2020

Machine Learning based RF Transmitter Characterization in the Presence of Adversaries

Debashri Roy
University of Central Florida

MACHINE LEARNING BASED RF TRANSMITTER CHARATERIZATION IN THE PRESENCE OF ADVERSARIES

by

DEBASHRI ROY
M.S. University of Central Florida, 2018

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2020

Major Professor: Mainak Chatterjee

ABSTRACT

The advances in wireless technologies have led to autonomous deployments of various wireless networks. As these networks must co-exist, it is important that all transmitters and receivers are aware of their radio frequency (RF) surroundings so that they can learn and adapt their transmission and reception parameters to best suit their needs. To this end, machine learning techniques have become popular as they can learn, analyze and even predict the RF signals and associated parameters that characterize the RF environment.

In this dissertation, we address some of the fundamental challenges on how to effectively apply different learning techniques in the RF domain. In the presence of adversaries, malicious activities such as jamming, and spoofing are inevitable which render most machine learning techniques ineffective. To facilitate learning in such settings, we propose an adversarial learning-based approach to detect unauthorized exploitation of RF spectrum. First, we show the applicability of existing machine learning algorithms in the RF domain. We design and implement three recurrent neural networks using different types of cell models for fingerprinting RF transmitters. Next, we focus on securing transmissions on dynamic spectrum access network where primary user emulation (PUE) attacks can pose a significant threat. We present a generative adversarial net (GAN) based solution to counter such PUE attacks. Ultimately, we propose recurrent neural network models which are able to accurately predict the primary users' activities in DSA networks so that the secondary users can opportunistically access the shared spectrum. We implement the proposed learning models on testbeds consisting of Universal Software Radio Peripherals (USRPs) working as Software Defined Radios (SDRs). Results reveal significant accuracy gains in accurately characterizing RF transmitters- thereby demonstrating the potential of our models for real world deployments.

To my mother Arati.

ACKNOWLEDGMENTS

I am grateful to my advisor Dr. Mainak Chatterjee for guiding, supporting and believing in me over the years of my Ph.D studies. Without his experienced mentoring, my dream of earning a Ph.D would not have come true. I would like to express sincere appreciation of my committee members Dr. Nazanin Rahnavard, Dr. Phil Zheng, and Dr. Cliff Zou for serving in my committee. Their constructive feedback and comments have helped me in improving my dissertation. I would like to thank Dr. Tathagata Mukherjee from University of Alabama in Huntsville and Dr. Eduardo Pasiliao from the Air Force Research Laboratory for the collaborative opportunities. I would also like to thank all my colleagues at the Computer Networks and Security laboratory and all my friends who have always inspired me while working as a Ph.D. student. I like to thank my colleagues at the University of Florida Research and Engineering Education Facility laboratory as well. In addition, I would like to thank the Department of Computer Science at the University of Central Florida for offering me with all these wonderful classes which enriched my knowledge along the course of my education. I would like to acknowledge the role of my parents and brother, Dr. Raju Hazari, Mr. Subrata Dutta, Mr. Hemanta Chakroborty and my school teachers for motivating and supporting my desire of pursuing higher studies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

The ubiquitous use of wireless services and applications have become ingrained in every aspect of our lives. We depend on wireless technologies not only for our smart phones but also for other applications like telemetry, surveillance, emitter location, radio navigation, jamming, anti-jamming, radar detection, UAV surveillance, navigation, and location tracking. With such large scale dependence on use of the radio frequency (RF) spectrum, it becomes imperative that we manage and use the limited available spectrum in the most efficient manner possible. In order to do that, one needs to better understand the ambient signal characteristics for optimal deployment of wireless infrastructure and efficient resource provisioning.

Design of new wireless technologies and deployment of wireless networks using those technologies must take into consideration several factors including detection and monitoring of encroachment, ability to predict RF propagation and coverage, techniques to mitigate noise, policies enabling spectrum sharing, characterization of frequencies and waveforms, coverage analysis for optimal deployment, detection and de-confliction and more importantly identification of adversarial RF signals. Furthermore, the advent of dynamic spectrum access enabled by the use of software defined radios is pushing the frontiers of wireless communications. These radios are expected to constantly monitor the radio environment and the resulting data can be used to *learn* about their surroundings so that they can intelligently adapt their RF parameters (e.g., operating frequency, bandwidth, waveform, modulation, noise mitigation) to meet their desired objectives.

In order to best use the radio resources in both the spatial and time domains, and to maximize the spectral efficiency, past and current knowledge of the RF signals are important. Though sensing mechanisms can be used to assess the current environment, learning techniques are typically used to analyze the past observations and predict future occurrences of events related to a signal. With

the proven success of machine learning (ML) techniques in various domains, such techniques are also being sought for characterizing and understanding the RF environment. Some of the goals of the learning techniques in the RF domain are emitter fingerprinting, emitter localization, modulation recognition, feature learning, attention and saliency, autonomous RF sensor configuration and waveform synthesis.

ML techniques allow the radios to learn and adapt their RF parameters so as to optimize their respective objectives. Such adaptation by the radios is achieved by exposing their configuration options to make the operational parameters flexible and tunable. As a consequence, the configurability and adaptability features open up avenues for manipulation as well where a radio can be induced to learn false information by adversaries [9]. This creates an unique set of challenges in the domain of Radio Frequency Machine Learning (RFML) systems which makes implementing ML algorithms for RF systems way more challenging.

In this chapter, we discuss the recent trends of facilitating learning in the RF domain using different machine learning approaches. These ML techniques have their own strengths and weaknesses depending on the context and the type of dataset being used. We start by broadly classifying the ML techniques into supervised and unsupervised learning, and highlight the various schemes that have been used in the RF domain. Then, we discuss five techniques that are currently being widely explored as they exhibit promising results for future implementations. These are: i) Support Vector Machine (SVM), ii) Deep Neural Networks (DNN), iii) Convolutional Neural Network (CNN), iv) Recurrent Neural Network (RNN), and v) Non-parametric Bayesian Classifier. SVM and the three neural networks (NN) perform better with continuous and multi-dimensional datasets, which could be leveraged when the RF signals contain multiple attributes. However, applicability of SVM and NN comes at a cost: they exhibit high variance sensitivity. On the other hand, Bayesian classifiers are advanced statistical techniques for classifying and identifying features. In spite of these developments, we argue that the ML techniques have their limitations in an adversarial setting

i.e., if adversarial RF signals are present during the learning and/or classification process. We show that approaches based on the relatively new generative adversarial nets (GAN) are well-suited for learning in adversarial RF environments and are able to distinguish between adversarial and trusted signals and sources. We also discuss the effectiveness if transmitter fingerprinting technique, the basics of primary user emulation attack in the dynamic spectrum access (DSA) regime.

1.1 Machine Learning for RF Signals

The basic work flow of any machine learning algorithm starts with digesting the feature space. Most often the quality of the model learned by the algorithm depends heavily on the quality of the features used as input to the algorithm which in turn depends on the problem. Thus for example, even though the RF data may consist of the RSSI values for a given problem, using the raw input in this form may not result in the optimal model. In such cases it might be helpful to transform the raw input features into a higher/lower dimensional feature space that succinctly captures the essence of the problem thereby making it easier to learn better ML models for the problem. Fig. 1.1 shows the classification of different machine learning algorithms that have been used for learning in the RF domain. All learning techniques broadly fall under either supervised or unsupervised training mechanism.

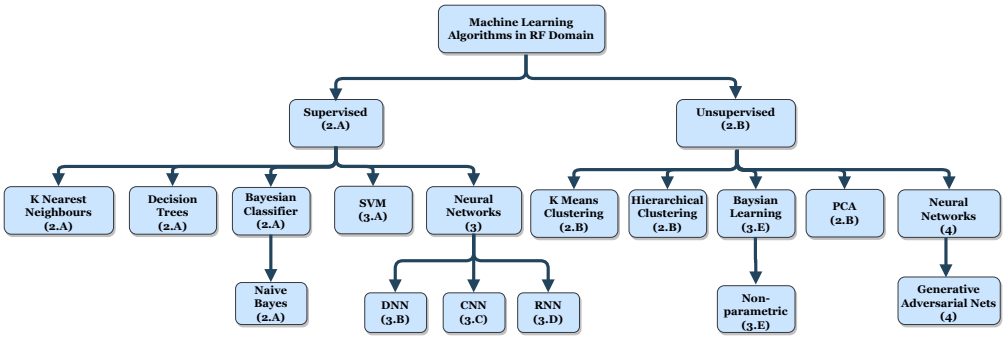


Figure 1.1: Classification of Machine Learning Algorithms in the RF Domain

1.1.1 Supervised Learning in RF Domain

Supervised learning algorithms are a set of learning algorithms where a set of mutually exclusive labeled data is used for building the learning model (also called training). One of the simplest supervised learning algorithm is based on the K-nearest neighbor (KNN) search. The KNN algorithm classifies previously unseen data based on the labels of nearest data samples that are included in the training set. Algorithms based on KNN searches are usually very computation intensive, specially in the higher dimensions where the dimension of the feature space factors into the running time of the known algorithms for computing KNNs. This also makes algorithms based on KNN search unsuitable for the RF domain. Furthermore, KNN may not provide optimal performance for higher dimensional dataset, which is often the case for RF signal data. Another set of supervised learning algorithms are based on the idea of decision trees. Decision trees are used to assign specific class labels to the items using predictive modeling based on the values of the features associated with the item of interest. The decision trees perform well with high dimensional data, which could be advantageous for handling multidimensional RF data, however longer training time and lack of accuracy could hinder usefulness in mission critical applications as is most often the case with RF processing. Another important class of supervised algorithms are based on the idea of Bayesian classifiers. Bayesian classifiers predict the probability of a given sample belonging to a particular class using *a priori* knowledge. A specific category of Bayesian classifier is the Naïve Bayes, where one assumes that each feature contributes towards the classification and are mutually correlated. Since different parameters from the same kind of radios can be considered as mutually correlated, Naïve Bayes can potentially be used successfully in the RF domain. Naïve Bayes also works reasonably well with limited training data and is less likely to suffer from overfitting. However, the predictions are less accurate compared to the other methods like neural networks. We defer our discussions on SVMs, DNN, CNN, and RNN to Section 1.2 as they have shown promising results in the RF domain and thus deserve to be discussed in details.

1.1.2 Unsupervised Learning in RF Domain

Unsupervised learning algorithms are a set of algorithms where there is no explicit training phase for building a model from labelled data as is done with the supervised algorithms. These algorithms make inference from the unlabelled data, most often exploiting the observed variance of the data and their association relative to each other. K-means clustering is one such learning algorithm where the observed data is partitioned into clusters using information about the distance between the data points (or more generally using the similarity between the observed data points). New data is assigned to a cluster such that the data point is closest to a particular cluster mean. Another set of clustering algorithms are based on the idea of *hierarchies*. Here the items of interest are progressively partitioned into a hierarchy of clusters, the clusters which are higher up in the hierarchy are coarse whereas the ones that are lower are more fine grained. Hierarchical clustering can be used for transmitter identification and classification where the number of entities are unknown. Another set of unsupervised algorithms concerns with the problem of *dimensionality reduction*. Dimensionality reduction aims to identify a subspace of the feature space such that the projection of the data into the subspace (which has a lower dimension than the feature space) would explain most of the variation observed in the data. Principal component analysis (PCA) is one such method for dimensionality reduction and data compression. PCA can be very useful for the multi-dimensional RF data as it can be leveraged to speed up the underlying classification or regression algorithms when faster online learning and processing is required.

The final and most important category of unsupervised algorithms for the RF domain is the Generative Adversarial Networks (GANs) [32]. It is a relatively new class of algorithms that have been shown to perform well in adversarial settings. GANs use a generative model which enables the realistic generation of samples from a given distribution which can then be used to train a discriminator for identifying real samples drawn from the distribution as opposed to fake ones obtained

from the generator.

1.2 Important ML Algorithms for RF Domain

In this section, we describe the ML techniques that have been most successful in characterizing the RF environment by identifying and differentiating signals from different kinds of transmitters like broadcast radio, local and wide area data and voice radios, radars etc. Support vector machine (SVM) was one of the earliest algorithms used due to the availability of different standard library packages and its applicability to multi-dimensional labeled dataset. However, neural network based classification have become popular in recent times as they are flexible with the learning parameters and also yield better accuracy. As far as unsupervised learning is concerned, there are a few efforts towards RF parameter identification, the non-parametric version of the Bayesian classifier probably being the most successful one.

1.2.1 Support Vector Machine

SVMs have been one of the most successful classical machine learning algorithms and have been applied to a vast array of problems. SVM is a discriminative classifier using supervised learning and generates an optimal hyperplane for data classification and identification. It has been applied to the RF domain for transmitter identification using fingerprinting. At its heart SVMs are binary classifiers that assume that the data is linearly separable and computes the optimal separating hyperplane by solving a quadratic program on the space defined by the training data.

SVMs have been successfully applied for the task of transmitter identification using the RF fingerprint of the transmitter. In [43], Kroon *et al.* presented a RF fingerprinting technique for banning prohibited transmitters from accessing the cellular base stations. They proposed a customized en-

semble classifier based on the probability density of a SVM classifier, which achieved 97% true positives and 80% true negative rates.

However in recent times, the research trends on RF fingerprinting is shifting towards using raw signal data as compared to using hand engineered features. This has also been facilitated by the availability of automatic feature learning systems like the multi layer perceptron. In [40], Youssef *et al.* investigated different ML strategies including SVM and neural networks using raw I/Q data, rather than using any hand-engineered features. I/Q data consists of complex-valued in-phase (I) and quadrature (Q) component in a signal data constellation. Their implementation produces good training accuracy of 87.6% but poor test accuracy of 67.6%.

SVM classifiers are relatively easy to implement and can be extended for higher dimensional data. However, achieving a high accuracy remains a challenge. Furthermore, SVM implementations do not typically consider any adversarial situation; thus, when malicious entities try to pass as a trusted device, the SVM classifier has no way of determining its true identity and thus would incorrectly classify it as one of the trusted devices. This makes the possibility of using SVMs in adversarial settings quite low. Moreover SVMs require hand engineered features even when raw I/Q data is used. As a result of these drawbacks and the recent resurgence of neural networks attention as turned to using the same for RFML applications.

1.2.2 Deep Neural Networks

Deep Neural Networks (DNN) have revolutionized the field of artificial intelligence in the last few years. The problems tackled by DNNs range over Computer Vision, Natural Language Processing, Speech Processing and so on. They have been shown to perform better than humans, for some of these problems. More recently, [65] and [67] have shown the efficacy of using DNNs in RF communication systems. In [65], Shea *et al.* presented a modulation recognition approach using

DNN, achieving nearly 82%-87% accuracy. They used a synthetic dataset consisting of 11 widely used modulations: 8 digital and 3 analog modulations. Their RF fingerprinting was based on the modulation type which was used as the primary input feature (for the network which then computes more complex features for model learning).

The automatic features learned by DNNs are often times better and more informative than the hand engineered features used for SVMs and hence the DNNs yield better accuracy. However DNNs do not perform as well for datasets with spatial and temporal correlations, which is the case for the RF domain. RF signals exhibit high spatial correlation (e.g., modulation schemes) or high temporal correlation (e.g., I/Q signal data) or both and in order for a system to work well with RF data these correlations need to be exploited by the learning system. Furthermore, DNNs have shown to be susceptible to malicious attacks and fails to distinguish rogue transmitters from trusted ones [68] in the presence of active adversaries.

1.2.3 Convolutional Neural Networks

Fully connected DNNs are like standard neural networks but with a lot more hidden layers. However, these networks can be augmented with *convolutional layers* for faster training and for enabling the network to learn more compact and meaningful representations. Deep Convolutional Neural Networks (DCNN) have been shown to be effective for several different tasks in communication. There have been quite a few attempts at using DCNN for learning different RF parameters. One such effort is presented in [68], where Shea *et al.* presented an optimized DCNN model (with 18 layers) for recognition of modulation schemes for a large synthetic dataset (consisting of 24 modulation schemes), as well as over-the-air data. 94% accuracy on synthetic data and 87% accuracy on over-the-air data, were achieved. Inspired from this and buoyed by the success of the application of CNNs in communication, Youssef *et al.* presented a CNN architecture [40] for deep feature embedding, transmitter identification and classification.

CNN works with several filters that capture the *spatial* correlation within the input features for the purpose of computing lower level features that effectively capture the spatial correlation between the input features. A cascade of such filters are used for propagation of these features through multiple layers, in effect computing more low level features, thereby giving the best solution for datasets having spatially correlated features. However, it does not perform well in general for datasets where the features are uncorrelated, for example the *rise time* which is a feature that can be computed for many RF datasets. CNNs may not perform well if the nature of the correlations change from the training data to the test data. This is because in such cases the features computed through the different filters on the training data will not be applicable to the test data. It is also not recommended for time series data where temporal correlations might exist. Moreover, CNNs have the same limitations as DNNs in regards to immunity from malicious attacks.

1.2.4 Recurrent Neural Networks

Recurrent neural networks (RNN) are capable of predictions with *time series data*. They have been shown to work well with speaker recognition tasks [74] and inspired by this and based on the fact that the raw RF signal data from the transmitter represent a time series, RNN can be considered as a potential model to build a system for learning transmitter embedding and classification. One variant of RNN is long short term memory (LSTM) [28], which has been successfully used for modeling *temporal data* such as speech. The problem of estimating the noise from the signal, requires analysis of the temporal data because the noise characteristics can only be estimated by looking at the received signal over time.

In order to estimate the noise, any system needs to "listen" to the underlying signal for sometime and "remember" the same, for noise estimation. Previously, neural networks lacked this capability. Another issue was the problem of *vanishing gradients*, when trying to use *back propagation* with

temporal data. However, both these problems were solved by the invention of gated units, such as the Long Short-Term Memory (LSTM), the Gated Recurrent Unit (GRU), and their variants.

In [73], Sreeraj *et al.* presented a 2-layer LSTM model to perform modulation recognition over the synthetic dataset of 11 modulation schemes. Accuracies close to 90% was achieved for data with high signal to noise ratio coupled with time domain data. This implementation establishes the feasibility of using recurrent neural network models for learning RF features related to time domain analysis. However, effectiveness of LSTM models for learning other RF parameters is still an open research challenge.

RNN performs well with temporally correlated dataset through the implementation of the concepts of "memory" and "gates". However it responds poorly to spatially correlated data. RNN implementations also incorrectly classify the rogue data as one of the trusted ones when active adversaries spoof the signals as coming from trusted sources in cases where they are not.

