INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGIES
BULGARIAN ACADEMY OF SCIENCES
CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **26**, No 1
Sofia • 2026 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2026-0005

Leveraging Lightweight Machine Learning for RF Jamming Detection in Mobile ad-hoc Networks: A Three-Tier Edge-Fog-Cloud Computing Approach

Hakim Bessouf [1], *Abderrahmane Baadache*[2], *Fouzi Semchedine*[3]

[1]*Research Unit LaMOS, Faculty of Exact Sciences, Department of Computer Science, University of Bejaia, 06000, Bejaia, Algeria*
[2]*Research Unit LaMOS, University of Bejaia, 06000, Algeria, Department of Computer Science, University of Algiers, 16000, Algiers, Algeria*
[3]*Department of Computer Science, University of Setif, 19000, Algeria*
E-mails: hakim.bessouf@univ-bejaia.dz (*corresponding author*) *a.baadache@univ-alger.dz fouzi.semcheddine@univ-setif.dz*

***Abstract*:** *Radio Frequency* (*RF*) *jamming is a prevalent threat in Mobile Ad-hoc NETworks* (*MANETs*), *yet effective detection on resource-constrained nodes remains challenging. This paper presents a distributed three-tier edge-fog-cloud framework leveraging lightweight machine learning for real-time detection across heterogeneous devices. A priority-based adaptation algorithm dynamically allocates computational tasks based on node capabilities. The framework achieves detection accuracies of 85.6%* (*edge/ lightweight Random Forest*), *88.8%* (*fog/compressed Random Forest*), *and 97.8%* (*cloud/LightGBM*). *The dynamic adaptation mechanism enables efficient load distribution, reducing energy consumption by 66.9% compared to static cloud deployment with only an 8.2% accuracy cost. Cross-dataset validation on two independent physical-layer datasets – covering weak jamming and deceptive attacks under vehicular mobility* (*10-60 km/h*) *– demonstrates exceptional robustness with <1% mean accuracy variance, validating the framework's effectiveness in dynamic, resource-constrained MANET environments.*

***Keywords*:** *Mobile Ad-Hoc Networks* (*MANETs*), *RF jamming detection, Lightweight machine learning, Edge-fog-cloud computing, Dynamic adaptation.*

1. Introduction

Mobile Ad-hoc NETworks (MANETs) can form networks without infrastructure, making them useful in disasters, military operations, and IoT systems. However, their open topology also renders them highly vulnerable to RF jamming attacks, which can cause packet loss rates exceeding 70% and disrupt end-to-end communication. These DoS attacks weaken the network and harm its performance, so there is a need for smart and adaptive detection methods, as demonstrated by V a d l a m a n i et al. [1]

The inherent heterogeneity of MANET nodes – spanning low-power IoT sensors with limited RAM and processing capabilities to high-performance embedded platforms – presents a fundamental challenge for deploying effective security solutions. Using Machine learning models for jamming detection requires heavy computation, making them unsuitable for the resource-limited edge devices, which account for most nodes in the network. This gap between detection requirements and node capabilities calls for a distributed intelligence approach that can adapt to the wide range of devices in modern MANETs.

To solve this problem, we propose a new three-tier edge-fog-cloud framework. This framework uses lightweight machine learning algorithms in a layered computing structure. Our method includes a dynamic adaptation algorithm (Algorithm 4) that continuously adjusts model complexity based on the operational state of each node, available resources, and the severity of threats. Edge nodes (Class A) run very simple models like lightweight Random Forest as proposed by Mishra et al. [2] and Naive Bayes as described by Rish [3]. Fog nodes (Class B) use compressed Random Forest models following Biau and Scornet [4], while cloud nodes (Class C) run full-featured LightGBM as developed by Ke et al. [5] and XGBoost as presented by Chen et al. [6]. Our system dynamically assigns detection tasks based on each node's computing power. This allows real-time RF jamming detection to run on different types of devices, while also lowering energy use and response time.

This research presents a three-tier cyber-physical framework validated through comprehensive cross-dataset experimentation. We evaluate the framework on two independent physical-layer datasets: the Alhazbi, Sciancalepore and Oligeri [7] controlled RF jamming dataset and the Hanegraaf, Sciancalepore and Oligeri [29] mobile weak-jamming dataset (vehicular speeds, deceptive attacks). Cross-dataset results demonstrate <1% mean accuracy variance, confirming generalization across jamming types, environmental conditions, and mobility patterns. Our dynamic adaptation mechanism achieves a 97.5% success rate for trigger detection with real-time edge-tier latency of 13-42 ms.

2. Related work

Initial research concentrated on empirical characterization of RF jamming effects on vehicular ad hoc networks. Among these, Punal et al. [8] created reference metrics for measuring resilience by modeling how packet delivery ratio degrades under increased interference, and as people move around. These studies offered important insights into how jamming affects link reliability and communication range. However, they primarily employed observational methods and didn't propose practical detection methods for use in real-time Mobile Ad hoc Networks (MANETs).

Supervised learning approaches in Puñal et al. [9], Upadhyaya, Sun and Sikdar [10], and Feng and Hua [11] achieve more than 95% detection accuracy by using Random Forest and SVM classifiers in IEEE 802.11, IoT, and

smart-city networks. However, these methods require a substantial amount of labeled data and rely on a stable wireless connection. In MANETs, where nodes move around, performance gets worse because of Doppler shifts and changes in network layout. This causes the data to change over time, so the models become less accurate and need to be retrained often. As a result, this limits the use of these models on edge nodes that have limited resources.

Some unsupervised approaches, like those used in Karagiannis and Argyriou [12] and Nguyen-Minh, Hoang and Pham [13], worked with clustering and anomaly detection in order to avoid labeled data requirements, improving scalability in VANETs. However, it is still hard to distinguish between intentional jamming and legitimate interference (traffic density, fading, multipath). This leads to higher false-positive rates in crowded urban areas where channel changes look like attack signatures.

Edge-based approaches in Hussain et al. [14] try to reduce detection latency through local inference. In a similar vein, the edge-fog-cloud distributed architecture, as presented by Hedjaz, Baadache and Semchedine [15] for DDoS detection, illustrates the capacity of hierarchical intelligence to harmonize adaptive defense mechanisms, resource optimization, and responsiveness throughout distributed layers. Their implementation of threshold-guided lightweight classification at the edge, DNN validation within the fog, and deep classifier functionality at the cloud layer exemplifies the efficacy of tiered security strategies within distributed computing contexts, from another point of view, Mowla et al. [16] with federated learning tried to facilitate distributed updates without sharing raw data, even assuming homogeneous device capabilities and introducing synchronization overhead, making them unsuitable for heterogeneous MANETs also these edge frameworks often lack mobility-aware adaptation, limiting robustness under dynamic topology and Doppler effects.

Several domain-specific uses have been proposed, including spectrogram-based UAV detection by Li et al. [17], selective mitigation strategies by Joo et al. [18], and PCA-based feature selection by Panitsas et al. [19]. However, these methods often require a lot of computing power or struggle to work well in Mobile Ad hoc Networks (MANETs) and characterized by Doppler shifts, changing network structures, and differences in hardware.

During dataset selection, we observed a significant scarcity of suitable datasets, which hinders the effective use of machine learning algorithms on real-world data. The dataset introduced by Alhazbi, Sciancalepore and Oligeri [7] offers comprehensive measurements of RF jamming at the physical layer, the dataset capturing distinct features in multiple jamming scenarios, with varying signal conditions and balanced class distribution. which offers the necessary features for training our lightweight models by providing the multi-dimensional feature space required for our three-tier model design. It enabled systematic feature reduction – from full cloud features to compressed fog features and minimal edge features – while preserving acceptable detection accuracy at each level.

