JAMMING DETECTION AND CLASSIFICATION VIA CONVENTIONAL MACHINE LEARNING AND DEEP LEARNING WITH APPLICATIONS TO UAVS

by

Yuchen Li

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Electrical and Computer Engineering

Department of Electrical and Computer Engineering

Hammond, Indiana

December 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

ACKNOWLEDGMENTS

Firstly, I would like to express my gratitude to my advisors and committee members Colin Elkin, Khair Al Shamaileh, Quamar Niyaz, and Sidike Paheding for their helpful instructions and guidance during the period of my Master's thesis research at Purdue University Northwest.

Meanwhile, I am truly grateful to Dr. Colin Elkin for funding me as a research assistant in the Department of Electrical and Computer Engineering. I am also grateful to Dr. Vijay Devabhaktuni for his insightful comments. In addition I would like to thanks Dr. Chenn Zhou for facilitating my admission as an exchange student.

Secondly, I would like to express my gratitude to Jered Pawlak, Joshua Allen Price, and Matthew Allen Wright, who had helped me in the early stages of the project.

Thirdly, I sincerely appreciate my parents for their financial support and encouragement while studying. I would like to thank my girlfriend for her help and encouragement.

Lastly, thanks to all professors and staff who helped me in the whole graduate career.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

With the constant advancement of modern radio technology, the safety of radio communication has become a growing concern for us. Communication has become an essential component, particularly in the application of modern technology such as unmanned aerial vehicle (UAV). As a result, it is critical to ensure that a drone can fly safely and reliably while completing duties. Simultaneously, machine learning (ML) is rapidly developing in the twenty-first century. For example, ML is currently being used in social media and digital marking for predicting and addressing users' varies interests. This also serves as the impetus for this thesis. The goal of this thesis is to combine ML and radio communication to identify and classify UAV interference with high accuracy.

In this work, a ML approach is explored for detecting and classifying jamming attacks against orthogonal frequency division multiplexing (OFDM) receivers, with applicability to UAVs. Four types of jamming attacks, including barrage, protocol-aware, single-tone, and successive-pulse jamming, are launched and analyzed using software-defined radio (SDR). The jamming range, launch complexity, and attack severity are all considered qualitatively when evaluating each type. Then, a systematic testing procedure is established, where a SDR is placed in the vicinity of a drone to extract radiometric features before and after a jamming attack is launched. Traditional ML methods are used to create classification models with numerical features such as signal-to-noise ratio (SNR), energy threshold, and important OFDM parameters. Furthermore, deep learning method (i.e., convolutional neural networks) are used to develop classification models trained with spectrogram images filling in it. Quantitative indicators such as detection and false alarm rates are used to evaluate the performance of both methods. The spectrogram-based model correctly classifies jamming with a precision of 99.79% and a false-alarm rate of 0.03%, compared to 92.20% and 1.35% for the feature-based counterpart.

1. INTRODUCTION

Unmanned aerial vehicle (UAV) has become an indispensable tool in modern society. It was initially applied in the military field, and then widely used in civil and scientific applications. Therefore, it is important to provide a safe and reliable environment for UAVs. The main contribution of this thesis is to investigate jamming detection and classification with applications to UAVs using realistic setup and attack seniors. By acquiring real-time modulation parameters, a traditional machine learning (ML) model can identify weather the UAV is jammed or not. Another deep learning method is also applied to improve the overall jamming detection and classification accuracy. In this method the spectrogram are used as a training dataset. A brief introduction of both UAV and ML is given in the subsequent sections.

1.1 Unmanned Aerial Vehicle's Challenges

A UAV (i.e., drone) does not have a human pilot, crew, or passengers on board. On the military side, it has been used in some extremely dangerous scenarios such as search and rescue missions to ensure the safety of military personnel. On the civil side, people use drones in a number of applications including climate monitoring, disaster management, item delivery, space research, and wildlife tracking [1]–[4].

The market of UAV is expected to expand from USD 27.4 billion in 2021 to USD 58.4 billion by 2026, according to a recent report [5]. The increasing demand for automation, as well as the rapid advancements in enabling technology, are primarily responsible for this predicted rise. Several initiatives have been launched to promote the control and navigation of UAVs [6]–[17]. However, few have addressed the associated cybersecurity challenges despite their potential in compromising UAVs performance, which may result in catastrophic consequences in some cases [18]. For instance, an attacker can construct a drone to sniff wireless signals from nearby targets, disconnect them from their legitimate networks, and assemble an army of zombie drones, as seen in one case [19]. The GPS jamming brought down 46 drones during a Hong Kong event and inflicted at least USD 127,000 in damage [20], which is another case of drone accident. Therefore, further research on the cybersecurity of UAVs that addresses the detection and mitigation of their associated cyberattacks is of grave significance. Here, jamming detection is of a particular interest and is tackled with two approaches that enable both attack detection and classification. Jamming mitigation, on the other hand, is outside the scope of this effort. Nonetheless, several techniques were reported in the literature, where the use of artificial intelligence (i.e., enforced learning), path planning and rescheduling were proposed [21]–[25].

1.2 Machine Learning

ML refers to the use of computers to understand the inherent regularity information in data in order to gain new experience and knowledge to increase the computer's intelligence and enable the computer to make judgments in the same way that people do. Specially, deep learning is a successful method of artificial neural network, which belong to ML. Some general ML algorithms are Linear Regression, Logistic Regression (LR), Naive Bayes (NB), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree (DT), and Random Forest (RF). The common ML algorithms have been widely used in finance, medical, environment, transportation and other fields, and can accurately identify and classify different events or results. Therefore, ML is adopted in this research. Here, we used a variety of traditional ML and deep learning to predict and classify jamming attacks. The former uses feature for detection and classification, while the latter uses spectrogram for the same purpose.

1.3 Literature Review

Cyber attacks against drones include data interception, data manipulation, and denial of service (i.e., jamming). Broadcast authentication and safe location verification can typically prevent data interception/manipulation threats. The former employs encryption and non-encryption techniques, whilst the latter validates the drone's location using distance bounds, group verification, Kalman filtering, multi-point placement, and traffic modeling. Although these solutions have showed promise in enhancing drone safety, the inclusion of hardware and/or software, as well as time stamp tweaks to current protocols, are the key limiting constraints that will prevent their widespread use in the near future. In addition, these approaches are ineffective in detecting interference. For example, disrupts communication between the drone and the controller, thereby posing security risks and halting information flow. An attacker may simply launch such interference utilizing off-the-shelf software-defined radio (SDR) to interfere with the drone's course, perhaps resulting in a collision. As a result, it is critical to build low-cost interference detection equipment that also fulfills existing requirements. These approaches must promote high detection rates while minimizing false alarms. Furthermore, they should allow for interference categorization in order to pick the optimal countermeasure processes and assure operational safety through sound judgments. It is worth mentioning that ML was proposed for satellite communications, vehicle Ad Hoc networks (VANETs), 5G networks, Internet of Things (IoT), and UAVs with applications including jamming detection [26]–[30], object detection, trajectory optimization, swarm communication, situational awareness, and malicious attack mitigation [31]–[33].

In this work, the impacts of four jamming types on UAV security were analyzed qualitatively (i.e., range, complexity, severity) and quantitatively with conventional ML algorithms. These algorithms were exploited for jamming detection/classification based on extracted signal features. Meanwhile, deep learning models, i.e., four configurations of convolutional neural networks (CNNs), are adopted based on spectrogram images. The spectrogram-based approach improved the classification accuracy from 92.2% (i.e., feature-based approach) to 99.79% and reduced the false-alarm rate from 1.35% to 0.03% as will be presented in greater detail in Chapter 3 and 4. Finally, this work contributes an additional dataset (i.e., spectrogram images) for training and testing ML classifiers. This dataset and the spectrogram-based approach proposed herein, were not provided nor explored in [34]. Also, this work differs from other existing techniques in the following aspects:

1. In contrast to imposing modifications to the existing protocols [35]–[42], readily available radiometric features and spectrogram images are used to develop ML models for detecting and classifying jamming.

2. In comparison to the simulation-based attack scenarios [43]–[51], this work utilizes SDR for launching jamming attacks that facilitate detection and classification with realistic environments and training datasets.

3. Here, jamming detection/classification via deep learning models is introduced. These models are trained and tested with spectrograms that characterize the jamming spectrum. This approach outperforms its feature-based counterpart in classification accuracy.

4. The datasets that are collected and used to develop the feature- and spectrogram-based classification models (i.e., features, images) are made publicly available.

1.4 Thesis Outline

This thesis is structured as follows:

- Chapter 2 introduces four different types of jamming attack and their generation methods. Then, data collection process (features/spectrograms) with and without the present of jamming are carried out.

- Chapter 3 discusses the establishment of conventional ML models (six algorithms), analyzes the characteristics of the input data (three cases), and finally summarizes the detection accuracy and false alarm rate results of the different algorithms.

- Chapter 4 presents the establishment of deep learning (four models), and finally summarizes the detection accuracy and false alarm rate results of these models.

- In Chapter 5, conclusions are made along with a discussion, reflection, and suggestions for future work.

2. JAMMING ATTACKS, EXPERIMENTAL SETUP, AND DATASET

This chapter introduces four types of interference attack scenarios, the lunching method, and experimental settings. First, the characteristics of the four types of interference, transmission and data types are introduced in detail. By adjusting the experimental distance and angle in the real environment, all possible interference situations of the UAV can be realized. Figure 2.1b, Holy Stone HS720E was used in this test, which has a small size and is convenient for this test. The UAV with a communication distance and transmission power of 1000 meters and 16 dBm respectively has the characteristics of small size and reliable performance. At the same time, it also uses IEEE 802.11 Orthogonal Frequency Division Multiplexing (OFDM) at 2.4 GHz 2.1a. The vast majority of UAV communication requirements provide a reliable basis for the universality of the experiment. The B210 SDR from National Instruments and GNU Radio, which were shown in Figure 2.1, was used to transmit the four different types of interference attacks within 40 MHz bandwidth to accommodate all subcarriers.

(a) Holy Stone HS720E

(b) USRP B210

Figure 2.1. Required hardware for the experiment.

2.1 Types of Jamming Attacks

As mentioned before, four types of jamming were applied in this experiment, which were Barrage, Single-tone, Successive-pulse, and Protocol-aware. All these jamming will be detailed and interpreted using spectrogram in the subsections. The spectrum is illustrated in Figure 2.2.

2.1.1 Barrage Jamming

In this type, noise from normal distribution is launched at the communication band to increase interference level at the receiver (i.e., UAV). Therefore, barrage is often used when the transmission frequency is unknown to the jammer. Barrage jamming is simple to launch; however, its efficiency reduces as the transmission bandwidth increases, as shown in Figure 2.2a.

2.1.2 Single-tone Jamming

Here, a high-power interference is launched to interfere with the center frequency that the target uses for data exchange, which is shown in Figure 2.2b. This interference signal is generally denoted as $J(t) = A_{\mathrm{j}} cos(2\pi f_0 t + \theta_{\mathrm{j}})$, where A_{j} is the jamming amplitude, f_0 is the center frequency, and θ_{j} is a phase shift.

2.1.3 Successive-pulse Jamming

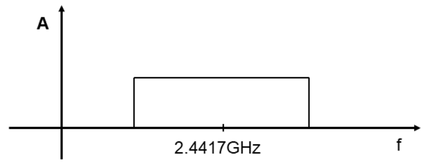
In this type, pulse-sequence is launched to interfere with the target's operation band, and is given as:

$$J(t) = A_{\mathrm{j}} \sum_{n=1}^{N_{\mathrm{j}}} \delta(t - nT), \tag{2.1}$$

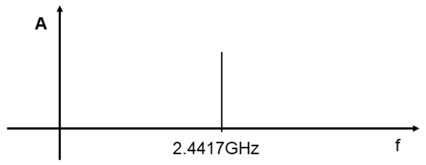
where N_{j} is the jamming tones. The period T is set such that 312.5 KHz frequency spacing is realized between generated pulses (i.e., subcarrier spacing in IEEE 802.11 OFDM). The spectrum is illustrated in Figure 2.2c .