1.2.5 Non-parametric Bayesian Classifier

A non-parametric Bayes classifier is an unsupervised learning strategy which uses a probability density estimator to determine the probability of an observation belonging to a particular class. Nguyen *et al.* presented a non-parametric Bayesian learning [59] approach to identify wireless devices by characterizing their fingerprints. They considered a device dependent feature space modeled as multivariate Gaussian distribution, which includes frequency difference and phase shift difference as dominant features. The non-parametric Bayesian learning approach is then employed over the generated distributions and used to identify the unknown number of clusters. Experimental results reveal that the proposed method was capable of clustering 1 to 4 Zigbee transmitting devices with a 100% hit rate. Next, they showed that the proposed RF fingerprinting approach can be applied for intrusion detection for Sybil and masquerade attacks by spoofing the MAC address.

There is still a dearth of methods that are resilient to any general type of attacks for RFML systems.

1.2.6 Comparison of Different ML Techniques

Analyzing the potential of different kinds of ML approaches in RF parameter learning, we summarize that (a) SVM struggles to achieve high accuracies for higher dimensional datasets, (b) DNN is best suited for fixed valued RF parameters (e.g., *rise time*), (c) CNN works well with RF parameters that exhibit high spatial correlations (e.g., *modulation techniques*), (d) RNN is the best option for RF parameters with high temporal correlations (e.g., *I/Q signal data*), and (e) Non-parametric Bayesian classifiers are limited to specific type of applications and datasets.

It must be noted that all the aforementioned ML techniques are susceptible to attackers. Once the attacker gets to know of the features used by the learning engine, it becomes easy for the attacker to mislead the learning process. The same applies in the RF domain where a RF transmitter can spoof the signals of others and remain undetected.

1.3 Adversarial Learning via GAN

The idea of a generative adversarial net (GAN) [32] in machine learning is based on synergistic application of ideas from game theory and unsupervised learning. It consists of two competing systems: a generator (adversary) and a discriminator. The input from the "adversaries" is used to build robust discriminative models that can operate in the presence of real adversaries. The overall training mechanism can be conceptualized as a min-max game with two players, namely the generator and the discriminator. They help each other to improve themselves through an iterative training process. Though in theory, the generator and discriminator should play the game indefinitely, in reality, depending on the ratio of data and model density, the discriminator overpowers

the generator in a finite amount of time, due to the vanishing gradient of the generator. In practice, this results in generating more accurate and robust ML models. Note that the discriminator implementation is made deeper than the generator in order to get a purposeful implementation of the GAN framework.

1.4 Transmitter Fingerprinting

The ubiquitous usage of wirelessly connected Internet-of-Things (IoT) [99] along with the deployment of wireless autonomous systems has ushered in a new era of industrial scale deployment of radio frequency (RF) devices. This prevalence of large scale peer-to-peer communication and the nature of the underlying ubiquitous network brings forth the challenge of accurately identifying a RF transmitter. Every device that is part of a large network needs to be able to identify its peers with high confidence in order to set up secure communication channels. One of the ways in which this is done is through the interchange of "keys" [84] for host identification. However, such schemes are prone to breaches by malicious agents [90] because often the actual implementations of such systems are not cryptographically sound. In order to get around the problem of faulty implementations, one can use the transmitter's intrinsic characteristics to create a "fingerprint" that can be used by a transmitter identification system. Every transmitter, no matter how similar, has intrinsic characteristics because of the imperfections in its underlying components such as amplifiers, filters, frequency mixers as well as the physical properties of the transmitting antenna; these characteristics are unique to a specific transmitter. The inaccuracies present in the manufacturing process and the idiosyncrasies of the hardware circuitry also contribute to the spatial and temporal characteristics of the signal transmitted through a particular device.

Those inherent heterogeneity can be exploited to create unique identifiers for the transmitters. One such property is the imbalance in the Inphase (I) and Quadrature (Q) phase components of the

signal (I/Q data). However, because of the sheer number of the transmitters involved, manually "fingerprinting" each and every transmitter is not a feasible task [21]. Thus, in order to build such a system, there needs to be an "automatic" method of extracting the transmitter characteristics and using the resulting "fingerprint" for the differentiation process. One way of achieving this is by learning the representation of the transmitter in an appropriate "feature space" that has enough discriminating capability so as to be able to differentiate between "apparently identical" transmitters.

Among the various approaches that can be used to discern this feature space, deep learning (DL [45]) based methods provide an efficient and automatic way of learning and characterizing the feature space. They are able to learn and analyze the inherent properties of large deployments and use it to predict and characterize the associated parameters for the task of automatic feature learning for classification (or regression). Deep neural networks have been shown to be effective for automatically learning discriminating features from data for various tasks [31]. With proper choice of the neural network architecture and associated parameters, they can compute arbitrarily good function approximations [48]. Since the task of classification is equivalent to learning the decision boundary, neural networks have been a natural candidate for a learning machine.

1.5 Primary User Emulation Attack

The opportunity for abundant usage of wireless devices has created an overly crowded radio spectrum and led to the scarcity in spectrum availability. In order to deal with this different pervasive measures are taken to deal with competitive nature of the spectrum availability. However, studies have shown that a large portion of licensed spectrum is unused at any given time or location [54]. To exploit such unused spectrum, the concept of dynamic spectrum access (DSA) was envisioned. The deployment of such spectrum management approach requires the use of intelligent systems at lower levels of the communication stack, even at the end users. Cognitive radio networks (CRN)

have proven its competence for such deployments [107]. The basic idea of DSA is to allow some unlicensed users (secondary users) to opportunistically access the spectrum of the licensed users (primary users) when the spectrum is not in use. The rules strictly restrict any harmful overlap or pretentious use of spectrum by secondary users (SUs) when primary users (PUs) are present. The cognitive properties of CRN, enable the radio devices to take decision on how to manage spectrum for both PUs and SUs. One important challenge in order to achieve the goal of ideal CRN deployment is ensuring the security of PUs.

Most of the research on such spectrum allocation and management techniques for DSA deployment, are build on the assumption that all participants are cooperative, honest, and the network is without the presence of adversaries. The Federal Communications Commission (FCC) [27] mandated that all SUs must release the occupied spectral band as soon as any PU starts to transmit in that band, ensuring full privacy and availability for the licensed users. However, since till now the CRN network deployments are lacking any measure to implement the security guidelines [10], a situation could arise where the PUs get denied the required spectrum due to the presence of a malicious SU. This threat could be categorized as denial-of service (DoS) attack [39]. *An adversarial SU could pose itself as a PU by transmitting the signal with characteristics identical to the PU.* Such malicious SUs could threaten the integrity of the CRN in two ways: (i) by preempting the existing SUs in any spectral band, by posing as a PU; and (ii) by fooling the spectrum manager to deny the PU, as the malicious SU is impersonating itself as a valid PU. Such malicious deployment of large scale can hijack the entire white space of any spectrum band, thus launching a "DoS" attack on the legitimate SUs and PUs. Such attacks were first described by Chen et al. [15] as primary user emulation (PUE) attacks.

1.6 Primary User's Activity Prediction

The ever-increasing demands for spectrum access from different emerging wireless applications have made it necessary to better manage and utilize the radio spectrum. The successful deployment of such spectrum management approaches requires intelligent and adaptive systems, in order to accurately assess the radio environment such that unlicensed (secondary) users are able to opportunistically access the spectrum of licensed (primary) users when such spectrum is not in use, thereby increasing the spectrum utilization. However, such spectrum management and deployment practices must take into account the fact that spectrum access policies prohibit any *interference violation* by the secondary users (SUs) when primary users (PUs) use the spectrum. The Citizens Broadband Radio Service (CBRS) [82] is an example of a DSA implementation, where variety of commercial users share the 3.5 GHz band with incumbent federal and non-federal licensed users.

The Federal Communications Commission (FCC) [27] mandated that all SUs must release the occupied spectral bands as soon as any PU starts to transmit on that band, ensuring uninterrupted availability to the licensed users. To ensure this, the SUs must have knowledge about the spectrum availability. This awareness is typically achieved by sensing the transmission activities on the target spectrum bands using various techniques like: use of beacons, geolocation database, and local energy sensing at the receivers [52, 108].

In this dissertation, we focus on spectrum sensing performed at a central spectrum sensor (SS), as the use of a centralized SS has broaden applications and lower infrastructure costs [103].

We illustrate the various aspects of spectrum sensing in Fig. 1.2. The first row represents the PU's ON-OFF activity and the remaining rows show the SU's activities that include sensing and transmissions when PU is in the OFF state.

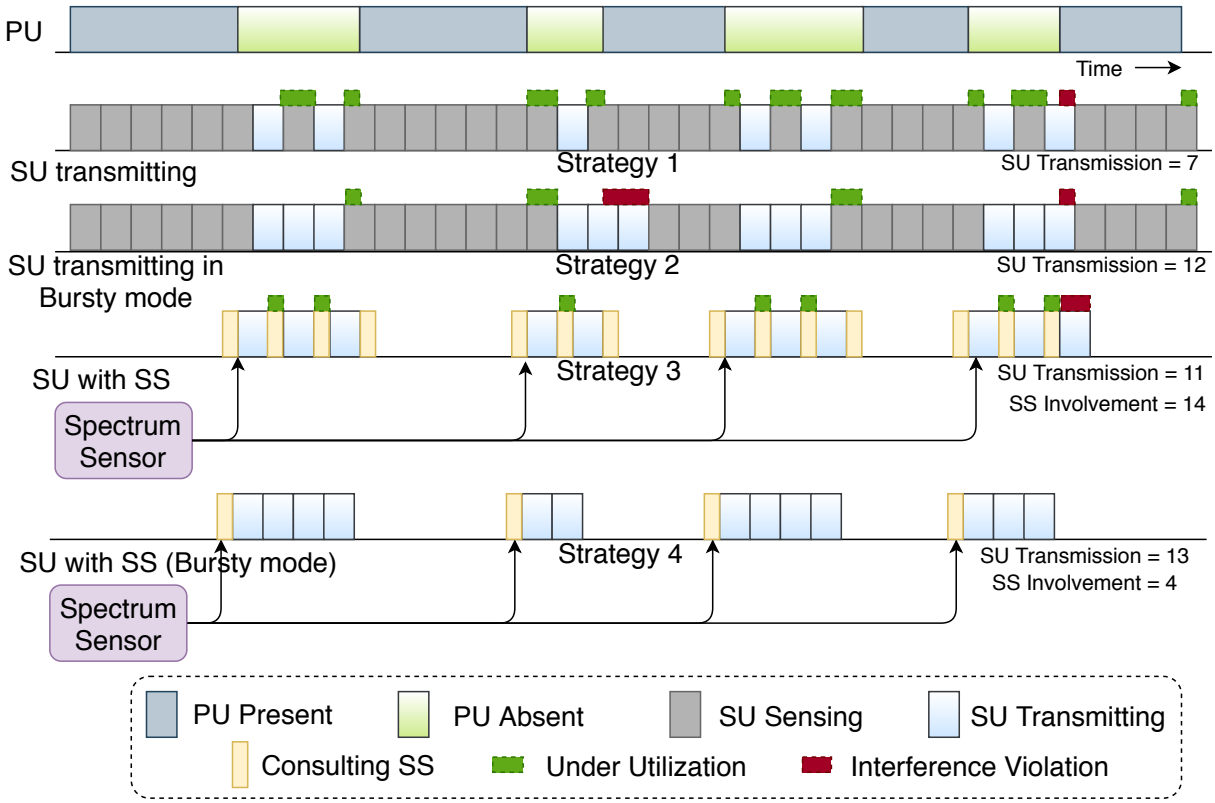


Figure 1.2: Typical Spectrum Sensing Scenarios

Note that the first two strategies do not involve separate SS and we also assume that the SUs sense the spectrum continuously. The last two strategies involve the SUs deciding to access the spectrum based on the information obtained from a local centralized SS. Thus in the first case, a conservative strategy is imposed on the SU for use of the channel. The cognitive SU continuously senses the channel and whenever it finds that the channel is free, it transmits in the next timestamp, going back to the sensing state after that. The second strategy is also conservative but here the SU transmits in bursty mode. Both these conservative strategies aim to avoid interference violations; however, under-utilization is high. As for the third strategy, the spectrum sensors send information to secondary user whenever it senses the channel to be free. The secondary user transmits from one timestamp and consults the spectrum sensor for the next timestamp. Note that here we assume

that the time taken to consult the spectrum sensor is less than the spectrum sensing time. This assumption coupled with the strategy minimizes the under-utilization of the previous two strategies. The fourth strategy is proposed in this dissertation. In this case, the spectrum sensor is intelligent and trained over the historical data of primary user activity. It can predict the primary use's activity for the next timestamp. The secondary user uses that prediction to transmit in bursty mode. It is evident from all the four strategies that long term prediction from the SS enables the SU to transmit efficiently over the shared channel by minimizing both the under-utilization and the interference violations.

1.7 Contribution of this Dissertation

This dissertation focuses on four approaches to address some of the fundamental challenges on how to apply different learning techniques in the RF domain. (1) First, we propose an adversarial learning based approach to detect malicious attacks in the RF domain. (2) We explore the spatio-temporal properties in RF data for fingerprinting of RF transmitters. (3) We propose a defense mechanism against primary user emulation attacks using generative adversarial network based learning for dynamic spectrum access (DSA) networks. (4) Lastly, we propose a machine learning model for primary user activity prediction in DSA networks using recurrent structures. We show the applicability of existing machine learning algorithms in RF domain is feasible. Those learning techniques can be leveraged to different cheap and robust security measures in dynamic spectrum access era of RF transmission in general. The designed models, and testbed evaluations show the significant accuracy over real collected dataset, therefore portraying feasible scenarios of real-world deployment. The main contributions of this dissertation are as follows:

1. We propose and implement the *Radio Frequency Adversarial Learning (RFAL) framework*

using a GAN with two primary components: i) a generative model that uses a deep neural network (DNN) for generating fake (aka, counterfeited) signals that closely resembles the real signals, by deducing the parameter space and replicating the time-invariant features and ii) a discriminative model that also uses a neural network (DNN) to distinguish trusted transmitters from rogue ones. During the learning phase, the outcome of the decision process is fed to the generative model, allowing the adversary (generator) to update its model. The generator thus serves as a compact front-end for mimicking a transmitter (rogue transmitter in this case).

2. Once RFAL detects the trusted transmitters from the rogue ones, RFAL uses standard NN models to *differentiate* between the different trusted transmitters. RFAL first uses a convolutional neural network (CNN) which leverages the correlation between the complex-valued I/Q data constellations. We also test RFAL using a fully connected deep neural network (DNN) that improves upon the accuracy of the CNN. Finally, a recurrent neural network (RNN) with both long short term memory (LSTM) and gated recurrent units (GRU) is used with RFAL that exploits the time series properties of the I/Q data.

3. Our models have been validated on a laboratory testbed consisting of several universal software radio peripheral (USRP) B210s [26] and a RTL-SDR receiver which is also a software defined radio (SDR) [60]. The USRPs transmitted signals on a particular frequency which were received by the RTL-SDR. We also collect I/Q data from a SDR made by another manufacturer, namely ADALM-PLUTO [23]. We show that the I/Q imbalance is more pronounced (and thus easier to exploit both explicitly as well as implicitly) when different types of SDRs (from different manufacturers) are used as transmitters. We also collect three more datasets of I/Q values from 8 USRP B210s with varying signal-to-noise-ratio (SNR). We use distance and multi-path as the defining factors for SNR variation during data-collection.

4. We train the proposed models and present a competitive analysis of the performance of our

models against the traditional techniques and state-of-the art techniques for transmitter identification (classification). Results reveal that the proposed methods out-perform the existing ones thus establishing the superiority and usefulness of the proposed models, more so considering the fact that the proposed models do not require any pre-processing of the raw I/Q data that feeds into the NN models.

5. We exploit the temporal properties of I/Q data and propose a supervised learning approach for transmitter identification using a recurrent network structure. We use two approaches: first, we exploit only the temporal property, then we exploit the spatio-temporal property. We propose RNNs with LSTM and GRU cells for the first approach. We propose a convLSTM model for the latter. Although transmitter fingerprinting has been studied earlier, but to the best of our knowledge this is the first work which leverages the spatio-temporal property of the over-the-air signal data.

6. We validate the proposed RNN models using the indoor testbed setup mentioned earlier. The novelty of our approach lies in accurately modeling and implementing the different types of RNNs to generate a robust transmitter fingerprinting system from over-the-air signal data, by exploiting the spatio-temporal correlations within the signal data.

7. We propose a generative adversarial net based solution towards the primary emulation attack. We impersonate SUs using two types of generator models: (a) dumb attacker, and (ii) smart attacker. The dumb attacker has no “prior" information about the signal characteristics of the PUs, but still tries to emulate the PUs. However, the smart attackers has sufficient information about the PU’s signal data and therefore can emulate the PU’s signal in an intelligent way. We also model two discriminators (neural networks) and train them over the PU data, and generated data. The “dumb discriminator” gets generated data from dumb generator attack model, and “smart discriminator” gets data from smart one. We show that the GAN training makes the discriminator able to distinguish between a wide array of possible

malicious entity types and therefore being able to detect the real adversaries with intention of PUE attack.

8. Using USRP, we collect the raw over-the-air signal data from PUs to train both the discriminators. We also extract the PU's signal characteristics from the collected data and train the smart generator. We show that the untrained discriminators have a $\sim$50% accuracy for detecting PUE attackers during the deployment phase. We present 100% training accuracy for both the generator models. The trained discriminators over dumb, and smart generator models exhibit testing accuracy of 98%, and 99.5% respectively during deployment phase, with real PUE attackers present.

9. We propose three machine learning based models for long-term prediction of the primary user's activity. They are: (i) Linear regression; (ii) Recurrent neural network (RNN) with Long Short Term Memory (LSTM) cells; and (iii) RNN using Convolutional LSTM (ConvLSTM) cells. Using a testbed, we record transmission activities of 8 software defined radios (USRP B210) [26], which are used as primary users. A central spectrum sensing module is trained using the proposed models on the collected dataset for multiple epochs by minimizing the mean squared error over the training data. The trained models are then used by the spectrum sensor and used for long-term prediction of the activities of the primary user, to be used by the secondary users during the deployment phase.

