 2024, doi: 10.3390/informatics11040071.

[2] Y. Sun and H. Jung, “Machine Learning (ML) Modeling, IoT, and Optimizing Organizational Operations through Integrated Strategies: The Role of Technology and Human Resource Management,” *Sustainability*, vol. 16, no. 16, p. 6751, Aug. 2024, doi: 10.3390/su16166751.

[3] V. Winkler, “Cloud Computing Architecture,” in *Elsevier eBooks*, 2011, pp. 29–53. doi: 10.1016/b978-1-59749-592-9.00002-6.

[4] M. Fahimullah, S. Ahvar, M. Agarwal, and M. Trocan, “Machine learning-based solutions for resource management in fog computing,” *Multimedia Tools and Applications*, vol. 83, no. 8, pp. 23019–23045, Aug. 2023, doi: 10.1007/s11042-023-16399-2.

[5] N. Z. Rizqullah, J. Alekhine, D. L. Yonia, R. M. Racel Purnomo and A. M. Shiddiqi, "Enhancing Intrusion Detection Systems with Adaptive Learning Techniques," *2024 IEEE International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)*, Bandung, Indonesia, 2024, pp. 1-6, doi: 10.1109/AIMS61812.2024.10513076.

[6] R. Dugyala *et al.*, “Secure cloud computing: leveraging GNN and leader K-means for intrusion detection optimization,” *Scientific Reports*, vol. 14, no. 1, p. 30906, Dec. 2024, doi: 10.1038/s41598-024-81442-7.

[7] Y. Wang and S. Xing, “AI-Driven CPU resource management in cloud operating systems,” *Journal of Computer and Communications*, vol. 13, no. 06, pp. 135–149, Jan. 2025, doi: 10.4236/jcc.2025.136009.

[8] Imane Briki, Rachid Ellaia, Maryam Alami Chentoufi, "A PSO-Driven Hyperparameter Optimization Approach for GRU-Based Traffic Flow Prediction," *Engineering Proceedings*, vol .112, no. 1, 2025, doi:10.3390/engproc2025112078.

[9] Y. Sanjalawe, S. Fraihat, S. Al-E’mari, M. Abualhaj, S. Makhadmeh, and E. Alzubi, “Smart load balancing in cloud computing: Integrating feature selection with advanced deep learning models,” *PLoS ONE*, vol. 20, no. 9, p. e0329765, Sep. 2025, doi: 10.1371/journal.pone.0329765.

[10] L. Abualigah, “Particle Swarm Optimization: advances, applications, and experimental insights,” *Computers, Materials & Continua/Computers, Materials & Continua (Print)*, vol. 82, no. 2, pp. 1539–1592, Jan. 2025, doi: 10.32604/cmc.2025.060765.

[11] P. K. Bal, S. K. Mohapatra, T. K. Das, K. Srinivasan, and Y.-C. Hu, “A Joint Resource Allocation, Security with Efficient Task Scheduling in Cloud Computing Using Hybrid Machine Learning Techniques,” *Sensors*, vol. 22, no. 3, p. 1242, Feb. 2022, doi: 10.3390/s22031242.

[12] D. Alqahtani, H. Imani and T. El-Ghazawi, “Enhanced Workload Prediction in Data Centers Using Two-Stage Decomposition and Hybrid Parallel Deep Learning,” in *IEEE Access*, vol. 13, pp. 64115-64132, 2025, doi: 10.1109/ACCESS.2025.3558743.

International Journal of Emerging Research in Engineering, Science, and Management
Vol. 5 | Issue 2 | pp. 148-163 | Apr-Jun 2026.
www.ijeresm.com | eISSN – 2583-4894
Received: 14 Mar 2026 | Revised: 16 Jun 2026 | Accepted: 22 Jun 2026 | Published: 26 Jun 2026

[13] S. Simaiya *et al.*, "A hybrid cloud load balancing and host utilization prediction method using deep learning and optimization techniques," *Scientific Reports*, vol. 14, no. 1, p. 1337, Jan. 2024, doi: 10.1038/s41598-024-51466-0.

