Indonesian Journal of Electrical Engineering and Computer Science
Vol. 44, No. 1, October 2026, pp. 293~306
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v44.i1.pp293-306

TinyML-based 1D-CNN intrusion detection for IoT physical layer

Qasim Oqlah, Saad Al- Mobark, Samer Rahmah, Wassim Al Juneidi, Yamen Hallak
Department of Computer Systems and Network Security, Faculty of Engineering, Syrian Private University (SPU), Damascus, Syria

Article Info

Article history:

Received Apr 30, 2026
Revised Aug 19, 2026
Accepted Sep 9, 2026

Keywords:

1D-CNN
Internet of things
Jamming detection
Physical layer security
Temporal window length
TinyML

ABSTRACT

The proliferation of internet of things (IoT) devices has introduced critical security vulnerabilities at the physical layer, where cryptographic mechanisms cannot address radio-frequency threats such as jamming. This paper investigates received signal strength (RSS) based intrusion detection using a lightweight one-dimensional convolutional neural network (1D-CNN) optimized for TinyML deployment. We identify the temporal window length as the dominant factor governing smart jamming detectability, a parameter treated as an incidental design choice in prior work. Using an empirical RSS dataset collected with software-defined radio hardware, we show that extending the window from 16 to 128 samples raises three-class accuracy from 73.17 ± 0.15% to 86.03 ± 0.63% and smart jamming recall from 30.36 ± 2.90% to 64.20 ± 3.80% across five independent runs. We attribute this to the duty-cycled nature of smart jamming: windows shorter than the jammer activity period fall within a single phase, rendering them indistinguishable from constant jamming or normal traffic. Unlike prior RSS-based work addressing binary classification, this study discriminates three classes, a distinction with operational significance because the attack types warrant different responses. Mapped onto the binary task without retraining, the model attains 95.49% accuracy. The model comprises 2,691 parameters and occupies 13.8 KB after quantization.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Qasim Oqlah
Department of Computer Systems and Network Security, Faculty of Engineering
Syrian Private University (SPU)
Damascus, Syria
Email: Qasim_Oqlah@spu.edu.sy

1. INTRODUCTION

The internet of things (IoT) has become an integral part of modern infrastructure, enabling seamless connectivity across various domains including smart cities, healthcare, and industrial automation [1]. However, the widespread deployment of IoT devices has exposed significant vulnerabilities, particularly at the physical layer of wireless communication. Due to the broadcast nature of wireless signals, IoT networks are increasingly susceptible to malicious interference, among which intelligent jamming poses a severe threat. Unlike traditional jamming, which transmits continuously and is therefore comparatively easy to identify, smart jamming attacks are intermittent and energy-efficient, making them difficult to detect using conventional threshold-based methods [2], [3].

The complexity of smart jamming attacks lies in their ability to mimic legitimate network behavior, often bypassing traditional security protocols that rely on static thresholds. These attacks target the synchronization and control channels of the IoT physical layer, leading to total communication failure or significant data loss. Consequently, there is an urgent need for adaptive detection frameworks that can

operate autonomously within the strict energy and memory constraints of IoT edge nodes. Addressing these challenges requires a paradigm shift from centralized cloud-based security to localized, intelligent processing at the device level [4], [5].

Ensuring robust security in resource-constrained IoT environments requires lightweight yet highly accurate detection mechanisms. Traditional intrusion detection systems (IDS) often demand substantial computational power, which is not feasible for most IoT edge devices [6]. Recent advances in artificial intelligence (AI) and machine learning (ML) have opened new possibilities for physical layer security [7]. Specifically, tiny machine learning (TinyML) offers a promising direction by allowing compact models to run directly on low-power microcontrollers [8], [9]. This paper proposes an IDS framework that utilizes a one-dimensional convolutional neural network (1D-CNN) [10] to identify and classify jamming patterns based on received signal strength (RSS) behavior.

Despite these advances, a gap remains in how the temporal dimension of the problem is treated. Most existing ML-based jamming detection studies formulate the task as binary discrimination between a normal and a jammed channel, and are validated on simulated interference rather than on measurements captured from real radio hardware [11]-[13]. Where a temporal window is used at all, its length is generally reported as an incidental implementation detail rather than examined as a determinant of performance. Among the few works combining multi-class jamming classification with edge deployment, the study by Hussain *et al.* [14] released an empirical RSS dataset but reported a binary accuracy of 86.33% together with documented overfitting, indicating that reliable multi-class discrimination and lightweight deployment have not yet been achieved together [15]. This paper addresses that gap. The main contributions of this paper are summarized as follows:

a) We identify the temporal window length as the dominant factor governing smart jamming detectability from RSS measurements alone. Extending the window from 16 to 128 samples raises overall three-class accuracy from 73.17 ± 0.15% to 86.03 ± 0.63% and smart jamming recall from 30.36 ± 2.90% to 64.20 ± 3.80% over five independent runs, and we provide a physical explanation grounded in the duty-cycled behavior of the smart jammer.
b) We address three-class discrimination between normal traffic, constant jamming, and smart jamming, whereas prior RSS-based approaches address the binary task. This distinction carries operational significance, since constant and intermittent jamming call for different mitigation responses. Mapped onto the equivalent binary task without retraining, the proposed model attains 95.49% accuracy.
c) We demonstrate deployability under TinyML constraints with a network of 2,691 trainable parameters occupying 13.8 KB after dynamic-range quantization, and we measure post-quantization accuracy directly on the held-out test set rather than assuming that it is preserved.
d) We contextualize the detection approach within a simulated smart-home IoT deployment in NS-3 [16], illustrating where such a model would operate within a broader network architecture, independently of the model's training and evaluation, which are based entirely on the empirical dataset described in section 3.2.

2. RELATED WORKS

Research on jamming in wireless and IoT networks has progressed along three broad lines: classical detection based on signal statistics, machine-learning-based detection, and, more recently, deployment of learned models on constrained edge hardware. This section reviews each line critically and identifies the specific limitations that motivate the present work.

2.1. Statistical and threshold-based detection

Early approaches modeled jamming as a departure from expected channel statistics, using metrics such as RSS, packet delivery ratio (PDR), or carrier sensing time, and flagged an attack when a fixed threshold was exceeded. Comprehensive surveys of these techniques [2], [3], [17] establish a consistent limitation, and reviews of physical-layer security for resource-constrained IoT devices reach the same conclusion [18], [19]: static thresholds are effective against constant jammers, whose energy signature is persistently elevated, but degrade sharply against intermittent or reactive jammers whose silent intervals are statistically indistinguishable from normal traffic. This limitation is structural rather than incidental, and it motivates the shift toward data-driven detectors that exploit the temporal structure of the signal rather than its instantaneous amplitude alone.