2.1.4 Protocol-aware Jamming

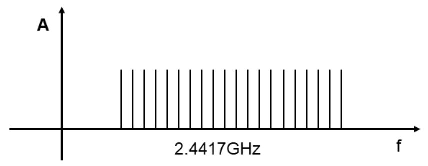
This type transmits low interference via shot-noise pulses to corrupt the ongoing transmissions while minimizing detection probability. In other words, the jammer simulates the transmitter of the targeted protocol without affecting other standards occupying the same bandwidth [52]. The spectrum is exemplified in Figure 2.2d.



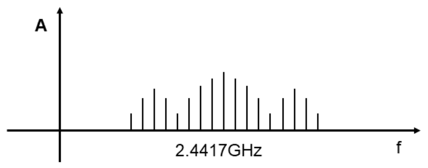
(a) Barrage Jamming



(b) Single-tone Jamming



(c) Successive-pulse Jamming



(d) Protocol-aware Jamming

Figure 2.2. Theoretical spectrum of different types of jamming.

2.2 Generating and Receiving Signals through GNU Radio

In this chapter, B210 SDRs and GNU Radio are mentioned to launch different jamming attacks and extract signal features. As described in the previous section, USRP B210 is not the only condition for transmitting interference. To use this SDR board, developers also need a modular model on the software side as a driver, which is GNU Radio. This software is normally installed in a Linux system, which is easier for developer to customize. Therefore, the following section will mention how this project generates the jamming and receives the features.

2.2.1 Jammer Side

Figures 2.3(a) shows simplified GNU Radio flow graphs for launching the attacks, respectively. Through the figure, we can clearly see that Barrage jamming is produced by superimposing the USRP Sink with Gaussian white noise, which is marked in the red dashed box. Similarly, Single-tone, Successful-pulse, and Protocol-Aware are marked in green, blue and pink dotted boxes, respectively. They are generated by signal source, vector source and random source superimposed on USRP sink respectively. All modules can be enabled and disabled. In this way, when the experiment is in progress, disabling the unnecessary modules and connecting the USRP B210 board can easily generate a specific interference. For example, in order to launch Barrage Jamming, developers can disable all modules except *Noise Source* and *USRP Sink*.

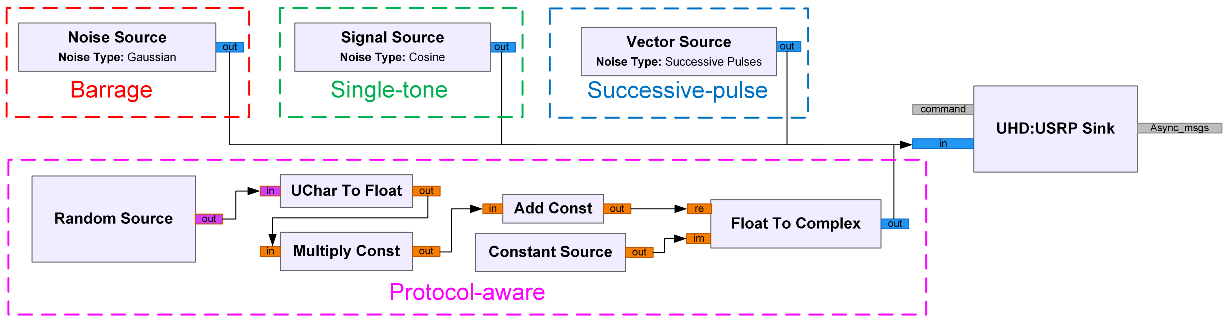


Figure 2.3. Simplified GNU Radio flow graph for launching the jamming attacks.

2.2.2 UAV Side

Figures 2.4 shows simplified GNU Radio flow graphs for extracting features. The system first collects the radio signals into GNU Radio through the USRP board, and finally obtains the parameters we need through a series of data processing. *USRP Source* is responsible for receiving radio data obtained from the air. A series of modules in the middle are responsible for processing data. Last, the *Null sink* is used for collecting data which can output as a text file. It is worthy to mention that *QT GUI Waterfall Sink* was used to collect spcetrogram images which can be used in Chapter 4. Three important blocks are *OFDM Estimator*, *Energy Detector*, and *SNR Estimator Probe*. The ① *OFDM Estimator* block shown in 2.4 is used to extract OFDM features [53], (i.e., *subcarrier length*, *cyclic prefix* (CP) *length*, *subcarrier spacing*, and *symbol time*). The ② *Energy Detector* block is used to extract the *average received power* and *threshold* [53]. Finally, three more features; namely, *signal-to-noise ratio* (SNR), *average signal power*, and *average noise power* are extracted from the ③ *SNR Estimator Probe* block.

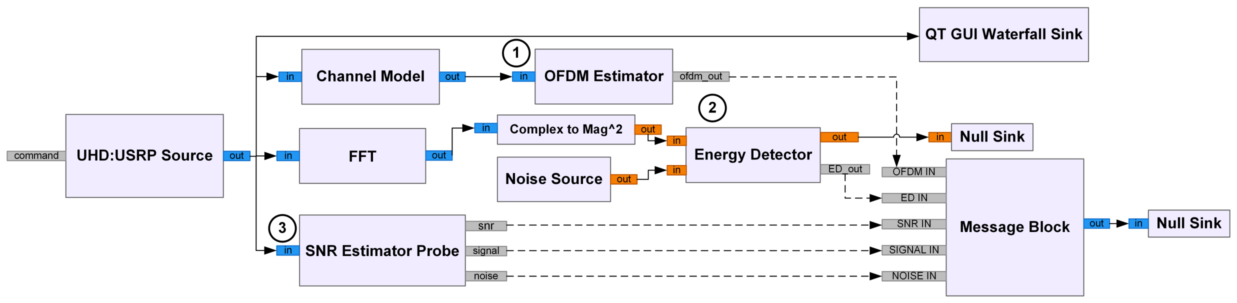


Figure 2.4. Simplified GNU Radio flow graph for extracting the radiometric features.

2.3 Experimental Setup

Two experimental environments are established to evaluate the qualitative and quantitative impacts of the jamming types. The qualitative evaluation analyzes severity, launch complexity, and effective jamming range. The quantitative evaluation entails radiometric extractions (i.e., signal features, spectrogram images) through data collection under different jamming scenarios. Data is used for training and validating ML algorithms for jamming detection and classification.

2.3.1 Qualitative Evaluation

The separation between the jammer (i.e., B210 SDR) and drone is fixed to 0.5 meter. To measure the effective jamming range, the separation between the jammer-drone pair and the transmitter is increased gradually for each jamming type in an unobstructed outdoor setup, as shown in Figure 2.5. Here, effective jamming is defined as a complete loss of signal and is reported in Table 2.1 for each type. Results indicate that barrage jamming has the most range among all types due to spreading interference over all OFDM subcarriers in comparison to interfering with the center (or selected) frequencies as in single-tone and successive-pulse jamming or transmitting shot-noise as in protocol-aware jamming.

Table 2.1. Measured range of a successful jamming attempt.

Type	**Barrage**	**Single-tone**	**Success.-pulse**	**P-aware**
Range (m)	80	145	350	155

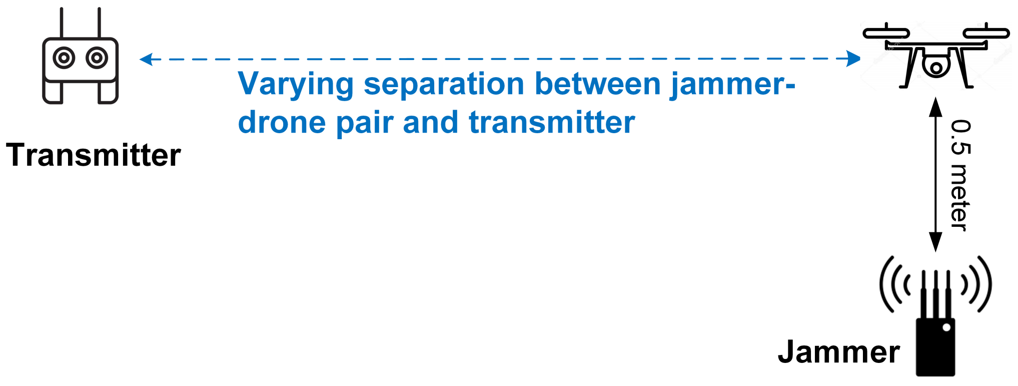

Transmitter
Varying separation between jammer-drone pair and transmitter
0.5 meter
Jammer

Figure 2.5. Experimental setup to obtain effective jamming range.

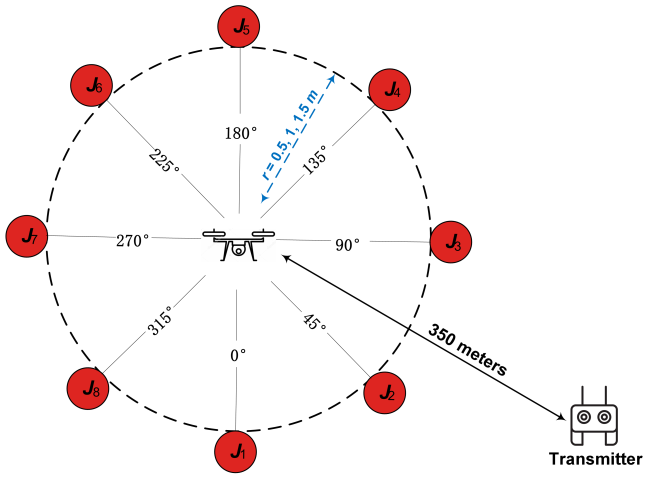
Table 2.2 depicts the qualitative findings for launch complexity and severity in a scale of 1 to 4, where 4 is the highest score. Barrage has the least launch complexity, as it does not require extensive knowledge about the communication bandwidth. Nonetheless, it has the highest severity. Single-tone jamming is relatively simple to launch. Nevertheless, this type is inefficient in scenarios in which multiple frequencies or subcarriers are used. Successive-pulse jamming with $N_{\mathrm{j}} = 64$ pulses has a moderate launch complexity as interference pulses need careful positioning with respect to the center and subcarrier frequencies. The output power, P_{j}, of the jammer is distributed on pulses in a way that the interference pulse power is $P_{\mathrm{j}}/N_{\mathrm{j}}$. Therefore, it has the least severity. Protocol-aware jamming has the highest launch complexity as it requires a thorough knowledge of the communication protocol. It also has a moderate severity since limited-power interference is launched at the transmission bandwidth to maintain low detection probability.

Table 2.2. Qualitative analysis for the four jamming types.