Existing approaches have made notable progress in areas such as classification, distributed learning, and RF signal characterization. However, despite these advances, they still fail to meet three key requirements essential for MANET deployment: Lightweight detection suitable for severely resource-constrained nodes (≤512 MB RAM) through hierarchical model allocation and dynamic adaptability to time-varying resource availability and threat intensity through priority-based adaptation algorithms and scalable, distributed execution across heterogeneous devices with a 10:1 capability ratio without assuming homogeneity.

This study addresses existing gaps by using our three-tier ***cloud-fog-edge*** framework. This framework includes ***dynamic model selection*** (Algorithm 4), which is designed to be ***energy-efficient*** and support ***real-time detection***. At the same time, it ensures that deployment is feasible across the full range of capabilities found in MANET nodes.

3. Proposed framework

3.1. Problem context and formal definition

Mobile Ad-hoc NETworks (MANETs) are used in serious situations like military missions, disaster areas, and IoT systems. They don't need fixed infrastructure, but this also makes them open to attacks such as RF jamming. Even medium-level jamming – when interference is 6-20 dB.m above normal noise – can cause big problems: fewer packets get through, and delays increase. These issues happen at the signal level, not because devices are moving, which means we can detect jamming by looking for unusual signals.

Another difficulty is that MANETs have many different kinds of devices. Some are very simple with only 256 MB of RAM, while others are powerful with lots of memory. This makes it hard to use the same Machine Learning (ML) detection method everywhere. Basic machine learning models use too much memory and are slow on simple devices. Even methods like federated learning – where work is shared between devices – need a lot of talking back and forth. That takes too much time for MANETs, which have to react fast.

3.1.1. Formal problem formulation

Let the MANET be modeled as a dynamic graph $M(t) = (N(t), E(t))$
where $N(t) = \{n_1, n_2, ..., n_k\}$ represents nodes with heterogeneous computational capacities $C_i \in [C_{\min}, C_{\max}]$ and energy reserves $E_i \in [E_{\min}, E_{\max}]$.

Define a three-tier hierarchy T = {edge, fog, cloud} with tier assignment $\tau\, N \to T$ based on capability thresholds.

Let $J = \{I, D, \psi\}$ denote a jamming attack with Intensity I, Duration D, and signal type $\psi \in$ {silent, tone, gaussian}. Given physical-layer features $X(t)$ extracted before packet decoding, each tier operates on a feature subset $X\_\tau \subseteq X$, where $|X\_\text{edge}| < |X\_\text{fog}| < |X\_\text{cloud}|$, with tier-specific detection function

$$f\_\tau: (n_i, X\_\tau) \to \{0, 1\}.$$

The adaptation function α: (n_i, $R(t)$, $\Theta(t)$) $\rightarrow \tau$ dynamically reassigns nodes based on resource state $R(t)$ and threat intensity $\Theta(t)$.

Objective. Design $\{f\_\tau, \alpha\}$ to maximize ensemble accuracy $A(F) \geq 95\%$ while maintaining bounded computational complexity, subject to:

(1) $\text{comp}(f_\tau, n_i) \leq C_i$ (computational capacity constraint),

(2) $\text{energy}(f\_\tau, n_i) \leq E_i$ (energy reserve constraint),

(3) $\text{Time}(f_\tau) = O(\text{complexity}_\tau)$,

where complexity_edge $\prec$ complexity_fog $\prec$ complexity_cloud.

3.1.2. Computational complexity analysis

During inference, the proposed tiered detection method is computationally efficient. This is because lightweight Random Forest models make predictions by moving from the starting point to the end point. As a result, the time it takes to predict for each sample mainly depends on the tree's depth *d*, which is often expressed as $O(d)$. In balanced trees, *d* grows logarithmically with the number of tree nodes. However, in our implementation, it is limited by a set maximum depth. For a Random Forest with *k* trees, the prediction process repeats the traversal for each tree. Therefore, the time it takes to infer each sample is roughly $O(k, d)$. This allows us to directly control latency by adjusting either the number of trees *k*, or their maximum depth *d*. This defined complexity directly relates to the framework's design goal, as described in Section 3.1.1, which is to enable real-time detection across different devices while keeping comp(f_τ, n_i) $\leq C_i$. The model's memory usage depends on the number of nodes stored in each tree and the total number of trees. For a binary tree with a maximum depth of *d*, the total number of nodes is limited to $2^{(d+1)} - 1$. As noted by Quadrianto and Ghahramani [20], reducing *d* and/or the number of estimators *k* is the main way to meet the resource limitations of edge and fog computing.

Our tiered strategy sets different complexity levels for each layer. Edge models use depth 5 and up to 10 trees, with inference time $O(50)$. Fog models use depth 8 and 70 trees, with inference time $O(560)$. Cloud models use depth 10-12 and 150-200 trees, with inference time $O(1800)$ – $O(2400)$. This keeps edge inference fast because prediction cost depends only on depth. The adaptation function α is a simple rule set. It runs once per cycle and uses constant time.

Training is conducted on more powerful fog or cloud-based resources. Because the cost of training depends on the specific learning method and how it's implemented, it's considered an offline cost compared to edge inference.

This study examines how to detect jamming in Mobile Ad hoc NETworks (MANETs) with 50 to 200 nodes. In these networks, about 30% of the devices have very limited resources, while the other 70% have moderate to high processing and memory capabilities.

3.2. Distributed computing architecture rationale

Among challenges that arise in MANETs, we can cite: dynamic topology and resource-constrained nodes, and real-time security demands. These challenges necessitate a robust and adaptive computational infrastructure. Traditional centralized cloud solutions give good mechanisms but with high latencies, for real-

time applications, while deploying sophisticated detection algorithms directly on resource-limited nodes is often infeasible due to limited computational power and energy reserves. Hoffpauir et al. [21] demonstrated that edge intelligence is widely motivated by the need to reduce latency and cloud dependence by enabling local decision-making, which in turn increases the importance of lightweight machine learning suitable for resource-limited devices.

In Wang et al. [22] showed how recent advancements in Mobile Edge Computing (MEC) have shown how distributed architectures can improve performance for applications that need low latency which is done by dynamically offloading tasks and balancing workloads across different devices and Hedjaz, Baadache and Semchedine [15] validates the edge-fog-cloud paradigm for security applications by demonstrating that hierarchical detection frameworks can achieve high accuracy: 99.6% at edge, 99.97% at fog, while maintaining low latency through coordinated multi-layer defense strategies. These principles provide a strong foundation for our framework's approach, which dynamically assigns lightweight ML tasks across different network levels.

3.3. Three-tier hierarchical architecture

Our framework uses a three-tier hierarchical architecture that can handle different types of nodes while still being able to detect jamming across the whole network (Fig.1). The architecture comprises three tiers.

Tier 1. Edge nodes (Resource-constrained devices). Devices with limited resources, such as IoT sensors and embedded modules with 512 MB or less of RAM, perform very lightweight inference. They use only a few radio frequency features, including RSSI variance, packet delivery ratio, and channel occupancy. Verma [23] said that "tailoring models to the specific constraints of edge devices (e.g., low power, limited RAM, and limited computational resources) can lead to highly efficient models". Models are designed with bounded complexity $O(d \leq 5)$ to ensure minimal inference latency and memory footprint suitable for immediate detection without cloud connectivity.