10. We deployed the trained version of all the models in a spectrum sensor and predicted the shared channel availability for the secondary users with 75%, 97%, and 99% accuracy respectively for linear regression, LSTM, and ConvLSTM. Note that intuitively the ConvLSTM based model is able to achieve this high accuracy by exploiting the spatio-temporal correlation present within the recurrent structure of the collected I/Q data. We also show through testbed evaluation that the proposed models can decrease interference violations by 0.2%, 99.3%, and 100% for linear regression, LSTM and ConvLSTM models, respectively.

Under-utilizations are decreased by 98.9%, 99.5%, and 100% respectively, for the aforementioned models.

1.8 Organization of the Dissertation

This dissertation is organized as follows. Chapter 2 presents the related work that is relevant to this dissertation. In Chapter 3, we propose a rogue transmitter detection technqiue using GAN based framework in adversarial learning. Chapter 4 presents the machine learning algorithms for classification of authentic transmitters using recurrent structures of I/Q data. A defense mechanism against the PUE attack is presented in Chapter 5. The primary user prediction modelling and testbed implementation is presented in Chapter 6. Finally, Chapter 7 presents the conclusions.

CHAPTER 2: LITERATURE REVIEW

In this chapter, we discuss the related works that are most relevant to this dissertation. We divide the discussion into i) transmitter fingerprinting, ii) use of recurrent neural network (RNN) for transmitter fingerprinting, (iii) existing defense against PUE attacks, and (iv) primary user's activity predictions.

2.1 Transmiter Fingerprinting

The problem of transmitter classification has been studied in the past. Here we first discuss a few traditional learning based approaches for transmitter classification. Finally, we discuss more recent transmitter identification methods based on the idea of automatic feature detectors (like neural networks). We also discuss the advantages of using automatic feature learning techniques over the traditional ones.

2.1.1 Traditional Learning based Techniques

Traditional transmitter classification methods are based on statistical learning techniques that use expert engineered features which leverage some unique characteristics of the transmitters. In [94], the authors proposed a genetic algorithm based solution for transmitter classification based on transients. A transient signal is transmitted when a transmitter is powered up or powered down. During this short period (typically few micro seconds), capacitive loads charge or discharge. A genetic algorithm generated the "transient times" from 5 different types of transmitters, which were later classified using a NN model yielding a 85% - 98% accuracy. It is to be emphasized that the experimental results were solely based on the synthetically generated transient values. Though this

work used NNs for the final classification, the features (transients) were empirically determined and hence we categorize this as an example of a traditional approach. A multifractal segmentation technique was proposed in [86] using the same concept of transients. The segmentation technique extracted significant features from transient signals and generated a compact multifactral model. Later a probabilistic NN classifier achieved 92.5% success rate over the extracted transient features in a simulated environment. Another transient based transmitter classification was proposed in [41]. A k-nearest neighbor discriminatory classifier was used to create a classification engine which leveraged transient signals for spectral feature selection. The authors achieved a 97% accuracy at 30 dB SNR and 66% accuracy at 0 dB SNR for classification of 8 transmitters.

A different approach for transmitter fingerprinting was proposed in [101], where the authors classified FM radio transmitters based on unique stray features extracted from spurious modulation characteristics. The proposed approach was able to classify samples (20 dB SNR) from 3 FM radio stations with 62%-71% accuracy. This method does not provide a competitive accuracy and is also constrained by the need to have knowledge of modulation technique. A particle swarm optimization (PSO) technique was proposed in [25], where two radar transmitter models were classified based on the radar pulse's time-frequency representation. An acceptable classification accuracy was reported with 20 dB SNR and relatively low component tolerances. In [106], the authors proposed a location-based transmitter fingerprinting approach by extracting signal characteristics (skewness and kurtosis) from wavelet transform. This transmitter location fingerprint was performed for 4 *stationary* transmitters in an indoor office environment. In [30], the authors proposed a dimensionality reduction method for extracting intrinsic features from bispectrum information of transmitters. They reported 99% accuracy for transmitter identification from the bispectrum matrices. However, deployment of such techniques in real-time is challenging due to the additional overhead for generating the bispectrum before the classifier can be invoked for identification.

As seen from the above discussion, there are some advantages to using traditional fingerprinting

techniques such as classifying the transmitters based on their unique identifications in that we are able to leverage the expertise of the "human-in-the-loop" using such techniques through feature engineering. However, these methods have extra overhead due to the feature extraction step and furthermore the quality of the solution is constrained by the knowledge of the expert making such techniques limited in scope. Moreover, as the features are signal and protocol dependent, any change in the nature of the transmission mandates a change in the underlying model, thus making them hard to generalize across different types of transmissions (having varying protocols, heterogeneous transmitters etc.).

2.1.2 Automatic Feature Learning based Techniques

In recent years, there has been some effort at using automatic feature learning techniques for fingerprinting RF transmitters. In [65] the authors presented a radio modulation classification method using naively learned features. They have shown that blind temporal learning on densely encoded time series using CNNs is a viable approach. However, this method did not perform well in the low signal to noise ratio (SNR) regime. In [92], the authors have presented an unsupervised learning technique using convolutional autoencoders, to learn the modulation basis functions and then leverage that to recognize different digital modulation schemes. They also proposed and evaluated quantitative metrics for evaluating the quality of the encoding using domain relevant performance metrics.

In [67] the authors have demonstrated the use of NN for modulation detection. Apart from the results, an interesting aspect of the work is the way I/Q values were used as input to the NN. More precisely, given N I/Q values, the authors used a vector of size $2N$ as an input to the NN, effectively using the I and Q components as a tuple representing a point in the complex plane. A method for modulation classification was proposed in [73], for a distributed wireless spectrum

sensing network. The authors used a recurrent neural network using long short term memory (LSTM) cells yielding 90% accuracy on a synthetic dataset [72].

An in-depth study on the performance of deep learning based radio signal classification was presented in [68]. The authors considered 24 modulation schemes with a rigorous baseline method that uses higher order moments and strong boosted gradient tree classification for detection. The authors also applied their method to real over-the-air data collected by Software Defined Radios (SDRs). In [66] an approach based on the idea of adversarial learning was proposed for synthesizing new physical layer modulation and coding schemes. The adversarial approach is used to learn the channel response approximations in any arbitrary communication system, enabling the design of a smarter channel autoencoder. All these approaches demonstrate the efficacy of using an "end-to-end" technique based on learning deep feature representations, for different tasks in the RF domain.

There have been quite a few studies that have used CNN based models for automatic feature learning [17,55,56,75,85,102,109] for the task of transmitter classification (or identification). The CNN models presented in [55,75,85] require some pre-processing on the raw signal data before it can be used as input. In [102], the authors compared several learning paradigms for the task of transmitter identification. More precisely, they looked at conventional deep neural nets, convolutional neural nets, support vector machines, and deep neural nets with multi-stage training. They showed that deep neural nets with multi-stage training worked best for the problem and achieved 100% accuracy with a novel dataset having 12 transmitters. On the other hand CNN models were proposed in [17, 56, 109] for existing datasets (ACARS [4], ADS-B [5], FIT/CorteXlab [53]). However, none of these models directly take the the raw signal data as input. Note that none of the methods that we have discussed till now in regards to the task of transmitter identification are resilient to the presence of active adversaries. This motivated us to propose a robust NN based model which would be resilient to the presence of active adversaries and at the same time provide an end-to-end

solution to the transmitter classification problem.

2.1.3 Comparison of Traditional and Deep Learning based Methods

All the traditional techniques that have been used for RF analysis lack flexibility and robustness. These approaches require an expert's involvement for determining which features (e.g., transients, spurious modulation, etc.) to extract and how to design an algorithm tuned to that feature. Even if a feature is identified, it is not necessary that this feature will be applicable for all scenarios. For example, location based fingerprinting [106] will work well for indoor environments but will fail for non-stationary transmitters. However, deep learning (DL) based methods are capable of learning the features automatically from the data and hence they do not require the feature engineering step. Furthermore it is possible to use DL techniques in conjunction with adversarial learning to build robust transmitter classification models that can function in the presence of active adversaries. In this work, we use the raw I/Q data as an input to the learning model. The model automatically discerns the features that can encode the information required to disambiguate the transmitters. Note that the features computed by the DL system can be implicitly based on the "I/Q imbalance" or some other intrinsic features of the transmitter or a combination of these.

2.1.4 I/Q Imbalance based Fingerprinting

"I/Q imbalance" based fingerprinting approaches provide more significant discriminant information than transient based or modulation-metrics based approaches [37]. Though the use of RF signal data in general (and I/Q data in particular) with machine learning algorithms has been limited in the past, more recently there has been several applications (see [57] and references therein).

An "I/Q imbalance" based fingerprinting approach was proposed in [37], where a subclass dis-

criminant analysis (SDA) ML method was used to estimate the distortion parameters from the I/Q constellations as features. The proposed method was tested on transmitted signals from 7 Time-division multiple access (TDMA) satellite terminals, relayed by a transparent transponder, giving 97% accuracy over 15 dB SNR. However, this method is not guaranteed to capture the difference between transmitter's "I/Q imbalances," as it is aggregated by the imbalance of the transponder. Similarly, a classifier based on Gaussian mixture models (GMM) was proposed in [70]. Though, the study showed $\sim$100% accuracy, the experiments were conducted on artificial data. A simulation-based transmitter authentication scheme was proposed in [33] using an "I/Q imbalance" matrix and multiple collaborating receivers. After analyzing these existing DL techniques, it is evident that there is still a lack of systematic "end-to-end" approaches, that can use the raw signal data from real transmitters and exploit the I/Q imbalance for fingerprinting. It must be noted that DL-based RF methods will not only exploit the "I/Q imbalances" but also extract and use other intrinsic features related to the transmitters that may or may not be directly related to I/Q imbalance. However, the end product will conceptually be able to differentiate between transmitters having different "I/Q imbalances."

2.1.5 Consideration of Adversaries

Motivated by the capacity of deep learning systems to automatically learn deep discriminative features, we focus on the problem of transmitter identification in the presence of adversaries using a generative adversarial network (GAN). The idea of training discriminative models via an adversarial process was first proposed by Goodfellow [32]. Since then, GANs have been adopted for solving problems in varied fields of applications and particularly for image processing where GANs have proved to be highly efficient for several different tasks [8,22,110]. We take inspiration from [67] for using the raw I/Q data for input to our networks.

2.2 Use of RNN for Transmitter Fingerprinting

Recurrent neural networks [31] have been used extensively for modeling *temporal data* such as speech [74]. There is limited amount of work that recognizes the potential of using recurrent structures in the RF domain and in general the use of deep learning in the RF domain has been limited in the past with only a few applications in recent times [65, 67].

In [62], O'Shea et. al. presented a recurrent neural network that extracted high level protocol information from the low level physical layer representation for the task of classification. A radio anomaly detection technique was presented in [64], where the authors used a LSTM based RNN as a time series predictor using the error component to detect anomaly from real signals. Another application of RNN was proposed in [83], where the authors used a deep recurrent neural network to learn the time-varying probability distribution of received powers on a channel and used the same to predict the suitability of sharing that channel with other users. A method for modulation classification was proposed in [73] for a distributed wireless spectrum sensing network. The authors proposed a recurrent neural network using long short term memory (LSTM) cell, yielding 90% accuracy on a synthetic dataset [72]. Bai et. al. proposed an end-to-end RF fingerprinting [6] method using two RNNs in order to learn the spatial or temporal pattern. Both simulated and real-world data was used to improve the positioning accuracy and robustness of moving RF devices.

2.3 Existing Defenses against PUE Attack

In this section we discuss the main premise of PUE attack and its properties. We also present different ideas for defending against these attacks and finally we discuss the existing research on these concepts.

PUE attack, first conceptualized and proposed by Chen et al. [15], is a DoS attack for CRNs. In a CRN, the PUs are prioritized over any SU. At the heart of a CRN, a spectrum manager makes the decision of preempting SUs when any PU is in need of transmission. For example, the PUs can be TV stations with a wide range, wireless microphones with a limited range [16], or mobile public safety devices surging with sudden transmissions during the times of emergency [14]. The SUs can be conceptualized as wireless devices connected to a WiFi network.

As per the policies mandated by FCC [27], SUs cannot cause any interference with PU's transmission, or hurt the PU's transmission in any other way. However, the malicious SUs can pretend to be a PU by "mimicking" certain features of PUs. One example is, where a malicious SU emulates itself to be a PU by using a low power commercial off-the-shelf TV transmitter [47] located near some legitimate PUs and starts to transmit on a particular spectrum band. The objective of the emulated primary user (EPU) could be of two types: (i) a selfish goal to maximize the spectrum usage for itself, (b) malicious goal with a tendency to prevent the legitimate SUs from detecting vacant spectrum bands, leading to a DoS attack. The selfish EPU attacker starts transmitting on a vacant frequency band without going into the waiting queue for SU selection by the central spectrum allocator. The malicious EPU attacker starts to "attack" over multiple vacant frequency bands randomly, resulting in starvation of PUs' and legitimate SUs.

Any security threat can be thwarted by several defense mechanisms through continuous research. Though there is still a dearth of opportunities to come up with a robust defense mechanism to overcome all technical challenges related to PUE attacks, there are few existing defense mechanisms which are available today. One such technique is matched filtering-based detection [13] of PUE attacker, which requires specialized hardware and software. On the other hand, energy detection-based schemes [61] poises high risk of missed false alarm and missed detection possibility. The location-based detection technique [15] is limited to stationary PUs with known coordinates. Another approach was the use of phase noise as a signature [105] to identify PUs and defend against

PUE attacks. The key idea behind this technique is to erase modulation from captured signal (which is modulated) and extract phase noise of local oscillator (LO) to work as a signature. This approach is also constrained over the prior knowledge of modulation scheme. However, the use of a cryptographic signature and wireless link signature [50] to detect PUE attacker requires an additional helper node in close physical proximity of each PU. Similarly, a cyclostationary calculation [71] based artificial neural network model needs prior knowledge of modulation schemes of PUs, and it is constrained to be different for the other users.

All the mentioned defense mechanisms are burdened with overheads or constraints of different types. To resolve this, we propose a GAN based robust PUE attack defense (in Chapter 5) which will involve only centralized deployment and can provide defense mechanism for both mobile and stationary PUs or legitimate SUs, without any such constraints.

2.4 Primary User's Activity Prediction

In this section, we discuss some previous important work in this area of primary user's activity prediction modeling that have used machine learning techniques. Various traditional methods of spectrum sensing and their applications have been presented in [103]. The idea of cooperative spectrum sensing was presented as the solution to security related problems in the spectrum sensing domain. Though the concept of estimating spectrum usage in multiple dimensions such as time, frequency and space, was introduced, no prediction attempts were made. In [91], the authors presented opportunistic spectrum access based on a Markovian model that accounted for the bursty nature of ON/OFF traffic models. Through empirical results, the authors have shown that the selection of a channel for the secondary user will depend on the probability distribution of the primary user's traffic as well the elapsed OFF time. This dissertation established the feasibility of using PU usage modeling for designing the spectrum access methods for SU bursty traffic.

There are few existing researches on spectrum sensing and prediction that use machine learning based techniques. An artificial neural network based approach for spectrum sensing in noisy environment was proposed in [93]. The simulation results accomplished better sensing reliability than traditional techniques for signals with low signal-to-noise ratio (SNR). A machine learning based spectrum prediction approach was proposed in [96]. A multi-layer perception (MLP) based neural network was used to predict the primary user's activity, the resulting model was validated through extensive simulations.

In [100], the authors give an analytical overview of various existing methods for spectrum prediction, that are based on: moving average, autoregressive models, hidden Markov models, Bayesian interference and static neighbor graph. It was established that all these models have limited scope for accurate long-term prediction. A deep cooperative sensing scheme was proposed by Lee *et al.* in [46]. A convolutional neural network (CNN) model was proposed to exploit both spectral and spatial correlation of individual sensing outcomes for multiple PUs and SUs. All these studies demonstrate the necessity of a practical long-term spectrum usage prediction method for the PUs that leverage machine learning techniques.

Our focus is on the effectiveness of recurrent neural networks [31] which have been used extensively for modeling *temporal data* such as speech [74]. There is limited amount of work that recognizes the potential of using recurrent structures in the Radio frequency (RF) domain. Though such use of I/Q data for building machine learning systems for communication has been limited in the past, recently it has been used in several applications [57, 65, 67, 79]. RF data (being a time series data) has both spatial and temporal property within the I/Q components. However, to the best of our knowledge there is no recurrent neural network based application which exploits both the spatial and temporal property of the RF data and use that for a long-term prediction model for PU's presence or absence.

CHAPTER 3: ADVERSARIAL LEARNING AND DETECTION

In this chapter we propose the *Radio Frequency Adversarial Learning (RFAL) framework* for building a robust system to identify rogue RF transmitters by designing and implementing a generative adversarial net (GAN). We first demonstrate the use of generative adversarial nets (GAN) to disambiguate trusted transmitters from rogue (fake) ones. After eliminating the rogue transmitters, we use different standard neural network (NN) models to identify (classify) the trusted transmitters based on their radio fingerprints.

We program 8 universal software radio peripheral (USRP) software defined radios (SDRs) as trusted transmitters and collect "over-the-air" raw I/Q data from them using a Realtek Software Defined Radio (RTL-SDR), in a laboratory setting. We also implement a discriminator model that discriminates between the trusted transmitters and the counterfeit ones with 99.9% accuracy and is trained in the GAN framework using data from the generator. Finally, after elimination of the adversarial transmitters, the trusted transmitters are classified using a convolutional neural network (CNN), a fully connected deep neural network (DNN) and a recurrent neural network (RNN) to demonstrate building of an end-to-end robust transmitter identification system with RFAL. Experimental results reveal that the CNN, DNN, and RNN are able to correctly distinguish between the 8 trusted transmitters with 81.6%, 94.6% and 97% accuracy respectively. We also show that better "trusted transmission" classification accuracy is achieved for all three types of neural networks when data from two *different* types of transmitters (different manufacturers) are used rather than when using the same type of transmitter (same manufacturer).

The chapter is organized as follows: Section 3.1 presents some background on feature selection for the proposing RFAL. The GAN architecture along with the generator and the discriminator models are proposed in Section 3.2. The various NN models used are discussed in Section 3.3. We present

the testbed setup and evaluation framework in Section 3.4. The results are presented in Section 3.5. The contents of this chapter appeared in [76, 77, 79].

