Jamming Detection in IoT Wireless Networks: An Edge-AI Based Approach

Ahmed Hussain*
Qatar University
Doha, Qatar
ahmed.hussain@qu.edu.qa

Nada Abughanam*
Qatar University
Doha, Qatar
nada.abughanam@qu.edu.qa

Junaid Qadir
Qatar University
Doha, Qatar
jqadir@qu.edu.qa

Amr Mohamed
Qatar University
Doha, Qatar
amrm@qu.edu.qa

ABSTRACT

Wireless enabling technologies in critical infrastructures are increasing the efficiency of communications. In the era of 5G and beyond, more technologies will be allowed to connect to mobile networks, enabling the Internet of Things (IoT) on a massive scale. Most of these technologies are vulnerable to physical-layer security attacks, namely jamming. Jamming attacks are among the most effective techniques to attack and compromise the availability of these wireless technologies. Jamming is an interfering signal that limits the intended receiver from correctly receiving the messages. Once the adversary deploys a jammer in a wireless network, jammer detection becomes difficult, if not impossible, due to the inaccessibility of the affected devices in the network. This paper extends the state-of-the-art jamming detection and classification methods by proposing an effective IoT Tiny Machine Learning (TinyML)-based approach, where a trained deep learning model is deployed on an IoT edge device, namely a Raspberry Pi. The model is built using TensorFlow and deployed on the IoT device using TensorFlow lite. The trained model encompasses two commonly known jamming types: constant and periodic, in addition to the normal channel state. The Raspberry Pi is connected to a Software Defined Radio (SDR) that continuously senses the WiFi channel and acquires Received Signal Strength (RSS) readings which the TinyML model evaluates to detect the presence of jamming and its type. We release both the procedure and collected dataset for the different types of jamming as open source. Finally, we conducted an extensive testing campaign to test, evaluate, and illustrate the effectiveness of the proposed TinyML-based detection on the edge scheme.

*Both authors contributed equally to this research.

IoT '22, November 7–10, 2022, Delft, Netherlands
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9665-3/22/11…$15.00
https://doi.org/10.1145/3567445.3567456

CCS CONCEPTS

• **Network security** → Mobile and wireless security; • **Computing methodologies** → *Machine learning approaches*; • **Security and privacy** → *Systems security*.

KEYWORDS

Jamming, Internet of Things, Wireless Communication, Deep Learning, Edge AI, Tensorflow, TinyML

ACM Reference Format:
Ahmed Hussain, Nada Abughanam, Junaid Qadir, and Amr Mohamed. 2022. Jamming Detection in IoT Wireless Networks: An Edge-AI Based Approach. In *Proceedings of the 12th International Conference on the Internet of Things (IoT '22), November 7–10, 2022, Delft, Netherlands.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3567445.3567456

1 INTRODUCTION

The development of fifth-generation (5G) and beyond wireless networks is proliferating, intending to connect practically all parts of life via a network with substantially faster speed, very low latency, and pervasive connection. Because of the extraordinary expansion in the provided services and the number of devices, the considerations for the security of 5G and beyond have increased considerably. It is estimated that the number of IoT devices by the end of 2022 to be more than 29 billion devices [6]. Jamming attacks can be performed on the intelligent sensing layer, and they are considered a significant threat that disrupts and can jeopardize fully connected networks. In addition, jamming could stall the operations where wireless nodes are performing data collection.

Jamming attacks are denial of service attacks that disrupt wireless radio communications by overlapping legitimate signals with a noise signal with significantly higher power to reduce the Signal-to-Noise Ratio (SNR) via a jamming device. These attacks prevent the transmitter from transmitting messages to a receiver by occupying the channel and making it appear busy, thus preventing any legitimate messages from being exchanged by the transmitter and receiver in a specific area. Examples of critical targets include locations such as airports and sensitive infrastructures [13]. Nowadays, jamming attacks can be implemented with low complexity and low-cost signal generator devices due to the evolution of SDRs technology. With the use of a single SDRs device, the implementation of several types of jamming attacks is possible with minimal implementation and modifications to software using tools such as GNU Radio. The availability of such devices in a cheap off-the-shelf manner has resulted in jamming attacks being easily executed. There is massive potential in the anti-jamming market, as market research shows that the market size was valued at USD 4.00 Billion in 2020, and is forecasted to grow in the period from 2021 to 2028 at a Compound Annual Growth Rate (CAGR) of 8.29%, reaching a value of USD 7.50 Billion [18]. At the time of writing, many solutions have been proposed in the last decade for jamming detection. However, none of the currently-available contributions entirely took and experimented with an actual implementation that involves jamming detection using a combination of Artificial Intelligence (AI) and IoT-edge devices. Operating under normal conditions in a wireless network makes it difficult to detect jamming attacks since the connection between the base station or the other devices could be affected by the environment or the jamming. A jamming detection procedure involves at least the deployment of one node that incorporates several strategies [14] which samples the jammer transmission to detect and classify the type of jammer.