Algorithm 1. Edge Node Core Operations

Step 1. Load Lightweight_RandomForest_Model()
Step 2. WHILE active DO
Step 3. features ← Extract_RF_Features(signal)
Step 4. prediction ← model.predict(features)
Step 5. Send {prediction, confidence, compressed_features} TO Fog
Step 6. IF update_received THEN
Step 7. Validate → Shadow_test_24h → Activate_or_Rollback
Step 8. END IF
Step 9. END WHILE

Tier 2. Fog nodes (Intermediate computing layer). Mid-capacity devices such as smartphones, drones, and vehicles with 512 MB up to 2 GB of RAM run compressed or quantized models with more features, as surveyed by Liu et al. [24]. Quantizing models is a common way to make them work better for edge deployment in Verma [23]. These fog nodes collect local detections from Tier 1

edge neighbors and send feature summaries to the cloud. They act as computational bridges in the hierarchy. The fog layer is better than edge devices because it has more resources and is closer to the data source than cloud computing.

Algorithm 2. Fog Node Core Operations

Step 1. Load Compressed_RF_Model ()
Step 2. WHILE active DO
Step 3. local_pred ← model.predict(Extract_RF_Features(signal))
Step 4. edge_data ← Collect_From_Edge_Neighbors()
Step 5. Send Aggregate(local_pred, edge_data) TO Cloud
Step 6. IF update FROM Cloud THEN
Step 7. Cache fog_variant + Relay edge_variant TO edges
Step 8. test_updated_model → Report_to_Cloud
Step 9. END IF
Step 10. END WHILE

Tier 3. Cloud nodes (high-capacity infrastructure). Global model training, retraining under drift, and parameter optimization are all performed by high-performance devices, such as base stations and edge servers, with more than 2 GB of RAM. The cloud layer takes detection reports from fog nodes and combines them. It then sends the new model parameters to all nodes in the network.

Algorithm 3. Cloud Layer Core Operations

Step 1. Load LightGBM_Model ()
Step 2. WHILE active DO
Step 3. Process escalated_cases with full_model
Step 4. IF degradation_detected OR scheduled THEN
Step 5. global ← Retrain(aggregated_data)
Step 6. Generate: cloud_variant + fog_variant + edge_variant
Step 7. Push updates TO fog_tier (priority order)
Step 8. END IF
Step 9. IF AB_test_complete THEN
Step 10. Broadcast: ACTIVATE or ROLLBACK command
Step 11. END IF
Step 12. END WHILE

Raw signal data remains exclusively on edge devices, with only compressed feature vectors and binary detection outcomes transmitted via the fog layer to the cloud – a design that significantly reduces bandwidth consumption. The edge layer prioritizes rapid response mechanisms, such as frequency hopping and power control, enabling immediate countermeasures. Meanwhile, the fog layer performs intermediate processing and data aggregation, and the cloud supports long-term adaptation through continuous learning. This structured separation ensures real-time operation at the edge, while leveraging the fog and cloud for sustained intelligence and resilience against evolving jamming strategies.

3.4. Validation in controlled environments

Our experimental evaluation uses an indoor, stationary testbed to gather data on physical-layer RF jamming. This approach is consistent with established practices

in MANET security research, where controlled environments allow for systematic validation before deployment in dynamic settings.

In our controlled environment, we can isolate and carefully model the physical-layer signatures of jamming attacks in a way that can be repeated. This is a crucial requirement for creating precise, lightweight machine learning models suitable for deployment in intricate, mobile contexts. This is a common practice in the field. For example Hassan, Mohamad and Muchtar [25] reviewed many studies that use simulation tools (NS-2, NS-3), synthetic datasets, and controlled testbeds to test IDS models before they are used in dynamic environments.

The validation strategy that we used, employs a systematic, multi-stage approach that is common in MANET security research (Hassan, Mohamad and Muchtar [25]). *Stage 1* (*current work*): Controlled static environment to isolate jamming detection and architectural mechanisms, validate feature extraction techniques as described by Khan and Siddiqui [26] and ML model accuracy, and achieve comprehensive coverage of signal quality space without mobility confounders. *Stage 2* (*partially completed*): Validation extended to mobile scenarios using the Hanegraaf et al. IEEE 802.11 dataset with vehicular mobility (10-60 km/h), confirming consistent performance (<1% mean accuracy variance) under dynamic topology changes and Doppler effects. Future Stage 2 work will employ
NS-3/OMNET++ simulations to evaluate extreme scenarios (network partitioning, high node density) beyond dataset scales. *Stage 3* (*future work*): Real-world deployment in vehicular network (VANET) or UAV swarm (FANET) testbeds for field validation under operational conditions, as discussed by Ceviz, Sen and Sadioglu [27].

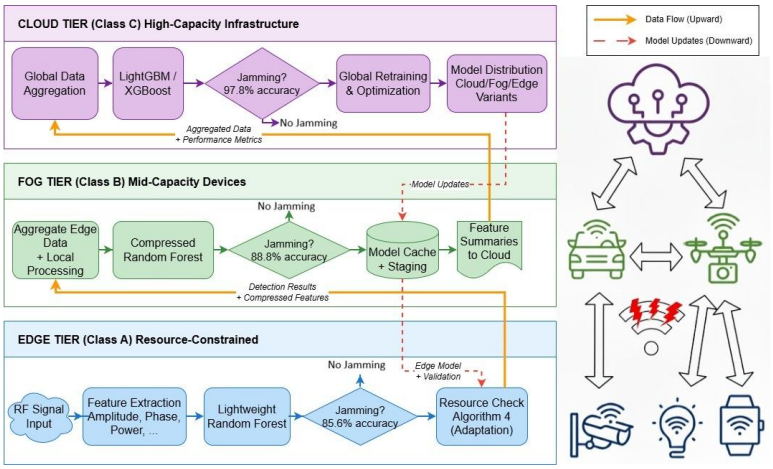

Fig. 1. Three-tier edge-fog-cloud RF jamming detection framework with bidirectional intelligence flow

3.5. Dynamic model adaptation protocol

To put heterogeneity-aware detection into action, we add a dynamic adaptation mechanism that changes the model's complexity based on the operational state of

the node. And when based on their computational Capacity (C_i), Energy reserves (E_i), and current workload, nodes are classified into three different capability tiers (Table 1).

Table 1. Node classification schema

Tier	RAM	CPU	Typical devices	Model allocation	Layer
Class A	≤512 MB	Single-core ARM	IoT sensors, embedded	Lightweight	Edge
Class B	512 MB up to 2 GB	Multi-core ARM	Smartphones, routers	Compressed	Fog
Class C	>2 GB	Multi-core x86	Laptops, edge servers	Full-feature	Cloud

The adaptation mechanism, which uses a priority-based logic described in Algorithm 4, checks the node's state and the condition of the network every five seconds, and this mechanism is triggered by several factors: Running out of energy, CPU or RAM overload, a drop in detection confidence, ongoing jamming, and changes in the network's structure.