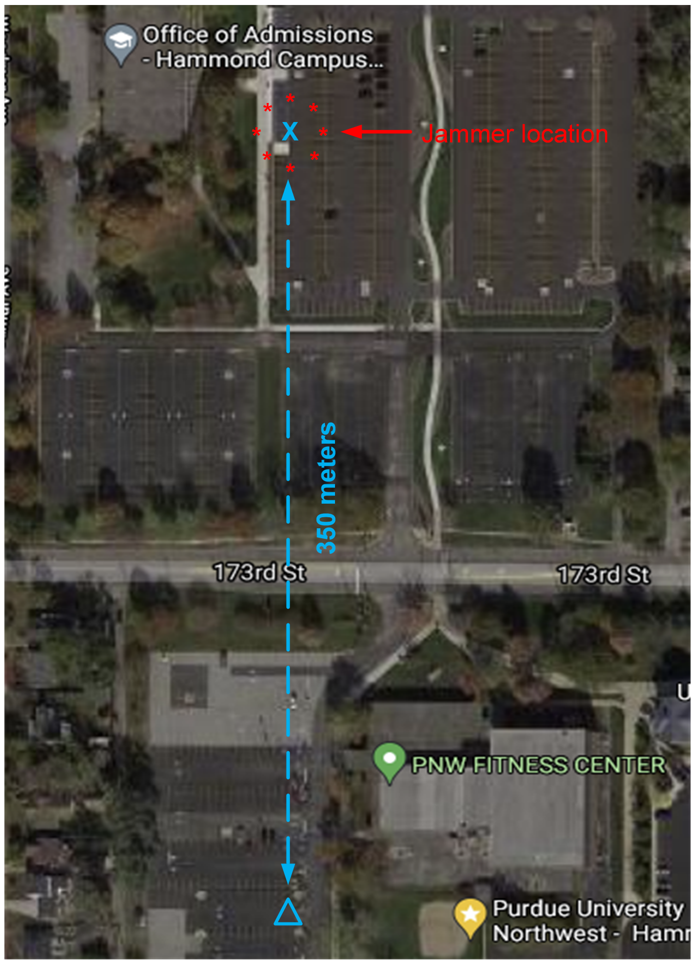
		Complexity			
		1	**2**	**3**	**4**
Severity	**1**			Success.-pulse	
	2				P-aware
	3		Single-tone		
	4	Barrage			

2.3.2 Quantitative Evaluation

Radiometric data (i.e., signal features, spectrogram images) are collected for ML training/classification. The goal here is to develop models that not only detect jamming, but also identify its type. To collect such data, the transmitter-drone separation is set to 350 meters, which is the minimum separation where all jamming types are effective. Then, without jamming presence, features and images are obtained at the drone with B210 SDR and GNU Radio modules. The same procedure is repeated in the presence of each of the jamming types, where a second SDR is utilized as jammer at eight locations $J_{\mathrm{i}}, \mathrm{i} = 1, 2, 8$, around the drone as shown in Figure 2.6. This procedure is performed for three radii r = 0.5, 1, and 1.5 meters.



(a)


Office of Admissions - Hammond Campus...
Jammer location
350 meters
173rd St
173rd St
PNW FITNESS CENTER
Purdue University Northwest - Hamm

(b)

Figure 2.6. Extraction of signal features and spectrogram images under no-jamming/jamming scenarios at different jammer locations: (a) testing setup and (b) testing location from Google maps. The $\triangle$, $*$, and $\times$ represent the trnasmitter, jammer, and drone, respectively.

2.4 Feature and Spectrogram Datasets

According to the description in the previous section, nine features are extracted as a dataset to train ML algorithms for detecting and classifying jamming attacks. They are listed below:

- *subcarrier length*
- *CP length*
- *subcarrier spacing*
- *symbol time*
- *average received power*
- *threshold*
- *signal-to-noise ratio*
- *average signal power*
- *average noise power*

The subcarrier length represents the number of subcarriers being used. The CP length is used to control symbol overlapping, and the subcarrier spacing is the frequency separation between subcarriers, which is the reciprocal of symbol time [54]. The threshold is a binary indicator that returns 1 once the *average received power* exceeds a certain level and returns 0 otherwise. It is paramount to point out that the *average received power* conveys noise energy, whereas the *average signal power* presents the estimated signal power excluding noise power. At the end of the experiment featured in Figure 2.6, a total of 23,565 signal samples are collected. These samples include 10,071 under no jamming and 13,494 in jamming presence, which are divided into 3,392, 3,367, 3,378, and 3,357 samples for barrage, single-tone, successive-pulse, and protocol-aware jamming, respectively. The complete training dataset is given in [55].

Meanwhile, spectrogram images were also collected by the block *QT GUI Waterfall Sink*, as mentioned in the previous section in Figure 2.4. Figure 2.7 depicts sample images under no jamming and others with jamming. At the end of the experiment, a total of 1578 images were collected. These images were separated to 762 clean spectrogram and 204 jamming spectrogram for each type of jamming. The complete image dataset is made available on [55].

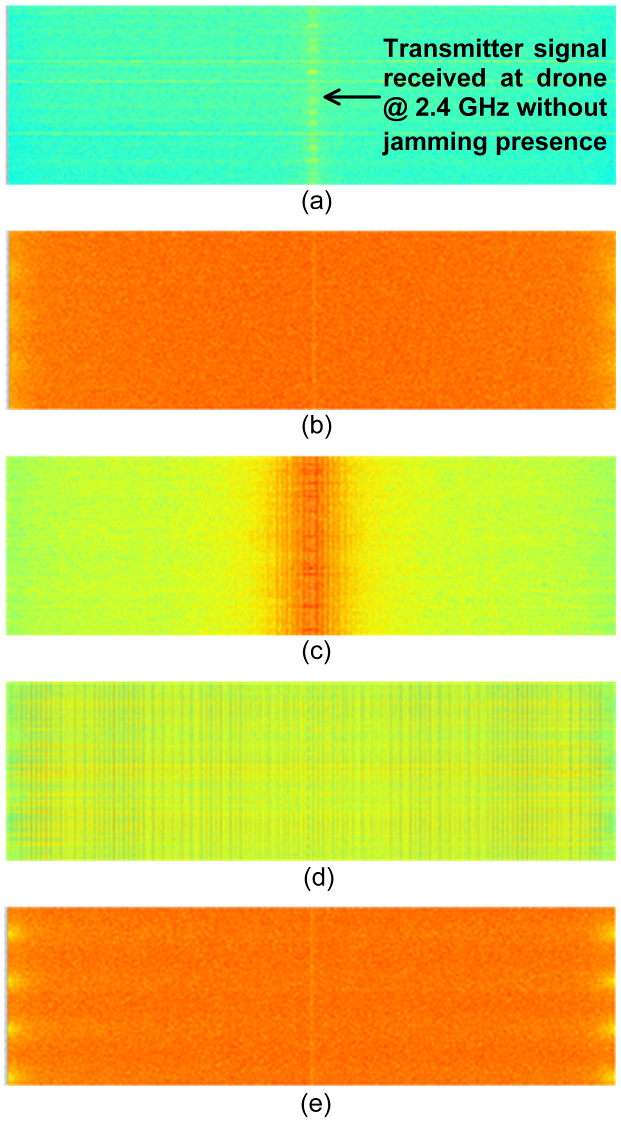


Figure 2.7. Spectrograms under (a) no jamming, (b) barrage, (c) single-tone, (d) successive-pulse, and (e) P-aware jamming.

3. FEATURE-BASED CLASSIFICATION

As discussed in Chapter 2, nine features are extracted through USRP B210 and GNU Radio to train ML algorithms for detecting and classifying jamming attacks. This chapter mainly discusses the structure four six ML model, the environment for running those model, the detailed training and testing procedures and result for conventional machine learning model.

3.1 Machine Learning Algorithms

Through the comparison and analysis of common machine learning algorithms, finally, six conventional algorithms were used to train the model, which are DT, KNN, LR, Multi-layer Perceptron (MLP), NB, and RF. They were all detailed in the following subsections.

3.1.1 Decision Tree

DT is based on the known probability of occurring of various events, and it uses a decision tree to calculate the chance that the expected value of the net present value is greater than or equal to zero, to evaluate project risk, and to decide feasibility. It is a graphical technique for using probability analysis intuitively. It is named as decision tree because this type of decision-making branch is drawn into a graph like the branches of a tree, like Figure 3.1. A decision tree is a prediction model in machine learning that depicts a mapping connection between object properties and object values. As described in the Figure 3.1, through different weather factors (outlook, humidity, or wind speed) the teacher can decide whether to continue or suspend the class.

3.1.2 K-Nearest Neighbors

The KNN classification algorithm is a theoretically mature approach that is also one of the most basic machine learning methods. The theory behind this technique is that if the majority of the k nearest (i.e., the nearest neighbors in the feature space) samples near a sample belong to a certain category, the sample should be classified to that category. As described in the Figure 3.2, circles and triangles are two different categories. To distinguish

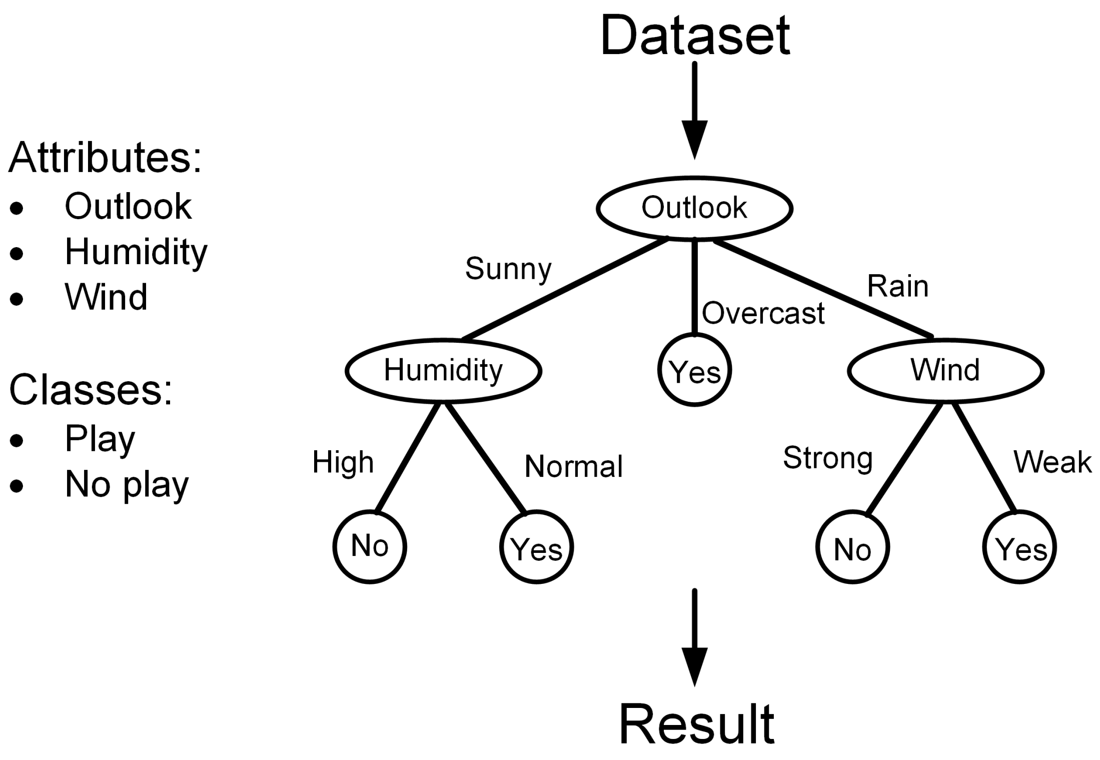


Figure 3.1. The structure of the decision tree.

the newly input element, you need to find the categories of some elements closest to it, which is K (K can be set as a positive integer). This process can predict the category of the input element with the largest number of nearest element's categories.

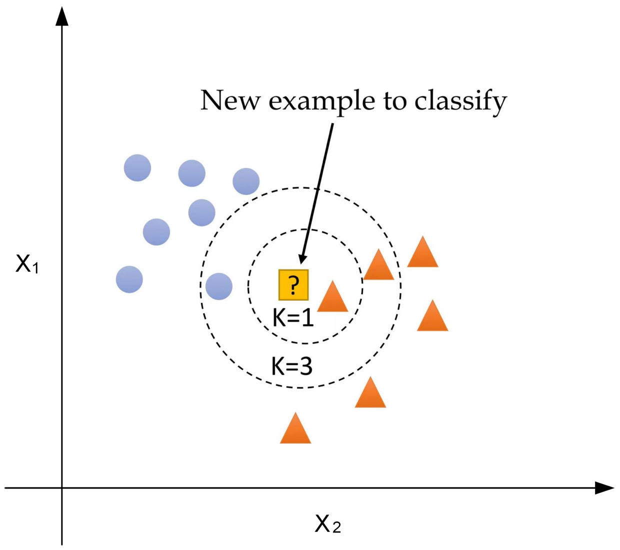


Figure 3.2. The description of the KNN.