Our work takes inspiration from the upcoming trend of TinyML, which promotes the incorporation of intelligence on low-power resource-constrained devices by enabling on-device machine learning and deep learning. TinyML is an emerging field in artificial intelligence that offers machine learning solutions closer to the source, on low-powered, reducing the power consumption and network traffic and providing high availability and low latency to the end-users [17]. Deep learning methods can be utilized on the deployed nodes to determine jamming attacks. The inference is performed on edge rather than sending it to the cloud, reducing the latency and saving bandwidth.

Thus, the existing literature currently lacks an actual implementation of a practical and portable IoT AI-based jammer detection that operates on the edge ensuring reliability and efficiency even in hostile scenarios. To the best of our knowledge, we are the first to introduce such a system that utilizes TinyML to predict the jamming type in a portable format.

Contribution. In this paper, we provide the following contributions:

(1) We extend the current state-of-the-art by proposing a portable, IoT TinyML-based edge system implementation that enables classifying the type of jamming attack based on the signal received by low cost and off-the-shelf hardware.
(2) We perform an RSS data collection campaign that encompasses two different types of jamming attacks with roughly 1,230 signals[1] and this dataset is publicly released [2].
(3) We utilize Deep Learning (DL) methods to train and build a model that classifies two different types of jamming signals by taking into account only the RSS characteristics of each type.
(4) We perform the detection based on a deployed IoT device, namely, Raspberry Pi 3. In particular, samples of the RSS received by the Pi are collected and evaluated against the trained model to determine the type of jamming.
(5) Finally, we evaluate the proposed implementation by deploying the lightweight model on the Raspberry Pi and performing different jamming attacks in the wild.

[1]The total number of RSS samples for the 1230 signals is 1,230,000 samples, where each signal is 1000 samples each. (Each type is 410 signal, and each signal consist of 1000 sample)

Paper Organization. The remainder of this paper is organized as follows: Section 2 addresses the background and related work relevant to the presented topic. Section 3 discusses the scenario and the threat model. Section 4 illustrates the system architecture proposed in this paper from the hardware and software perspective. Section 5 discusses the implementation architecture of the deep learning model and its deployment using the TinyML framework. In Section 6, we explain and evaluate the performance of the proposed methodology. Finally, Section 7 wraps up the findings and results discussed in the paper.

2 BACKGROUND AND RELATED WORK

This section addresses the contributions found in the literature concerning the detection of jamming attacks and other topics related to the implementation. We divide the discussion based on the subject topic, i.e., Jamming Attacks in IoT networks, Machine Learning (ML)-based detection methods, TinyML, and conventional neural networks.

2.1 Jamming Attacks in IoT Networks

Jamming has been and is still a significant concern in research due to the availability of cheap and easy-to-use jamming methods. To this end, research has been conducted on anti-jamming techniques on many different networks, such as Wireless Networks, Wireless Sensor Network (WSN) [20], and Vehicular Ad Hoc Networks (VANET) [3]. There exist several types of jamming attacks. We focus on two types in this work, namely, constant and periodic jammers. Constant jamming is the act of continuously injecting Additive White Gaussian Noise (AWGN) into the wireless channel, and this affects the communication for an indefinite amount of time. On the other hand, periodic jamming injects the AWGN periodically into the channel, i.e., jamming for 3 seconds and sleeping for 3 seconds. Jamming detection is a challenging task as the majority of the devices on edge are only deployed to perform measurements or data collection. That makes it overhead to detect and countermeasure these types of attacks. Furthermore, the hardware capabilities of these edge devices are very limited to perform the detection in protocols operating in the 5/6G networks, e.g., LoRa, LoRaWAN, ZigBee, and several others [12].

2.2 ML-based Jamming Detection Methods

Anti-jamming techniques include jamming localization, jamming detection, and others [8]. Jamming attacks can be detected with different methods, including machine learning-based methods. Much research has focused on identifying different types of jamming attacks in wireless networks using machine and deep learning techniques. Some of the current work utilizes detection methods that include machine learning methods such as tree-based classifiers [5, 7], support vector machine (SVM) [1], K-Nearest Neighbour (KNN), and gradient boosting [10]. Additionally, a limited number of contributions addressed the use of deep learning methods [9, 15]. The metrics typically used are RSS [19], Packet Delivery

Ratio (PDR) [15], or combination of both [9]. None of the current research has attempted to perform a hardware implementation to detect attacks in real time. The aforementioned techniques have been applied in simulated environments or using existing datasets. We propose a system implementation that enables the detection of jamming attacks in a WSN using deep learning. This system is deployed on an edge device to observe the performance against two different types of jamming attacks.