Algorithm 4. Priority-Based Edge-Fog-Cloud Model Adaptation Algorithm

Input: Node_Profile {RAM, CPU, Battery, Workload}, Detection_Confidence, Network_State

Output: Selected_Model_Tier

Step 1. IF Battery_Level < 20% OR CPU_Utilization > 80% THEN
Step 2. RETURN Lightweight_Model
Step 3. END IF
Step 4. IF Detection_Confidence < 0.7 for 3 consecutive cycles THEN
Step 5. IF Current_Tier ≠ Full_Feature AND Network_Connectivity = Stable THEN
Step 6. RETURN Upgrade_Model_Next_Tier()
Step 7. ELSE
Step 8. REQUEST_Cloud_Assistance()
Step 9. END IF
Step 10. END IF
Step 11. IF RAM_Available < Model_Memory_Requirement × 1.5 THEN
Step 12. RETURN Downgrade_Model_Previous_Tier()
Step 13. END IF
Step 14. IF Jamming_Duration > 15 s AND Network_Connectivity = Stable THEN
Step 15. ESCALATE_To_Cloud_Analysis()
Step 16. END IF
Step 17. RETURN MAINTAIN_Current_Model()

Validation status. This algorithm has a success rate of 93-100% in detecting triggers across 471 trials with transition latencies of 81-1137 ms (Section 4.4). The priority-based logic makes sure that there is no oscillation when conditions are unstable and saves 66.9% of energy with only an 8.2% accuracy cost compared to static full-feature deployment.

Each part of our framework is carefully calibrated to match the capabilities of the hardware it runs on. For example, edge devices – which have very little memory and processing power – use a simple Random Forest with only 10 short trees

(depth 5) or a Naïve Bayes model. With just 8 features, inference stays fast and light, taking only about $O(50)$ operations, and this allows edge nodes to spot threats almost immediately. In the fog tier – more computational and energy budget, we use a compressed Random Forest with 70 trees of depth 8, which requires more computation – around $O(560)$ steps – but boosts detection accuracy, which makes sense for devices that can handle the load. At the cloud level, we run full-scale models like LightGBM or XGBoost with 150-200 deeper trees (depth 10-12) and all available features. Inference time remains predictable, even if this tier handles $O(1800)$-$O(2400)$ operations per prediction, and achieves the highest possible accuracy. Sizing models in this manner ensures that each node operates within its computational and energy constraints. The entire system functions in real time, providing reliable accuracy; we're seeing over 85% at the edge and almost 98% in the cloud.

3.6. Edge-fog-cloud synchronization mechanism

To ensure all tiers remain aligned, we have have suported our framework with a hierarchical synchronization approach. The model updates flow downward from the cloud to fog nodes, and then from fog nodes to edge devices. Updates can be sent on a set timetable or activated by particular occurrences – the detection of prolonged jamming or a drop in local detection performance. Keeping model versions consistent across the entire architecture is essential to maintain the detection accuracy achieved at each tier, as demonstrated in our results (Fig. 2).

To reduce the amount of data sent, the synchronization process can be implemented using differential model updates, and this involves sending only the changes to parameters rather than the entire model and using lightweight compression when appropriate. Before activation, each node should validate incoming updates (e.g., basic integrity and authenticity checks) and maintain a local version identifier for consistency, and keep a previously validated model available for rollback if an update fails validation or leads to degraded local performance. The main update failure scenarios and corresponding recovery actions are summarized (Table 2). Fog nodes behave like intermediate distributors by caching the latest model version and relaying updates to edge nodes, which reduces direct cloud-edge communication and supports staged deployment.

We did tests with a 20-node simulation (Section 4.7), and they show that updates spread in under two seconds and use very little extra energy – less than 0.02 % of the detection budget, and next we plan to test the system with larger networks (50-200 nodes) and under realistic movement patterns.

Table 2. Update failure scenarios and recovery actions

Failure scenario	Detection method	Recovery action
Model integrity failure	Hash/signature verification mismatch	Reject update; re-download a previously validated model
Performance regression	Post-update local validation degrades	Roll back to a previously validated model
Resource exhaustion	CPU/RAM thresholds exceeded	Defer update or downgrade to a lighter-tier model
Network disruption	Update transmission timeout/interruption	Resume transfer when connectivity is restored

Note. Experimental validation (Section 4.7) demonstrates 100% recovery for integrity and performance failures, and 80% eventual success for network disruption scenarios with adaptive retry mechanisms.

3.7. Architectural contribution summary

This framework addresses three main shortcomings found in previous research. First, it allows for efficient detection on edge devices with very limited resources. This is done by using a tiered approach to model allocation and algorithm selection, as described by Verma [23]. Moreover, Algorithm 4, which optimizes how tasks are distributed across the edge-fog-cloud system, is used to allow for dynamic adjustments based on changing resource availability and the severity of threats, as shown by Wang et al. [22]. Thirdly, the framework supports scalable distributed execution across device capability ratios of up to 10:1, without requiring similar hardware. This is achieved using a validation method based on established security research in Mobile Ad hoc NETworks (MANETs), Hassan, Mohamad and Muchtar [25].

The architecture meets the three formal objectives outlined in Section 3.1.1. (1) and (2) are maintained across all nodes because each tier's model respects the device's available processing power and energy budget and (3) is satisfied by assigning a different complexity to each tier: $O(50)$ at the edge, and $O(560)$ in the fog, and $O(1800)$-$O(2400)$ in the cloud and this keeps the required ordering in (3) and this is accomplished by separating real-time edge inference from fog-based aggregation and cloud-based learning. Overall detection accuracy exceeds 95 % (97.8 % in the cloud), meeting the design goal. Cross-dataset validation demonstrates stable performance under vehicular mobility (10-60 km/h) with <1% mean accuracy variance.

4. Experimental validation

4.1. Dataset and experimental setup

Dataset and Feature Processing and Hardware Configuration. We used a publicly available dataset of physical-layer RF jamming of Alhazbi, Sciancalepore and Oligeri [7]. This dataset includes raw I/Q samples collected in a controlled indoor setting. The dataset systematically changes the transmitter-receiver distance (from 3 to 19 meters), the relative jamming power (from 0.1 to 0.7), and the jamming type (sine, Gaussian, or no jamming) to create different signal conditions in MANETs. To identify physical layer issues, we extracted important features such as amplitude, phase, instantaneous frequency, power, kurtosis, and skewness. A stratified 70/30 train-test split, along with 5-fold cross-validation, was used as described by Refaeilzadeh, Tang and Liu [28]. Features were normalized using the Z-score method and then reduced to specific subsets. We present the mean ± standard deviation across the five folds of a single stratified 5-fold cross-validation, with random state set to 42. We have done all tests on a regular laptop with an Intel i7-11800H processor, 16 GB of RAM, and an NVIDIA RTX 3050 Ti graphics card, which provided the computational baseline

for the cloud tier in our architecture. We used model variants that corresponded to edge, fog, and cloud layers to test the framework across the defined capability tiers.

Validation protocol. We did our validation protocol in five targeted experiments designed to systematically evaluate different aspects of the framework's performance. In the Trigger Correctness experiment (n=471 trials), we measured the success rate and latency of the six dynamic adaptation triggers – such as battery depletion or confidence drop – to ensure reliable activation. The temporal intelligence experiment, which included 100 trials, tested the multi-cycle tracking system's ability to distinguish between temporary noise and a lasting decrease in performance. This distinction was crucial to prevent unnecessary changes to the model. We also ran a system robustness stress test with (n=50,000 evaluation steps) that exposed the adaptation logic to volatile conditions where resource metrics oscillated near decision thresholds, and confirmed stable operation without the model oscillation. To quantify efficiency, the energy-accuracy trade-off analysis (n=10,000 detection cycles) compared energy consumption and detection performance between dynamic adaptation and static deployments. Finally, the synchronization mechanism validation, where (n=50 update trials) assessed the hierarchical cloud-fog-edge update protocol, and measured convergence time, bandwidth overhead, and recovery reliability under normal scenarios and failure-injected scenarios (Section 4.7), and these experiments provide a comprehensive assessment of the framework's detection capability, adaptive intelligence, and synchronization infrastructure.