3.1.3 Logistic Regression

LR is an algorithm that is widely used by people because it is easy to understand and very efficient without much calculation. The logistic regression algorithm does not need to scale the input features, does not require any adjustment, and outputs a well-calibrated prediction probability. Like linear regression, both of them are often used in binary classification algorithms, and they are both a kind of generalized linear model. Logistic regression assumes that the dependent variable y follows a Bernoulli distribution, while linear regression assumes that the dependent variable y follows a Gaussian distribution.

3.1.4 Multi-layer Perceptron

MLP is an artificial neural network with a forward structure that maps a set of input vectors to a set of output vectors. MLP can be regarded as a directed graph, composed of multiple node layers, and each layer is fully connected to the next layer. Except for the input node, each node is a neuron (or processing unit) with a nonlinear activation function. The structure of the MLP is shown in Figure 3.3. In this example, only one hidden layer is involved. The input has only three variables [*x1, x2, x3*] and a bias *b*, and the output layer has three neurons.

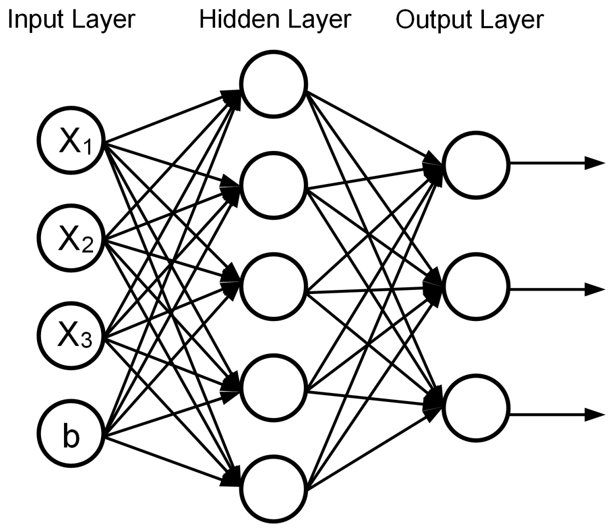


Figure 3.3. The structure of the multi-layer perceptron.

3.1.5 Naive Bayes

NB is a method based on Bayes' theorem and assumes that the feature conditions are independent of each other. The joint probability distribution of the output, based on the learned model, input X to find the output Y that maximizes the posterior probability.

Here's the Bayes's theorem:

$$p(Y \mid X) = \frac{p(X \mid Y)p(Y)}{p(X)} \tag{3.1}$$

The denominator is removed from the calculation $p(X)$ as it is a constant:

$$p(Y \mid X) = p(X \mid Y)p(Y) \tag{3.2}$$

So:

$$p(y_{\mathrm{j}} \mid x_1, x_2, ..., x_n) = p(x_1, x_2, ..., x_n \mid y_{\mathrm{j}})p(y_{\mathrm{j}}) \tag{3.3}$$

These independent conditional variables can be then multiplied together:

$$p(y_{\mathrm{j}} \mid x_1, x_2, ..., x_n) = p(x_1 \mid y_{\mathrm{j}})p(x_2 \mid y_{\mathrm{j}})p(x_3 \mid y_{\mathrm{j}})...p(x_n \mid y_{\mathrm{j}})p(y_{\mathrm{j}}) \tag{3.4}$$

Finally, the label with the maximum posterior probability can be selected as the prediction for the given instance.

$$\begin{aligned} MAP_y &= \underset{y_{\mathrm{j}} \in Y}{\operatorname{argmax}}\, p(y_{\mathrm{j}} \mid x_1, x_2, ..., x_n) \\ &= \underset{y_{\mathrm{j}} \in Y}{\operatorname{argmax}}\, p(x_1 \mid y_{\mathrm{j}})p(x_2 \mid y_{\mathrm{j}})p(x_3 \mid y_{\mathrm{j}})...p(x_n \mid y_{\mathrm{j}})p(y_{\mathrm{j}}) \\ &= \underset{y_{\mathrm{j}} \in Y}{\operatorname{argmax}} \prod_{\mathrm{i}=1}^{n} p(x_1 \mid y_{\mathrm{j}})p(y_{\mathrm{j}}) \end{aligned} \tag{3.5}$$

3.1.6 Random Forest

RF refers to a classifier that uses multiple decision trees to train and predict samples. Its advantage is that it can produce a high-precision classifier and can handle a large number of input variables. The disadvantage is that due to the larger parameters, the actual running time is longer. As Figure 3.4 shows, the dataset can be fed into different decision trees to get different results, which looks like a forest. By majority voting of or averaging N results, RF can get a more accurate final result.

3.2 Training Environment and Cases

For training and testing models, a reliable and convenient environment is essential. For traditional machine learning, a faster CPU can complete the task, so all classifiers for this experiment are executed on a 64-bit Windows 8 machine with Intel®Core™i7-6900K CPU @ 3.20 GHz processor and 128 GB memory. Install anaconda in the existing hardware environment and load the machine learning libraries, Tensorflow, with Keras API to complete the environment.

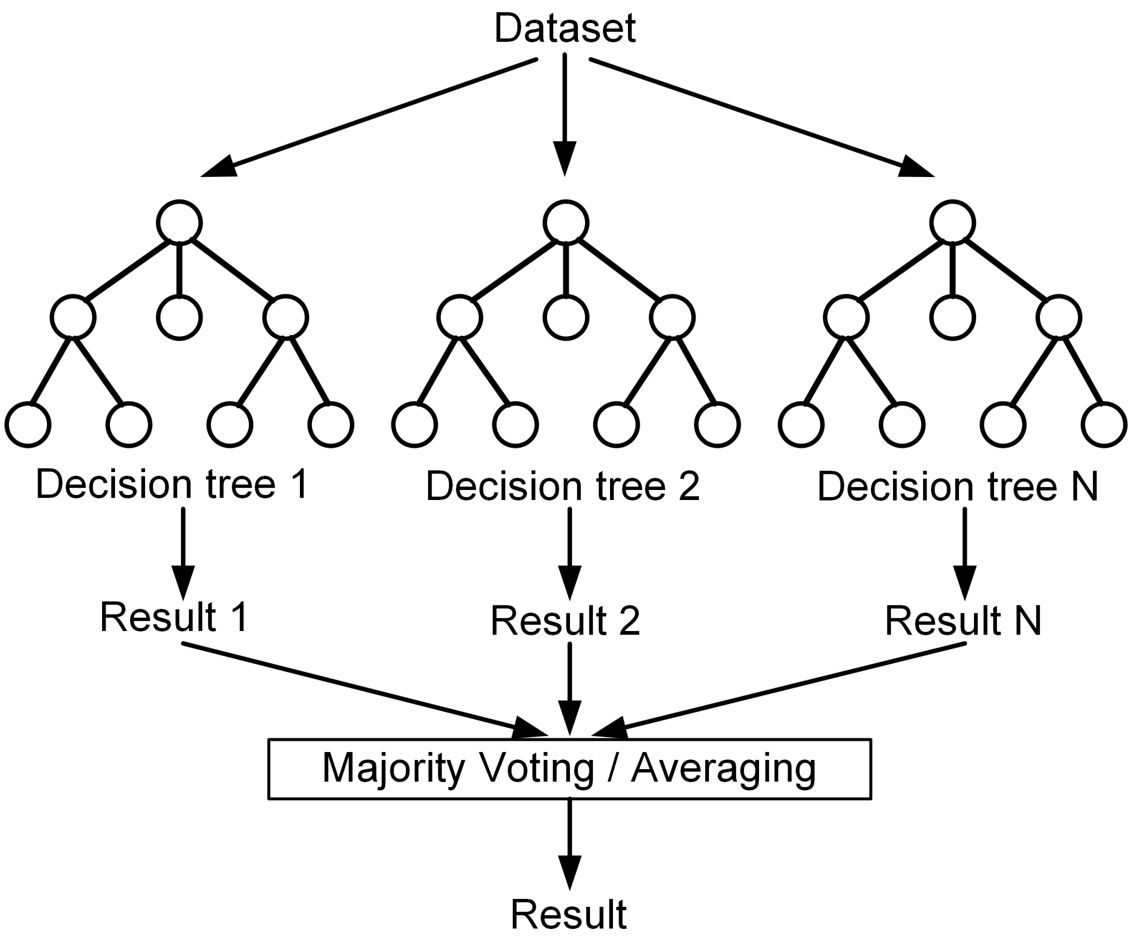


Figure 3.4. The structure of the random forest.

Also, it is found that the (*symbol time, subcarrier length*) and (*threshold, average noise power*) feature pairs are highly correlated. Thus, different ML models are explored by reducing the dimension of the features dataset. In Case 1, all the features were used for machine learning. In Case 2, *symbol time* is eliminated; whereas *symbol time* and *average noise power* are eliminated in Case 3. The list of features in each case is given in Table 3.1.

Table 3.1. List of features used in each case.

	Features		
Case	**OFDM Estimator**	**Energy Detector**	**SNR Probe**
1	Subcarrier Spacing Symbol Time Subcarrier Length CP Length	Avg Received Power Threshold	Avg Signal Power Avg Noise Power SNR
2	Subcarrier Spacing Subcarrier Length CP Length	Avg Received Power Threshold	Avg Signal Power Avg Noise Power SNR
3	Subcarrier Spacing Subcarrier Length CP Length	Avg Received Power Threshold	Avg Signal Power SNR

The During model development, the dataset is split into 70% for training and 30% for testing. Due to the uncertainty of machine learning results, we have applied each algorithm multiple times and calculated the average of various results, especially the running time. Because this experiment is an applied experiment, the more meaningful is timeliness and applicability. Therefore, we measured the training and testing time of each algorithm five times, and obtained more reliable data through repeated training.

3.3 Evaluation Methods and Results

In this section, detailed the parameters and methods for evaluating the performance of the ML model are described. At the same time, the results of this experiment are compared and discussed.

3.3.1 Evaluation Methods

This part demonstrated the performance metrics for ML model evaluation, which are detection Rate (DR), precision, recall, F-score (FS), and false-alarm rate (FAR). DR denotes the percent of correctly detected samples over total dataset samples.

$$DR = \frac{Correctly\ Predicted\ Samples}{Samples\ in\ the\ Dataset} \tag{3.6}$$

Precision is defined as the number of positive samples predicted as positive (i.e. true positive) divided by the sum of true positive and negative samples predicted as positive (i.e. false positive).

$$Precision = \frac{True\ Positive\ Samples}{True\ Positive\ + False\ Positive\ Samples} \tag{3.7}$$

Recall is the number of true positive samples divided by the sum of true positive and positive samples predicted as negative (i.e. false negative).

$$Recall = \frac{True\ Positive\ Samples}{True\ Positive\ + False\ Negative\ Samples} \tag{3.8}$$

F-score is computed from precision and recall to represent their harmonic mean.

$$F - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.9}$$

Lastly, FAR is the number of false positive samples divided by the sum of false positive and true negative samples predicted by the model.

$$FAR = \frac{False\ Positive\ Samples}{False\ Positive\ + True\ Negative\ Samples} \tag{3.10}$$

Two- and five-class ML models are created for each of the cases summarized in Table 3.1. The two-class models predict whether a jamming attack is launched or not; whereas the five-class models detect the jamming attack and identify its type (i.e., barrage, single-tone, successive-pulse, and P-aware). Also, 10-fold cross-validation is used during the training/validation stages. Once a model is trained, evaluation is performed on the test set; and the DR, F-score, and FAR are computed. Grid search is used to find the optimal hyper-parameters for each algorithm.