2.2. Machine learning approaches

Classical ML was subsequently applied to the problem. Puñal *et al.* [13] evaluated decision-tree and rule-based classifiers on IEEE 802.11 measurements, Upadhyaya *et al.* [11] proposed a passive anchor-node

detection scheme for wireless IoT networks, and Arjoune *et al.* [12] compared random forest (RF), support vector machine (SVM), and neural-network classifiers on RSS and PDR features. Feng and Hua [20] and Kasturi *et al.* [21] extended this line to the classification of several jammer categories. These studies collectively demonstrate that learned decision boundaries outperform fixed thresholds. However, two limitations recur. First, the majority of these works are evaluated on simulated interference, which does not reproduce hardware-specific distortion or environmental multipath behavior. Second, feature-engineered classifiers such as SVM and RF require handcrafted statistical descriptors, and their inference cost scales unfavorably with feature dimensionality, limiting their suitability for continuous on-node operation.

Deep learning removed the dependence on handcrafted features. Erpek *et al.* [22] applied deep learning to both launching and mitigating jamming attacks, Premkumar and Sundararajan [23] proposed a deep-learning defense mechanism for denial-of-service attacks in wireless sensor networks, and more recent work has addressed 5G and aerial scenarios: Hachimi *et al.* [24] combined deep learning with a kernelized SVM in cloud radio access networks, Bousalem *et al.* [25] targeted 5G V2I/V2N links, and Li *et al.* [26] classified jamming in OFDM-based UAV communications using spectrogram-tailored models. Abhishek and Gurusamy [27] specifically addressed the energy cost of detection in vehicular networks. While accuracy improves across this progression, model capacity also grows, and few of these studies report the parameter count, memory footprint, or inference latency that would determine whether the model can run on a microcontroller-class device.

2.3. TinyML and edge deployment

TinyML addresses precisely this constraint by compressing trained networks for execution on low-power microcontrollers [8], [9], [15], [28]. The dominant compression techniques—pruning and post-training quantization—were formalized in the deep-compression framework [29] and are now standard. Critically, however, compression is not free: reducing weight precision can degrade accuracy, and the extent of that degradation depends on the architecture being compressed. A network with substantial redundancy tolerates quantization well; a network that has been fitted tightly to its training data may not.

This trade-off is illustrated concretely by Hussain *et al.* [14], the study most directly comparable to the present work. The authors collected an empirical RSS dataset using software-defined radio hardware, trained a 1D-CNN with two 128-filter convolutional layers to classify three channel states, and deployed the model on a Raspberry Pi. The work is valuable in that it provides genuine hardware measurements and an actual edge deployment. Nevertheless, the authors themselves report three limitations: a test accuracy of 86.33%, overfitting beginning after approximately 18 training epochs, and—most significantly for TinyML—a collapse of accuracy to 33.3% when post-training quantization was applied, which the authors attribute to the overfitted model. The consequence is that the compressed model was not usable in practice.

2.4. Research gap

Taken together, the literature reveals two gaps. First, high detection accuracy, multi-class discrimination, empirical validation, and compactness have each been demonstrated, but not simultaneously within a single framework. Second, and more specifically, the temporal window over which RSS is aggregated is treated throughout this body of work as an implementation detail: window lengths are reported, but their effect on detectability is not examined. For duty-cycled interference this omission is consequential, since the relationship between the window length and the jammer activity period determines whether the alternation between active and silent phases is observable at all. The present work addresses both gaps by treating window length as the primary experimental variable, evaluating a deliberately small 1D-CNN across four window lengths on the same empirical three-class dataset, and reporting the full experimental configuration together with repeated-run statistics.

3. METHOD

This section describes the deployment context in which the detector is intended to operate, the empirical dataset used for training and evaluation, the preprocessing and windowing procedure, the network architecture, and the training and quantization configuration. The experimental protocol is reported in full to support reproducibility.

3.1. Smart home network topology

To frame the detection scenario in a realistic deployment context, a smart-home network was modeled using the NS-3 simulator [16], as illustrated in Figure 1. The topology consists of ten diverse IoT nodes, including smart sensors and a central gateway, operating on Wi-Fi and LoRa protocols, with an attacker node introduced ten seconds after the simulation start.

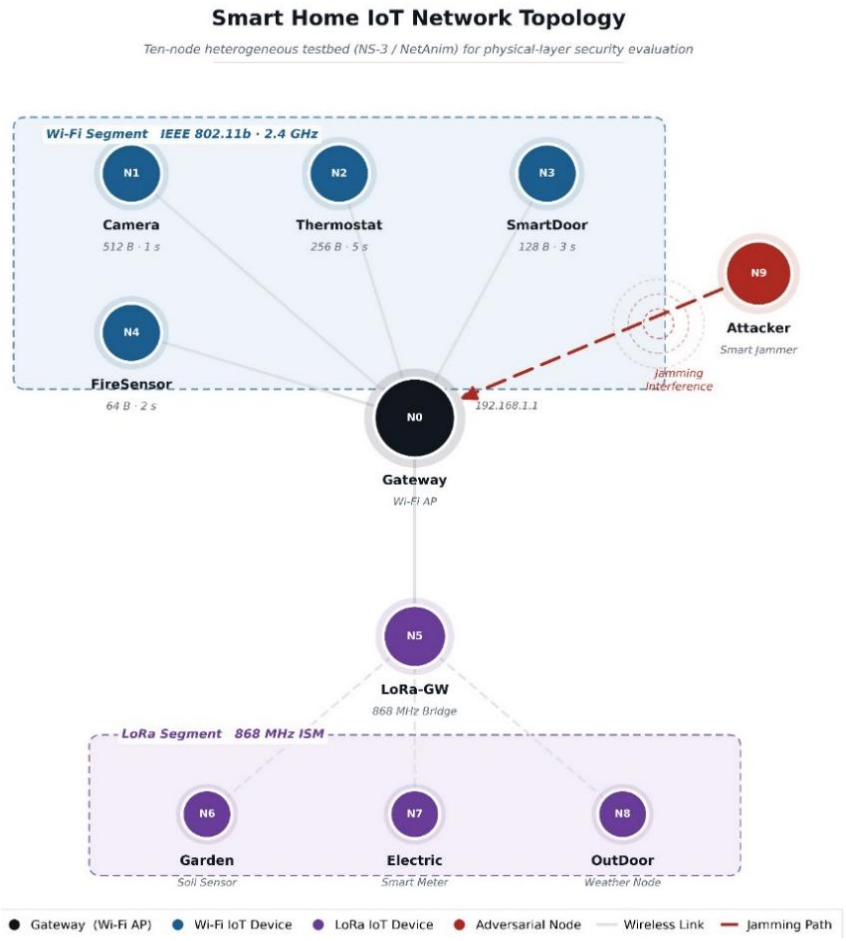


Figure 1. NS-3 simulation topology of the smart-home IoT network, showing Wi-Fi and LoRa device placement relative to the attacker node

This simulation establishes deployment context only. The classification model described in the remainder of this section was trained and evaluated exclusively on the empirical RSS dataset introduced in section 3.2; no data were generated by the NS-3 simulation for that purpose.