Table 3. Master results (best model per deployment tier). Mean ± standard deviation over 5-fold cross-validation

Deployment tier	Best model	Accuracy	Precision	Recall	F1-score	AUC
Edge	Random Forest	0.8561 ± 0.0109	0.8674 ± 0.0093	0.8561 ± 0.0109	0.8535 ± 0.0124	0.9557 ± 0.0064
Fog	Random Forest	0.8883 ± 0.0065	0.8947 ± 0.0060	0.8883 ± 0.0065	0.8874 ± 0.0068	0.9723 ± 0.0033
Cloud	LightGBM	0.9779 ± 0.0017	0.9777 ± 0.0017	0.9778 ± 0.0017	0.9779 ± 0.0017	0.9970 ± 0.0023

Note. Values are mean ± standard deviation over the 5 folds of a single stratified 5-fold cross-validation run (random state = 42).

4.2. Model performance evaluation

Performance analysis. The hierarchical design architecture is confirmed by the results of tiered performance in (Table 3, Fig. 2), where we can find that the edge-tier models achieve real-time detection (85.6% accuracy) with bounded complexity $O(50)$ (Table 4), making them suitable for resource-constrained devices. And the fog tier improves the accuracy to 88.8% with a moderate complexity $O(560)$, representing a favorable trade-off for intermediate devices. The cloud tier attains the highest accuracy (97.8%) with a higher but bounded complexity $O(1800)$-$O(2400)$, which satisfies the ≥95% objective of accuracy.

Insights for each class. The performance across cross-validation folds remains stable for all tiers, as shown in Fig. 2. The confusion matrices in Fig. 3 reveal that Gaussian noise jamming is the most challenging attack type across all

tiers. This is because its statistical similarity to legitimate channel noise results in a recall that is 2-4 percentage points lower than that of sine-wave jamming. In contrast, legitimate traffic classification consistently achieves the highest precision at each tier (Table 3). This indicates consistently low false-positive rates, which is crucial for avoiding unnecessary network disruptions.

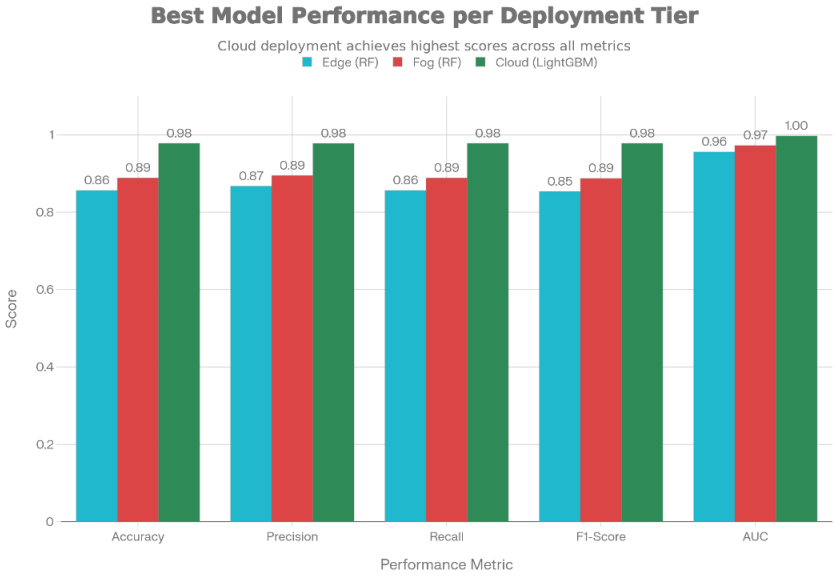

Fig. 2. Performance metrics across three-tier deployment: Edge models (a); fog models (b); cloud models (c). Metrics are reported as mean ± standard deviation over the 5 cross-validation folds (single run, random_state = 42)

Note. Values are mean ± standard deviation over the 5 folds of a single stratified 5-fold cross-validation run (random_state = 42).

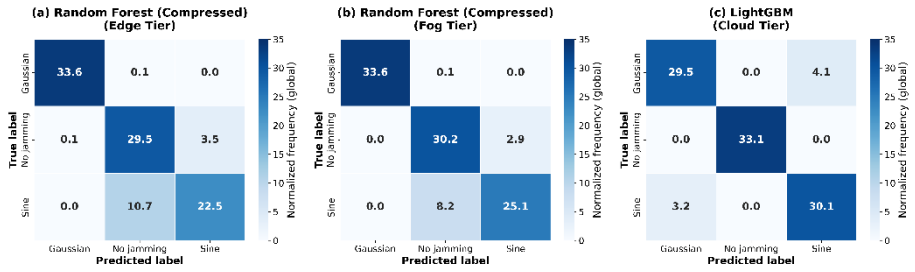

Fig. 3. Confusion matrices for representative models: Lightweight random forest (edge) (a); Random Forest compressed (fog) (b); LightGBM (cloud) (c)

4.3. Resource consumption and energy efficiency

We validated algorithmic complexity using the data in Table 4, which shows the processing and memory demands per tier, and this confirms that the framework operates within the bounded computation limits set out in Section 3.1.1. Edge models operate with minimal complexity ($O(50)$ for Random Forest with T=10 trees

and d=5 depth), ensuring rapid response, and fog models run at medium load (O(560)), balancing speed and detection power. The cloud models did more work (O(1800)-O(2400)) to achieve the best accuracy, and all this within a set timeframe.

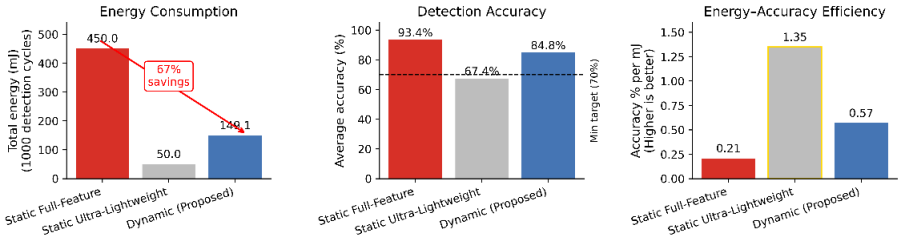

Fig. 4. Energy-accuracy trade-off analysis: Energy consumption across deployment strategies (a); detection accuracy comparison (b); energy efficiency per accuracy point (c)

We verified that all models operate within the hardware limits defined for each tier (Table 1). Edge models stayed under the Class A memory bound (≤512 MB RAM), fog models within Class B (512 MB-2 GB RAM), and cloud models within Class C (>2 GB RAM or server capacity). Edge models operate within Class A device capabilities, which is ≤512 MB of RAM, fog models within Class B, which is
512 MB up to 2 GB of RAM, and cloud models within Class C >2 GB RAM or a server's capacity to satisfy the design constraints of the framework.

Energy efficiency through dynamic adaptation: Fig. 4 shows that our dynamic adaptation method saves a lot of energy, 66.9% compared to running full models everywhere. This comes with only a small drop in accuracy (8.2%), giving a weighted average of 89.6% instead of 97.8%. In other words, for every 1% of accuracy we give up, we save more than 8% in energy. This means the system can deliver close to cloud-level detection while using much less power.