2.3 TinyML

TinyML is an emerging field at the intersection of machine learning and embedded systems. It facilitates deploying and running models on small, low-cost, and low-powered devices such as microcontrollers [21]. This gives the advantage of less data transmission, as the results of predictions will be sent instead of transmitting the raw data to the server for processing, which is considered costly in terms of bandwidth and energy. This way allows data analytics to be performed on the IoT device directly with low power requirements and low latency. Developing a TinyML application can be divided into generating a trained model, developing, and deploying a firmware that executes on an embedded system. To summarize, TinyML is used as it enables the deployment of the model on devices that reside on edge to perform classification on edge without the need to send the data to the cloud for processing. Thus, reducing latency and bandwidth.

2.4 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a type of deep neural network that has one or more convolutional layers (i.e., layers that perform convolution operations) [11]. Convolution is a linear procedure that involves sliding a parametric-sized filter across the input representation (usually a visual image). A feature map is generated by applying the same filter to different overlapping filter-sized parts of the input. Filters, also known as operators, come in various shapes and sizes. Each filter attempts to recognize a particular characteristic inside the input representation, such as corners or edges. One of the most significant characteristics of CNNs, and the reason for their widespread acceptance, is their capacity to derive specific characteristics in every feature of the input by applying a large number of filters in parallel. Rather than using the output of all the neurons in the previous layer like fully connected perceptrons, CNNs use a hierarchical model that enables them to construct complicated features utilizing tiny and basic patterns. One of the most important considerations to make when creating a CNN, or more broadly a neural network, is how to encode the input data. Several input representations are accessible in the literature, each with benefits and downsides. Our model is developed using multiple one-dimensional (1D) CNN layers used for feature extraction from the acquired signal. The filters of each 1D layer move in one direction to calculate the output. This allows the extraction of essential features that enables the model to analyze the signal and detect the type of jammer with higher accuracy.

3 SCENARIO AND THREAT MODEL

The adversarial scenario presented in this work is shown in Figure 1. We assume that a powerful jammer is deployed in a WSN to

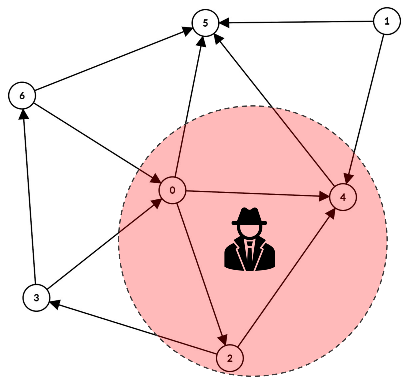

Figure 1: The scenario assumed in this work. An adversary carries a jamming attack in a dense IoT wireless network to block the communications over the wireless channel. The surrounding nodes are able to detect the jamming and perform channel measurements to evaluate the data with a pre-trained Tensorflow lite model to determine the jamming type.

disrupt all types of communications on a particular frequency (e.g., 2.4GHz), through broadcasting AWGN. In addition, we assume that the adversary deployed the jammer in a hidden place that cannot be easily identified. Further, the jammer is of an unknown type. We assume that this signal mainly blocks all communications between the nodes and, hence, disables any communication between the surrounding nodes located within the jamming radius. Furthermore, we emphasize that the jamming effects do not only affect the communications between the nodes; in fact, we consider that the strength of the signal is extremely powerful that it makes it impossible for the nodes to establish communications in the network. We aim to deploy a Raspberry Pi with an SDR attached that can effectively detect and identify the jamming type. This will facilitate the sensing of the WiFi channel and acquire RSS readings; then, these readings will be evaluated against the onboard TinyML model to identify the type of jamming carried within the WSN.

4 SYSTEM ARCHITECTURE

In this section, the proposed system architecture is discussed in detail. It encompasses two major components: hardware and software implementation.

4.1 Hardware Setup

HackRF SDR: HackRF One is an SDR peripheral that can transmit and receive radio signals ranging from 1 MHz to 6 GHz. HackRF One is an open-source hardware platform that may function as a USB peripheral or stand-alone device. The main reason for using an SDR rather than using the existing WiFi module onboard the edge device is due to the sensitivity of the transceiver. The SDR has a higher sensitivity when compared to the regular WiFi chip and can process up to 20 million samples per second. Unlike regular edge devices' WiFi chips, SDRs support a wider range of frequencies. Initially, two HackRFs' were used to instantiate the setup. One HackRF is connected to a laptop running Ubuntu 21.10 through

Table 1: Constant and Periodic jammer parameters.

Parameter	Value
Frequency	2.412 GHz
Sampling Rate	32K
Radio Frequency Gain	40 dB
Intermediate Frequency	40 dB
Baseband Gain	40 dB
Bandwidth	40 MHz

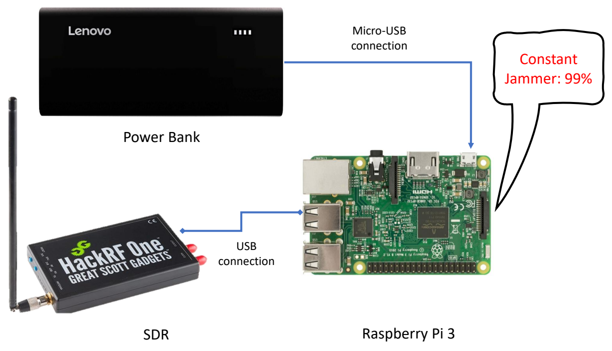

Figure 2: Jamming detection module experimental setup. This setup constitutes a Raspberry Pi connected to a HackRF and powered by a power bank.