3.3.2 Conventional Machine Learning Results

The performance of the developed classifiers for the two- and five-class models are given in Table 3.2. All the six algorithms for two-class model classifiers achieved almost 100% DR and validation accuracy (VA), which means to classifying records into “no-jamming” or “presence of jamming”. Moreover, LR and RF perform well on all three cases, which means they are the most suitable algorithm for this experiment. On the other hand, the RF model has the highest VA of 91.80%, 92.20%, and 86.23% for Cases 1, 2, and 3, respectively, among the five-class models. Also, Random Forest achieved the highest DR and F-score in almost all cases with a DR of 92.11%, 92.20%, and 85.95% as well as an F-score of 0.92, 0.92, and 0.86 for Cases 1, 2, and 3, respectively. Finally, RF results in the highest training and testing times of 5.4s and 0.410s, respectively, in comparison to the other conventional algorithms due to the associated large number of decision trees. It is noteworthy to point out that eliminating the symbol time from the dataset (i.e., Case 2) has a marginal effect in improving classification. However, eliminating both symbol time and average noise power (i.e., Case 3) degrades the performance significantly.

Figures 3.5, 3.6, and 3.7 show the confusion matrices of the five-class RF model for each case. It is worth pointing out that none of the clean (i.e., non-jamming) records are misclassified as jamming records. Rather, mis-classification occurs only among the jamming types; particularly, barrage and protocol-aware, which is attributed to the similarity in their spectral properties (i.e., interference in these types targets the entire transmission bandwidth, but at different intensity levels).

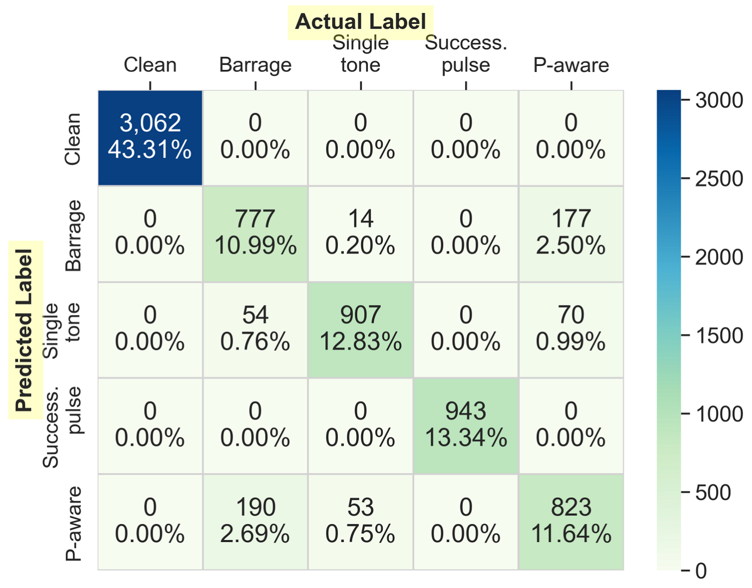


Figure 3.5. Confusion matrix of the five-class RF model for nine features.

Finally, the weighed FAR values are obtained from Figures 3.5, 3.6, and 3.7 to be 1.35% for Case 1, 1.33% for Case 2, and 2.38% for Case 3. There is no false-alarm in the two-class models regardless of the number of features used in training/validation.

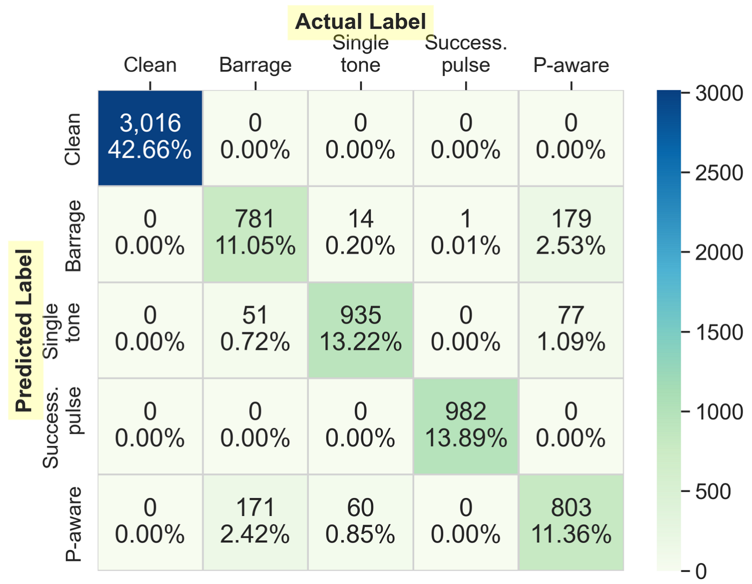


Figure 3.6. Confusion matrix of the five-class RF model for eight features.

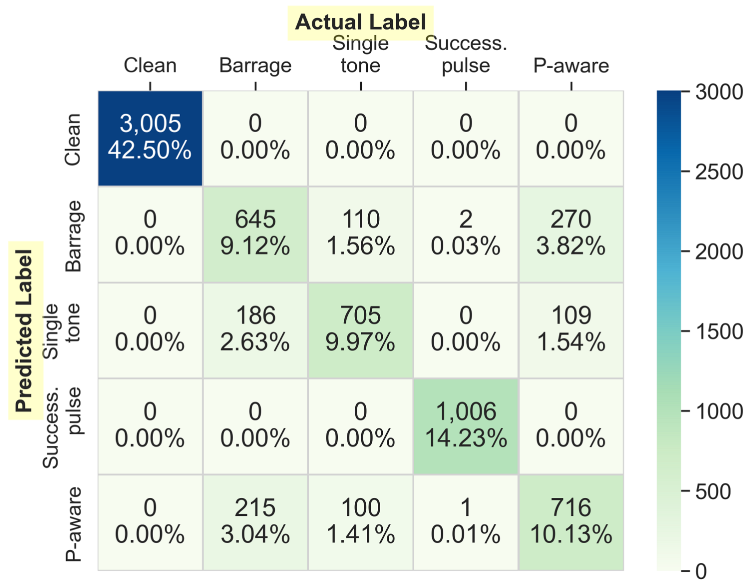


Figure 3.7. Confusion matrix of the five-class RF model for seven features.

Table 3.2. Metrics for the two- and five-class jamming detection models (VA: Validation Accuracy, DR: Detection Rate, FS: F-score, CTR: CPU Training Time, CTE: CPU Testing Time).

Performance metrics for five-class models											
	Case 1: Nine Features			**Case 2: Eight Features**			**Case 3: Seven Features**			**Time (Case 2)**	
ML Classifier	**VA (%)**	**DR (%)**	**FS**	**VA (%)**	**DR (%)**	**FS**	**VA (%)**	**DR (%)**	**FS**	**CTR(sec)**	**CTE(sec)**
LR	82.45 (± 0.65)	82.90	0.82	82.75 (± 0.67)	82.73	0.82	79.42 (± 0.76)	78.95	0.79	0.860	0.002
KNN	84.47 (± 0.74)	84.23	0.84	84.87 (± 0.74)	83.50	0.84	83.70 (± 0.72)	83.40	0.83	0.131	0.130
NB	79.30 (± 0.80)	78.74	0.79	79.40 (± 0.80)	78.33	0.78	77.50 (± 0.79)	77.80	0.77	0.002	3.550
DT	91.60 (± 0.70)	92.52	0.93	91.90 (± 0.64)	91.75	0.92	84.96 (± 0.75)	84.75	0.85	0.058	≈ 0
RF	**91.80 (± 0.06)**	**92.11**	**0.92**	**92.20 (± 0.60)**	**92.20**	**0.92**	**86.23 (± 0.79)**	**85.95**	**0.86**	**5.404**	**0.411**
MLP	78.02 (± 1.70)	79.60	0.79	77.50 (± 2.13)	76.25	0.75	77.46 (± 1.80)	75.60	0.72	1.807	0.005
Performance metrics for two-class models											
LR	**100.00 (± 0.00)**	**100.00**	**1.00**	**100.00 (± 0.00)**	**100.00**	**1.00**	**100.00 (± 0.00)**	**100.00**	**1.00**	**0.022**	**0.003**
KNN	99.92 (± 0.07)	99.89	1.00	99.93 (± 0.06)	99.94	1.00	99.93 (± 0.06)	99.96	1.00	0.135	0.135
NB	99.80 (± 0.09)	99.79	1.00	99.77 (± 0.12)	99.85	1.00	99.77 (± 0.11)	99.86	1.00	0.006	≈ 0
DT	100.00 (± 0.02)	99.98	1.00	100.00 (± 0.02)	99.98	1.00	99.98 (± 0.03)	100.00	1.00	0.009	≈ 0
RF	**100.00 (± 0.00)**	**100.00**	**1.00**	**100.00 (± 0.00)**	**100.00**	**1.00**	**100.00 (± 0.00)**	**100.00**	**1.00**	**2.344**	**0.203**
MLP	99.72 (± 0.60)	99.98	1.00	99.23 (± 2.50)	99.98	1.00	99.70 (± 0.50)	99.89	1.00	1.112	0.001

4. SPECTROGRAM-BASED CLASSIFICATION

To improve the five-class classification accuracy in Chapter 3, deep learning models trained with spectrogram images are developed. Four deep learning models, the environment for running those model, the detailed training and testing procedures, and result for deep learning models are described in this chapter.

4.1 Deep Learning Models

Deep learning models have multiple processing layers that use backpropagation to model the parameters of complex datasets (e.g., image, speech), thereby facilitating precise classification [56]. Here, CNNs are used for their leading advantage in processing images by not only efficiently extracting image properties (e.g., size, color, pattern), but also pooling a large number of pixels to reduce calculations.

The configuration of CNNs consists of input layer, convolution layer, pooling layer, fully-connected layer, and output layer. The input layer feeds images to the hidden layers. The convolution layer contains convolution kernels for extracting features, and their size gradually decreases, or remains constant, as more convolution layers are added. The pooling layer retains the highest-scoring features and discards others with low scores. It also reduces model parameters; thus, reduces computations at later layers. The fully-connected layer is similar to a regular neural network (i.e., neurons in one layer are connected to those in the next layer). The output layer returns the probability of each class. Weights are adjusted in the network via backpropagation. Spectrogram-based classification is realized with four deep learning configurations: AlexNet, VGG-16, ResNet-50, and EfficientNet-B0, which are shown in the next subsections. Figure 4.1 shows their structures and Table 4.1 details their parameters.