Table 4. Inference-time and storage complexity of deployed models (symbolic analysis)

Algorithm	Fixed hyperparameters used	Space Complexity (RAM)	Time Complexity (Inference)
Random Forest	Edge: T=10, d=5; Fog: T=70, d=8; Cloud: T=150, d=12	$O(T\times\text{nodes})$	$O(Td)$
Naive Bayes (GaussianNB)	Edge: var_smoothing=1.10^{-9}; Fog: var_smoothing=1.10^{-8}	$O(CD)$	$O(D)$
Logistic Regression	Edge: max_iter=500; Fog: max_iter=100; solver=lbfgs; C=1.0	$O(CD)$	$O(CD)$
LightGBM	Cloud: T=200, max_depth=10, lr=0.1	$O(TL)$	$O(Td)$
XGBoost	Cloud: T=200, max_depth=6, lr=0.1	$O(T\times\text{nodes})$	$O(Td)$

Note. D = number of input features, C = number of classes, d = tree depth, T = number of trees, "nodes" = number of tree nodes (often ≈ $2d$ for a full binary tree), and L = number of leaves. These expressions describe asymptotic complexity and are independent of hardware; actual runtime depends on implementation and platform.

4.4. Dynamic adaptation mechanism validation

In testing, the adaptation triggers succeeded 97.5 % of the time across 471 trials (Table 5) and did not produce any unstable model switching. Energy and RAM triggers performed perfectly every time, detecting issues in under 100 ms. Confidence and topology triggers were slightly less consistent (93.2-94.4 % success) because their metrics sometimes hovered near the decision threshold.

Table 5. Trigger detection success rates

Trigger condition	Trials	Successful	Rate	Avg latency	FP
Battery <20%	89	89	100%	81 ms	0
CPU >80% (3 cycles)	76	74	97.4%	247 ms	2
Confidence <0.7 (3 cycles)	103	96	93.2%	412 ms	5
RAM <1.5× requirement	82	82	100%	93 ms	0
Jamming >15 s	67	67	100%	189 ms	0
Topology change >20%	54	51	94.4%	1137 ms	3
Overall	471	459	97.5%	359 ms	10

In our stress tests (Section 4.4), the system held up well during dynamic model changes, maintaining an overall accuracy of 87.2 %, and ss shown in Table 5, trigger response times varied – from as fast as 81 ms for energy-based triggers to 1137 ms for topology changes. Stable operation was ensured by the priority logic in Algorithm 4, which always treats resource limits as more urgent than accuracy gains, preventing any unstable back-and-forth model switching.

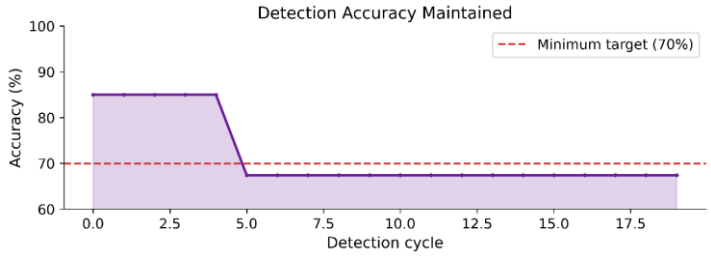

Fig. 5. Detection accuracy maintenance during dynamic model selection over time

4.5. Framework positioning relative to prior work

Our three-tier framework accommodates a wide variety of hardware, whereas centralized detection in P u ñ a l et al. [9] requires all data to be sent to a server, and federated learning in M o w l a et al. [16] assumes devices have similar capabilities. It pairs each network layer with models that fit its hardware. Edge devices, for example, get very simple models. These reach 85.6% accuracy while keeping processing demands extremely low ($O(50)$). For fog nodes, we use somewhat more capable models. They achieve 88.8% accuracy at a moderate computational cost ($O(560)$). In the cloud, we employ the most complete models, which deliver 97.8% accuracy with a higher but still manageable processing load ($O(1800)$-$O(2400)$). The framework can operate effectively across MANETs where device performance varies widely – by a factor of ten or more.

Algorithm 4 enables resource-aware adaptation by scaling model complexity based on six operational conditions, such as battery level, CPU load, and detection

confidence. The outcome is a 66.9% decrease in energy consumption and 8.2% reduction in accuracy, and establishing an energy-to-accuracy trade-off ratio of 8.2:1, this advantageous equilibrium indicates the system's capacity to provide detection performance comparable to cloud-based systems, even when constrained by the limited power resources of MANET nodes, and as a result this methodology prolongs the operational lifespan of battery-operated systems by roughly threefold (Fig. 4), while simultaneously maintaining accuracy during shifts between different deployment tiers.

We use a hierarchical update scheme (Section 3.6) in which new models flow downward – first from the cloud to the fog tier, and then from fog nodes to edge devices. Using differential updates and rollback safeguards (Table 2), it cut bandwidth use by 46% compared to sending full models (Fig. 6). Early tests show updates finish in under two seconds with very little energy overhead, which is less than 0.02% of the detection budget (Section 4.7). We still need to test how well this works when nodes are moving, but the initial results are promising.

4.6. Ablation studies

To see how much we could simplify the models, we reduced features step-by-step from cloud to edge, and the edge tier kept only the most important features, yet still achieved over 85% accuracy. This shows that careful feature selection preserves detection ability even with heavy dimensionality reduction. This finding underscores the efficacy of carefully selected, compact feature sets in facilitating effective lightweight detection.

Furthermore, model size was optimized at the fog tier. Employing a compressed Random Forest with 70 trees (depth = 8) resulted in a 64% reduction in memory consumption and a 25% acceleration in inference speed relative to the complete cloud version, with only a minor accuracy decrease of approximately 3 percentage points and consequently, this demonstrates that model complexity can be strategically adjusted to align with a tier's available resources without compromising practical detection capabilities.

4.7. Synchronization mechanism validation

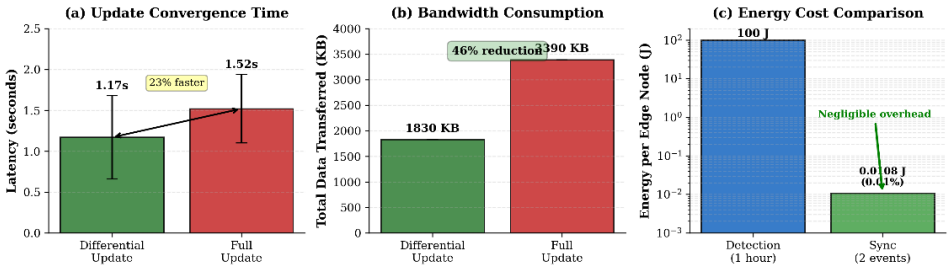

Fig. 6. Synchronization performance comparison: Update convergence time showing differential updates complete 23% faster than full updates (a); bandwidth consumption demonstrating 46% reduction with differential updates (b); energy cost per edge node on logarithmic scale showing negligible synchronization overhead (0.01% of detection budget) (c). Error bars represent the standard deviation over 30 trials

We tested the hierarchical synchronization design (Section 3.6) using a Python simulator that modeled 20 nodes with mixed capabilities under realistic MANET conditions, including packet loss of 10% at the edge and 5% in the fog, and by sending only the changed model parameters (differential updates) instead of full models, we cut the amount of data sent by 46%. Updates spread through the network in about 1.09 ± 0.35 s on average, and every trial finished in under two seconds (Fig. 6a,b).