USB2.0. This laptop runs GNU Radio, which is used to program the HackRF with the different types of jamming attacks investigated. The second HackRF is connected to the Raspberry Pi to acquire the RSS readings and feed them to the TinyML model. Both devices are placed 1 meter apart. The jammer is set to transmit Adaptive White Gaussian Noise with an amplitude of 4 on the 2.412 GHz. Table 1 lists all the jammer parameters. Finally, the configurations are the same for each type of jamming attack. Figure 3 illustrates the hardware setup.

Raspberry Pi 3 B+: The Raspberry Pi is an open-source, Linux-based, low-cost, small-sized computer board. It can be used for many applications. It has wireless LAN and Bluetooth connectivity, ideal for powerful connected designs. The Raspberry Pi runs a "PiSDR" image pre-loaded with multiple SDR software. Figure 2 illustrates the Raspberry Pi, which is connected to a HackRF to facilitate sensing and reading the channel.

4.2 Software Setup

As previously discussed, the jammer is a laptop that is connected to a HackRF that is programmed to continuously broadcast AWGN to emulate the constant jammer. As for the periodic jammer, the HackRF is programmed to transmit AWGN for a specific amount of time, after which the jammer is put to sleep for a period of time,

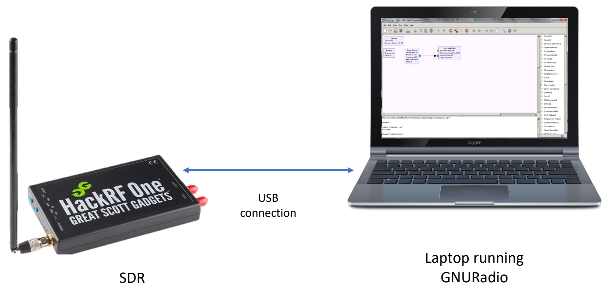

Figure 3: Jamming source (adversary) experimental setup. The adversarial setup consists of a portable laptop that runs GNU Radio and is connected to a HackRF.

with the cycle repeating indefinitely. We set the jamming period to 2 seconds and the sleeping time to 3 seconds for this experiment.

The receiver side consists of a Raspberry Pi connected to a HackRF and has the TensorFlow lite classification model. The HackRF is programmed to sense the channel and record the RSS. Despite applying different techniques in GNU Radio (i.e., implementing a custom block or modifying the C code of some of the existing blocks), it is worth mentioning that it was extremely challenging to be able to extract the RSS value from the sink due to the lack of documentation of the included blocks. However, after multiple attempts, we utilized the "Log Power FFT "block to perform the conversion operation. Further, we made the code for this available [2].

4.3 Data Collection

Despite obtaining existing datasets [4, 16], it was convenient to generate our data as it was challenging to find a dataset that fulfills our requirements in addition to being collected from an actual jamming setup, not simulation-based. Our solution considers using the RSS as the main metric for training and classification, and this was hardly available in the existing datasets. We recorded the data for each jamming type for 30 minutes based on the software setup. The considered setup is made of two modules, Raspberry Pi and the jamming source (Figure 2, 3). Both modules are deployed 1 meter apart. The jamming module injects the channel with high power AWGN with the settings previously mentioned (recall Table 1). A shell script is coded to facilitate emulating periodic jammer, and the corresponding code is released with the dataset. This resulted in having 1230 signals for each type of jamming and the normal channel. Figure 5 illustrates the RSS and the shape of the signal associated with the collected data for each type.