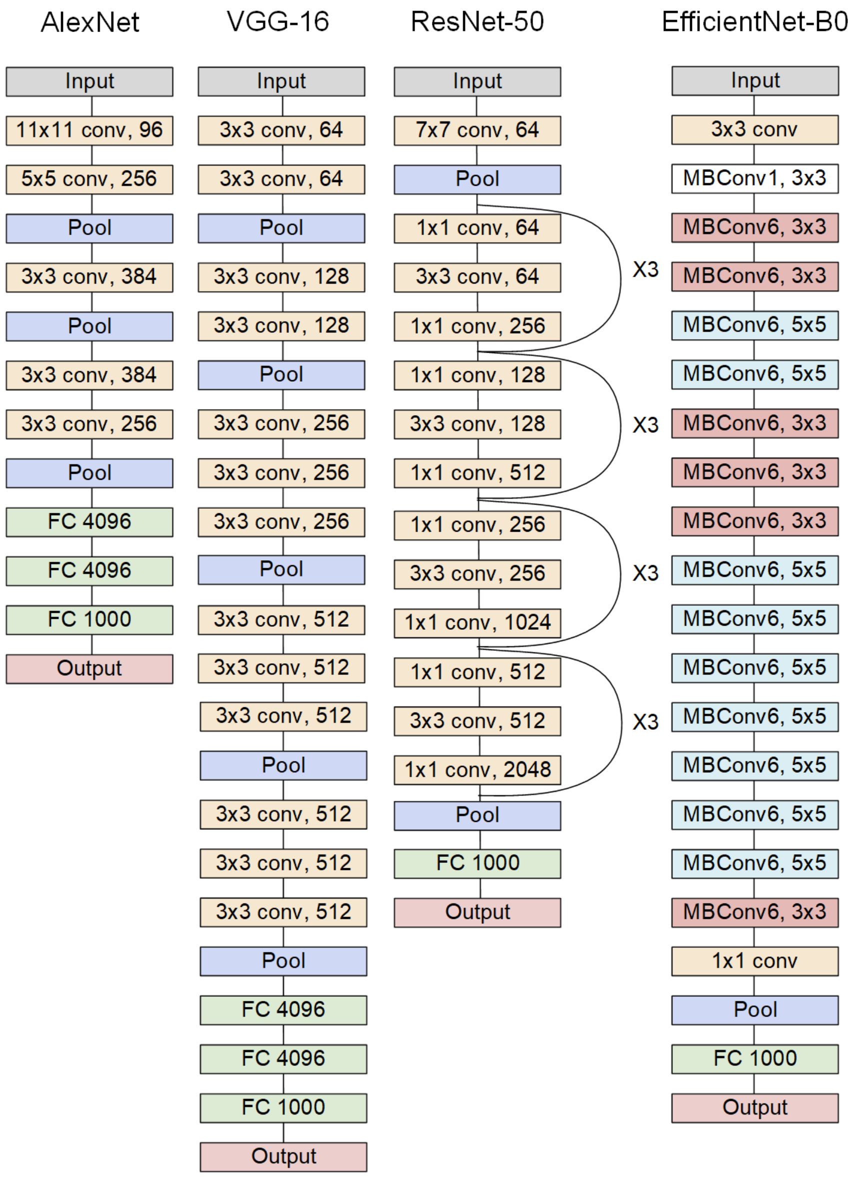


Figure 4.1. The configurations of the four CNN-based classifiers.

4.1.1 AlexNet

AlexNet uses ReLu activation function and dropout method [57]. ReLu increases training speed and the dropout is added in the first two fully-connected layers to minimize overfitting. It starts with a convolution layer of 11×11 kernel size and 96 filters, which reduces to 5×5 and 256 filters. It also consists of three convolution layers with 3×3 kernel size and three pooling layers. These layers are followed by three fully-connected layers and an output layer.

Table 4.1. Parameters of the images and deep learning algorithms. Stochastic gradient descent solver with 100 epochs is considered.

Case	**Parameter**	**Value**
Raw image	image size	$1688 \times 990 \times 3$
	image type	.jpg
Pre-processing	image size	$422 \times 248 \times 3$
	image type	.jpg
AlexNet	Learning rate	0.001
	Kernel size	11×11, 5×5, 3×3
	Kernel stride	4, 2, 1
	Batch size	64
VGG-16	Learning rate	0.0001
	Kernel size	3×3
	Kernel stride	2, 1
	Batch size	32
ResNet-50	Learning rate	0.0001
	Kernel size	7×7, 3×3, 1×1
	Kernel stride	2, 1
	Batch size	32
EfficientNet-B0	Learning rate	0.001
	Kernel size	5×5, 3×3, 1×1
	Kernel stride	2, 1
	Batch size	32

4.1.2 VGG16

The VGG configuration adds more convolution layers to facilitate accuracy via deep neural networks [58]. However, an excessive addition of such layers potentially leads to gradient dispersion that results in training divergence. Here, VGG-16 is used for image training with five groups of two or three convolution layers of 3×3 kernel size together with five pooling layers, three fully-connected layers, and an output layer.

4.1.3 ResNet-50

The ResNet configuration addresses the vanishing gradient problem by exploiting batch normalization and by skipping connections among convolution layers [59]. It also comes in different structures including ResNet-18/34/50/101/152. Here, ResNet-50 is adopted, which consists of a 7×7 convolution layer and groups of 1×1, 3×3, and 1×1 convolution layers. It also has two pooling, one fully-connected, and output layers.

4.1.4 EfficientNet-B0

Lastly, EfficientNet improves accuracy through model scaling and branches into B0–7 [60]. In this work, EfficientNet-B0 is used for its compact architecture, which is characterized by a 3×3 convolution layer followed by moving reverse bottleneck convolution (MBConv) layers with either 3×3 or 5×5 kernels. It also conveys 1×1 convolution, pooling, fully-connected, and output layers.

4.2 Training Environment and Input Datasets

4.2.1 Training Environment

The training and testing of the four CNN models is performed in two systems. The first uses a 64-bit Windows 8, Intel® Core™i7-6900K CPU @ 3.20 GHz processor and 128 GB RAM. The second uses Google Colab with 16 GB RAM and Tesla P100 GPU. All Python code uses Tensorflow with Keras interface.

4.2.2 Input Datasets

Spectrogram dataset (Figure 2.7) is collected with USRP B210 and *QT GUI Waterfall Sink* block shown in Figure 2.4. Python scripts are developed to capture real-time screenshots during the testing procedure. A total of 1578 images are collected, which are divided into 762 images under no jamming and 204 images for each of the jamming types. The standard image size is $1688 \times 990 \times 3$ pixels, which is scaled down to $422 \times 248 \times 3$ to reduce training time. These images are separated into 70% training and 30% testing.

4.3 Deep Learning Results

Table 4.2 shows the DR, VA, F-score, and the training/testing times for the CNN classifiers. EfficientNet-B0 has the highest DR of 100% and 99.79% for the two- and five-class models, respectively.

AlexNet results in the lowest training/testing times, highest VA, and fastest convergence rate as shown in Figure 4.2(a)-(d). It is also found that the training and testing times for the CNN models are significantly higher than those obtained by the conventional ML algorithms, which is attributed to the CNNs deep and complex architectures. However, since detection times (i.e., GTE, CTE) result from classifying 472 images, the average processing times of the five-class EfficientNet-B0 model to classify an image are 0.005s with GPU and 0.066s with CPU, enabling real-time jamming detection and classification.

Figure 4.3 shows the receiver operating characteristic (ROC) of the two-class models and indicates that EfficientNet-B0 outperforms other classifiers in jamming detection.

Lastly, the weighted FARs are computed from the confusion matrices, presented in Figure 4.4, 4.5, 4.6, and 4.7, to be 0.60% for AlexNet, 1.55% for VGG-16, 1.86% for ResNet-50, and 0.03% for EfficientNet-B0. It is noteworthy to mention that complexity and severity of a given jamming type have no contribution to its classification accuracy. For example, barrage jamming is the simplest to launch, whereas protocol-aware has the most launch complexity. Yet, their feature- and spectrogram-based misclassifications are nearly 2.5% and 0%, respectively. Similarly, barrage has the highest severity among the four jamming types, whereas successive-pulse has the lowest severity. Nonetheless, their feature- and spectrogram-

Table 4.2. Performance metrics of the CNN models (VA: Validation Accuracy, DR: Detection Rate, FS: F-score, GTR: GPU Training Time, GTE: GPU Testing Time, CTR: CPU Training Time, CTE: CPU Testing Time).

Performance metrics for five-class models							
ML Classifier	**VA (%)**	**DR (%)**	**FS**	**GTR (sec)**	**GTE (sec)**	**CTR (sec)**	**CTE (sec)**
AlexNet	100.00	99.36	0.99	174	0.82	6765	4.90
VGG-16	94.03	94.50	0.94	1479	5.81	70932	63.30
ResNet-50	99.82	98.10	0.98	1118	2.72	58359	31.84
EfficientNet-B0	**98.55**	**99.79**	**1.00**	**1530**	**2.53**	**39476**	**31.22**
Performance metrics for two-class models							
ML Classifier	**VA (%)**	**DR (%)**	**FS**	**GTR (sec)**	**GTE (sec)**	**CTR (sec)**	**CTE (sec)**
AlexNet	100.00	99.15	0.99	171	0.76	6048	4.86
VGG-16	99.91	99.36	0.99	1478	5.77	52837	63.43
ResNet-50	100.00	99.36	0.99	1114	2.47	52334	32.00
EfficientNet-B0	**99.91**	**100.00**	**1.00**	**1489**	**2.28**	**39351**	**31.55**

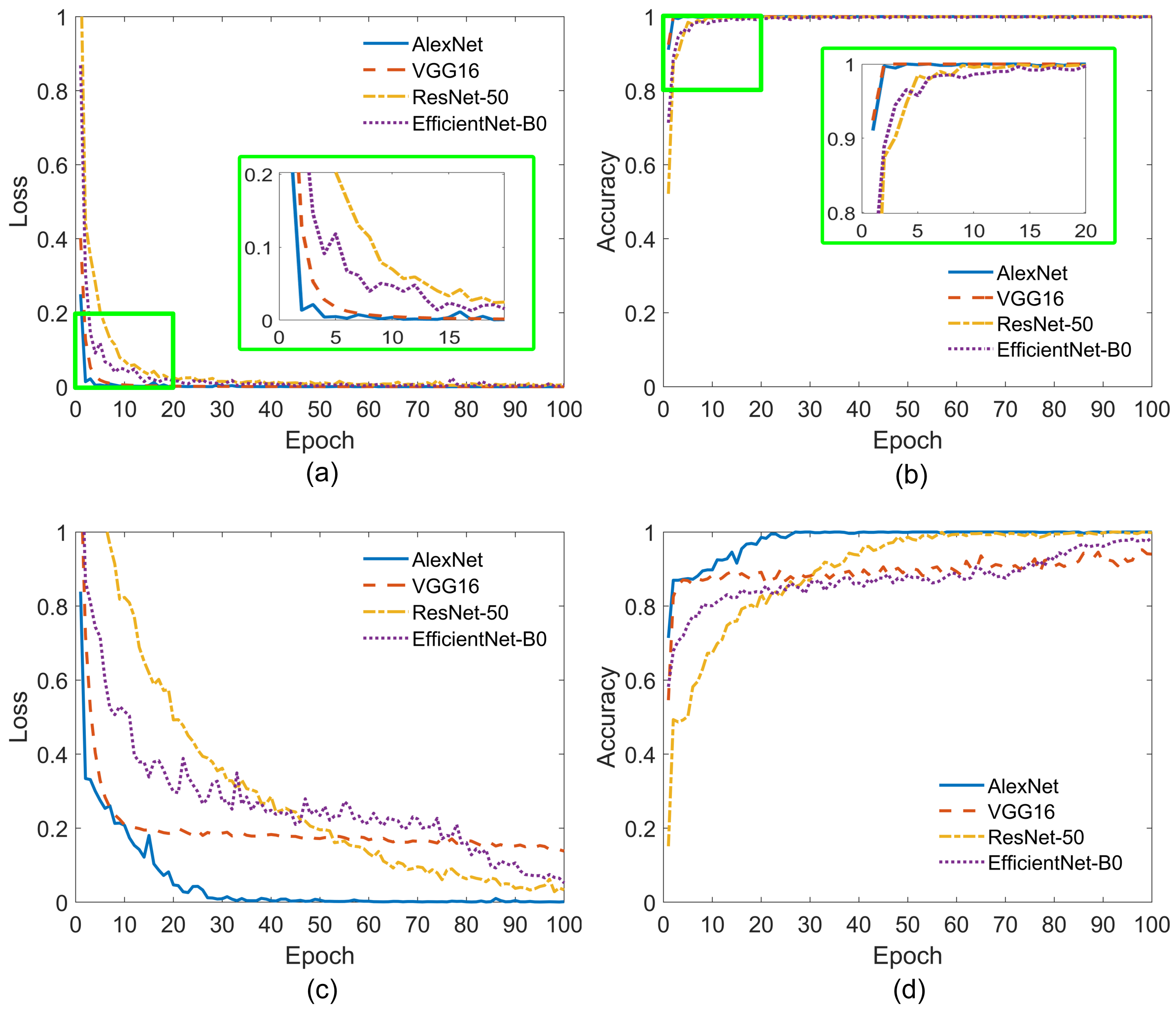


Figure 4.2. Two-class models (a) loss and (b) accuracy. Five-class models (c) loss and (d) accuracy.