3.2. Dataset

The dataset consists of real-world RSS measurements acquired with software-defined radio hardware and released publicly by Hussain *et al.* [14]. Data acquisition did not form part of the present work; the pre-collected dataset is used directly. It comprises three recordings corresponding to the three channel states considered here: an unjammed channel, a constant jammer, and a periodic jammer, the latter treated as the smart-jamming class because its duty-cycled transmit and sleep behavior is representative of reactive strategies designed to evade continuous-threshold detectors. Table 1 summarizes the statistical characteristics of each class.

The standard deviations in Table 1 already indicate the difficulty of the task. Constant jamming occupies a narrow, strongly elevated amplitude band, whereas smart jamming exhibits the largest dispersion of the three classes because its recording alternates between an active phase resembling constant jamming and a silent phase resembling normal traffic. Figure 2 shows the corresponding RSS traces, and Figure 3 reproduces the signal-acquisition flowgraph reported by the dataset authors.

Table 1. Statistical characteristics of the three RSS classes in the empirical dataset [14]

Label	Scenario	Mean RSS (dBm)	Standard deviation
0	Normal channel	−80.48	12.63
1	Constant jamming	−30.73	5.55
2	Smart (periodic) jamming	−59.96	25.54

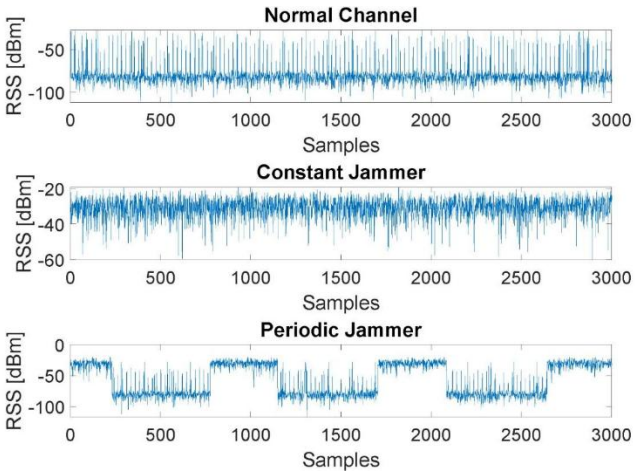


Figure 2. RSS signal patterns for the normal, constant jamming, and smart (periodic) jamming classes, as recorded in the empirical dataset [14]

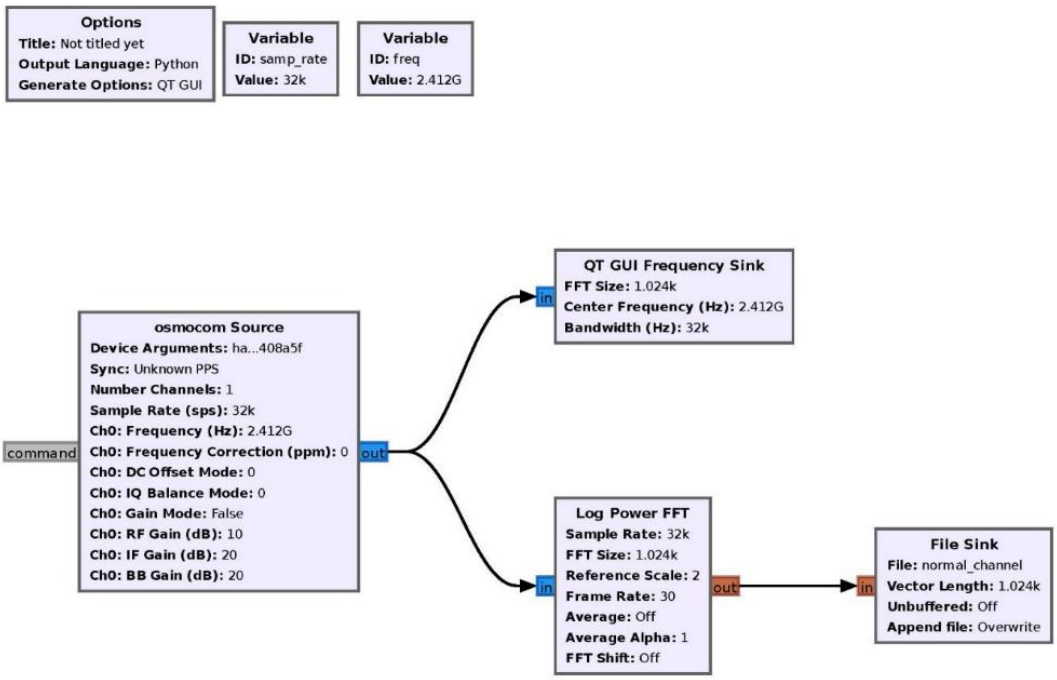


Figure 3. Software-defined radio acquisition flowgraph used by Hussain *et al.* [14] to collect the dataset, data collection was performed by the dataset authors and did not form part of the present work

From each recording, 120,000 samples were extracted to obtain an exactly balanced three-class set, and raw values were clipped to the physically plausible range [−105, −20] dBm to remove sensor artifacts. Compared with widely used simulated intrusion-detection datasets for wireless sensor networks such as WSN-DS [30], these recordings retain hardware-specific distortion and environmental multipath behavior.

3.3. Preprocessing and window extraction

A single RSS reading carries insufficient temporal context to separate smart jamming from normal communication, since during its silent intervals the intermittent jammer produces amplitudes that coincide with the unjammed range. The model therefore operates on short temporal windows: W consecutive RSS measurements are aggregated into one input vector using a stride of W/2, giving 50% overlap. Because the choice of W is the central experimental variable of this study, values of 16, 32, 64, and 128 samples were each evaluated, and W = 128 is adopted as the primary configuration on the basis of the results in section 4.1.