3.1 Feature Selection

In order for ML techniques to be effective for the task of emitter identification, one must choose an attribute or feature that is unique to a transmitter, irrespective of the signals it transmits. A feature that is commonly used for this task is the "*I/Q imbalance*" that is generated by the random noise introduced into the transmitter manufacturing process due to the use of uncharacterized mixers, oscillators and unbalanced low pass filters [97]. However, building robust learning systems using traditional methods that only uses the "*I/Q imbalance*" is hard due to the underlying characteristics of the RF channel. This when coupled with the presence of active adversaries, renders the use of traditional learning algorithms in RF channels moot.

3.1.1 I/Q Imbalance

The low price of radio devices comes with a trade-off; namely the presence of "almost undetectable" hardware impairments in the radio units due to the use of inexpensive bulk produced commercial off-the-shelf (COTS) components during manufacturing. One such impairment is the imbalance between the in-phase (I) and quadrature (Q) components of the transmitted signal, commonly known as the "I/Q imbalance," that is unique to different radio hardware and are caused by imperfections in local oscillators and mixers. As a result of this, the I and Q components of the modulator are not orthogonal. When a signal is transmitted using a particular radio transmitter having an I/Q imbalance, this is imposed over the complex-valued I/Q data that is being transmitted [44], as shown in Fig. 3.1. The presence of I/Q imbalance leads to performance degradation

for higher order modulations because the symbol rotation becomes more sensitive with increasing number of constellations, towards both the I and Q components [49].

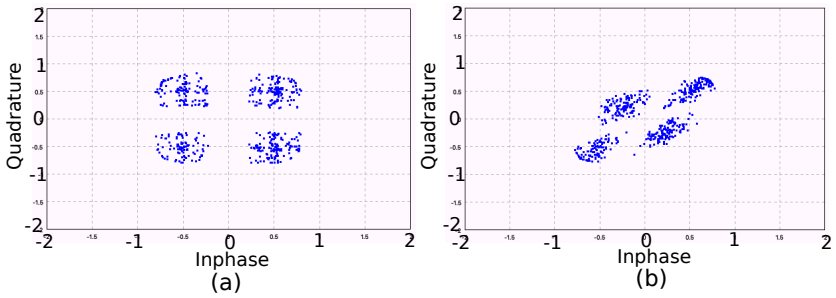


Figure 3.1: I/Q Imbalance for QPSK: (a) Before (b) After 45° Phase Imbalance

The number of resources having information about the I/Q imbalance of real systems is limited. Some prior works [3, 7, 12, 20, 36, 49, 95] on I/Q imbalance estimation and compensation have reported amplitude imbalance test values ranging from 0.02 to 0.82 and phase imbalance test values between 2° and 11.42° [44]. These estimates can in turn be used to compensate for the imbalance. The in-phase and quadrature component of baseband signal is generated from RF signal for signal processing convenience. Ideally, I and Q baseband signals should be orthogonal to each other with same amplitude. However, different channel environment and transmitters configuration poise various numbers of imbalances, e.g. magnitude imbalance, phase imbalance, quadrature offset, inphase offset, DC offset, in the transmitted signal.

Therefore, the transmitted signal, along with different imbalances and impairment from the transmitter radio, can be written as following:

$$I'(t) = \alpha \cos(\omega t) + \beta_I \tag{3.1}$$

$$Q'(t) = \sin(\omega t + \psi) + \beta_Q \tag{3.2}$$

where β_I and β_Q are DC offsets on two baseband component, α and ψ represent amplitude and phase imbalance respectively.

In spite of these techniques for compensation of the imbalance, the fact remains that all transceivers exhibit this *unique* I/Q imbalance. Furthermore, the I/Q imbalance depends on the choice of the hardware components used, and is an unwanted byproduct of the manufacturing process that is hard to imitate. Hence the I/Q imbalance can be used as a basis for feature engineering (implicit or otherwise) for transmitter identification and recognition.

3.2 GAN for Rogue Transmitter Detection

As pointed out earlier, most of the traditional ML techniques are susceptible to malicious attacks. The susceptibility increases once the attacker knows the features used by the learning algorithm. With the knowledge of the feature space (and hence the underlying feature distribution) the attacker becomes smart enough to mislead the learning process. Thus, for the problem of transmitter identification, the target of the adversaries would be to learn the probability distribution of the training data used for model creation, given a sample of the same. With this knowledge, the adversaries can use a *generative model* to generate signals so as to spoof the transmission of known transmitters. A GAN [32] uses a generative model which enables the realistic creation of samples from a given distribution which can then be used to train a discriminator for identifying real samples from false/counterfeited ones obtained from the generator. The GAN training makes the model resilient over the trained adversarial data and thus intuitively helps it to be prepared for the "yet-to-be-seen" adversaries.

3.2.1 Proposed GAN Architecture

The proposed GAN framework, as shown in Fig. 3.2, has two primary components: a generative model ($\mathcal{G}$) that generates false data using a given data distribution and a discriminative model ($\mathcal{D}$) that estimates the probability that a sample came from the training data (that is known transmitters) rather than $\mathcal{G}$. The adversary generates random modulation scheme ($m(t)$), signal amplitude ($r(t)$) and phase ($l(t)$) and mixes additive white Gaussian noise (AWGN) ($n(t)$) with the false signal. The generated signal ($g(t)$) which is initially random in nature improves over time as the generator learns from the discriminator and improves on its ability to imitate real data. On the other hand, the discriminator ($\mathcal{D}$) gets input from both the generator ($\mathcal{G}$) and `Trusted` transmitters. This helps it to learn to differentiate between real and false inputs. The known transmitter data is collected and fed to the discriminator ($\mathcal{D}$) as raw I/Q values.

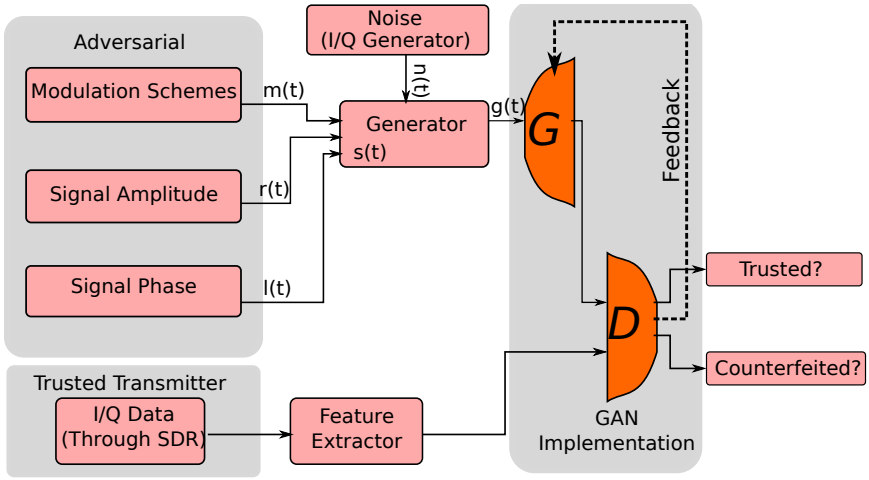


Figure 3.2: Proposed RFAL GAN architecture

Overall, our objective is to train $\mathcal{G}$ in such a way that will maximize the probability of $\mathcal{D}$ making a mistake. $\mathcal{G}$ tunes its hyper parameters with the feedback from $\mathcal{D}$. We argue that GAN is an efficient

way to generate correlated data samples and thereby approximate an accurate generative model, something the adversary aims to achieve. Once the model is trained, RFAL synthesizes signals using the generator to mimic adversarial transmitters based on the learned probability distribution on the sample space of I/Q signal data from the known transmitters.

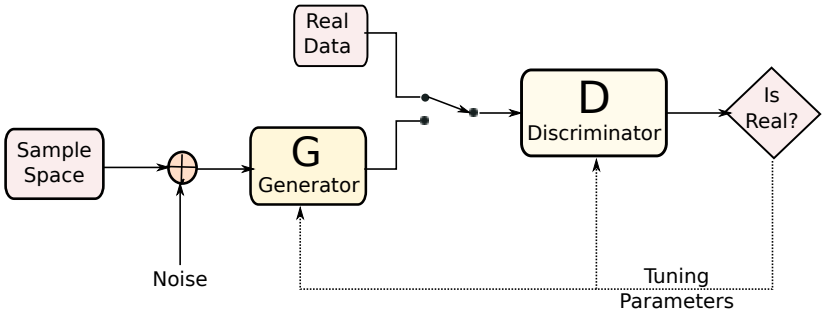


Figure 3.3: A Simplified View of GAN Implementation

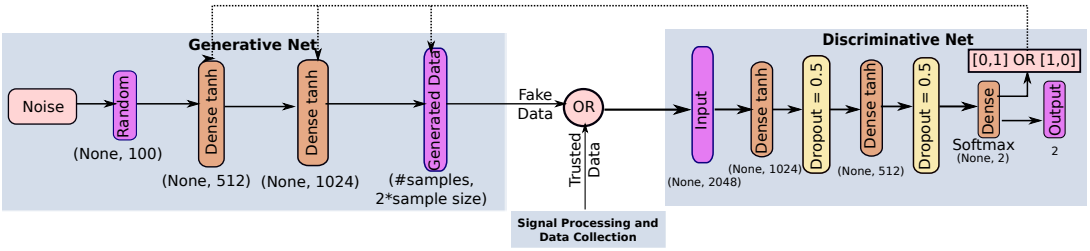


Figure 3.4: GAN Implementation for Rogue Transmitter Detection

3.2.2 The Generative Model

In order to build the generator, we treat the overall problem as an N-class decision problem where the input is a complex base-band time series representation of the received signal. That is, the training data consists of the in-phase and quadrature components of a radio signal obtained at discrete time intervals through analog to digital conversion with a carrier frequency to obtain a

$1 \times N$ complex valued vector. Classically, this is written as:

$$s(t) = c_1 m(t) + c_2 r(t) + c_3 l(t) \tag{3.3}$$

where $s(t)$ is a continuous time series signal modulated onto a sinusoid with either varying frequency, phase, amplitude, trajectory, or some permutation of these parameters. Here, $m(t)$, $r(t)$, and $l(t)$ are the time series continuous signals for modulation, amplitude, and phase respectively, selected randomly by the generator. The coefficients c_1, c_2, and c_3 are some path loss or constant gain terms associated with $m(t)$, $r(t)$, and $l(t)$ respectively. The output $g(t)$ is obtained as:

$$g(t) = s(t) + n(t) \tag{3.4}$$

where $n(t)$ is the AWGN. The output $g(t)$ is then fed to a generator which is used as an unsupervised learning tool as a part of the GAN framework. The generator learns the probability distribution $p_g(\mathbf{x})$ over sample space ($\mathbf{x}$) of the I/Q input.

3.2.3 The Discriminative Model

The discriminative model learns by minimizing a cost function during training. The cost function, $C(\mathcal{G}; \mathcal{D})$, depends on both the generator ($\mathcal{G}$) and the discriminator ($\mathcal{D}$). It is formulated as $C(\mathcal{G}; \mathcal{D})$ = $\mathbb{E}_{p_{data}(\mathbf{x})} \log \mathcal{D}(\mathbf{x})$ + $\mathbb{E}_{p_g(\mathbf{x})} \log(1 - \mathcal{D}(\mathbf{x}))$, where $p_g(\mathbf{x})$ is the generator's distribution over $\mathbf{x}$, $p_{data}(\mathbf{x})$ is the data distribution over $\mathbf{x}$, $\mathcal{D}(\mathbf{x})$ is the probability that $\mathbf{x}$ came from $p_{data}(\mathbf{x})$ than $p_g(\mathbf{x})$ [32]. The training is formulated as:

$$\max_{\mathcal{D}} \min_{\mathcal{G}} C(\mathcal{G}; \mathcal{D}) \tag{3.5}$$

For the GAN framework, there is an unique optimal discriminator for a fixed generator, $\mathcal{D}^*(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(x) + p_g(\mathbf{x})}$ [32]. It is also inferred that $\mathcal{G}$ is optimal when $p_g(\mathbf{x}) = p_{data}(\mathbf{x})$, i.e., the generator is optimal when the discriminator cannot distinguish real samples from false ones. Similarly, the $\mathcal{D}$ is optimal when the discriminator can recognize each real sample from the false ones.

3.2.4 GAN Implementation

For implementing the GAN, we use "over-the-air" signal data collected from the trusted transmitters. (The testbed setup is discussed in Section 3.4.) The generator ($\mathcal{G}$) generates counterfeit data from the same sample space to impersonate a trusted transmitter. True and counterfeit I/Q data are fed to the discriminator ($\mathcal{D}$) with equal probability. We design the discriminator and generator separately, as shown in Fig. 3.3. The overall GAN implementation is shown in Fig. 3.4.

The generator starts by generating random data within $(-\infty, +\infty)$. As the training evolves, the generator learns that the sample space of real data is $[-1, 1]$. Thus, the generator will gradually generate I/Q values within $[-1, 1]$ in turn reducing the parameter space. The time-invariant features being automatically learned by the discriminator should implicitly capture the inherent imbalance within the I/Q data. The generator gradually learns the real data distribution over multiple training epochs and starts to replicate the I/Q imbalances in the generated I/Q values.

Once the initial random values are generated, they are passed through two *dense* layers of size 512 and 1024 respectively with *tanh* [11] activation. Then a single *dense* layer of twice the sample size (2048 in our case) is invoked with the *sigmoid* activation function [11]. $\mathcal{G}$ continues to learn the data distribution (p_g) and generates counterfeit samples of size 2048 within the sample space of the I/Q values.

$\mathcal{D}$ consists of one input layer of 2048 nodes, two hidden layers of 1024 and 512 nodes respectively,

and finally a *softmax* output layer of 2 nodes to classify an input as either `Counterfeited` or `Trusted`. We used *tanh* as activation function at the hidden layers and added *Dropout* [89] of 0.5 in between these layers for regularization. We train both the generator and discriminator through iterative sequential learning to strengthen the generative model over time.

Once the discriminator recognizes the trusted transmitters from the counterfeit ones, we feed the trusted transmitter data into another DNN (either convolutional or fully connected (dense)) for further classification into a number of classes (as determined by the number of trusted transmitters). Next we discuss the NN architectures we use for this purpose.

3.3 Proposed Neural Networks for Transmitter Classification

Recent advances in NNs have made it possible to obtain robust models with low generalization errors by training "deep" neural architectures efficiently. The "depth" signifies the number of iterative operations performed on the input data using each layer's transfer function and deeper architectures allow the network to learn more robust feature representations from the input data. Though such techniques require higher computation power and involve complicated layer-by-layer back-propagation, nevertheless, most DL systems are able to efficiently learn deep feature representations from the training data using back-propagation and some variation of gradient decent, with adaptive learning rates (e.g., *Adam* [42]) and regularization to avoid overfitting (e.g., *Dropout* [89]). Next, we present our proposed NN models for classification of `Trusted` transmitters. We use a total of 8 transmitters, the details of which are given in Section 3.4.

3.3.1 CNN Model

In order to capture the correlation between I/Q values, we started by implementing a convolutional neural network (CNN). We implement the CNN with three *conv2D* layers with 1024, 512 and 256 filters respectively, a *Flatten* operation and three fully connected (*FC*) layers of size 512, 256 and 8 nodes [18] respectively, as shown in Fig. 3.5. We use *Dropout* [89] of 0.25 and 0.5 after each *conv2D* and *dense* layer respectively. We also use kernel size of (2,3) and stride of (2,2) at each of the *Conv2D* layers. We apply a pooling layer *MaxPooling2D* after each *conv2D* layer with pool size of (2,2) and stride of (2,2). We use *ReLU* [58] activation for all convolution and fully connected layers, other than the last, where we use *softmax*. Note that, the number of nodes in the last layer is changed based on the number of classes the dataset has. We use *Adam* [42] (with learning rate of 10^{-3}) based optimization with categorical cross-entropy training.

It is to be noted that we design the CNN with only 3 convolution layers and 4 fully connected layers for faster training [34], since no significant increase in the testing accuracy was observed after increasing the number of layers.

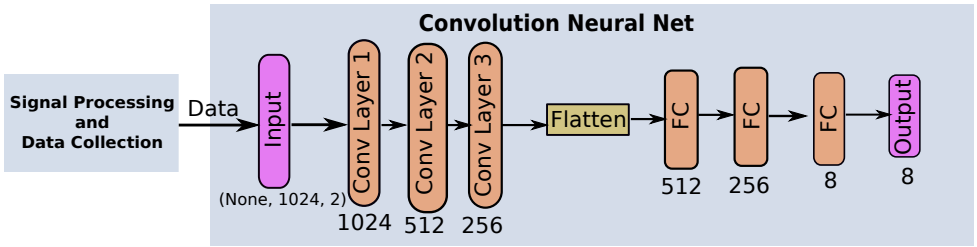


Figure 3.5: CNN Implementation for Transmitter Classification

3.3.2 DNN Model

Apart from the CNN discussed above, we also use a fully connected (dense) DNN for the task of trusted transmitter classification. The implementation of the DNN is similar to the discriminator model of GAN and is shown in Fig. 3.6. The only difference is that the *softmax* output layer has 8 nodes to recognize the 8 transmitters (or more generally k nodes if there are k transmitters). We implement a DNN with 5-layers with 1 input layer and 4 dense layers. We use *tanh* [11] activation function for the *Dense* layers in this model. We apply biases and regularization to avoid under- and over-fitting. In this case also, use *Adam* [42] based optimization with learning rate of 10^{-3}, for categorical cross-entropy training.

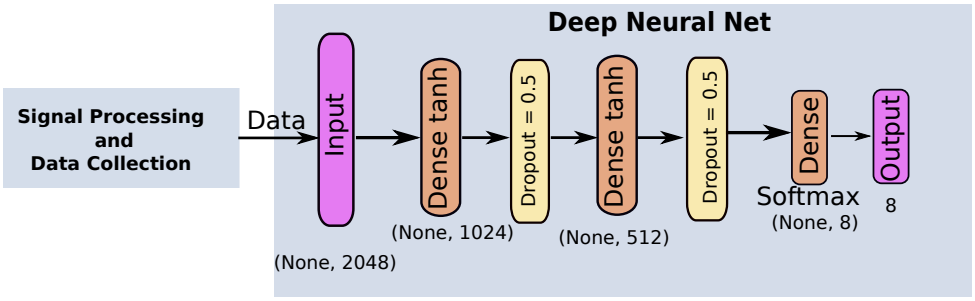


Figure 3.6: DNN Implementation for Transmitter Classification

3.3.3 Recurrent Neural Network (RNN)

In order to exploit the the time-series property of the collected data, we use RNNs with LSTM and Gated Recurrent Unit (GRU) cells, as both avoid the "vanishing" or the "exploding" gradient problems [24].