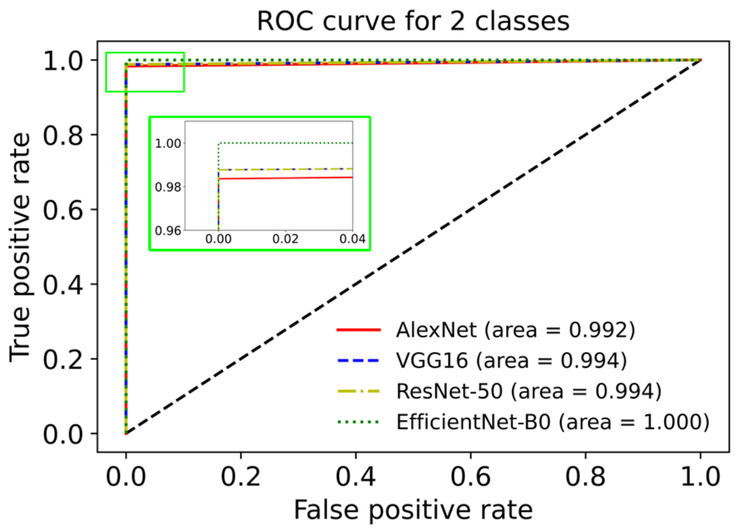
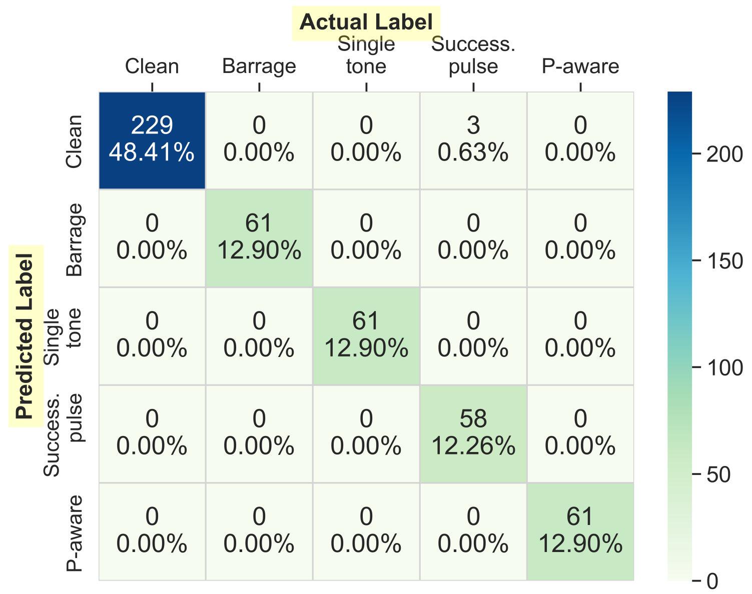


Figure 4.3. ROC curve of the two-class CNN models.

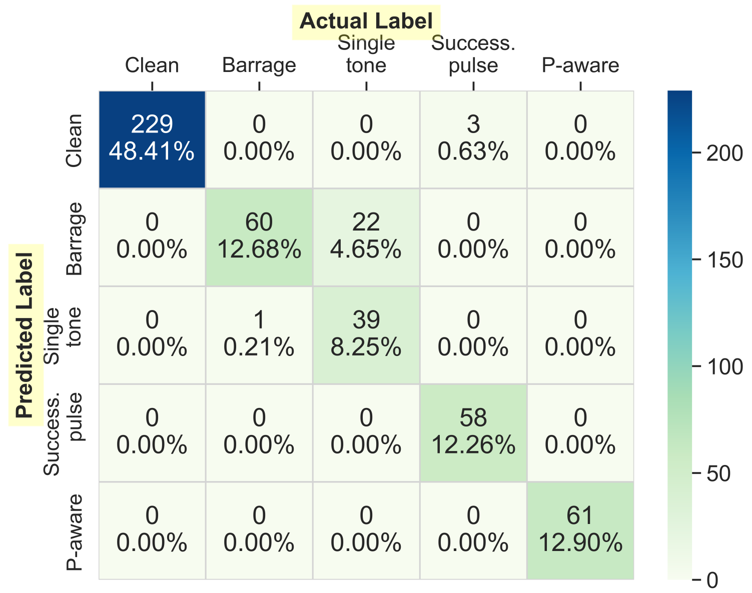
based misclassifications are $< 1\%$ and 0%, respectively, as demonstrated in the confusion matrices in Figures 4 and 9.

Table 4.3 shows a comparison between the proposed approach and those reported in the literature in detecting and/or classifying jamming attacks with applications to satellite communications, OFDM, VANETs, and 5G/IoT networks. This work entailed four different jamming attacks with the highest detection and classification accuracy. Furthermore, six conventional and four deep learning models are trained and tested with realistic datasets of extracted signal features and images obtained after rigorous measurement routines.



Predicted Label \ Actual Label	Clean	Barrage	Single tone	Success. pulse	P-aware
Clean	229 48.41%	0 0.00%	0 0.00%	3 0.63%	0 0.00%
Barrage	0 0.00%	61 12.90%	0 0.00%	0 0.00%	0 0.00%
Single tone	0 0.00%	0 0.00%	61 12.90%	0 0.00%	0 0.00%
Success. pulse	0 0.00%	0 0.00%	0 0.00%	58 12.26%	0 0.00%
P-aware	0 0.00%	0 0.00%	0 0.00%	0 0.00%	61 12.90%

Figure 4.4. Confusion matrix of the five-class CNN models for AlexNet.



Predicted Label \ Actual Label	Clean	Barrage	Single tone	Success. pulse	P-aware
Clean	229 48.41%	0 0.00%	0 0.00%	3 0.63%	0 0.00%
Barrage	0 0.00%	60 12.68%	22 4.65%	0 0.00%	0 0.00%
Single tone	0 0.00%	1 0.21%	39 8.25%	0 0.00%	0 0.00%
Success. pulse	0 0.00%	0 0.00%	0 0.00%	58 12.26%	0 0.00%
P-aware	0 0.00%	0 0.00%	0 0.00%	0 0.00%	61 12.90%

Figure 4.5. Confusion matrix of the five-class CNN models for VGG16.

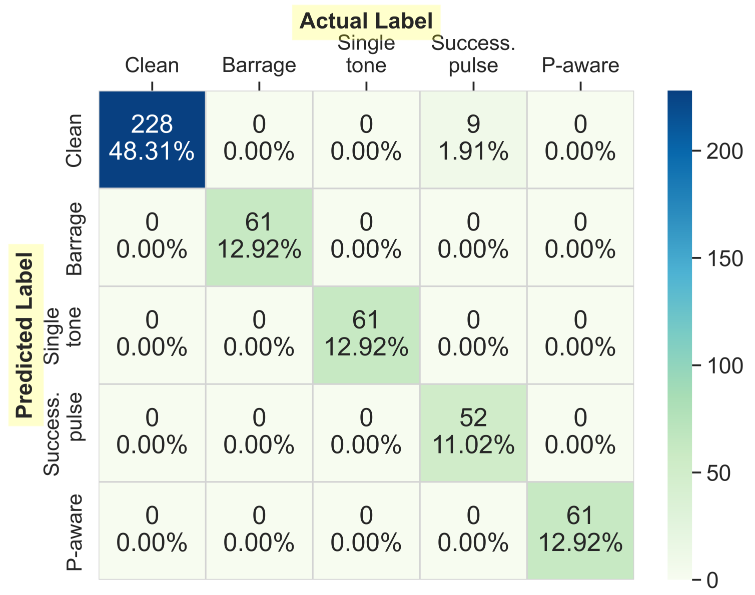


Figure 4.6. Confusion matrix of the five-class CNN models for ResNet-50.

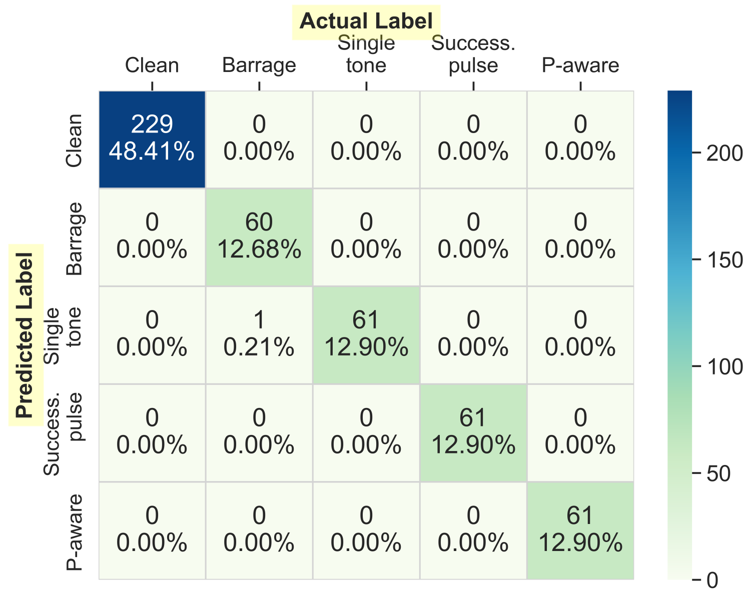


Figure 4.7. Confusion matrix of the five-class CNN models for EfficientNet-B0.

Table 4.3. Comparison between the proposed approach and other state-of-the-art approaches.

Ref.	Dataset Type	Dataset Source	ML Type	DR (%)	Application	Jamming Type
[26]	Spectrograms	Simulations	CNN & SVM	93.10	Satellites	Barrage, Pilot-tone, Intermittent (Detection)
[27]	Spectrograms	Measurements	CNN & RNN	86.10	OFDM	Barrage, Reference Signal (Detection and Classification)
[28]	Features	Measurements	DT, AdaBoost, SVM	97.00	OFDM	Constant, Reactive (Detection)
[29]	Features	Simulations	DT, RF, SVM	99.06	IoT Networks	Intermittent (Detection)
		Measurements	DT, RF, SVM, KNN	89.70		
[30]	Features	Borrowed	MLP, MLP&SVM	94.51	5G Networks	Constant, Random, Deceptive, Reactive (Detection and classification)
[43]	Features	Simulations	K-means	-	VANET	Constant, Smart (Detection)
[44]	Features	Simulations	RF, SVM, MLP	97.50	5G Networks	Barrage (Detection)
This Work	Features	Measurements	LR, KNN, NB, DT, RF, MLP	92.20	UAVs	Barrage, Single-tone, Success.-pulse, P-aware (Detection and classification)
	Spectrograms	Measurements	CNN	99.79		

5. CONCLUSION

An ML approach is proposed to detect and classify four types of jamming attacks on OFDM-based receivers with application to UAVs. Each attack is built using B210 SDR and launched against a drone to qualitatively analyze its impacts considering severity, complexity, and jamming range. Then, an SDR is used in proximity to the drone in systematic testing scenarios to record key OFDM parameters, threshold, signal power, noise power, and SNR for the feature-based approach as well as spectrogram images for the spectrogram-based approach. The former is explored with six algorithms, whereas the latter is realized with four different deep learning CNN algorithms to achieve higher jamming detection and classification accuracy. All ML models are validated quantitatively with metrics including detection and false alarm rates, and showed that jamming is detected with 92.2% and 99.79% confidence following the feature-based and spectrogram-based classifiers, respectively. This approach requires the integration of a data extraction module with the UAV receiver for obtaining real-time signal features and/or images to facilitate the detection and classification routines. This integration may impose the need for interface circuitry adjoined with a further analysis of power aspects and hardware imperfection. Future work could be installing a small processor (i.e., Raspberry Pi) on the drone. By loading the ML and deep learning model into this processor, the drone would have the ability to detect and classify jamming in real time, which can further advance this technology to UAV industry. Future work could also entail exploring more jamming types (e.g., deceptive, reactive), incorporating maximum-likelihood-based classification and advanced SNR probing, and investigating UAV-specific anti-jamming solutions (e.g., flight scheduling, path optimization).