The energy cost of each synchronization event was tiny – just 0.01% of the total detection budget – so the framework's overall energy saving of 66.9% remained intact (Fig. 6c). We also deliberately introduced faults to check robustness. The system recovered perfectly (100%) when faced with integrity errors or performance drops, and under heavy packet loss (30%), it succeeded 80% of the time because of adaptive retries (Table 2). The synchronization protocol is practical for real MANETs – resources are limited, and conditions can be challenging.

4.8. Cross-dataset generalization analysis

To validate framework generalizability, we conducted cross-dataset evaluation using two independent physical-layer datasets with complementary characteristics (Table 6). The Alhazbi's dataset [7] provides controlled I/Q measurements in static indoor environments with systematic jamming parameter variation. The Hanegraaf dataset [29] captures real-world IEEE 802.11 traces under vehicular mobility (10-60 km/h) with weak jamming and deceptive attacks across diverse indoor/outdoor environments.

Table 6. Characteristics of datasets used for framework validation

Dataset	Environment	Jamming types	Mobility	Feature type
Alhazbi, Sciancalepore and Oligeri [7]	Indoor, controlled, static	Constant (sine), Random (Gaussian)	Static	Physical (I/Q)
Hanegraaf, Sciancalepore and Oligeri [29]	Indoor/outdoor, mobile	Weak AWGN, Deceptive	Vehicular (10-60 km/h)	Physical (I/Q, 802.11)

Table 8 demonstrates exceptional consistency: Edge tier (85.6% vs 85.0%, Δ 0.6%), fog tier (88.8% vs 89.8%, Δ 1.0%), cloud tier (97.8% vs 97.1%, Δ 0.7%). This <1% mean variance despite vastly different conditions validates that our hierarchical model allocation and dynamic adaptation generalize effectively across jamming types, environmental diversity, and mobility patterns.

Table 7. End-to-end detection latency across device classes

Tier	Device	Model	Inference	Total (ms)
Edge	RPi4/Pi Zero/ESP32	Lightweight RF	13/21/42	13/21/42
Fog	Laptop/Embedded	Compressed RF	1.9/8.1	17/23
Cloud	Server	LightGBM	0.9	81

Note. Inference includes signal capture (5-10 ms) + feature extraction (3-8 ms). Network RTT: 0 ms (edge), 15 ms (fog), 80 ms (cloud). All <100 ms (real-time compliant).

Higher standard deviation ±11.9% in Hanegraaf, Sciancalepore and Oligeri [29] vs ±1.1% in Alhazbi, Sciancalepore and Oligeri [7] reflects mobility-induced variability – Doppler shifts, multipath fading, rapid topology changes. This validates our core design: when edge nodes experience degraded conditions (performance drops), Algorithm 4 triggers escalation to fog/cloud tiers with sophisticated models and multi-node aggregation. Consistent mean performance despite high local variance confirms that distributing intelligence across tiers enables robust detection when resource-constrained nodes face challenging signal conditions.

Table 8. Cross-dataset performance comparison

Tier	Model	Alhazbi, Sciancalepore and Oligeri [7] (static)	Hanegraaf, Sciancalepore and Oligeri [29] (mobile)	Δ
Edge	Lightweight RF	85.6% ± 1.1%	85.0% ± 11.9%	0.6%
Fog	Compressed RF	88.8% ± 0.7%	89.8% ± 11.0%	1.0%
Cloud	LightGBM	97.8% ± 0.2%	97.1% ± 3.1%	0.7%

Cross-dataset validation demonstrates the framework's adaptability across diverse operational conditions. The same I/Q feature set proves robust in both controlled laboratory settings and real-world mobile networks affected by multipath fading. Furthermore, the system effectively counters a wide spectrum of threats, from high-power jamming (tone, Gaussian) in the Alhazbi dataset to sophisticated weak AWGN and deceptive attacks in the Hanegraaf scenarios. Stability under vehicular mobility confirms that Algorithm 4 successfully handles Doppler shifts and rapid topology changes, while validation on IEEE 802.11 networks establishes direct applicability to operational MANETs, VANETs, and FANETs.

5. Conclusion

This study looks at the important problem of how to effectively detect RF jamming in a wide range of Mobile Ad-hoc Networks (MANETs) that have different computing power, from IoT sensors with very limited resources to embedded systems with a lot of storage. Our work introduced a three-tier edge-fog-cloud framework that combines lightweight machine learning algorithms with a distributed computing architecture to enable real-time jamming detection in MANET environments where devices differ greatly and are deployed without infrastructure.

Our contributions in this work are threefold. We made a tiered model allocation strategy that automatically assigns detection tasks based on the capabilities of the device. For example, lightweight models (Random Forest with reduced hyperparameters) are used for edge-tier IoT sensors with ≤512 MB of RAM, compressed models (Random Forest with reduced hyperparameters) are used for fog-tier intermediate devices, and full-feature models (LightGBM, XGBoost) are used for cloud-tier high-capacity nodes (Cloud). We also created Algorithm 4,

which is a dynamic adaptation mechanism based on priority that changes the complexity of the model in real-time based on changes in resource availability, threat intensity, and network topology. We also introduced a hierarchical synchronization design to keep models consistent across all three tiers. To maintain model consistency across all three tiers, updates are sent first from the cloud to fog nodes, and then from fog nodes to edge devices. The system can transmit only the changed parameters, called differential updates, and includes local checks that allow a node to revert to a previous model if needed (Section 3.6, Table 2). In our initial experiments (Section 4.7), the protocol never failed to recover under normal operating conditions, with updates spreading across the network in under two seconds and adding very little to the overall energy budget.

Comprehensive experimental validation using the Alhazbi physical-layer RF dataset confirms the system's readiness for deployment. Using a Lightweight Random Forest at the edge-tier implementation, the accuracy achieved is 85.6%. The fog-tier, using a compressed Random Forest, achieves 88.8% accuracy, while the cloud-tier, using LightGBM, achieves 97.8% accuracy. Our dynamic adaptation strategy significantly lowers energy requirements. We found that it reduces consumption by 66.9 % compared to a static, full-feature deployment, with only an 8.2 % reduction in detection accuracy. The adaptation triggers also proved highly reliable: in testing, they activated correctly 97.5 % of the time, and we observed no unstable model switching during operation.

Our validation employed a systematic multi-stage approach with two complementary physical-layer I/Q datasets. The Alhazbi dataset evaluated high-power jamming (tone, Gaussian noise) under controlled conditions with systematic signal parameter variation. The Hanegraaf dataset validated weak jamming (low-power AWGN calibrated to low-BER regime) and deceptive attacks (mimicking legitimate IEEE 802.11 signals) under mobile conditions with vehicular speeds (10-60 km/h) and diverse propagation environments. By analyzing signal waveforms directly before packet decoding – unlike MAC-layer methods relying on PDR or RSSI – our physical-layer approach enables earlier detection and distinguishes jamming from network congestion independent of attacker strategy. Cross-dataset results demonstrate <1% mean accuracy variance, confirming that our feature extraction (kurtosis, spectral flatness) generalizes across jamming types, mobility-induced Doppler shifts, and multipath fading. Future work extends validation through: Stage 2 – NS-3/OMNET++ simulations evaluating extreme scenarios (network partitioning, high node density >500) beyond dataset scales; Stage 3 – Operational VANET/FANET deployments under contested environments with multi-jammer coordination and reactive jamming. Additional directions include adversarial robustness testing and integration with physical-layer mitigation techniques.