To prevent data leakage, the window dataset was partitioned before any augmentation was applied. Stratified sampling produced training (70%), validation (15%), and test (15%) subsets, so that the validation and test subsets contain only original, unaugmented windows. Augmentation was then applied exclusively to the training subset and consisted of additive zero-mean Gaussian noise with a standard deviation of 5.0 dBm applied to each training window, doubling the size of the training set while emulating realistic multipath variability. Z-score normalization parameters were computed from the augmented training subset alone and applied unchanged to the validation and test subsets.

Because the stride scales with W, the number of extracted windows decreases as the window lengthens: from 44,997 windows in total at W = 16 to 5,622 at W = 128, of which 6,750 and 843 respectively form the held-out test subsets. The consequences of this trade-off between temporal context and sample count are examined in sections 4.1 and 5.

3.4. Network architecture

The classifier is a compact 1D-CNN comprising two convolutional blocks followed by global average pooling and a dense classification head. Figure 4 illustrates the architecture and Table 2 gives the complete layer specification for the primary configuration, W = 128.

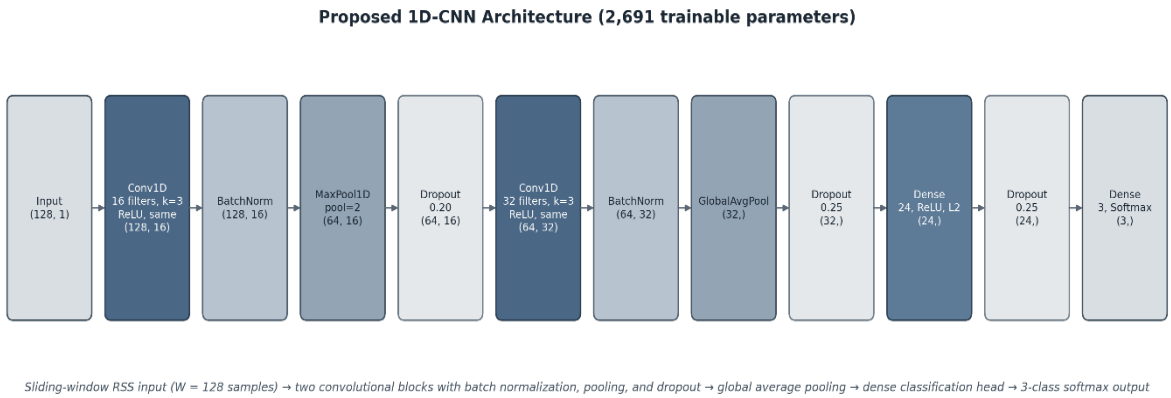


Figure 4. Layer-by-layer architecture of the proposed 1D-CNN, shown for the primary configuration W = 128, with output shapes at each stage

Table 2. Layer configuration of the proposed 1D-CNN for W = 128 (2,691 trainable parameters in total)

#	Layer	Configuration	Output shape	Parameters
1	Input	—	(128, 1)	0
2	Conv1D	16 filters, kernel 3, ReLU, same	(128, 16)	64
3	BatchNormalization	—	(128, 16)	64
4	MaxPooling1D	pool size 2	(64, 16)	0
5	Dropout	rate 0.20	(64, 16)	0
6	Conv1D	32 filters, kernel 3, ReLU, same	(64, 32)	1,568
7	BatchNormalization	—	(64, 32)	128
8	GlobalAveragePooling1D	—	(32,)	0
9	Dropout	rate 0.25	(32,)	0
10	Dense	24 units, ReLU, L2 = 0.001	(24,)	792
11	Dropout	rate 0.25	(24,)	0
12	Dense (output)	3 units, Softmax	(3,)	75
	Total	—	—	2,691

Global average pooling is used in place of a flattening layer before the dense head. At W = 128 a flattening layer would present 2,048 activations to the first dense layer, requiring approximately 49,000 parameters at that layer alone, whereas global average pooling reduces the same tensor to 32 activations and 792 parameters. This single design choice accounts for the greater part of the model's compactness and is the principal reason the network remains within TinyML memory budgets at a 128-sample input length.

3.5. Training configuration

The model was trained with the Adam optimizer at a learning rate of 0.001 using sparse categorical cross-entropy loss, with a batch size of 128 over a maximum of 60 epochs. Early stopping monitored

validation accuracy with a patience of 12 epochs and restored the best weights; the learning rate was halved after six consecutive epochs without improvement in validation loss, subject to a floor of 1×10^{-6}. Regularization comprised dropout at rates of 0.20 in the convolutional block and 0.25 in the dense head, L2 weight decay with a coefficient of 0.001 on the dense layer, and the training-set augmentation described in section 3.3. Table 3 summarizes these settings, which were applied unchanged across all window lengths evaluated.

Table 3. Training configuration and hardware, applied identically across all evaluated window lengths

Parameter	Value
Samples per class	120,000
Window stride	W/2 (50% overlap)
Dataset split (train / validation / test)	70% / 15% / 15%, stratified
Optimizer	Adam
Learning rate	0.001
Batch size	128
Maximum epochs	60
Early stopping patience	12 epochs (validation accuracy)
LR reduction (factor / patience / floor)	0.5 / 6 epochs / 1×10^{-6}
Dropout (convolutional / dense)	0.20 / 0.25
L2 coefficient	0.001
Augmentation noise σ	5.0 dBm (training subset only)
Hardware	Intel Core i9 (13th Gen), 16 GB RAM, NVIDIA RTX 4060

3.6. TinyML conversion

After training, the model was converted to TensorFlow Lite format using post-training dynamic-range quantization, which maps the 32-bit floating-point weights to 8-bit integers while retaining floating-point activations and requires no representative calibration dataset. This strategy was preferred over full-integer quantization for its implementation simplicity and because it avoids the need for a calibration set. Rather than assuming that quantization preserves accuracy, the converted model was re-evaluated directly on the same held-out test subset; the measured outcome is reported in section 4.5.

4. RESULTS AND DISCUSSION

This section reports the effect of temporal window length on detection performance, verifies the stability of that effect across independent runs, analyzes per-class behavior and the structure of the residual errors, compares the model with prior work on the equivalent binary task, and reports the measured cost of TinyML quantization.

4.1. Effect of temporal window length

Table 4 reports classification performance for four window lengths, each evaluated on its own held-out test subset. All other hyperparameters were held fixed, so the only variable is the amount of temporal context available to the network.

Table 4. Detection performance as a function of temporal window length

Window W (samples)	Test windows	Overall accuracy	Smart jamming recall
16	6,750	72.99%	31.51%
32	3,375	77.21%	42.40%
64	1,687	80.68%	54.71%
128	843	85.77%	60.14%

Both metrics increase monotonically with window length. Overall accuracy rises by 12.8 percentage points between W = 16 and W = 128, but the effect on smart jamming recall is far larger: detection of the smart jammer nearly doubles, from 31.51% to 60.14%. The gain is therefore concentrated almost entirely in the class that short windows fail to characterize, which is the central empirical observation of this work.