REFERENCES

[1] M. Messinger and M. Silman, “Unmanned aerial vehicles for the assessment and monitoring of environmental contamination: An example from coal ash spills,” *Environmental pollution*, vol. 218, pp. 889–894, 2016.

[2] A. Bhardwaj, L. Sam, Akanksha, F. Martín-Torres, and R. Kumar, “Uavs as remote sensing platform in glaciology: Present applications and future prospects,” *Remote sensing of environment*, vol. 175, pp. 196–204, 2016.

[3] R. Allison, J. Johnston, G. Craig, and S. Jennings, “Airborne optical and thermal remote sensing for wildfire detection and monitoring,” *Sensors*, vol. 16, no. 8, p. 1310, 2016.

[4] J. Qi, D. Song, H. Shang, *et al.*, “Search and rescue rotary-wing uav and its application to the lushan ms 7.0 earthquake,” *Journal of Field Robotics*, vol. 33, no. 3, pp. 290–321, 2016.

[5] *Unmanned aerial vehicles UAV market*, https://www.marketsandmarkets.com/Market-Reports/unmanned-aerial-vehicles-uav-market-662.html Accessed on December 9, 2021.

[6] J. Paredes, C. Jacinto, R. Ramírez, I. Vargas, and L. Trujillano, “Simplified fuzzy-pd controller for behavior mixing and improved performance in quadcopter attitude control systems,” in *2016 IEEE ANDESCON*, IEEE, 2016, pp. 1–4.

[7] P. Oettershagen, T. Stastny, T. Mantel, *et al.*, “Long-endurance sensing and mapping using a hand-launchable solar-powered uav,” in *Field and Service Robotics*, Springer, 2016, pp. 441–454.

[8] J. Braga, H. Velho, G. Conte, P. Doherty, and É. Shiguemori, “An image matching system for autonomous uav navigation based on neural network,” in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, IEEE, 2016, pp. 1–6.

[9] J. Tiemann, F. Schweikowski, and C. Wietfeld, “Design of an uwb indoor-positioning system for uav navigation in gnss-denied environments,” in *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, IEEE, 2015, pp. 1–7.

[10] M. Mullins, M. Holman, K. Foerster, N. Kaabouch, and W. Semke, “Dynamic separation thresholds for a small airborne sense and avoid system,” in *AIAA Infotech @ Aerospace (I@A) Conference*, 2013, p. 5148.

[11] M. Mullins, K. Foerster, N. Kaabouch, and W. Semke, “Incorporating terrain avoidance into a small uas sense and avoid system,” in *Infotech@ Aerospace 2012*, 2012, p. 2504.

[12] M. Mullins, K. Foerster, N. Kaabouch, and W. Semke, "A multiple objective and behavior solution for unmanned airborne sense-and-avoid systems," *AUVSI's Unmanned Systems North America*, 2012.

[13] F. Martel, M. Mullins, W. Semke, N. Kaabouch, *et al.*, "Cooperative miniature collision avoidance system flight testing for small unmanned aircraft systems," in *Proceedings of AUVSI conference*, 2011.

[14] M. Mullins, K. Foester, and N. Kaabouch, "Traffic alerting system for manned-unmanned aircraft airspace conflicts," in *ND EPSCoR/IDeA State Conference*, 2017.

[15] K. Foerster, B. Whitney, J. Hahn, N. Kaabouch, and W. Semke, "A health monitoring system for uas utilizing a miniature airborne sense and avoid system," in *AIAA Infotech@ Aerospace Conference*, 2013, p. 4654.

[16] H. Reyes, N. Gellerman, and N. Kaabouch, "A cognitive radio system for improving the reliability and security of uas/uav networks," in *2015 IEEE Aerospace Conference*, 2015, pp. 1–9. DOI: 10.1109/AERO.2015.7119159.

[17] H. Reyes, N. Kaabouch, W. Semke, and S. Salle, "Fuzzy logic method for link loss detection during unmanned aerial vehicle flights," in *Infotech@ Aerospace 2012*, 2012, p. 2574.

[18] *Drones are quickly becoming a cybersecurity nightmare*, https://threatpost.com/drones-breach-cyberdefenses/143075 Accessed on December 9, 2021.

[19] C. Albanesius, *SkyJack Software Finds and Hijacks Drones*, https://uk.pcmag.com/security-devices-2/8285/skyjack-software-finds-and-hijacks-drones Accessed on December 9, 2021.

[20] S. McCarthy, *HK$1 million in damage caused by GPS jamming that caused 46 drones to plummet during Hong Kong show*, https://sg.news.yahoo.com/hk-1-million-damage-caused-080848555.html Accessed on December 9, 2021.

[21] K. Ibrahim, S. Ng, I. Qureshi, A. Malik, and S. Muhaidat, "Anti-jamming game to combat intelligent jamming for cognitive radio networks," *IEEE Access*, vol. 9, pp. 137 941–137 956, 2021.

[22] G. Han, L. Xiao, and H. Poor, "Two-dimensional anti-jamming communication based on deep reinforcement learning," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2017, pp. 2087–2091.

[23] B. Duan, D. Yin, Y. Cong, H. Zhou, X. Xiang, and L. Shen, "Anti-jamming path planning for unmanned aerial vehicles with imperfect jammer information," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2018, pp. 729–735.

[24] H. Wang, J. Chen, G. Ding, and J. Sun, "Trajectory planning in uav communication with jamming," in *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, IEEE, 2018, pp. 1–6.

[25] L. Xiao, X. Lu, D. Xu, Y. Tang, L. Wang, and W. Zhuang, "Uav relay in vanets against smart jamming with reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4087–4097, 2018.

[26] S. Gecgel and G. Kurt, "Intermittent jamming against telemetry and telecommand of satellite systems and a learning-driven detection strategy," in *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*, 2021, pp. 43–48.

[27] S. Gecgel, C. Goztepe, and G. Kurt, "Jammer detection based on artificial neural networks: A measurement study," in *Proceedings of the ACM Workshop on Wireless Security and Machine Learning*, 2019, pp. 43–48.

[28] O. Puñal, I. Aktaş, C. 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*, IEEE, 2014, pp. 1–10.

[29] 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)*, IEEE, 2019, pp. 1–5.

[30] 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)*, IEEE, 2020, pp. 1–5.

[31] P. Bithas, E. Michailidis, N. Nomikos, D. Vouyioukas, and A. Kanatas, "A survey on machine-learning techniques for uav-based communications," *Sensors*, vol. 19, no. 23, p. 5170, 2019.

[32] Q. Wu, H. Wang, X. Li, B. Zhang, and J. Peng, "Reinforcement learning-based anti-jamming in networked uav radar systems," *Applied Sciences*, vol. 9, no. 23, p. 5173, 2019.

[33] X. Lu, L. Xiao, C. Dai, and H. Dai, "Uav-aided cellular communications with deep reinforcement learning against jamming," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 48–53, 2020.

[34] J. Pawlak, Y. Li, J. Price, *et al.*, "A machine learning approach for detecting and classifying jamming attacks against ofdm-based uavs," in *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*, 2021, pp. 1–6.

[35] M. Strohmeier, V. Lenders, and I. Martinovic, "On the security of the automatic dependent surveillance-broadcast protocol," *IEEE Communications Surveys Tutorials*, vol. 17, no. 2, pp. 1066–1087, 2015. DOI: 10.1109/COMST.2014.2365951.

[36] M. Manesh and N. Kaabouch, "Analysis of vulnerabilities, attacks, countermeasures and overall risk of the automatic dependent surveillance-broadcast (ads-b) system," *International Journal of Critical Infrastructure Protection*, vol. 19, pp. 16–31, 2017.

[37] K. D. Wesson, T. Humphreys, and B. Evans, *Can cryptography secure next generation air traffic surveillance?* http://users.ece.utexas.edu/~bevans/papers/2015/nextgen/ Technical Report, Accessed on December 9, 2021, 2021.

[38] C. Giannatto Jr, "Challenges of implementing automatic dependent surveillance broadcast in the nextgen air traffic management system," 2015.

[39] B. Danev, H. Luecken, S. Capkun, and K. El Defrawy, "Attacks on physical-layer identification," in *Proceedings of the third ACM conference on Wireless network security*, 2010, pp. 89–98.

[40] S. Brands and D. Chaum, "Distance-bounding protocols," in *Workshop on the Theory and Application of of Cryptographic Techniques*, Springer, 1993, pp. 344–359.

[41] B. Xiao, B. Yu, and C. Gao, "Detection and localization of sybil nodes in vanets," in *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, 2006, pp. 1–8.

[42] M. Sliti, W. Abdallah, and N. Boudriga, "Jamming attack detection in optical uav networks," in *2018 20th International Conference on Transparent Optical Networks (ICTON)*, IEEE, 2018, pp. 1–5.

[43] D. Karagiannis and A. Argyriou, "Jamming attack detection in a pair of rf communicating vehicles using unsupervised machine learning," *Vehicular Communications*, vol. 13, pp. 56–63, 2018.

[44] Y. Arjoune, F. Salahdine, M. 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)*, IEEE, 2020, pp. 459–464.

[45] L. Mokdad, J. Ben-Othman, and A. Nguyen, "Djavan: Detecting jamming attacks in vehicle ad hoc networks," *Performance Evaluation*, vol. 87, pp. 47–59, 2015.

[46] A. Nguyen, L. Mokdad, and J. Ben Othman, "Solution of detecting jamming attacks in vehicle ad hoc networks," in *Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems*, 2013, pp. 405–410.

[47] J. Grover, N. K. Prajapati, V. Laxmi, and M. Gaur, "Machine learning approach for multiple misbehavior detection in vanet," in *International conference on advances in computing and communications*, Springer, 2011, pp. 644–653.

[48] H. Liu, B. Lang, M. Liu, and H. Yan, "Cnn and rnn based payload classification methods for attack detection," *Knowledge-Based Systems*, vol. 163, pp. 332–341, 2019.

[49] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.

[50] X. Wang, X. Wang, and S. Mao, "Rf sensing in the internet of things: A general deep learning framework," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 62–67, 2018.

[51] C. Liu, J. Wang, X. Liu, and Y. Liang, "Deep cm-cnn for spectrum sensing in cognitive radio," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2306–2321, 2019.

[52] A. Hussain, N. Saqib, U. Qamar, M. Zia, and H. Mahmood, "Protocol-aware radio frequency jamming in wi-fi and commercial wireless networks," *Journal of communications and networks*, vol. 16, no. 4, pp. 397–406, 2014.

[53] S. Müller and C. Richardson, *GitHub - gnuradio/gr-inspector: Signal Analysis Toolbox for GNU Radio*, https://github.com/gnuradio/gr-inspector Accessed on December 9, 2021.

[54] Y. Cho, J. Kim, W. Yang, and C. Kang, "Introduction to ofdm," in *MIMO-OFDM Wireless Communications with MATLAB®*. 2010, pp. 111–151. DOI: 10.1002/9780470825631.ch4.

[55] *GitHub: UAVs Jamming Detection and Classification*, https://github.com/michaelevol/uavs_jamming_detection Accessed on December 9, 2021.

[56] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[57] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[58] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[59] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[60] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 6105–6114.

PUBLICATIONS

A Machine Learning Approach for Detecting and Classifying Jamming Attacks Against OFDM-based UAVs; Jered Pawlak, Yuchen Li, Joshua Price, Matthew Wright, Khair AI Shamaileh, Quamar Niyaz, Vijay Devabhaktuni, Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning. 2021.

Jamming Detection and Classification in OFDM-based UAVs via Feature- and Spectrogram-tailored Machine Learning; Yuchen Li, Jered Pawlak, Joshua Price, Khair AI Shamaileh, Quamar Niyaz, Vijay Devabhaktuni, IEEE Access. 2021. (Second round review)