3.3.3.1 Long Short Term Memory (LSTM) Cell Model

Though LSTM cells can be modeled and designed in various ways depending on the need, we implement the cells as shown in Fig. 3.7. In each LSTM cell, there are (i) three types of gates: input (i), forget (f) and output (o); and (ii) a state update of internal cell memory. The most interesting part of the LSTM cell is the "forget" gate, which at time t is denoted by f_t. The forget gates decide whether to keep a cell state memory (c_t) or not. The forget gates are designed as per the equation (3.6) on the input value of x_t at time t and output (h_{t-1}) at time $(t-1)$.

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{3.6}$$

where, σ denotes the *sigmoid* activation function, W_{xf} and b_f represent the associated weight and bias respectively, between the input (x) and the forget gate (f). Once f_t determines which memories to forget, the input gates (i_t) decide which cell states ($\widetilde{c_t}$) to update as per equations (3.7) and (3.8).

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{3.7}$$

$$\widetilde{c_t} = tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_{c_{t-1}}) \tag{3.8}$$

In equation (3.9), the old cell state (c_{t-1}) is updated to the new cell state (c_t) using forget gates (f_t) and input gates (i_t):

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c_t} \tag{3.9}$$

where, $\circ$ is the Hadamard product. Finally, RFAL filters the output values through output gates (o_t) based on the cell states (c_t) as per equations (3.10) and (3.11).

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_t + b_o) \tag{3.10}$$

$$h_t = o_t \circ tanh(c_t) \tag{3.11}$$

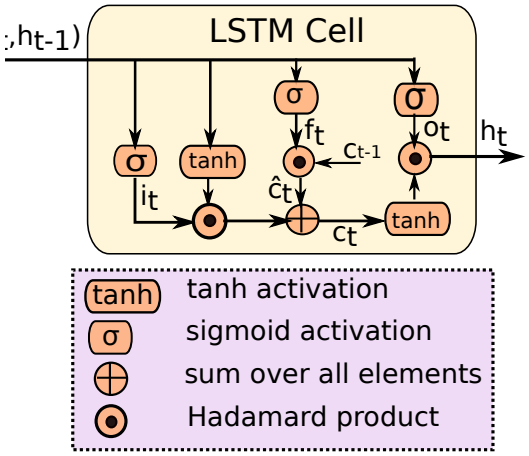


Figure 3.7: LSTM Cell Architecture Used in the RNN Model

We design and implement a RNN with LSTM, the structure of which is shown in Fig. 3.8. We use 2 LSTM layers with 1024 and 256 units sequentially. Next we add 2 fully connected layers with 512 and 256 nodes respectively. We use *batch normalization* on the output and pass it through a *dense* layer of 8 nodes. We use *ReLU* [58] as activation function for the LSTM layers and *tanh* [11] as activation for the *dense* layers. Lastly, we use *stochastic gradient descent (SGD)* [11] (with learning rate of 10^{-3}) based optimization with categorical cross-entropy training.

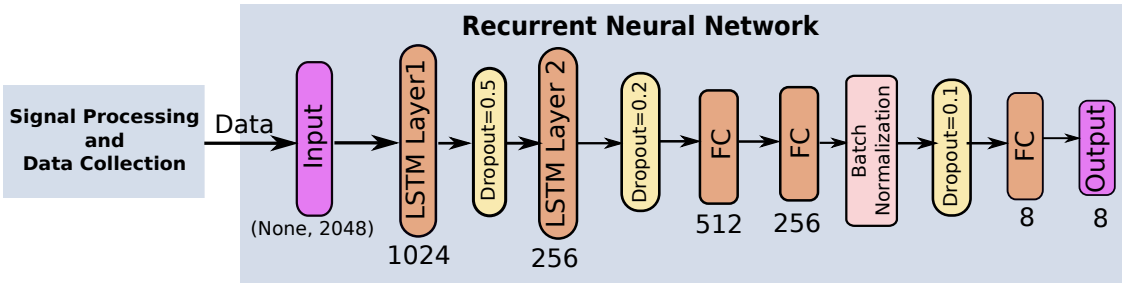


Figure 3.8: RNN Implementation for Transmitter Classification

3.3.3.2 Gated Recurrent Unit (GRU) Model

The main drawback of using LSTM cells is the need for additional memory. GRUs [19] have one less gate than LSTMs for the same purpose, thus having a reduced memory and CPU footprint. The GRU cells control the flow of information just like the LSTM cells, but without the need for a memory unit. It just exposes the full hidden content without any control. It has a "reset gate" (z_t), an "update gate" (r_t), and a cell state memory (c_t) as shown in Fig. 3.9. The reset gates determine whether to combine the new input with a cell state memory (c_t) or not. The update gate decides how much of c_t to retain. The equations (3.12), (3.13), (3.14), and (3.15) related to different gates and states of GRU are given below.

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \tag{3.12}$$

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \tag{3.13}$$

$$c_t = tanh(W_{xc}x_t + W_{hc}(r_t \circ h_{t-1})) \tag{3.14}$$

$$h_t = (1 - z_t) \circ c_t + z_t \circ h_{t-1} \tag{3.15}$$

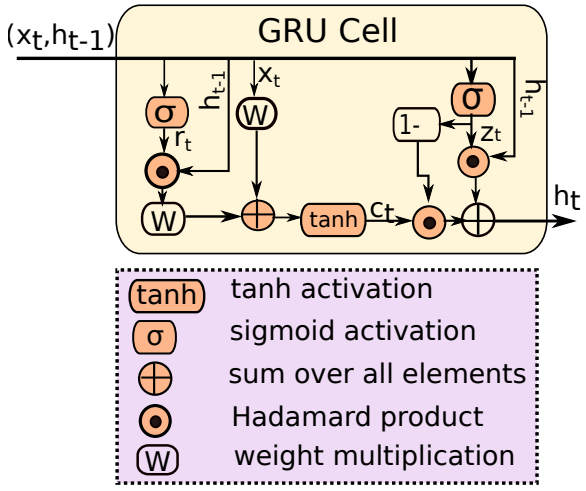


Figure 3.9: GRU Cell Architecture Used in the RNN Model

We implemented the recurrent model with GRU cells and used the same architecture as the LSTM implementation (GRU cells instead of LSTM cells at the first two layers). The proposed GRU network needs fewer parameters than the LSTM model. In this case, we also use *SGD* [11] based optimizer with a learning rate of 10^{-3}. An quantitative comparison of the results is discussed in Section 3.5.6.

3.4 Testbed Setup and Evaluation

In order to validate the proposed models, we wanted to use our own "over-the-air" RF data collected from different transmitters, instead of synthetic or publicly available data. We implemented

the generator and discriminator (of the GAN) to detect an unknown (adversarial) transmitter and then used the NN models to classify the known ones, the details of which are discussed next.

3.4.1 Signal Generation and Data Collection

In order to learn the discriminating fingerprints (features) of *similar* transmitters, we used 8 universal software radio peripheral (USRP) radios of the same kind, namely B210 from Ettus Research [26]. The overall setup for signal generation and reception is shown in Fig. 3.10. The B210s were programmed to transmit random data on 904 MHz using Quadrature Phase Shift Keying (QPSK) modulation. We used GNURadio [29] for signal processing and data transmission. The flow graph is presented in Fig. 3.11. The modulated signal was transmitted through the USRP sink block. For the receiver, we used a RTL-SDR [60] which captured "over-the-air" raw I/Q data and wrote it onto a file.

3.4.2 Analysis of Data Collection Environment

We set up the data collection testbed in an indoor lab environment with a direct line of sight between the transmitter and the receiver with a distance of 10 ft. Thus, the underlying channel can be modeled as a Rician fading channel. There was also multi-path effects due to the reflections from the walls. We measured the signal to noise ratio (SNR) using a RTL-SDR [60] dongle and Spekrtum [69] which is an open source spectrum analyzer available for both Windows and Linux. We decided to calibrate the SNR using the *Spekrtum* software (rather than using a spectrum analyzer) due to cost and portability issues. We found that the noise floor in the lab was between -20 dB and -30 dB. The signal strength for the 200 KHz (from 903.9 MHz to 904.1 MHz) channel was between 0 dB and 10 dB. We set the transmitter gain to 45 dB and calculated the SNR as the difference between the noise floor and the signal strength. Our calculated SNR was 5 dB - (-25 dB)

= 30 dB, with a 45 dB transmitter gain. It is to be noted that the signal strengths (in dB) of noise and signal measured by the Spektrum is relative, but the difference between them is absolute.

We also collected data for different SNR values by varying the transmitter-receiver distance and hindering the line-of-sight in the laboratory. As mentioned earlier, we obtained a SNR of 30 dB by keeping the transmitter and receiver at 10 feet from each other. Similarly we collect 3 more datasets with SNR of 20 dB, 10 dB and 0 dB at 20 feet, 30 feet, and 45 feet respectively. It is to be noted that the SNR of the transmitter frequency was measured at the receiver with Spektrum.

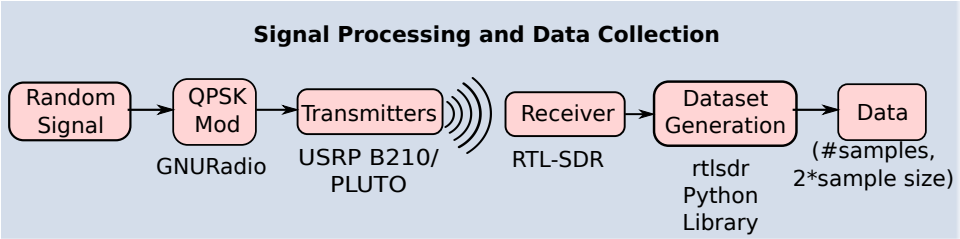


Figure 3.10: Signal Generation and Data Collection Setup

3.4.3 I/Q Datasets

We collected raw I/Q signal data with a sample size of 1024, i.e., each sample consists of 1024 I and 1024 Q values. The choice of 1024 as the sample size was sufficient to capture the unique pattern of I/Q imbalances and at the same time it was not computationally expensive. We experimented with other sample sizes as well: smaller sample size yields degraded performance whereas larger sample size does not improve the accuracy. We collected 40,000 training samples from each transmitter to avoid the data skewness problem observed in ML. The configuration parameters used are given in Table 3.1. We collected two different datasets at 30 dB SNR and three datasets with three different SNRs, as discussed below.

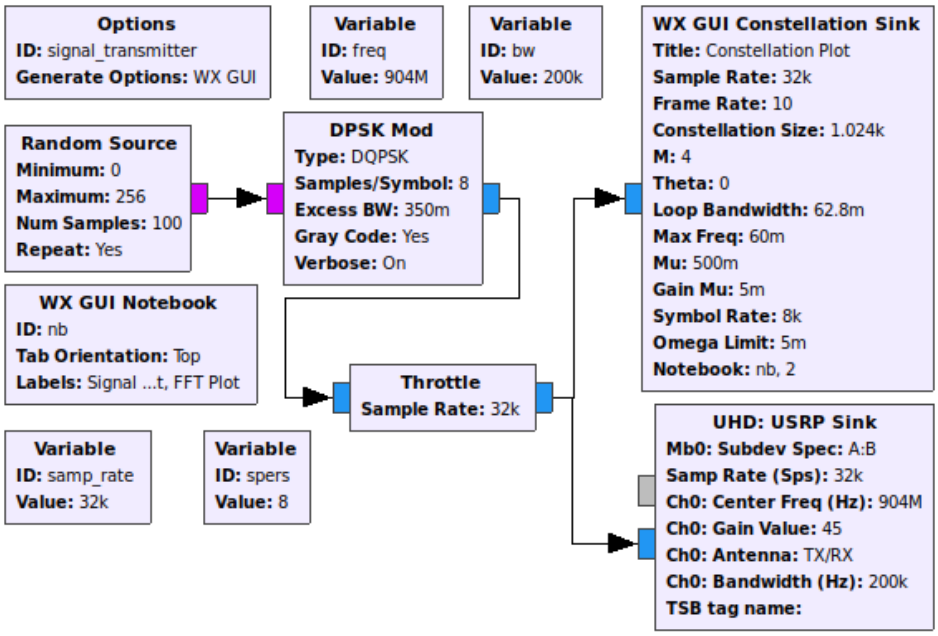


Figure 3.11: GNU Radio Flow Graph for Data Collection for USRPs

Table 3.1: Transmission Configuration Parameters

Parameters	Values
Transmitter Gain	45 dB
Transmitter Frequency	904 MHz (ISM)
Bandwidth	200 KHz
Sample Size	1024
Samples/Transmitter	40,000
# Transmitters	2 to 8

3.4.3.1 Homogeneous Dataset

For the "homogeneous" dataset, we use only one type of radio, namely, the USRP B210 from Ettus Research. We collected two sets of data: (i) using 4 USRP B210 transmitters: 6.8 GB size, 160K rows and 2048 columns and (ii) using 8 USRP B210 transmitters: 13.45 GB size, 320K rows and 2048 columns. Note that the SNR was 30 db.

3.4.3.2 Heterogeneous Dataset

In order to investigate the performance of the proposed classification methods, when different types of transmitters (from different manufacturers) are present, we use PLUTO SDR [23] apart from the B210s when collecting the data. The GNURadio flow graph for signal generation is similar to Fig. 3.11 with a different sink block for the PLUTO SDR. Note that the SNR in this case was also 30 db. The 'heterogeneous' datasets were obtained using (i) 2 USRP transmitters: 3.31 GB size, 80K rows and 2048 columns and (ii) 1 USRP B210 and 1 PLUTO transmitter: 2.85 GB size, 80K rows and 2048 columns.

3.4.3.3 Varying SNR Datasets

We collected 3 more datasets with 8 USRP B210 transmitters and SNRs of 20 dB, 10 dB, and 0 dB respectively. Each dataset is of size $\sim$13 GB with 320K rows and 2048 columns.

3.4.4 Correlation in Dataset

Correlation between each data sample plays a crucial role in transmitter classification. Given T training samples (for T timestamps) and a sample size of M for each time stamp, where each sample is a vector $(I, Q) \in \mathcal{C}$ representing a number in the complex plane, we create a new vector $X(t) = [I_i, Q_i; i = 1, 2, \cdots, M]^t \in \mathcal{C}^2 M; t = 1, 2, \cdots, T$ for timestamp t, and use it as an input to the NN. As mentioned before in our case ($M = 1024$). Thus we represent the I and Q values of each training sample at timestamp (t) as: $[I_0 Q_0 I_1 Q_1 I_2 Q_2 I_3 Q_3 I_4 Q_4 \cdots I_{1023} Q_{1023}]^t$. We used QPSK modulation [88] , which generates a constellation plot like Fig. 3.1. This signifies that the correlation should be between every fourth value, i.e., between I_0 and I_4, and Q_0 and Q_4 and so on. Hence we calculate the correlation coefficient of between $I_0 I_1 I_2 I_3$, $I_4 I_5 I_6 I_7$ and similarly, between

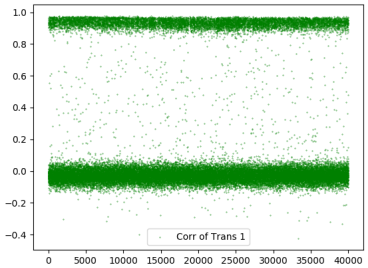
$Q_0Q_1Q_2Q_3$ and $Q_4Q_5Q_6Q_7$ and so on. We take the average of all the correlation coefficients for each sample.

We use *numpy.corrcoef* for this purpose which uses Pearson product-moment correlation coefficients, denoted by r. The Pearson's method for a sample is given by:

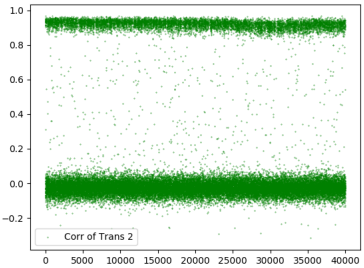
$$r = \frac{\sum_{i=0}^{(M-1)} (I_i - \overline{I})(Q_i - \bar{Q})}{\sqrt{\sum_{i=1}^{(M-1)} (I_i - \overline{I})^2} \sqrt{\sum_{i=0}^{(M-1)} (Q_i - \bar{Q})^2}} \tag{3.16}$$

where, M is the sample size, I_i and Q_i are the sample values indexed with i. The sample mean is $\bar{I} = \frac{1}{M} \sum_{i=0}^{(M-1)} I_i$ and similarly for the Q values.

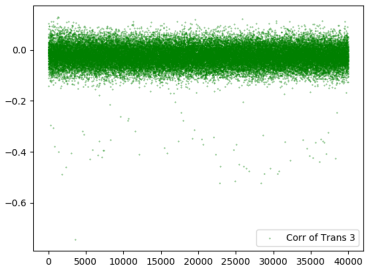
The correlations for all the 40,000 samples for each transmitter are shown in Fig. 3.12. We observed that around 75% of the samples' correlation coefficients are between -0.1 and 0.1 and the remaining 25% are close to 0.9. However, for transmitter ID 3, all the samples' correlation coefficients are between -0.1 and 0.1. This behavior of transmitter ID 3 is an obvious example of manufacturing differences. However, this observation gives us intuition about the difficulty of CNN with 2D convolutions for the task of transmitter classification. CNNs capture the correlation (local features) in the data. However, in this case, as the nature of the correlations are the same for all but one transmitter, hence there is not enough *discriminative* information in the correlations to disambiguate between all the transmitters, thus leading to the conclusion that CNNs with simple two dimensional convolutions will not be effective for the classification task. This was later corroborated via our testbed implementation (Section 3.5).