It is worth noting that the longer windows achieve this with substantially fewer training and test examples, since the stride scales with W. Performance improves despite an eightfold reduction in sample count between W = 16 and W = 128, which indicates that the limiting factor at short window lengths is the information content of each individual window rather than the quantity of windows available.

4.2. Stability across independent runs

To confirm that the trend in Table 4 is not an artifact of a particular weight initialization or data partition, the full experiment was repeated five times with different random seeds. Table 5 reports the mean and standard deviation for the two endpoint configurations, and Figure 5 shows the corresponding trend with error bars.

Table 5. Mean ± standard deviation over five independent runs with different random seeds

Window W (samples)	Overall accuracy	Smart jamming recall
16	73.17 ± 0.15%	30.36 ± 2.90%
128	86.03 ± 0.63%	64.20 ± 3.80%

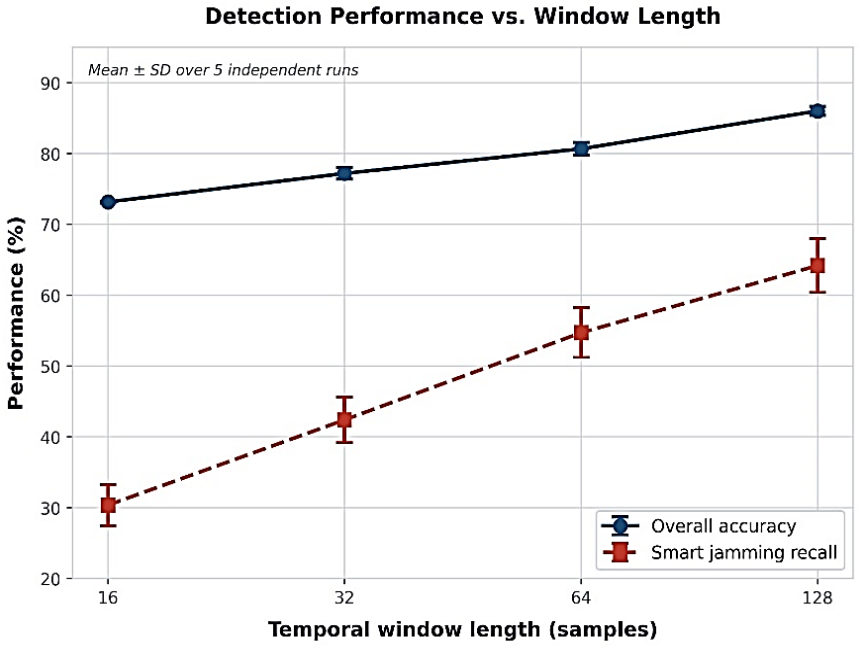


Figure 5. Overall accuracy and smart jamming recall as a function of temporal window length, error bars denote the standard deviation over five independent runs

The accuracy gap of 12.9 percentage points exceeds the observed standard deviations by more than an order of magnitude, and the increase in smart jamming recall of 33.8 points is roughly nine times the larger of the two standard deviations. The effect is therefore systematic rather than incidental. The standard deviations themselves are informative: overall accuracy is highly reproducible (0.15% and 0.63%), whereas smart jamming recall varies considerably more across seeds (2.90% and 3.80%), which is consistent with this class lying closest to the decision boundary and therefore being the most sensitive to initialization.

4.3. Per-class performance

Table 6 reports per-class metrics for the primary configuration at W = 128, taken from a representative run. The three classes behave very differently, and the pattern is physically interpretable. Constant jamming is detected without a single omission (100% recall), which follows from its statistical profile in Table 1: a mean of −30.73 dBm with a standard deviation of only 5.55 dB places it in a region of the amplitude space that neither other class occupies.

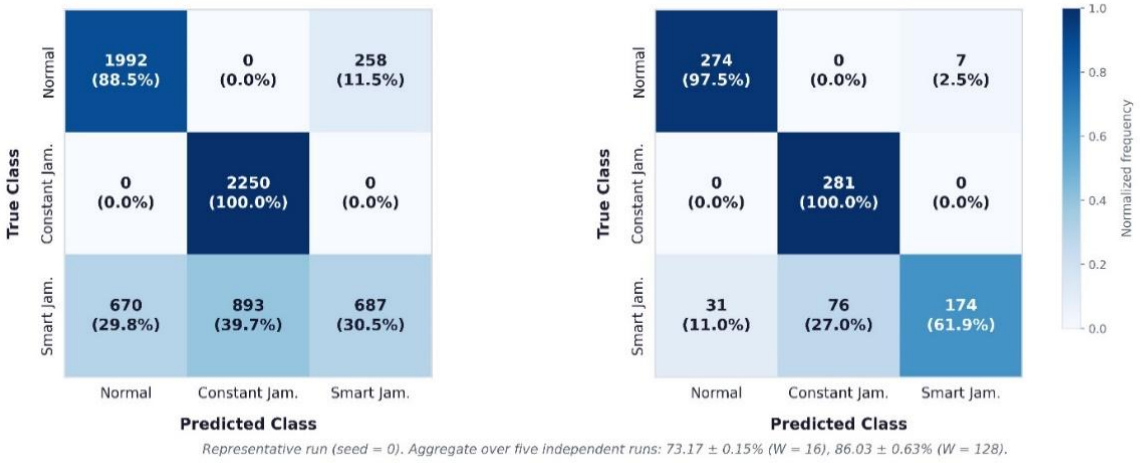
Table 6. Per-class performance at W = 128 (representative run, overall accuracy 86.48%)

Class	Precision	Recall	F1-score	Support
Normal	89.84%	97.51%	93.52%	281
Constant jamming	78.71%	100.00%	88.09%	281
Smart jamming	96.13%	61.92%	75.32%	281
Macro average	88.23%	86.48%	85.64%	843

Smart jamming shows the opposite asymmetry: precision is the highest of the three classes at 96.13%, while recall is the lowest at 61.92%. When the model reports smart jamming, it is almost always correct, but it fails to flag a substantial fraction of smart jamming windows. For an intrusion detection system this asymmetry is the preferable one, since high precision corresponds to a low false-alarm rate, and a jamming episode spanning many consecutive windows would still be detected on the majority of them. The corresponding cost appears as reduced precision for constant jamming (78.71%), which absorbs most of the misdirected smart jamming windows.