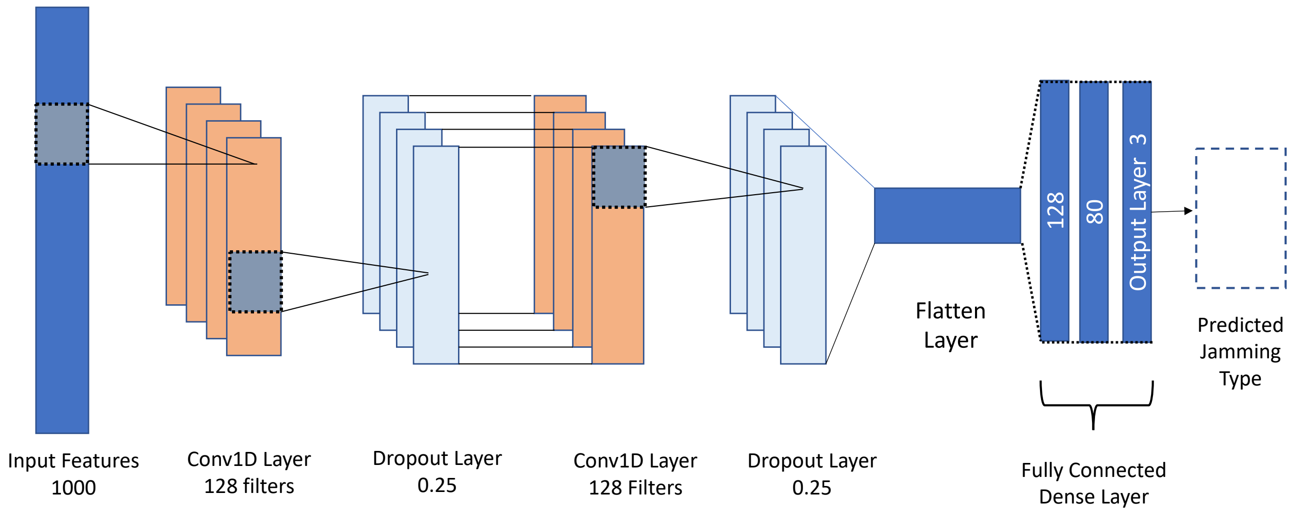

Figure 4: Structure and details of the implemented Convolutional Neural Network. The structure consists of two one-dimensional convolutional layers with a dropout layer. The output is then flattened with a flattened layer, followed by a fully connected dense layer and a softmax layer.

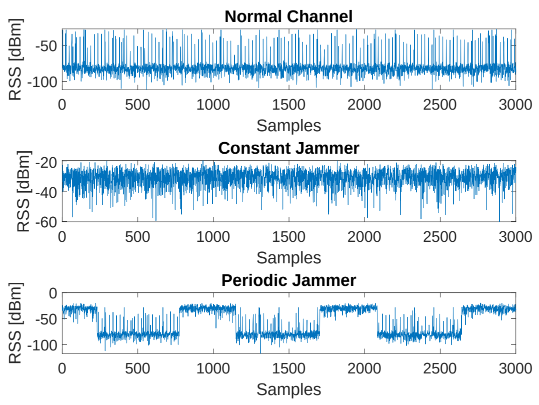

Figure 5: Acquired RSS readings associated with the different types of jamming (constant and periodic) and the normal channel.

5 TINYML-BASED JAMMING DETECTION METHODOLOGY

This section discusses the Deep Learning architecture implementation and the TinyML implementation and deployment.

5.1 Deep Learning Architecture and Procedures

To classify the different types of jamming attacks, a multi-layer 1D CNN model is built and trained. Table 2 summarizes the training options of our model, the details of which are discussed in the text below.

Data Preprocessing: From the 410 signals per jamming type, the first 310 signals are considered for the training set, while the remaining 100 signals are part of the test set. Each signal consists of 1000 samples. The signals for each jamming type in the training and testing were then concatenated, resulting in a training set and test set of size 930 and 300, respectively. The training and test data sets were then assigned labels. One hot encoding was applied to the labels to ensure that the model does not presume that higher labels have higher importance than others.

Data Validation: The training set is separated into training and validation sets at a ratio of 70/30. The validation set estimates the model's performance after each epoch during the tuning of the hyperparameters.

Optimizer: The optimizer works to reduce the loss function by tuning the parameters of the network. Here, the optimizer Adam is used, which is widely popular.

Activation Function: After each convolution, our neural network relies on the Rectified Linear Units (ReLU) activation function.

Option	Value
Optimizer	Adam
Loss Function	Categorical Cross-entropy
Learning Rate	0.001 (Default)
Epochs	60
Shuffle	Every Epoch
Validation Data	Random 30% of the data

Table 2: Training Parameters of the network.

Loss Function: The loss function used is the *Categorical cross-entropy* which is a loss function that is employed for multi-class classification, where the average difference between the predicted probability distribution and the actual probability distribution is calculated and minimized.

Network Architecture: The overall architecture of the CNN model is shown in Figure 4, which consists of two 1D convolutional layers with ReLu activation, then three fully connected layers that feed into a 3-way softmax layer.

Output: The softmax function is used as the output layer, which scales its inputs into a probability distribution between 0 and 1 and adds up to 1, which can be used as probabilities.

5.2 TinyML Implementation and Deployment

In this work, the proposed implementation relies on deploying the developed CNN model using Tensorflow on IoT edge devices. Deploying regular CNN models on small devices is very challenging due to (i) the size of the model, and (ii) when decreasing the size of the model, the accuracy is dramatically reduced. To overcome these challenges when deploying the model on edge devices, we convert it using Tensorflow lite. This will result in a model of size 62.7 MBs that falls within the memory budget of our Edge device. Several optimization methods can be applied to reduce the model's size, such as quantization, pruning, and clustering, intended to reduce the model size and latency with minimal or no loss in accuracy. These optimization techniques are beneficial when deploying the model on devices with very constrained resources. In our scenario, the Raspberry Pi provides enough storage to host models of bigger sizes. To deploy the trained model on the Raspberry Pi, we convert the model using Tensorflow lite. The conversion process is as follows:

(1) Using the TFlite converter, we convert the saved model to a ".tflite" model.
(2) The converted model is then deployed on the Raspberry Pi.
(3) On the Raspberry Pi, the interpreter loads the model, allocates memory for the input tensors, then invokes the model to perform inference.