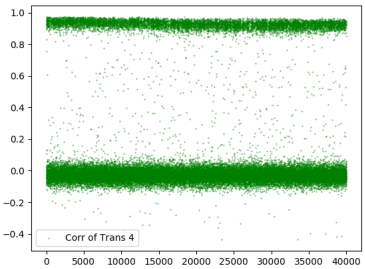
(a) Trans ID 1



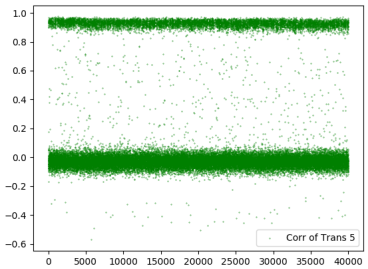
(b) Trans ID 2



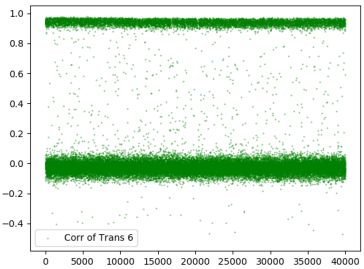
(c) Trans ID 3



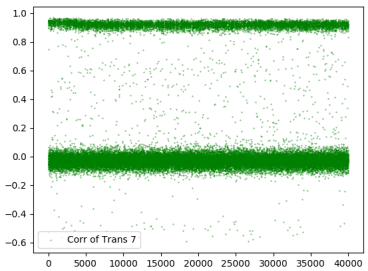
(d) Trans ID 4



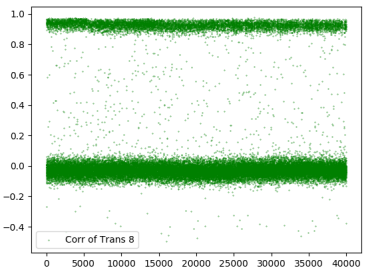
(e) Trans ID 5



(f) Trans ID 6



(g) Trans ID 7



(h) Trans ID 8

Figure 3.12: Correlation Plot for Different Transmitters in the Collected Dataset

3.4.5 Machine Learning Libraries Used

There are several libraries and tools that implement learning frameworks with support for immensely concurrent GPU architectures, that reduce the burden of programming the traditional GPU routines for training of larger NNs. We use *Keras* [18] as the frontend with *Tensorflow* [1] as the backend. *Keras* is an overlay on neural network primitives with Tensorflow [1] or Theano [2] that provides a customizable interface for quick deployment of complex NNs. We also use *Numpy*, *Scipy*, and *Matplotlib* Python libraries.

3.4.6 Performance Metric

To measure the effectiveness of any NN architecture, "classification accuracy" is used as the typical performance metric. However, "classification accuracy" as it is defined can sometimes be misleading and incomplete when the data is skewed. A confusion matrix overcomes this problem by showing how the classification model performs when it comes to erroneous detections (false alarms), and correct "counterfeit" classifications. It provides more insights on the performance by identifying not only the number of errors, but more importantly the types of errors. As a result we use confusion matrices to display and analyze the results of our experiments.

3.5 Implementation Results and Discussions

In this section, we present the results of i) adversarial transmitter detection using GAN and ii) transmitter classification using different NN architectures. We conducted the experiments on a Ryzen 8 Core system with 64 GB RAM and a GTX 1080 Ti GPU unit with 11 GB memory. For the sake of being robust and statistically significant, we present the experimental results for each

model after several runs of each implementation. We focused on four main aspects:

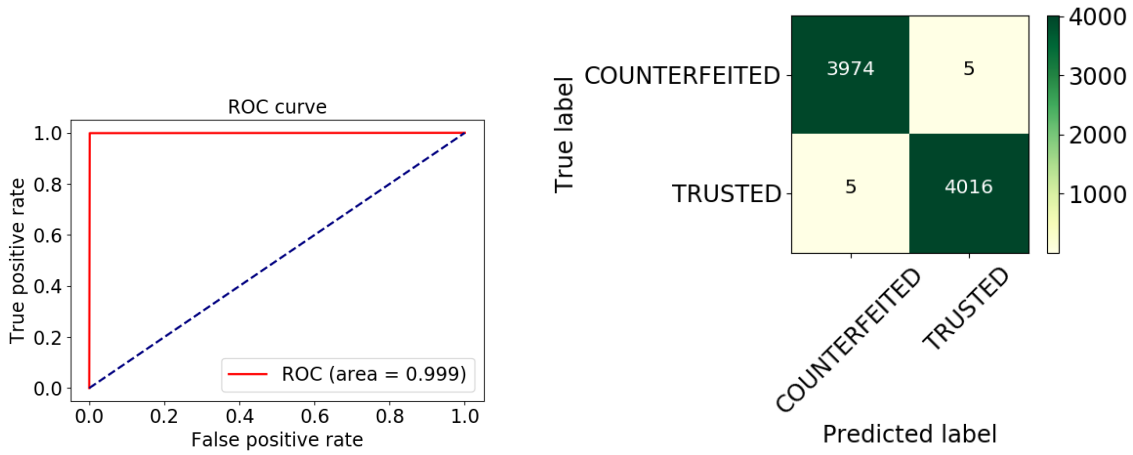
- Implementing a GAN to distinguish adversarial transmitters from trusted ones.
- Implementing a CNN with 2D convolutions to exploit the correlation in collected signal data of the trusted transmitters for trusted transmitter identification.
- Implementing a DNN to classify the trusted transmitters.
- Implementing RNN with both LSTM and GRU cells to improve the accuracy of trusted transmitter classification by exploiting the temporal aspect of the signal data.

3.5.1 GAN Results

In order to detect the adversarial transmitters, we implemented a GAN based model as described in Section 3.2.4. We used categorical cross-entropy training and *Adam* [42] for gradient based optimization. We notice that the discriminator was able to detect the adversarial transmitters with 50% accuracy before the GAN based training. Once the GAN goes through several epochs (< 50) of adversarial training, the optimal discriminator ($\mathcal{D}^*$) is able to detect the `Counterfeit` transmissions with about 99.9% accuracy as shown by the receiver operating characteristic (ROC) curve and confusion matrix in Fig. 3.13. Note that one epoch consists of a forward pass and a backward pass through the GAN over the entire dataset. It is clear from the confusion matrix that the number of false negatives and false positives are very low and well within an acceptable range [11]. The testing accuracy of the GAN implementation on datasets with different SNRs is presented in Table 3.2 . The "parameters" represent the total number of hyper-parameters required for the respective model.

Table 3.2: Accuracy of GAN for 4 and 8 Transmitters

Dataset	SNR	#Trans	#Parameters	Acc (%)
6.8 GB	30	4	3.6 M ($\mathcal{G}$) 6.8 M ($\mathcal{D}$) 10.4 M (GAN)	99.9
13.45 GB	30	8	3.6 M ($\mathcal{G}$) 6.8 M ($\mathcal{D}$) 10.4 M (GAN)	99.9
13.45 GB	0	8	3.6 M ($\mathcal{G}$) 6.8 M ($\mathcal{D}$) 10.4 M (GAN)	99.9

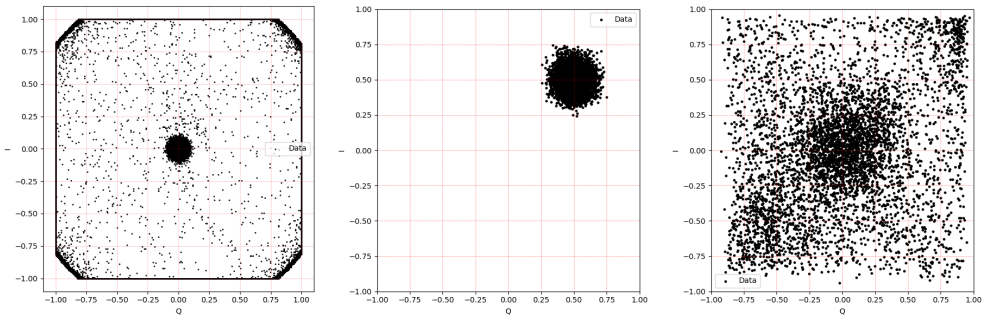


(a) ROC Curve (b) Confusion Matrix

Figure 3.13: ROC Curve and Confusion Matrix of Counterfeit Transmitter Detection from RFAL

In Fig. 3.14, we present three plots to represent how the proposed generator behaves. Note that we first show the plots using randomly selected 128 samples from both the real and generated datasets. We choose the number 128 as it is also used as the batch size for the training. In Fig. 3.14(b), we see that the data generated before the GAN training does not represent the distribution of the real data as shown in Fig. 3.14(a). It is evident that initially the I/Q values are randomly generated between 0 and 1. However, once the generator is trained over multiple iterations, it starts generating more

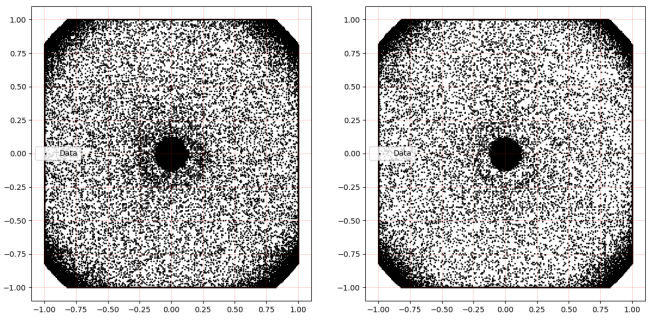
realistic data as shown in Fig. 3.14(b) which shows that the distribution of the generated data is starting to resemble the real data shown in Fig. 3.14(a). Once the GAN training converges, the generator has learned the data distribution of the I/Q values from the known transmitters and hence it starts to generate counterfeit data that succinctly captures the actual distribution of the I/Q values. Thus now the generated I/Q values are distributed within the range of [-1,1] as is the case with the real data. Figure 3.15 shows the plots for the real data and the generated data after *full* GAN training. These images are obtained by plotting 2000 I/Q samples. Also we do not discuss the theory of why the generator is able to generate realistic data because though there is some idea in the research community as to why GANs work, it is not fully understood yet. Furthermore, since this is a more applied work we refrain from adding such theory into the dissertation. Rather we provide references which interested readers may consult to learn more about the GANs.



(a) Real Data (b) Generated Data before GAN Training (c) Generated Data after GAN Training

Figure 3.14: Output from the Proposed Generator (Plot for 128 samples)

Note that though this chapter is not about trusted transmitter classification (but rather about using adversarial learning using GANs for identifying rogue transmitters), we decided to explore the capability of systems based on neural networks for the task of trusted transmitter classification as well. Next we present the results of this endeavor.

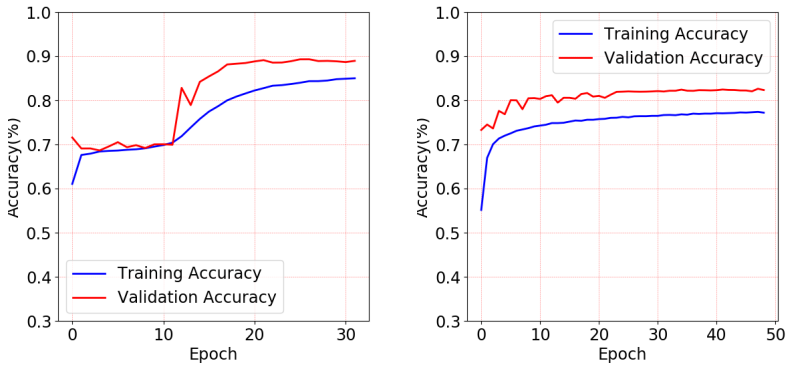


(a) Real Data

(b) Generated Data after Full GAN Training

Figure 3.15: Final Output from the Proposed Generator (2000 samples)

3.5.2 CNN Results



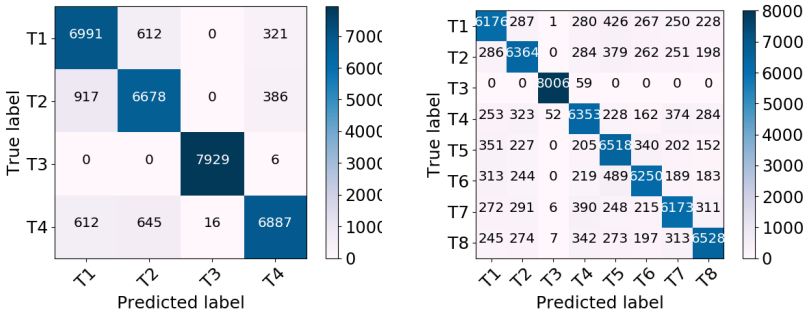
(a) 4 Transmitters

(b) 8 Transmitters

Figure 3.16: Accuracy Plots for Transmitter Classification using CNN

Once the rogue transmitters are detected and eliminated, our system classifies the "trusted" transmitters using a neural network. First we used a CNN built according to the implementation details provided in Section 3.3.1 for trusted transmitter classification. We obtain 89.07% and 81.6% accuracy for 4 and 8 transmitter classifications respectively. The accuracy plots and confusion matrices for both the cases are presented in Figs. 3.16 and 3.17. We note that both the training and validation

accuracy increases with the number of epochs. However for our CNN implementation the number of false positives and false negatives are somewhat high. Intuitively, this shows that the convolutional filters that were used with the network were not able to identify and encode discriminative features for this task. Since we know that there is at least one discriminative characteristic that distinguishes between the transmitters (namely the I/Q imbalance), we conclude that the input representation that we used with the CNN did not effectively encode these characteristics and hence the system could not efficiently learn them.

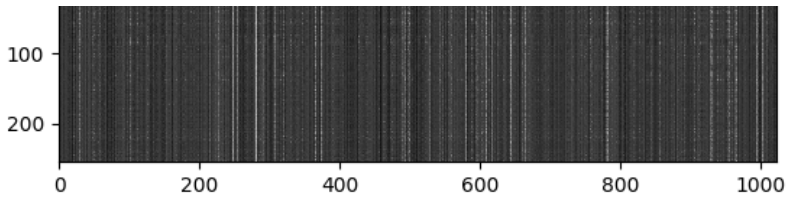


(a) 4 Transmitters (b) 8 Transmitters

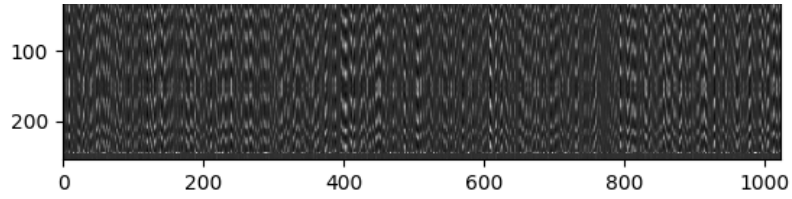
Figure 3.17: Confusion Matrices for Transmitter Classification using CNN

In order to understand the inner workings of the CNN better, we present the feature maps obtained from the first convolution layer of the CNN, for the case of four transmitter classification, in Fig. 3.18. It can be seen that each of these feature maps encodes a different pattern, one for each of the four different transmitters. However, we also notice that the convolutions fail to capture the highly discriminative patterns from the I/Q data. Thus for example, none of the feature maps resemble feature characteristic of the data shown in 3.15(a). Thus even though the features are different for each transmitter, they do not fully encode discriminative features present in the I/Q samples. Intuitively this is because of the fact that the I/Q samples does not have significant spatial correlation that can be leveraged through the use of the convolution operation. Since further tuning of the CNN parameters was unsuccessful, we decided to build a fully connect DNN to try and

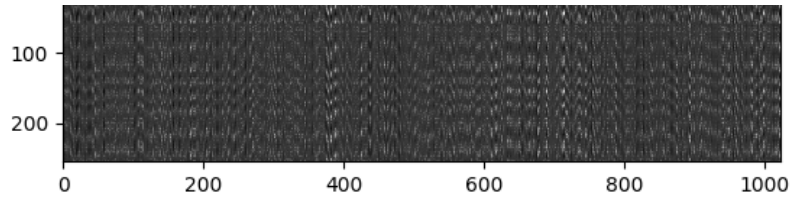
achieve better accuracy for this task.



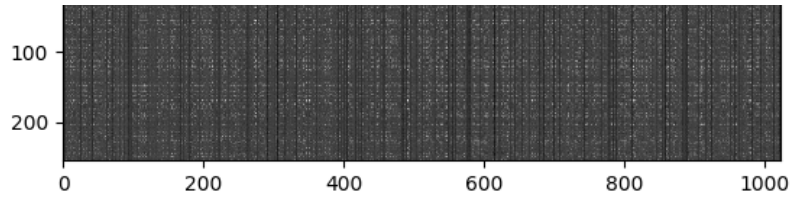
(a) Radio 1



(b) Radio 2



(c) Radio 3



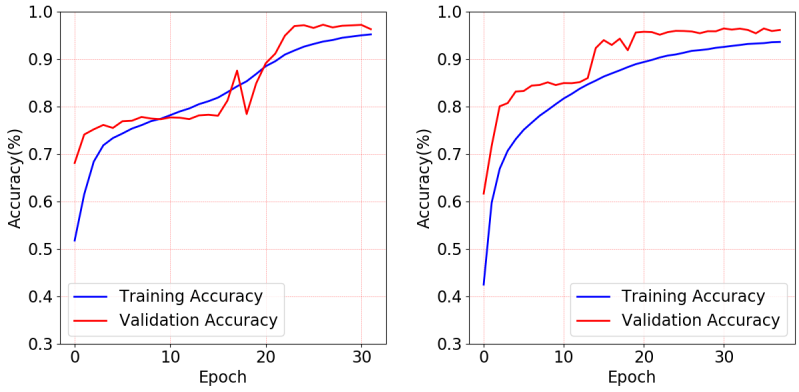
(d) Radio 4

Figure 3.18: Feature Maps for the First Convolution Layer of the Proposed CNN Model

3.5.3 DNN Results

To overcome the deficiencies of the CNN, we use a DNN as described in Section 3.3.2. The DNN yields an accuracy of 96.49% for 4 transmitters and 94.60% for 8 transmitters. The accuracy

plots and confusion matrices are shown in Figs. 3.19 and 3.20 respectively. It is evident that the number of false positives and false negatives in the confusion matrices are significantly low for the DNN as compared to the CNN and thus intuitively the DNN can capture and learn better features from the I/Q samples than the CNN for the task of transmitter classification. Note that from the perspective of the features learned by the DNN, it is hard to explain the types of features that are learned by these systems. However since the task of classification boils down to learning a *decision boundary* and recently it has been shown that DNNs are capable of learning approximations of such boundaries efficiently [48], the possible reason for the better performance of the DNN might be tied to learning a better approximation to the underlying function representing the decision boundary for this task.