4.4. Error analysis and the duty-cycle explanation

Figure 6 presents the confusion matrices at both endpoint window lengths and reveals not merely that errors become fewer as W increases, but that they change in kind. At W = 16 the smart jamming windows are distributed almost evenly across all three predicted classes: 687 of 2,250 are classified correctly, 893 are assigned to constant jamming, and 670 to normal traffic as shown in Figure 6(a). The model is effectively unable to characterize this class. At W = 128 the residual confusion is both smaller and asymmetric: of 281 smart jamming windows, 174 are correct, 76 are assigned to constant jamming, and only 31 to normal traffic as shown in Figure 6(b).



(a) (b)

Figure 6. Confusion matrices for (a) W = 16 and (b) W = 128, representative run (seed 0), aggregate performance over five independent runs: 73.17 ± 0.15% and 86.03 ± 0.63% respectively

This shift is precisely what the duty-cycle hypothesis predicts. A smart jammer alternates between an active phase and a silent phase. A window shorter than the jammer's activity period falls entirely within one phase: if within an active phase it presents the amplitude statistics of constant jamming, and if within a silent phase those of an unjammed channel. No classifier can separate these cases from the genuine classes on amplitude information alone, because the observations are in fact identical. A window long enough to span a complete duty cycle contains both phases and therefore carries the alternation itself as a discriminative feature.

The nature of the remaining errors at W = 128 supports this reading. Misclassification toward constant jamming dominates over misclassification toward normal traffic by a factor of roughly 2.5. In other words, the model correctly recognizes that interference is present in the large majority of cases and errs mainly in identifying which type of interference it is. From an operational standpoint this is a considerably less serious failure than missing the attack altogether: merging the two jamming classes, only 31 of 843 windows are wrongly reported as normal.

4.5. Comparison with prior work and TinyML deployment

Hussain *et al.* [14] report 86.33% accuracy on this dataset using a convolutional architecture with two 128-filter layers, noting the onset of overfitting beyond epoch 18 of a 60-epoch schedule. Their evaluation addresses the binary Normal-versus-Jamming problem, whereas the present work addresses the harder three-class problem, so the two figures are not directly comparable.

To enable a like-for-like comparison, the three-class predictions of the proposed model were mapped onto the binary task by merging both jamming classes, without any retraining or task-specific tuning. Table 7 reports the outcome alongside the baseline.

On the binary task, the proposed model reaches 95.49% accuracy, exceeding the baseline by approximately nine percentage points while using an order of magnitude fewer convolutional filters. On the harder three-class objective, it attains 86.03 ± 0.63%, which is comparable to the baseline's binary result. Two qualifications apply. First, the baseline figures are taken from its authors' published report rather than from an independent re-implementation, so the comparison is documentary rather than a controlled experiment. Second, the binary figures reported here are derived from the three-class confusion matrix of a representative run rather than from a separately trained binary classifier.

Figure 7 shows the training and validation curves for the primary configuration. Validation accuracy tracks training accuracy without the divergence characteristic of overfitting, and validation loss does not rise after an initial minimum. The early instability in the validation curves reflects the small validation subset at W = 128 rather than a training pathology, and resolves within approximately ten epochs.

Table 7. Comparison with the documented baseline evaluated on the same empirical dataset. Binary figures for the proposed model are derived from the three-class confusion matrix in Figure 6(b) by merging the two jamming classes

Aspect	Baseline CNN [14]	Proposed model
Task reported	Binary (normal vs. jamming)	Three-class, mapped to binary
Convolutional filters	128 and 128	16 and 32
Trainable parameters	Not reported	2,691
Binary accuracy	86.33%	95.49%
Three-class accuracy	Not reported	86.03 ± 0.63%
Reported overfitting	Yes, beyond epoch 18	Not observed (Figure 7)

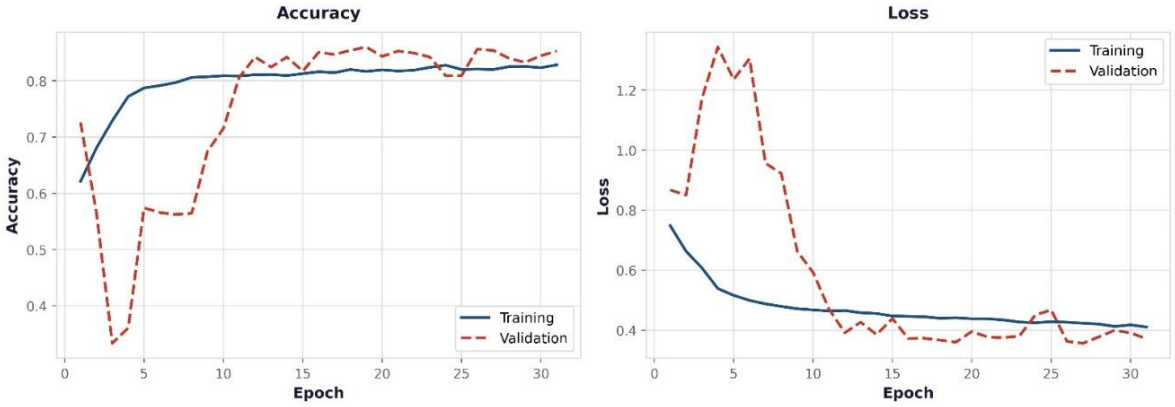


Figure 7. Training and validation accuracy and loss for the primary configuration (W = 128)

The trained model comprises 2,691 trainable parameters. Following conversion to TensorFlow Lite with dynamic-range quantization, it occupies 14,168 bytes, or 13.8 KB. Rather than assuming that accuracy survives this conversion, the quantized model was evaluated directly on the same held-out test subset, yielding 86.36% against 86.48% for the full-precision model, a degradation of 0.12 percentage points. Table 8 summarizes these measurements.

Table 8. Measured characteristics of the proposed model after TinyML dynamic-range quantization

Metric	Value
Trainable parameters	2,691
Weight precision after conversion	int8 (from float32)
Quantized model size	14,168 bytes (13.8 KB)
Full-precision test accuracy	86.48%
Quantized test accuracy (measured)	86.36%
Accuracy cost of quantization	0.12 percentage points

A 13.8 KB model with fewer than three thousand parameters falls well within the flash and RAM budgets of common microcontroller-class devices, and the measured accuracy cost of quantization is negligible. It should nonetheless be stated that no deployment on physical embedded hardware was carried out in this work, so on-device latency and energy consumption remain unmeasured; this is identified as a limitation in section 5.