6 PERFORMANCE EVALUATION

In this section, we discuss the performance of the deep learning model in terms of training, validation, and testing. Besides, we evaluate the performance of the deployed TinyML model by detecting and classifying jamming in the wild.

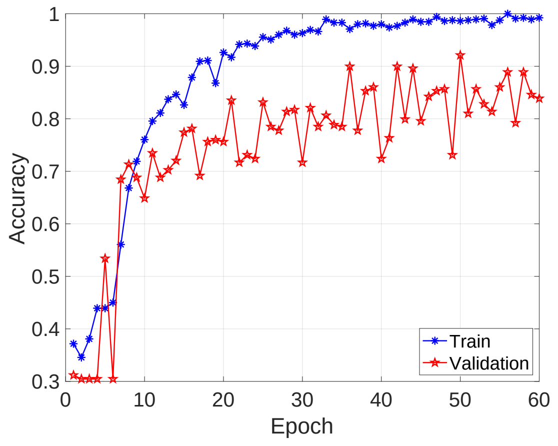

Figure 6: Accuracy as a function of the epochs.

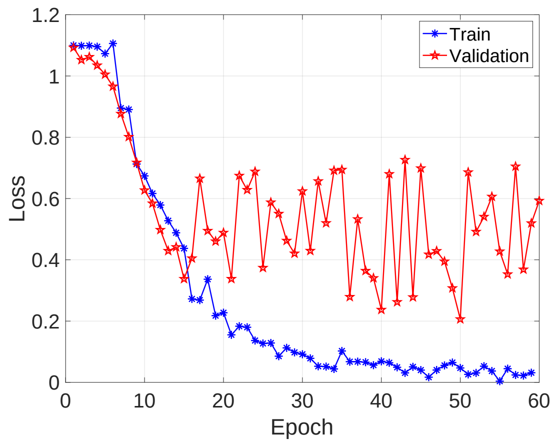

Figure 7: Loss as a function of the epochs.

6.1 Model Training

Training the model for several epochs with a large number of labeled training data, the model's internal parameters are updated to minimize the loss (error) and maximize the accuracy. Additionally, to prevent the over-fitting of the model, dropout was applied. The accuracy and loss for each epoch can be seen in Figure 6 and Figure 7.

Figure 6 shows that the training accuracy increases with increasing the number of epochs; this indicates that the model is capable of further learning when increasing the number of training epochs. The training accuracy at the last epoch was 99.39%, while the validation accuracy was around 91.76%. Additionally, in Figure 7, the training and validation loss curves of the model are illustrated. We can see that the loss does not decrease and shows that the model over-fits. To further verify the model's accuracy, the model performed inferences using the test data, obtaining a testing accuracy

Ref.	Network Type	Purpose	Type of Jammer	Data Source	Method	Metrics	Detection in the Wild	Edge-based Solution	Accuracy [%]
[7]	Wireless Network	Classification	Constant, Random, and Reactive	Simulation	KNN, Decision Trees, and Random Forests	RSS, PDR, and Noise	✗	✗	79, 75, and 81
[19]	Wireless IoT Network	Detection	Random	Simulation and Real Dataset	SVM, Decision Trees, and Random Forests	RSS	✗	✗	98 and 89.7
[10]	Wireless Network	Classification	Constant, Random, and Reactive	Simulation	KNN, Decision Trees, Random Forests, and Gradient Boost	RSS and PDR	✗	✗	67.6, 73.03, 88.86, and 94.03
[9]	WSN	Detection	Random	Simulation	Deep Neural Network	RSS, PDR, etc	✗	✗	–
[5]	Wireless Network	Detection	Constant	Dataset	Random Forest, cubic SVM, and NN	RSS, PDR	✗	✗	97.5, 96.4, and 97.1
This work	**Wireless Network (802.11)**	**Detection and Classification**	**Constant and Periodic**	**Real-time Acquired Dataset**	**1D CNN, TinyML**	**RSS**	✓	✓	**86.3**

Table 3: The proposed solution in comparison with existing solutions.

of 86.33%. Figure 8 illustrates the confusion matrix for each class. Indeed, the prediction accuracy for the constant jammer is 100% due to the nature of the constant RSS. On the one hand, the model does not accurately classify the periodic jammer (72.0%). This drop inaccuracy is due to several mismatching samples with the behavior of the normal channel. On the other hand, classifying the normal channel resulted in 13 instances that were mismatched with the periodic jammer, resulting in an accuracy of 87.0%. Table 4 provides more insights and evaluation metrics for the model.