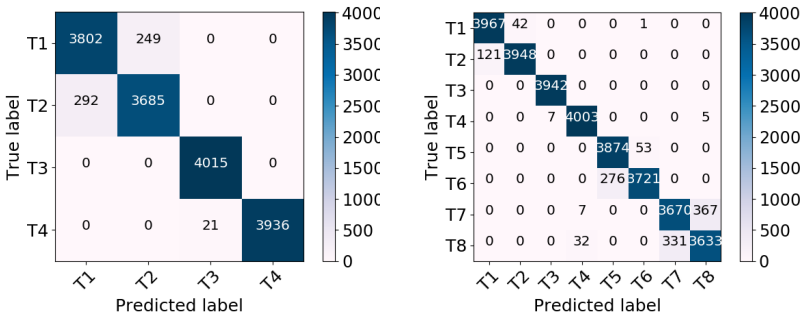


(a) 4 Transmitters (b) 8 Transmitters

Figure 3.19: Accuracy Plots for Transmitter Classification using DNN

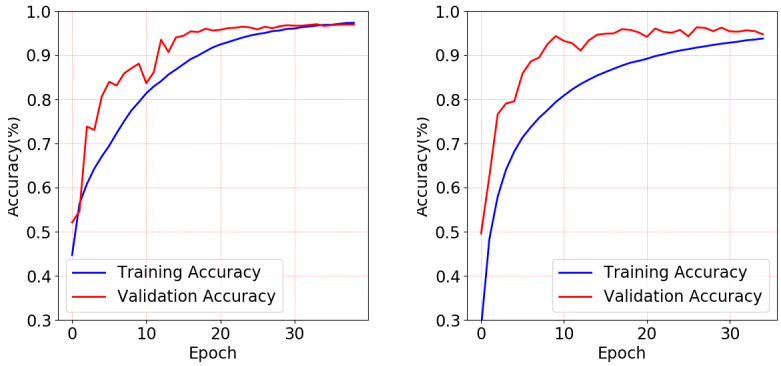
3.5.4 RNN (with LSTM Cells) Results

For the RNN, we first implement the LSTM cells as described in Section 3.3.3.1. We achieved 97.40% and 95.78% testing accuracy for 4 and 8 transmitters respectively. The accuracy plots and confusion matrices are given in Figs. 3.21 and 3.22 respectively.



(a) 4 Transmitters (b) 8 Transmitters

Figure 3.20: Confusion Matrices for Transmitter Classification using DNN

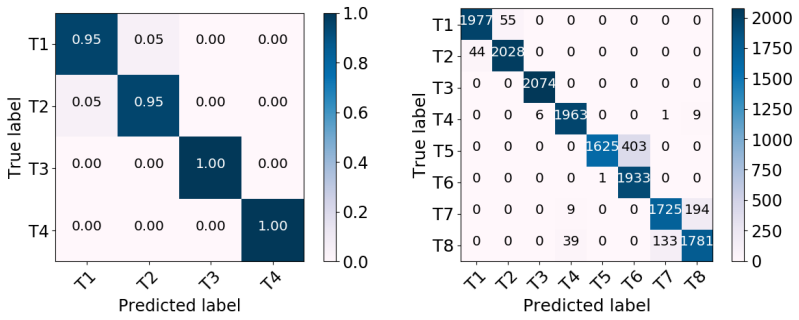


(a) 4 Transmitters (b) 8 Transmitters

Figure 3.21: Accuracy Plots for Transmitter Classification using LSTM Cells

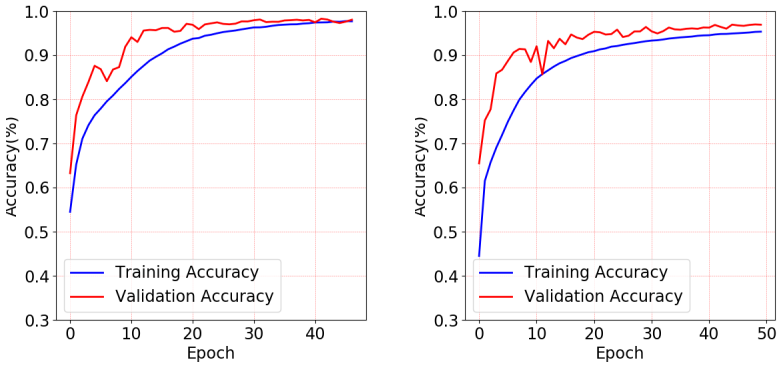
3.5.5 RNN (with GRU Cells) Results

Finally, we implement RNN with GRU cells as described in Section 3.3.3.2. We achieved 97.85% and 97.06% testing accuracy for 4 and 8 transmitters respectively. The accuracy plots and confusion matrices are shown in Figs. 3.23 and 3.24 respectively. It must be noted that the GRU implementation achieves better accuracy than the one using LSTM cells.



(a) 4 Transmitters (b) 8 Transmitters

Figure 3.22: Confusion Matrices for Transmitter Classification using LSTM Cells

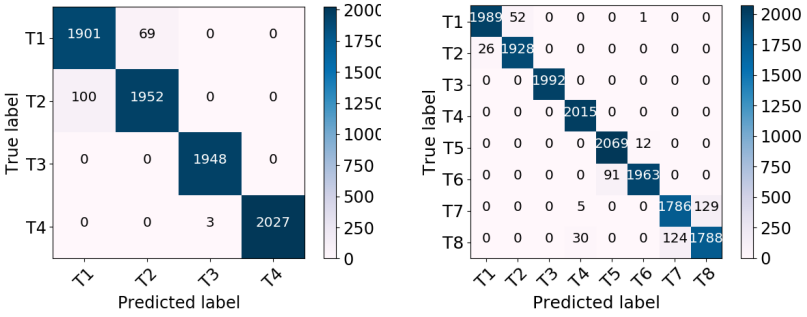


(a) 4 Transmitters (b) 8 Transmitters

Figure 3.23: Accuracy Plots for Transmitter Classification using GRU Cells

During the training phase, the hyper-parameters get adjusted depending on the categorical cross-entropy loss. Sometimes, with such adjustments, the model tends to over-fit the training data. To avoid such scenarios, we used *Dropout* [89] regularization in our models. Another way to monitor and possibly avoid over-fitting is through the use of cross validation during the training phase. This way, we ensure that the proposed model gets trained fairly so that it generalizes well during the testing phase. Note that the fluctuations in the validation curve show that for certain epochs with a particular set of hyper-parameter values, the model tends to over-fit the data, but in

later epochs the hyper-parameter values get adjusted so as to counter the effect of the overfitting. Furthermore, the training can be tweaked using the results of the validation phase. Note that since the I/Q samples represent a *time series* data, it is natural to model them using a RNN. Since RNNs learn the temporal correlation between the time series data, they can encode the transmitter specific variations in the I/Q data and intuitively this makes the RNNs better at the task of trusted transmitter classification than the CNNs.



(a) 4 Transmitters (b) 8 Transmitters

Figure 3.24: Confusion Matrices for Transmitter Classification using GRU Cells

3.5.6 Classification Comparison of CNN/DNN/RNN

Once a transmitter is found to be `Trusted` via the proposed GAN, we used a CNN, DNN, and RNN respectively, to uniquely identify it. We used 90%, 5%, and 5% to train, validate and test respectively. The overall accuracy of the different NNs for the task of trusted transmitter classification is shown in Table 3.3. We see that the CNN does not exhibit the best performance for transmitter classification, which intuitively is due to the lack of distinguishing *spatial* correlation in the data from each transmitter. In Fig. 3.26, we show a graphical representation of the classification accuracy obtained using the different neural network architectures.

We present the observed empirical training time for each model as the last column in Table 3.3. It is evident that the training time for the CNN is almost double compared to the rest of the proposed models, as convolution operations are significantly more complex than other neural network computations. We also conducted experiments by varying the number of transmitters from 2 to 8, using the proposed DNN. In Fig. 3.25, we show how the training and testing accuracy varies as the number of transmitters are increased. We note that the training accuracy decreases when there are more classes (more number of transmitters). However the testing accuracy of the model is stable and does not change significantly with the increase in the number of transmitters. This establishes the efficacy of these methods for the task of transmitter classification, as for production systems one wants the test accuracy to be stable no matter the number of classes involved in the problem.

Table 3.3: Comparison of the Various Implementations

#Trans	Models	#Parameters	Acc (%)	Time (min)
4	CNN	38 Million	89.07	∼25
4	DNN	6.8 Million	96.49	∼12
4	RNN - LSTM	14.2 Million	97.40	∼12
4	RNN - GRU	10.7 Million	97.85	∼12
8	CNN	38 Million	81.59	∼30
8	DNN	6.8 Million	94.60	∼15
8	RNN - LSTM	14.2 Million	95.78	∼16
8	RNN - GRU	10.7 Million	97.06	∼16

In our testbed evaluation one of the goals was to explore different types of NNs to find the architecture (and model) having the best possible accuracy within the constraints of the training time (which was not more than 30 minutes in our test-bed setup for all models). The performance of any NN architecture depends, among other things, on the values of the hyper-parameters. Furthermore, as a given hyper-parameter setting may be optimal for one network but not for another, we used different hyper-parameter values to train different networks. Intuitively, this dependence on the values of the hyper-parameters is due to the fact that the underlying optimization problem and

hence the solution space, is different for different types of networks. We observed that decreasing the learning rate of the CNN to 10^{-4} increases the accuracy by 7-8%. However, decreasing the learning rate to less than 10^{-3} for RNN and DNN does not improve the testing accuracy by a significant amount (even by 0.5%). In Table 3.4 we record the values which gave the best possible accuracy under the constraints of the training time. During each training, we set the maximum epoch to 50 with an early stopping condition, such as, if there is no improvement of validation loss for five consecutive epochs, then the training is stopped. We observed through multiple runs of training, that each of the models converged within a given range of the maximum number of epochs, as presented in Table 3.4.

Table 3.4: Comparison of Configuration Settings for Different Models

Models	#Layers	Learning Rate	Batch Size	Epochs	Optimizer
CNN	7	10^{-4}	128	45-50	*Adam* [42]
DNN	5	10^{-3}	128	35-40	*Adam* [42]
RNN-LSTM	6	10^{-3}	128	30-35	*SGD* [11]
RNN-GRU	6	10^{-3}	128	30-35	*SGD* [11]

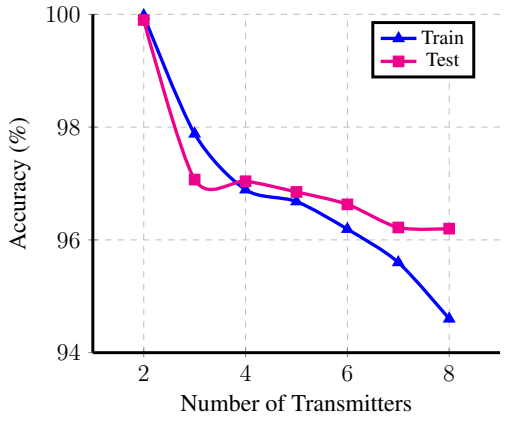
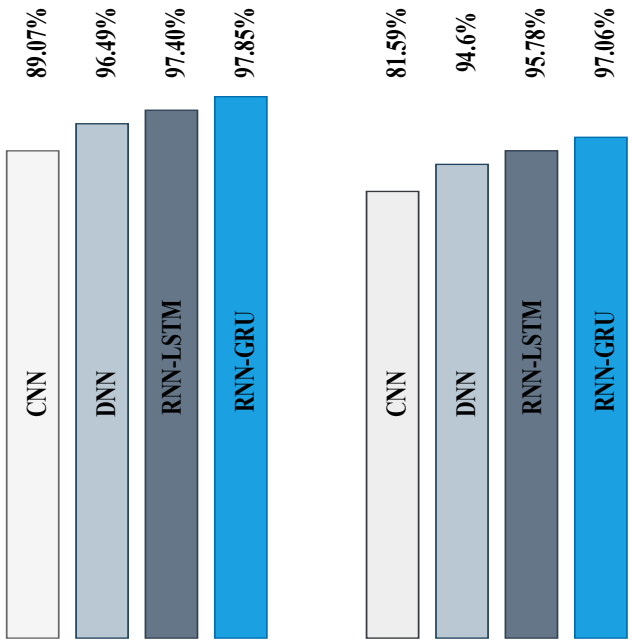


Figure 3.25: Training and Accuracy with Increasing numbers of Transmitters

It is clear from the the results presented above that the GAN based NN is effective for the task of rogue transmitter identification whereas DNN and RNN are effective for the task of trusted transmitter classification. In summary we have established the following:



(a) 4 Transmitters (b) 8 Transmitters

Figure 3.26: Detection Accuracies of the Neural Networks

1. A GAN is able to distinguish between `Trusted` and `Counterfeited` RF transmitters.

2. CNN yields 81%-86% accuracy for trusted transmitter classification proving the inefficacy of spatial correlation as a discriminative attribute for transmitter classification.

3. DNN yields 94-97% accuracy for known transmitter classification.

4. RNN yields an accuracy of 97% for known transmitter classification using GRU cells.

5. Comparing the accuracies, we can conclude that I/Q data of radio signals exhibits more temporal correlation than spatial correlation.

6. DNN or RNN can be used for transmitter fingerprinting for identifying trusted transmitters and in conjunction with GAN this can be used to create an end-to-end robust system for transmitter identification.

3.5.7 Computational Complexities

In this section we present the computational time complexity for the training phase only, as the trained model gives the output within constant time ($O(1)$) during the deployment phase. Understanding the time complexity of training a NN is still an evolving research area. In [51], the authors proved that a NN of depth δ can be learned in $poly(s^{2^{\delta}})$ time, where s is the dimension of the input, and $poly(.)$ takes a constant time depending on the configuration of the system. However, the convolution operations of CNN add additional time complexity along with the forward and back-propagation operations. In [34], the authors mentioned that the time complexity for training all the convolutional layers is: $O(\sum_{\tau=1}^{\zeta}(\eta_{\tau-1}\nu_{\tau}^2.\eta_{\tau}\rho_{\tau}^2)$, where ζ is the number of convolutional layers, τ is the index of a convolutional layer, $\eta_{\tau-1}$ is the number of input channels of the τ^{th} layer, ν_{τ} is the spatial size of the filters at the τ^{th} layer, η_{τ} is the number of filters at the τ^{th} layer and ρ_{τ} is the size of the output features of the τ^{th} layer. In the proposed CNN model, we have 3 convolutional layers, and 4 fully connected layers and hence we add in additional time complexity for training the three convolutional layers.

The time complexities for each implemented NN model is presented in Table 3.5, using the aforementioned results on time complexity of neural network training. The numbers within the parenthesis in the second column represents the total number of layers for a particular model. Note that we have two different datasets of dimensions 160K and 320K and as mentioned earlier, we use 95% of data for training and validation purpose. For example, the complexity for GAN with 8 layers using 95% of $160e3$ data samples for training and validation, is $poly(0.95 \times 160e3^{2^8})$. On another note we should mention that for quick estimates of the time complexity of training neural networks, the number of hyper-parameters (as presented in Table 3.3) can also be used as a measure of the required training time. Thus, the more the number of hyper-parameters, the more the training time required.

Table 3.5: Time Complexities for Training of the Various Implementations

#Trans	Models	Complexity
4	GAN (8)	$poly(0.95 \times 160e3^{2^8})$
4	CNN (7)	$poly(0.95 \times 160e3^{2^3})$ $\times O(\sum_{\tau=1}^{3}(\eta_{\tau-1}.\nu_{\tau}^2.\eta_{\tau}.\rho_{\tau}^2)$ + $poly(0.95 \times 160e3^{2^4})$
4	DNN (5)	$poly(0.95 \times 160e3^{2^5})$
4	RNN - LSTM (6)	$poly(0.95 \times 160e3^{2^6})$
4	RNN - GRU (6)	$poly(0.95 \times 160e3^{2^6})$
8	GAN (8)	$poly(0.95 \times 320e3^{2^8})$
8	CNN (7)	$poly(0.95 \times 320e3^{2^3})$ $\times O(\sum_{\tau=1}^{3}(\eta_{\tau-1}.\nu_{\tau}^2.\eta_{\tau}.\rho_{\tau}^2)$ + $poly(0.95 \times 320e3^{2^4})$
8	DNN (5)	$poly(0.95 \times 320e3^{2^5})$
8	RNN - LSTM (6)	$poly(0.95 \times 320e3^{2^6})$
8	RNN - GRU (6)	$poly(0.95 \times 320e3^{2^7})$

3.5.8 Experiments with Heterogeneous Dataset

So far, we have used the proposed trusted transmitter identification models on "homogeneous" datasets in that the transmitters were implemented using the SDRs from the same manufacturer. However, in reality the trusted transmitters can be from several different manufacturers. Now we want to explore how the accuracy of the trusted transmitter identification system would change if "heterogeneous" data obtained from different types of transmitters (manufacturers) (as was discussed in Section 3.4.3) was used. From the testing accuracy as shown in Table 3.6 we observe that all the NNs perform better when the transmitters are from different manufacturers and hence are fundamentally of different types. This confirms the intuition that radios manufactured using different processes (from different manufacturers) contain easily exploitable characteristics in their I/Q samples, that can be implicitly learned using a NN. We also observe that the CNN can exploit the spatial correlation better for the heterogeneous dataset, yielding a 11.2% increase in the testing

accuracy compared to the homogeneous dataset.

Table 3.6: Comparison of testing Accuracies for Different Classification Models for Homogeneous and Heterogeneous Datasets

Models	USRP-USRP	PLUTO-USRP
CNN	89.91 (%)	99.91 (%)
DNN	99.9 (%)	100 (%)
RNN	99.95 (%)	100 (%)

3.5.9 Existing Transmitter Classification Techniques Comparisons

Though RFAL demonstrates the use of adversarial learning for rogue transmitter identification, a part of our work has been devoted to the problem of "trusted transmitter classification" after elimination of the rogue transmitters. "Trusted transmitter" identification systems when augmented with adversarial learning systems, as was done in RFAL, results in robust transmitter identification systems that are immune to presence of adversarial transmissions. Though our focus was not on improving or building novel "trusted transmitter" classification systems, we have explored the use of deep learning for building the same using I/Q data from the received signal. In this section we present a comparative study of our approach to "trusted transmitter classification with I/Q data" against some existing techniques for "transmitter classification" and the results of this endeavor is shown in Tables 3.7 for traditional approaches (low accuracy) and 3.8 for "state-of-the-art" (high accuracy) respectively. Note that here the "Inputs" column refers to the type of inputs used for the classification algorithms. It is to be noted that all the traditional methods use some form of extracted features (obtained through pre-processing of the data) as inputs ([41, 86, 94, 101]), or work with synthetic dataset ([73]). A few existing work on modulation recognition [65, 67, 68] using NN based approaches also work on synthetic datasets [63, 72] and hence they do not yield to a fair comparison with our "trusted transmitter classification" approach as well.