[14] M. S. Lakshmi *et al.*, "Optimizing resource allocation and link reliability in IoT–FOG–Cloud networks using machine learning and Multi-Objective algorithms," *Journal of Robotics and Control (JRC)*, vol. 6, no. 4, pp. 1817–1832, Jul. 2025, doi: 10.18196/jrc.v6i4.25026.

[15] Y. Chen, E. Feng, and Z. Ling, "Secure resource allocation optimization in cloud computing using deep reinforcement learning," *Journal of Advanced Computing Systems*, vol. 4, no. 11, pp. 15–29, Nov. 2024, doi: 10.69987/jacs.2024.41102.

[16] C. Dong, S. Hu, X. Chen and W. Wen, "Joint Optimization With DNN Partitioning and Resource Allocation in Mobile Edge Computing," in *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 3973-3986, Dec. 2021, doi: 10.1109/TNSM.2021.3116665.

[17] S. Wang, J. Li, G. Wu, H. Chen and S. Sun, "Joint Optimization of Task Offloading and Resource Allocation Based on Differential Privacy in Vehicular Edge Computing," in *IEEE Transactions on Computational Social Systems*, vol. 9, no. 1, pp. 109-119, Feb. 2022, doi: 10.1109/TCSS.2021.3074949.

[18] Anand Singh Rajawat, S. B. Goyal, Manoj Kumar, Varun Malik, "Adaptive resource allocation and optimization in cloud environments: Leveraging machine learning for efficient computing," in *Applied Data Science and Smart Systems*, CRC Press, 2024.

[19] S. Ahmadi, "Optimizing Data Warehousing Performance through Machine Learning Algorithms in the Cloud," *International Journal of Science and Research (IJSR)*, vol. 12, no. 12, pp. 1859–1867, Dec. 2023, doi: 10.21275/sr231224074241.

[20] K. B. Rajesh Naidu and J. Kumaran Kumar, "Deep Learning and Optimization Algorithms for Intelligent Cloud Resource Allocation: A Survey," *2025 5th International Conference on Trends in Material Science and Inventive Materials (ICTMIM)*, Kanyakumari, India, 2025, pp. 1487-1494, doi: 10.1109/ICTMIM65579.2025.10988100.

[21] S. Priyadarshini, T. N. Sawant, G. B. Yadav, J. Premalatha, and S. R. Pawar, "Enhancing security and scalability by AI/ML workload optimization in the cloud," *Cluster Computing*, vol. 27, no. 10, pp. 13455–13469, Jun. 2024, doi: 10.1007/s10586-024-04641-x.

[22] Q. W. Ahmed *et al.*, "AI-Based Resource Allocation Techniques in Wireless Sensor Internet of Things Networks in Energy Efficiency with Data Optimization," *Electronics*, vol. 11, no. 13, p. 2071, Jul. 2022, doi: 10.3390/electronics11132071.

[23] E. Al. T. Kamble, "Predictive resource allocation strategies for cloud computing environments using machine learning," *Journal of Electrical Systems*, vol. 19, no. 2, pp. 68–77, Jan. 2024, doi: 10.52783/jes.692.

[24] M. Lopez-Martin, A. Sanchez-Esguevillas, J. I. Arribas and B. Carro, "Network Intrusion Detection Based on Extended RBF Neural Network With Offline Reinforcement Learning," in *IEEE Access*, vol. 9, pp. 153153-153170, 2021, doi: 10.1109/ACCESS.2021.3127689.

[25] L. Jovanovic *et al*., "The XGBoost Tuning by Improved Firefly Algorithm for Network Intrusion Detection," *2022 24th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Hagenberg / Linz, Austria, 2022, pp. 268-275, doi: 10.1109/SYNASC57785.2022.00050.

[26] A. N. Suvarna and P. Kanchan, "PSO and GRU or Intrusion Detection in IoT: An Optimization Based Approach," *2025 International Conference on Emerging Technologies in Electronics and Green Energy (ICETEG)*, MYSORE, India, 2025, pp. 1-6, doi: 10.1109/ICETEG66194.2025.11473133.