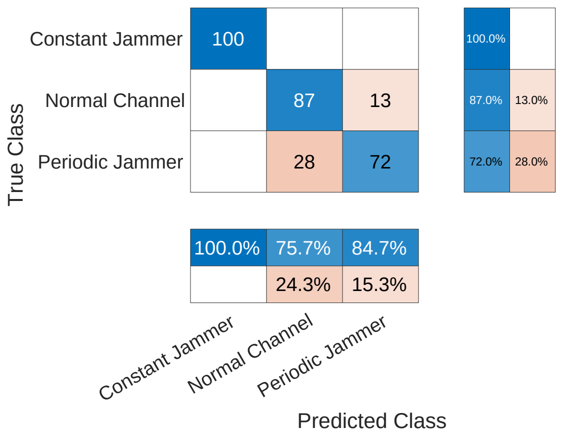

Figure 8: Confusion Matrix of the CNN Model associated with the classification of the data

6.2 Detection in the Wild

To evaluate the performance of the TinyML model, we deploy the model on the Raspberry Pi and perform several experiments to test the detection accuracy. On startup, a Python script is executed on the Raspberry Pi. This script runs the GNU Radio code for acquiring the samples from the spectrum and computing the RSS. Once 1000 samples are acquired, the data is reshaped and fed to the TinyML model. The model evaluates the data and classifies the jamming type accordingly. This operation is performed continuously and indefinitely. The average accuracy achieved is 86% for 10 experiments. This shows that the trained model can effectively detect the type of jamming deployed with a confidence rate above 80%.

Type	Precision	Recall	F1 Score	Support
Normal	0.80	0.70	0.75	100
Constant Jammer	1.00	1.00	1.00	100
Periodic Jammer	0.73	0.83	0.78	100
Accuracy			0.84	300
Macro avg	0.85	0.84	0.84	300
Weighted avg	0.85	0.84	0.84	300

Table 4: Model Evaluation Metrics

6.3 Comparison with Existing Solutions

Table 3[2] illustrates the existing solutions in terms of several parameters against the solution proposed in this paper. Despite the low accuracy when compared to the existing solutions, we highlight several points that we consider crucial:

(1) Existing solutions rely mainly on data acquired during simulation to perform the classification. In contrast, the data used to train our model is obtained from an actual jamming source, i.e., an actual setup where the jamming attack is carried out.
(2) The majority of the contributions perform detection of jamming attacks, while our solution performs detection and classification of different types of jamming attacks.

[2]In [7], noise is considered as the power measured on the channel during the idle time of the transmitting end.

(3) Further, one major benefit of our solution is portability. Portability enables the deployment of the IoT-edge device anywhere for detection. It can even be connected to a 5G modem to communicate with a base station to report the classification results. None of the listed solutions consider an IoT-edge device that performs detection while being portable.

6.4 TinyML Model Optimization

To optimize our TinyML model, we resorted to using two techniques. Namely, post-training quantization and pruning. When quantization is applied, the model's size is indeed reduced by 73.88%, corresponding to 15.7MB. This, however, resulted in a very low inference accuracy of 33.3%. This means that the model is unable to predict the type of jamming. Additionally, after pruning the model, there was no reasonable reduction in the size. For quantization, this drop in accuracy could result from the model overfitting.

6.5 Limitations

The first limitation is related to the deep learning model. The designed and implemented model is over-fitting. When training the model, it over-fits after 18 epochs. This can be resolved by collecting more data to train the model with. The second limitation is associated with the model mismatching between the periodic jammer and the normal channel status. This can be resolved by using advanced signal processing and slicing techniques and preprocessing the data in such a way as to reduce the noise.

7 CONCLUSION AND FUTURE WORK

In this paper, we presented an IoT TinyML-based jamming detection and classification system implementation that improves on the existing solutions from several aspects, in particular, portability. We emulated two types of jamming attacks, namely, constant and periodic, using commonly available off-the-shelf equipment such as HackRF. The proposed implementation includes a HackRF connected to an IoT device, Raspberry Pi, which senses the channel and records RSS readings. The Raspberry Pi has a trained deep learning model capable of detecting and classifying the type of jamming attacks carried in the network by analyzing the recorded RSS data fed to the model. As reported by our performance evaluation, we demonstrated that a jamming type could be detected with an accuracy of 86.3%. The portable nature of the proposed system enables the deployment in various scenarios. For example, jamming attacks could be carried out in VANETs or a swarm of drones. The proposed implementation could be easily deployed to help identify jamming attacks in such mobile situations. For future work, we consider conducting more experiments with more devices, different jamming settings including various amplitudes, and frequencies. As well as at different distances. Additionally, future research activities in this direction intend to profoundly investigate the novelty of the proposed approach and evaluate the performance achieved herein. We believe that this contribution paves the direction for further research directions in both academia and the industrial sector.

ACKNOWLEDGMENTS

This work was made possible by Qatar National Research Fund (a member of Qatar Foundation), NPRP grant # NPRP12S-0119-190006. The work of Amr Mohamed was partially supported by NPRP grant # NPRP13S-0205-200265. The findings achieved herein are solely the responsibility of the authors.