5. CONCLUSION

This paper investigated RSS-based intrusion detection for the IoT physical layer and identified the temporal window length as the dominant factor governing smart jamming detectability. Extending the analysis window from 16 to 128 samples raised three-class accuracy from 73.17 ± 0.15% to 86.03 ± 0.63% and, more strikingly, smart jamming recall from 30.36 ± 2.90% to 64.20 ± 3.80% across five independent runs. The gains far exceed the run-to-run variability and are concentrated in precisely the class that short windows cannot characterize.

The explanation is physical rather than architectural. A smart jammer alternates between active and silent phases; a window shorter than its activity period falls wholly within one phase and is therefore genuinely indistinguishable from constant jamming or from an unjammed channel on amplitude information alone. Only a window spanning a complete duty cycle carries the alternation itself as a discriminative feature. The confusion matrices support this reading: at the longer window the residual errors concern which type of interference is present rather than whether interference is present at all, with only 31 of 843 windows wrongly reported as normal.

Unlike prior RSS-based approaches addressing binary classification, this work performs three-class discrimination. Mapped onto the equivalent binary task without retraining, the model reaches 95.49% accuracy, exceeding the documented baseline by roughly nine percentage points while using an order of magnitude fewer convolutional filters. With 2,691 trainable parameters and a quantized footprint of 13.8 KB, at a measured post-quantization accuracy cost of 0.12 percentage points, the model is compatible with TinyML memory budgets.

Several limitations should be stated explicitly. First, smart jamming recall remains at 64.20 ± 3.80% even at the best window length, indicating that RSS amplitude alone carries incomplete information about duty-cycled interference. Second, extending the window beyond 128 samples did not yield reliable further improvement: accuracy at 256 and 512 samples measured 79.15% and 87.68% respectively, but the corresponding test subsets contained only 422 and 211 windows, so these differences are statistically inconclusive; resolving the question would require a substantially larger underlying dataset. Third, no deployment on physical embedded hardware was performed, so on-device latency and energy consumption remain unmeasured. Fourth, the comparison with prior work is documentary, drawing on figures published by the baseline authors rather than an independent re-implementation, and the binary figures reported here are derived from the three-class confusion matrix of a representative run rather than from a separately trained binary classifier. Fifth, no experimental comparison against classical classifiers such as SVM or RF was carried out. Sixth, the evaluation rests on a single publicly available dataset collected in a controlled environment, so generalization to deployments with different propagation characteristics remains to be established. Finally, the NS-3 smart-home simulation provides deployment context but was not integrated with the classifier as a single end-to-end pipeline.

Future work follows directly from these limitations. The most immediate priority is deployment on embedded hardware to obtain measured latency, memory, and energy figures under realistic operating conditions. Beyond that, the finding that window length governs detectability suggests that features encoding temporal structure more explicitly, such as spectral or higher-order statistical descriptors, may raise smart jamming recall further without requiring longer windows and the associated loss of sample count. Controlled benchmarking against classical and deep-learning baselines trained under identical conditions, evaluation on additional jammer types including reactive and deceptive strategies, and validation on larger and more diverse datasets would together establish the generality of the approach.

ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to the Faculty of Engineering at the Syrian Private University (SPU).

FUNDING INFORMATION

The authors state that no funding was involved in this research.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration. All authors have read and approved the final version of the manuscript. Qasim Oqlah is the corresponding author and is responsible for all communication regarding this paper.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Qasim Oqlah	✓	✓				✓								✓
Saad Al-Mobark			✓					✓		✓				
Samer Rahmah			✓				✓		✓		✓			✓
Wassim Al Juneidi	✓			✓	✓							✓	✓	
Yamen Hallak		✓												✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors state that there is no conflict of interest.

INFORMED CONSENT

This study did not involve human participants or personal data requiring informed consent, as all experiments were conducted using technical signal data and network simulations.

ETHICAL APPROVAL

This research does not involve any studies with human participants or animals performed by any of the authors. All experiments were conducted using publicly available technical datasets and network simulations, in accordance with the institutional ethical guidelines of the Syrian Private University.

DATA AVAILABILITY

The empirical RSS dataset analyzed in this study was released as open source by Hussain *et al.* [14] and is publicly available at https://github.com/AMHD/Jamming-Detection-in-IoT-Wireless-Networks-An-Edge-AI-Based-Approach All derived results supporting the findings of this study are reported within the article.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: a survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/comst.2015.2444095.

[2] H. Pirayesh and H. Zeng, "Jamming attacks and anti-jamming strategies in wireless networks: a comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 767–809, 2022, doi: 10.1109/comst.2022.3159185.

[3] S. Vadlamani, B. Eksioglu, H. Medal, and A. Nandi, "Jamming attacks on wireless networks: A taxonomic survey," *International Journal of Production Economics*, vol. 172, pp. 76–94, Feb. 2016, doi: 10.1016/j.ijpe.2015.11.008.

[4] K. B. Letaief, Y. Shi, J. Lu, and J. Lu, "Edge artificial intelligence for 6G: vision, enabling technologies, and applications," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 5–36, Jan. 2022, doi: 10.1109/jsac.2021.3126076.

[5] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT security techniques based on machine learning: how do IoT devices use AI to enhance security?," *IEEE Signal Processing Magazine*, vol. 35, no. 5, pp. 41–49, 2018, doi: 10.1109/msp.2018.2825478.

[6] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (IoT) security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020, doi: 10.1109/comst.2020.2988293.

[7] L. Sun and Q. Du, "A review of physical layer security techniques for internet of things: challenges and solutions," *Entropy*, vol. 20, no. 10, p. 730, 2018, doi: 10.3390/e20100730.

[8] P. Warden and D. Situnayake, "TinyML: machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers. sebastopol," *CA, USA: O'Reilly Media*, 2019.

[9] P. P. Ray, "A review on TinyML: State-of-the-art and prospects," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 4, pp. 1595–1623, Apr. 2022, doi: 10.1016/j.jksuci.2021.11.019.