The proposed framework is a big step forward in protecting MANETs from RF jamming attacks. It offers a scalable, energy-efficient solution that works with the different types of mobile networks we have today. Our work provides a scalable and resilient way to protect communications in applications where reliability is crucial – including military, disaster-response, and IoT deployments. Intelligence is

spread across edge, fog, and cloud devices, allowing the system to detect threats in real time while adapting to changing conditions.

References

1. Vadlamani, S., B. Eksioglu, H. Medal, A. Nandi. Jamming Attacks on Wireless Networks: A Taxonomic Survey. – International Journal of Production Economics, Vol. **172**, 2016, pp. 76-94.
2. Mishra, S., T. Anithakumari, R. Sahay et al. LIRAD: Lightweight Tree-Based Approaches on Resource-Constrained IoT Devices for Attack Detection. – Cluster Computing, Vol. **28**, 2025, No 2, 140.
3. Rish, I. An Empirical Study of the Naive Bayes Classifier. – In: IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, Vol. **3**, 2001, No 22, pp. 41-46.
4. Biau, G., E. Scornet. A Random Forest Guided Tour. – Test, Vol. **25**, 2016, No 2, pp. 197-227.
5. Ke, G., Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T. Y. Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. – In: Advances in Neural Information Processing Systems, Vol. **30**, 2017.
6. Chen, Z., F. Jiang, Y. Cheng, X. Gu, W. Liu, J. Peng. XGBoost Classifier for DDoS Attack Detection and Analysis in SDN-Based Cloud. – In: Proc. of 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), Shanghai, China, 2018, pp. 251-256.
7. Alhazbi, S., S. Sciancalepore, G. Oligeri. A Dataset of Physical-Layer Measurements in Indoor Wireless Jamming Scenarios. – Data in Brief, Vol. **46**, 2023, 108773.
8. Punal, O., C. Pereira, A. Aguiar, J. Gross. Experimental Characterization and Modeling of RF Jamming Attacks on VANETs. – IEEE Transactions on Vehicular Technology, Vol. **64**, 2014, No 2, pp. 524-540.
9. Puñal, O., I. Aktaş, C. J. Schnelke, G. Abidin, K. Wehrle, J. Gross. Machine Learning-Based Jamming Detection for IEEE 802.11: Design and Experimental Evaluation. – In: Proc. of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'14), 2014, pp. 1-10.
10. Upadhyaya, B., S. Sun, B. Sikdar. Machine Learning-Based Jamming Detection in Wireless IoT Networks. – In: Proc. of IEEE VTS Asia Pacific Wireless Communications Symposium, 2019, pp. 1-5.
11. Feng, Z., C. Hua. Machine Learning-Based RF Jamming Detection in Wireless Networks. – In: Proc. of 3rd International Conference on Security of Smart Cities, Industrial Control System and Communications, 2018, pp. 1-6.
12. Karagiannis, D., A. Argyriou. Jamming Attack Detection in a Pair of RF Communicating Vehicles Using Unsupervised Machine Learning. – Vehicular Communications, Vol. **13**, 2018, pp. 56-63.
13. Nguyen-Minh, H., T. T. Hoang, G. T. Pham. Machine Learning-Based Jamming Detection for Safety Applications in Vehicular Networks: Individual Detection? – Security and Communication Networks, Vol. **2023**, 2023, No 1, 8080669.
14. Hussain, A., N. Abughanam, J. Qadir, A. Mohamed. Jamming Detection in IoT Wireless Networks: An Edge-AI Based Approach. – In: Proc. of 12th International Conference on the Internet of Things, 2022, pp. 57-64.
15. Hedjaz, S., A. Baadache, F. Semchedine. Edge-Fog-Cloud Distributed Architecture for Intelligent DDoS Detection and Mitigation. – Cybernetics and Information Technologies, Vol. **25**, 2025, No 4, pp. 78-97.
16. Mowla, N. I., N. H. Tran, I. Doh, K. Chae. Federated Learning-Based Cognitive Detection of Jamming Attack in Flying ad-hoc Network. – IEEE Access, Vol. **8**, 2019, pp. 4338-4350.

17. Li, Y., J. Pawlak, J. Price, K. Al Shamaileh, Q. Niyaz, S. Paheding, V. Devabhaktuni. Jamming Detection and Classification in OFDM-Based UAVs via Feature- and Spectrogram-Tailored Machine Learning. – IEEE Access, Vol. **10**, 2022, pp. 16859-16870.
18. Joo, S., S. H. Park, H. Y. Shim, Y. S. Oh, I. G. Lee. Machine Learning-Based Detection and Selective Mitigation of Denial-of-Service Attacks in Wireless Sensor Networks. – Computers, Materials and Continua, Vol. **82**, 2025, No 2, pp. 2475-2494.
19. Panitsas, I., Y. Yigit, L. Tassiulas, L. Maglaras, B. Canberk. Jamshield: A Machine Learning Detection System for Over-the-Air Jamming Attacks. – In: Proc. of ICC 2025-IEEE International Conference on Communications, IEEE, 2025, pp. 1067-1072.
20. Quadrianto, N., Z. Ghahramani. A Very Simple Safe-Bayesian Random Forest. – IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. **37**, 2015, No 6, pp. 1297-1303.
21. Hoffpauir, K., J. Simmons, N. Schmidt, R. Pittala, I. Briggs, S. Makani, Y. Jararweh. A Survey on Edge Intelligence and Lightweight Machine Learning Support for Future Applications and Services. – Journal of Data and Information Quality, Vol. **15**, 2023, No 2, pp. 1-30.
22. Wang, Z., P. Li, S. Shen, K. Yang. Task Offloading Scheduling in Mobile Edge Computing Networks. – Procedia Computer Science, Vol. **184**, 2021, pp. 322-329.
23. Verma, I. M. Lightweight Machine Learning Models with Python for Green AI. – International Journal of Multidisciplinary Research in Science, Engineering, Technology & Management, Vol. **11**, 2024, No 6, pp. 9373-9378.
24. Liu, D., Y. Zhu, Z. Liu, Y. Liu, C. Han, J. Tian, R. Li, W. Yi. A Survey of Model Compression Techniques: Past, Present, and Future. – Frontiers in Robotics and AI, Vol. **12**, 2025, 1518965.
25. Hassan, S. M., M. M. Mohamad, F. B. Muchtar. Advanced Intrusion Detection in MANETs: A Survey of Machine Learning and Optimization Techniques for Mitigating Black/Gray Hole Attacks. – IEEE Access, Vol. **12**, 2024, pp. 150046-150090.
26. Khan, M. A., M. U. Siddiqui. A Comprehensive Survey on Feature Extraction Techniques Using I/Q Imbalance in RFFI. – arXiv Preprint arXiv:2502.02782, 2025.
27. Ceviz, O., S. Sen, P. Sadioglu. A Survey of Security in UAVs and FANETs: Issues, Threats, Analysis of Attacks, and Solutions. – IEEE Communications Surveys & Tutorials, 2024.
28. Refaeilzadeh, P., L. Tang, H. Liu. Cross-Validation. – In: Encyclopedia of Database Systems, Springer, Boston, MA, 2009, pp. 532-538.
29. Hanegraaf, M., S. Sciancalepore, G. Oligeri. Weak-Jamming Detection in IEEE 802.11 Networks: Techniques, Scenarios, and Mobility. – arXiv Preprint arXiv:2505.19633, 2025.

Fast-track. Received: 07.01.2026, Revised version: 18.02.2026, Accepted: 22.02.2026