REFERENCES

[1] Nalam Venkata Abhishek and Mohan Gurusamy. 2021. JaDe: Low Power Jamming Detection Using Machine Learning in Vehicular Networks. *IEEE Wireless Communications Letters* 10, 10 (2021), 2210–2214.

[2] Ahmed Hussain, Nada Abughanam, Junaid Qadir, and Amr Mohamed. 2022. Jamming Detection in IoT Wireless Networks using TinyML — Dataset. Retrieved October 2022 from https://github.com/AMHD/Jamming-Detection-in-IoT-Wireless-Networks-An-Edge-AI-Based-Approach

[3] Mohammed Saeed Al-Kahtani. 2012. Survey on security attacks in vehicular ad hoc networks (VANETs). In *2012 6th international conference on signal processing and communication systems*. IEEE, 1–9.

[4] Iman Almomani, Bassam Al-Kasasbeh, and Mousa Al-Akhras. 2016. WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks. *Journal of Sensors* 2016 (2016).

[5] Youness Arjoune, Fatima Salahdine, Md Shoriful Islam, Elias Ghribi, and Naima Kaabouch. 2020. A novel jamming attacks detection approach based on machine learning for wireless communication. In *2020 International Conference on Information Networking (ICOIN)*. IEEE, 459–464.

[6] Ericsson. n.d. Ushering In A Better Connected Future. Retrieved April 2022 from https://www.ericsson.com/en/about-us/company-facts/ericsson-worldwide/india/authored-articles/ushering-in-a-better-connected-future

[7] Zhutian Feng and Cunqing Hua. 2018. 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)*. IEEE, 1–6.

[8] Kanika Grover, Alvin Lim, and Qing Yang. 2014. Jamming and anti–jamming techniques in wireless networks: a survey. *International Journal of Ad Hoc and Ubiquitous Computing* 17, 4 (2014), 197–215.

[9] E Jayabalan and R Pugazendi. 2021. Deep learning model-based detection of jamming attacks in low-power and lossy wireless networks. *Soft Computing* (2021), 1–22.

[10] GS Kasturi, Ansh Jain, and Jagdeep Singh. 2020. Detection and Classification of Radio Frequency Jamming Attacks using Machine learning. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.* 11, 4 (2020), 49–62.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (Lake Tahoe, Nevada) *(NIPS'12)*. Curran Associates Inc., USA, 1097–1105.

[12] Khaled B Letaief, Yuanming Shi, Jianmin Lu, and Jianhua Lu. 2021. Edge Artificial Intelligence for 6G: Vision, Enabling Technologies, and Applications. *IEEE Journal on Selected Areas in Communications* 40, 1 (2021), 5–36.

[13] Aristides Mpitziopoulos, Damianos Gavalas, Charalampos Konstantopoulos, and Grammati Pantziou. 2009. A survey on jamming attacks and countermeasures in WSNs. *IEEE Communications Surveys & Tutorials* 11, 4 (2009), 42–56.

[14] Hossein Pirayesh and Huacheng Zeng. 2022. Jamming attacks and anti-jamming strategies in wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials* (2022).

[15] M Premkumar and TVP Sundararajan. 2020. DLDM: Deep learning-based defense mechanism for denial of service attacks in wireless sensor networks. *Microprocessors and Microsystems* 79 (2020), 103278.

[16] Oscar Puñal, Carlos Pereira, Ana Aguiar, and James Gross. 2014. CRAWDAD dataset uportorwthaachen/vanetjamming2014 (v. 2014-05-12). Downloaded from https://crawdad.org/uportorwthaachen/vanetjamming2014/20140512. https://doi.org/10.15783/C7Q306

[17] Partha Pratim Ray. 2021. A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University-Computer and Information Sciences* (2021).

[18] Verified Market Research. 2021. Global Anti-Jamming Market Size By Receiver Type (Commercial Transportation Grade and Military & Government Grade), By Anti-Jamming Technique (Beam Steering Technique, Civilian Techniques, and Nulling Technique), By Application (Flight Control, Surveillance & Reconnaissance, Position, Navigation & Timing, Targeting), By Geographic Scope And Forecast. Retrieved Aug 18, 2022 from https://www.verifiedmarketresearch.com/product/global-anti-jamming-market-size-status-and-forecast-to-2026/#:~:text=Anti%2DJamming%20Market%20Size%20And,8.29%20%25%20from%202021%20to%202028

[19] Bikalpa Upadhyaya, Sumei Sun, and Biplab Sikdar. 2019. Machine learning-based jamming detection in wireless IoT networks. In *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*. IEEE, 1–5.

[20] Satish Vadlamani, Burak Eksioglu, Hugh Medal, and Apurba Nandi. 2016. Jamming attacks on wireless networks: A taxonomic survey. *International Journal of Production Economics* 172 (2016), 76–94.

[21] Pete Warden and Daniel Situnayake. 2019. *TinyML: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers.* O'Reilly Media.