[10] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mechanical Systems and Signal Processing*, vol. 151, p. 107398, Apr. 2021, doi: 10.1016/j.ymssp.2020.107398.
[11] B. Upadhyaya, S. Sun, and B. Sikdar, "Machine learning-based jamming detection in wireless IoT networks," in *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, Aug. 2019, pp. 1–5, doi: 10.1109/vts-apwcs.2019.8851633.
[12] Y. Arjoune, F. Salahdine, M. S. Islam, E. Ghribi, and N. Kaabouch, "A novel jamming attacks detection approach based on machine learning for wireless communication," in *2020 International Conference on Information Networking (ICOIN)*, Jan. 2020, pp. 459–464, doi: 10.1109/icoin48656.2020.9016462.
[13] O. Punal, I. Aktas, C.-J. Schnelke, G. Abidin, K. Wehrle, and J. Gross, "Machine learning-based jamming detection for IEEE 802.11: Design and experimental evaluation," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, 2014, pp. 1–10, doi: 10.1109/wowmom.2014.6918964.
[14] A. Hussain, N. Abughanam, J. Qadir, and A. Mohamed, "Jamming detection in IoT wireless networks: an edge-AI based approach," in *Proceedings of the 12th International Conference on the Internet of Things*, Nov. 2022, pp. 57–64, doi: 10.1145/3567445.3567456.
[15] L. Capogrosso, F. Cunico, D. S. Cheng, F. Fummi, and M. Cristani, "A machine learning-oriented survey on tiny machine learning," *IEEE Access*, vol. 12, pp. 23406–23426, 2024, doi: 10.1109/access.2024.3365349.
[16] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*, Springer Berlin Heidelberg, 2010, pp. 15–34.
[17] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "A survey on jamming attacks and countermeasures in WSNs," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 4, pp. 42–56, 2009, doi: 10.1109/surv.2009.090404.
[18] A. Mukherjee, "Physical-layer security in the internet of things: sensing and communication confidentiality under resource constraints," *Proceedings of the IEEE*, vol. 103, no. 10, pp. 1747–1761, Oct. 2015, doi: 10.1109/jproc.2015.2466548.
[19] E. Illi *et al.*, "Physical layer security for authentication, confidentiality, and malicious node detection: a paradigm shift in securing IoT networks," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 1, pp. 347–388, 2024, doi: 10.1109/comst.2023.3327327.
[20] Z. Feng and C. Hua, "Machine learning-based RF jamming detection in wireless networks," in *2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC)*, Oct. 2018, pp. 1–6, doi: 10.1109/ssic.2018.8556709.
[21] G. S. Kasturi, A. Jain, and J. Singh, "Detection and classification of radio frequency jamming attacks using machine learning," *ournal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 11, no. 2, pp. 49–62, 2020.
[22] T. Erpek, Y. E. Sagduyu, and Y. Shi, "Deep learning for launching and mitigating wireless jamming attacks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 1, pp. 2–14, Mar. 2019, doi: 10.1109/tccn.2018.2884910.
[23] M. Premkumar and T. V. P. Sundararajan, "DLDM: Deep learning-based defense mechanism for denial of service attacks in wireless sensor networks," *Microprocessors and Microsystems*, vol. 79, p. 103278, Nov. 2020, doi: 10.1016/j.micpro.2020.103278.
[24] M. Hachimi, G. Kaddoum, G. Gagnon, and P. Illy, "Multi-stage jamming attacks detection using deep learning combined with kernelized support vector machine in 5G cloud radio access networks," in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, Oct. 2020, pp. 1–5, doi: 10.1109/isncc49221.2020.9297290.
[25] B. Bousalem, V. F. Silva, A. Boualouache, R. Langar, and S. Cherrier, "Deep learning-based smart radio jamming attacks detection on 5G V2I/V2N communications," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, Dec. 2023, pp. 7139–7144, doi: 10.1109/globecom54140.2023.10437442.
[26] Y. Li *et al.*, "Jamming detection and classification in OFDM-Based UAVs via feature- and spectrogram-tailored machine learning," *IEEE Access*, vol. 10, pp. 16859–16870, 2022, doi: 10.1109/access.2022.3150020.
[27] N. V. Abhishek and M. Gurusamy, "JaDe: low power jamming detection using machine learning in vehicular networks," *IEEE Wireless Communications Letters*, vol. 10, no. 10, pp. 2210–2214, Oct. 2021, doi: 10.1109/lwc.2021.3097162.
[28] R. H. Jhaveri, H. R. Chi, and H. Wu, "TinyML for empowering low-power IoT edge consumer devices," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 4, pp. 7318–7321, Nov. 2024, doi: 10.1109/tce.2024.3482353.
[29] S. Han *et al.*, "Deep compression and EIE: Efficient inference engine on compressed deep neural network," Aug. 2016, doi: 10.1109/hotchips.2016.7936226.
[30] I. Almomani, B. Al-Kasasbeh, and M. AL-Akhras, "WSN-DS: a dataset for intrusion detection systems in wireless sensor networks," *Journal of Sensors*, vol. 2016, pp. 1–16, 2016, doi: 10.1155/2016/4731953.

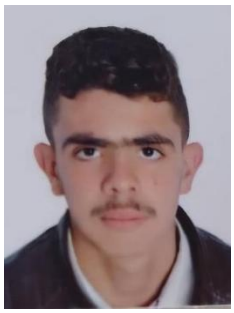
BIOGRAPHIES OF AUTHORS



Qasim Oqlah is currently a student in the Faculty of Artificial Intelligence Engineering, majoring in Smart Information Systems Security, at the Syrian Private University (SPU). Simultaneously, he is pursuing Postgraduate Diplomas in Cybersecurity and Computer Networks at the Syrian Virtual University (SVU). He can be contacted at email: Qasim_Oqlah@spu.edu.sy.



Saad Al-Mobark is currently a student in the Faculty of Artificial Intelligence Engineering, majoring in Smart Information Systems Security, at the Syrian Private University (SPU). He can be contacted at email: saad_almobark@spu.edu.sy.



Samer Rahmah is currently a student in the Faculty of Artificial Intelligence Engineering, majoring in Smart Information Systems Security, at the Syrian Private University (SPU). He can be contacted at email: samer_rahmah@spu.edu.sy.



Prof. Dr. Wassim Al Juneidi received his Engineering degree in Telecommunication (2000), his M.Sc. degree in Radio Networks (2013), and his Ph.D. degree in Wireless Sensor Networks with High Honors (2016) from the Higher Institute for Applied Sciences and Technology (HIAST), Syria. He is an Associate Professor across several academic institutions, including the Syrian Virtual University (SVU), HIAST, and the Syrian Private University (SPU). Throughout his career, he has held key leadership positions, serving as Director of Master Programs (MNT and NIST) at SVU and General Manager of the National Agency for IT Services (NAITS). He is a certified ISO 27001 trainer and auditor, specializing in cybersecurity, wireless sensor networks, intrusion detection systems, cloud computing, and IoT architecture. Prof. Dr. Aljuneidi has authored academic textbooks and published research papers in international peer-reviewed journals focusing on telecommunications, wireless networks, and embedded systems. He can be contacted at email: Wassiem_jnidi@spu.edu.sy.



Yamen Hallak is a scientific and academic researcher in cybersecurity. He is a faculty member at the Syrian Private University (SPU) and a teacher assistant at Damascus University. He can be contacted at email: yamen.hallak@damascusuniversity.edu.sy.