For the "state-of-the-art" approaches, we observe that the test bed experiments have used various types of SDRs for obtaining over-the-air data, or used datasets from actual infrastructure transmitters (ACARS etc.). We present comparison with: (i) [75], where same SDR (USRP B210) was used, (ii) [102] and [85], where different SDRs from the same manufacturer (USRP N210, USRP X310) were used, (iii) [55], where different types of devices (Zigbee) were used, and (iv) [109], [17] and [56] where different datasets (ACARS [4], ADS-B [5], FIT/CorteXlab [53]) were used.

The RFAL "trusted transmitter identification" models outperform [55] where the authors achieved 91.38% accuracy for classifying 7 Zigbee devices. The CNN proposed in [75] achieved 98% accuracy for 5 USRP B210 with preprocessed data (from MATLAB Communication Systems Toolbox). Similarly, in [85], Sankhe *et al.* presented a CNN classifier with 16 X310 radios with 99.5% accuracy, but that method used demodulated symbols rather than raw signal data. Youssef *et al.* presented a multi-stage training model to achieve 100% accuracy to classify 12 USRP N210s. Multi-stage training is a complex procedure and needs a easily parallelizable training environment, whereas CNN and DNN use a first-order update rule (stochastic gradient) and are comparatively simple procedures. For the sake of generality, we compare their proposed DNN method (which has the best accuracy compared to other implemented ML methods) in Table 3.8. The CNN models presented in [109], [17] and [56] achieved between 96% and 99.9% accuracy for existing datasets (ACARS [4], ADS-B [5], FIT/CorteXlab [53]).

We need to point out that none of methods discussed above are robust enough to work in *adversarial* settings. Thus if there is an adversarial transmitter then *all* the aforementioned methods will fail. However RFAL, due to its adversarial training will be able to identify the adversarial transmitter and eliminate it from consideration before classifying the "trusted transmitters", thus being more resilient and robust to such interference. Even if we just compare the "trusted transmitter" classification of RFAL with the methods discussed above, our method for "trusted transmitter

classification" achieves the same accuracy (of 97%) using just the raw I/Q data as input, thereby paving the way for real-time deployment of "transmitter fingerprinting" systems.

Considering the state-of-art, and to the best of our knowledge, our work is the first to:

1. propose a GAN based model to detect rogue transmitters from authentic ones;
2. demonstrate a high accuracy (97%) to classify 8 USRP B210s using an RNN model considering the temporal property of RF data;
3. present a testbed evaluation for classifying transmitters from different manufacturers;
4. provide an end-to-end solution of RF transmitter classification without any preprocessing of raw data. The raw data can be captured through any SDR and is identified by the proposed models at once;
5. propose RNN based models which needed half the training time than CNN models for the same experimental training time. Our proposed RNN model entails to be faster than all state-of-the art CNN models for the same experimental environment.

Table 3.7: Comparison of the Our Implementation with the Traditional ones

Approach	#Trans	SNR (dB)	Acc (%)	Inputs
Genetic Algorithm [94]	5	25	85-98	Transients
Multifractal Segmentation [86]	8	Not mentioned	92.5	Transients
Orthogonal Component Reconstruction (OCR) [101]	3	20	62 - 71	Spurious Modulation
k-NN [41]	8	30	97	Transients
RNN [73]	-	20	90	Synthetic Dataset [72]
RFAL (Ours)	8	30	97.04	Raw Signal

Table 3.8: Comparison of the Our Implementation with State-of-the-art

Approach	#Trans	SNR (dB)	Acc (%)	Input
CNN [55]	7	30	91.38	Preprocessed data from MATLAB
CNN [75]	5	50	98	Preprocessed data from MATLAB
CNN [109]	-	-	99.67	ACARS data [4]
DNN [102]	12	-	84.4	Raw signal
Inception ResNet [17]	-	-	98.1 & 96.3	ACARS [4] & ADS-B [5]
CNN [85]	16	30	99.5	Demodulated symbols
CNN [56]	21	-	99.99	FIT/CorteXlab [53]
RNN (Ours)	8	30	97.04	Raw signal

3.5.10 Performance Comparison for Varying SNR

In this section, we present the results of RFAL "trusted transmitter classification" for varying SNR values. We compare the accuracy for the proposed NN models having 8 USRP B210s with 30 dB SNR, with 3 other datasets collected at 0 dB, 10 dB, and 30 dB SNRs having the same number of transmitters (8 B210s) as shown in Table 3.9. It is seen that we achieve better accuracy with all the models for higher SNR values, which is intuitive. It is to be noted that the proposed RNN (with GRU cell) model gives more than 92% accuracy at 0 dB SNR too, whereas CNN and DNN models fail to achieve that.

It must be pointed out that the proposed NN models can be a trained using raw signal data from any type of radio transmitter. We would also like to point out that though our data was collected in a lab setting, we had no control over the RF environment: there were other transmissions, uncontrolled movement of people, and multi-path effects due to the location and layout of the lab. Moreover, the power of the transmitters was low which compounded the problem further. Thus, though we mention that the data was collected in a lab environment, in reality it was an uncontrolled RF environment reflective of our surroundings. We can safely argue that the proposed methods will

perform equally well in any real world deployment of large scale radio networks.

Table 3.9: Accuracies for Different Neural Network Models with Varying SNRs

SNR(dB)	**Accuracy (%)**		
	CNN	DNN	RNN (GRU)
0	51.53	85.12	92.3
10	78.64	92.24	95.64
20	81.3	94.60	97.02
30	81.59	94.60	97.06

3.6 Summary

In this chapter, we address the problem of building a robust and resilient model for identifying similar RF transmitters in the presence of adversaries. We argue that non-adversarial machine learning techniques would not be effective in adversarial settings and that breakthroughs in generative adversarial nets (GANs) can be instrumental in building such systems for detection of rogue transmitters and subsequent accurate identification of known ones in such settings. We propose and implement RF Adversarial Learning (RFAL) framework which includes a discriminative model for identifying rogue transmitters trained with data generated from a generative model. RFAL also contains a "trusted transmitter" identification system that for categorizing the known transmitters once the adversarial transmitters have been identified and eliminated from consideration. We collected over-the-air raw I/Q data using USRP B210s and used that to train the GAN. The discriminator was able to detect rogue transmitters with an accuracy of $\sim$99.9%. As for the subsequent trusted transmitter classification, we first implemented a CNN (accuracy $\sim$89%) for exploiting the spatial correlation between the I/Q data. Then we designed and implemented a fully connected DNN and RNNs both of which obtained an accuracy of around 97% for trusted transmitter identification. We also show how the proposed NN models for "trusted transmitter classification" (especially CNN)

worked when radios from different manufacturers were used. RFAL will be able to detect any active attack that involves a secondary device to pose as an authentic emitter, such as replay attacks, but it will not be able to detect passive attackers, such as traffic sniffers. Going forward we would like to use these methods for identification of actual infrastructure transmitters (for example FM, AM or GSM) in contested real world settings.

CHAPTER 4: TRANSMITTER FINGERPRINTING USING RECURRENT STRUCTURES

In this chapter, we propose a transmitter fingerprinting technique for radio device identification using recurrent structures, by exploiting the temporal property of the received radio signal. We design and implement three recurrent neural networks (RNNs) using different types of cell models: (i) long short term memory (LSTM); (ii) gated recurrent unit (GRU) and (iii) convolutional long short term memory (ConvLSTM), for this task. We program 8 universal software radio peripheral (USRP) software defined radios (SDRs) as transmitters and collect over-the-air raw in-phase (I) and quadrature (Q) (I/Q) *time series* data from them using a DVB-T RTL-SDR receiver, in a laboratory setting. We exploit both the temporal variations as well as the inherent spatial dependencies in the collected I/Q time series data, to learn unique feature representations and use these as "fingerprints" for identifying the transmitters. Experimental results reveal that the RNNs with LSTM, GRU, and ConvLSTM cells are able to correctly distinguish between the 8 transmitters with 95%, 97%, 98% accuracy respectively.

The chapter is organized as follows: in the first section, we present a survey of existing machine learning based transmitter identification techniques. In section 4.1, we propose different RNN models. In section 4.2, we present the testbed setup and experiments that we conduct to evaluate the proposed models. We present the experimental results in section 4.3. The contents of this chapter appeared in [76, 81].

4.1 Proposed RNN Models for Classification

Neural networks have previously been used for transmitter identification [65, 68, 79] and are particularly attractive since they can generate accurate models without knowledge of the *apriori* data distribution. Neural networks have been shown to be able to predict modulation techniques [65] and identify transmitters [79] by only considering the spatial correlations within the actual [68] or synthetic RF data [63]. It is to be noted that all prior works have only exploited the spatial correlation of the signal data, though a continuous signal can be represented as a time series, having both temporal and spatial properties [98].

Inspired by the success of deep learning systems for the task of characterizing RF environments [67] and the successful use of recurrent neural networks (RNN) for the task of analyzing time series data [74], we propose to use deep recurrent structures for learning transmitter "fingerprints" for the task of transmitter identification. Recurrent Neural Networks (RNNs) [31] have been shown to be useful for capturing and exploiting the temporal correlations of time series data. There are a few variants of recurrent neural networks: (i) Long Short-Term Memory (LSTM) [35], (ii) Gated Recurrent Unit (GRU) [19], and (iii) Convolutional Long Short-Term Memory (ConvLSTM) [87]. All these variants are designed to learn the long term temporal dependencies and are capable of avoiding the "vanishing" or "exploding" gradient problems [24].

In order to estimate the noise in a RF channel, the system needs to "listen" to the underlying signal for sometime and "remember" the same. Previously, neural networks lacked this capability when used in the context of temporal data. Another issue with using neural networks with temporal data was the problem of *vanishing gradients*, when trying to use *back propagation*. Both these problems were solved by the introduction of Recurrent Neural Networks (RNN) [45].

4.1.1 Formulation of temporal property of RF data

Given T training samples (for T timestamps) where each training sample is of size of M and consists of a vector of tuples of the form $(I, Q) \in \mathcal{C}$ representing a number in the complex plane, we represent a single sample as $x_t = [[(I, Q)_i]^t; i = 1, 2, \cdots, M] \in \mathcal{C}^M$ for each timestamp $t = 1, 2, \cdots, T$, and we use it as an input to the neural network. We use a sample size (M) of 1024 as a default. We want to find the probability of the input vector for next time step (x_{t+1}) to belong to class C_k, where $k \in 1, 2, \ldots, K$, K being the number of classes. The probability $P(C_k|x_{t+1})$ can be written as

$$P(C_k|x_{t+1}) = \frac{P(x_t|C_k)P(C_k)}{P(x_t x_{t+1})} \tag{4.1}$$

where ($P(x_t|C_k)$) is the conditional probability of x_t given the class C_k and ($P(x_t x_{x+1})$) is the probability of x_t and x_{t+1} occurring *in order*.

The details of used LSTM and GRU cell models are described in Section 3.3.3.1 and 3.3.3.2 in Chapter 3. So we next describe the details of convolutional LSTM network.

4.1.2 Convolutional LSTM Network Model

The recurrent neural networks with LSTM or GRU cells, do not consider the spatial information encoded in the the input-to-state or state-to-state transitions. To mitigate this problem, we use a convolution within the recurrent structure of the RNN. We first discuss the spatio-temporal property of RF data and then model a convolutional LSTM network to exploit the same.

4.1.2.1 Formulation of Spatio-temporal property for RF data

Suppose that a radio signal is represented as a time varying series over a spatial region using R rows and C columns. Here R represents the time varying nature of the signal and as such in our case it represents the total number of time stamps at which the signal was sampled (T in our case). C on the other hand represents the total number of features sampled at each time stamp (in our case its 2048 since there are 1024 features sampled each of dimension 2). Note that each cell corresponding to one value of R and one value of C represents a particular feature (I or Q) at a given point in time.

In order to capture the temporal property only, we use a sequence of vectors corresponding to different timestamps $1, 2, \cdots, t$ as $x_1, x_2, \cdots, x_t$. However, to capture both spatial and temporal properties, we introduce a new vector $\chi_{t,t+\gamma}$, which is formulated as: $\chi_{t,t+\gamma} = [x_t, x_{t+1}, \cdots, x_{t+\gamma-1}]$. So the vector $\chi_{t,t+\gamma}$ eventually preserves the spatial properties with an increment of γ in time. So, we get a sequence of new vectors $\chi_{1,\gamma}, \chi_{\gamma,2\gamma}, \cdots \chi_{t,t+\gamma}, \cdots, \chi_{t+(\beta-1)\gamma,t+\beta\gamma}$, where β is $\lfloor R/\gamma \rfloor$, and the goal is to create a model to classify them into one of the K classes (corresponding to the transmitters). We model the class-conditional densities given by $P(\chi_{t-\gamma,t}|C_k)$, where $k \in 1, \cdots, K$. We formulate the probability of the next γ-length sequence to be in class C_k as per equation 4.2. The marginal probability is modeled as $P(\chi_{t,t+\gamma})$.

$$P(C_k|\chi_{t,t+\gamma}) = \frac{P(\chi_{t-\gamma,t}|C_k)P(C_k)}{P(\chi_{t,t+\gamma})} \tag{4.2}$$

4.1.2.2 The ConvLSTM Model

The cell model is similar to an LSTM cell, but the input transformations and recurrent transformations are both convolutional in nature [87]. We formulate the input values, cell state and hidden

states as a 3-dimensional vector, where the first dimension is the number of measurements which varies with the time interval γ and the last two dimensions contain the spatial information (rows (R) and columns (C)). We represent these as: (i) the inputs: $\chi_{1,\gamma}, \chi_{\gamma,2\gamma}, \cdots \chi_{t,t+\gamma}, \cdots, \chi_{t+(\beta-1)\gamma,t+\beta\gamma}$ (previously stated); (ii) cell outputs: $\mathcal{C}_1, \cdots, \mathcal{C}_t$, and (iii) hidden states: $\mathcal{H}_1, \cdots, \mathcal{H}_t$. We represent the gates in a similar manner as in the LSTM model. The parameters t, i_t, f_t, o_t, W, b hold the same meaning as in Section 3.3.3.1 in Chapter 3. The key operations are defined in equations 4.3, 4.4, 4.5, 4.6, and 4.7. The probability of the next γ-sequence to be in a particular class (from equation 4.2) is used within the implementation and execution of the model.

$$i_t = \sigma(W_{xi}\chi_{t,t+\gamma} + W_{hi}\mathcal{H}_{t-1} + b_i) \tag{4.3}$$

$$f_t = \sigma(W_{xf}\chi_{t,t+\gamma} + W_{hf}\mathcal{H}_{t-1} + b_f) \tag{4.4}$$

$$\mathcal{C}_t = f_t \circ \mathcal{C}_{t-1} + i_t . \tanh(W_{xc}\chi_{t,t+\gamma} + W_{hc}\mathcal{H}_{t-1} + b_c) \tag{4.5}$$

$$o_t = \sigma(W_{xo}\chi_{t,t+\gamma} + W_{ho}\mathcal{H}_{t-1} + b_o) \tag{4.6}$$

$$\mathcal{H}_t = o_t \circ \tanh(\mathcal{C}_t) \tag{4.7}$$

4.2 Testbed Evaluation

In order to validate the proposed models, we collected raw signal data from 8 different universal software radio peripheral (USRP) B210s [26]. We collected the data in an indoor lab environment

with a signal-to-noise ratio of 30 dB, and used the dataset to distinguish between 4 or 8 transmitters, as mentioned in [79].

4.2.1 Signal Generation and Data Collection

In order to evaluate our methods for learning the inherent spatio-temporal features of a transmitter, we used eight USRPs of the same type, namely B210 from Ettus Research [26], as transmitters. The signal generation and reception are shown in Fig. 4.1. We used GNURadio [29] to randomly generate signal and modulated the same with Quadrature Phase Shift Keying (QPSK). We programmed the USRP B210s to transmit the modulated signal over the air and sensed the same using a DVB-T dongle (RTL-SDR) [60]. We generated the entire dataset from "over-the-air" data as sensed by the RTL-SDR using the *rtlsdr* python library.

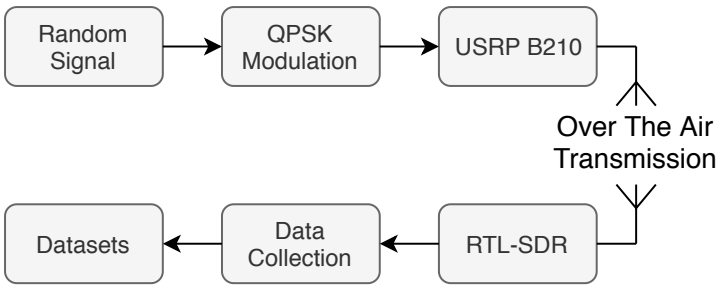


Figure 4.1: Over the Air Signal Generation and Data Collection Technique

We collected I/Q signal data with a sample size of 1024 at each time stamp. Each data sample had 2048 entries consisting of the I and Q values for the 1024 samples. Note that a larger sample size would mean more training examples for the neural network. Our choice of 1024 samples was sufficient to capture the spatial-temporal properties while at the same time the training was not computationally intensive. We collected 40,000 training examples from each transmitter to avoid the data skewness problem observed in machine learning. The configuration parameters that were

used are given in Table 4.1. We collected two sets of data: (i) using 4 transmitters: 6.8 GB size, 160K rows and 2048 columns and (ii) using 8 transmitters: 13.45 GB size, 320K rows and 2048 columns.

Table 4.1: Transmission Configuration Parameters

Parameters	**Values**
Transmitter Gain	45 dB
Transmitter Frequency	904 MHz (ISM)
Bandwidth	200 KHz
Sample Size	1024
Samples/Transmitter	40,000
# Transmitters	4 and 8

4.2.2 Spatial Correlation in the Dataset

The wokring principle of correlation calculation in the collected RF data is already described in Section 3.4.4 in Chapter 3. The spatial correlations of all the samples for the different transmitters are shown in Fig. 4.2. We observe that for most of the transmitters, the correlation is $\sim$0.42, with a standard deviation of $\sim$0.2. However, transmitter 3 exhibits minimal correlation between these samples, which implies that the spatial property of transmitter 3 is different from the other transmitters. As a result Transmitter 3 should be easily distinguishable from the others. This claim will be validated later in the experimental result section where we see 0% false positive and false negative for transmitter 3 for all the three proposed models. This observation gives us the motivation to exploit the spatial property as well as the temporal property for the collected *time-series* data.

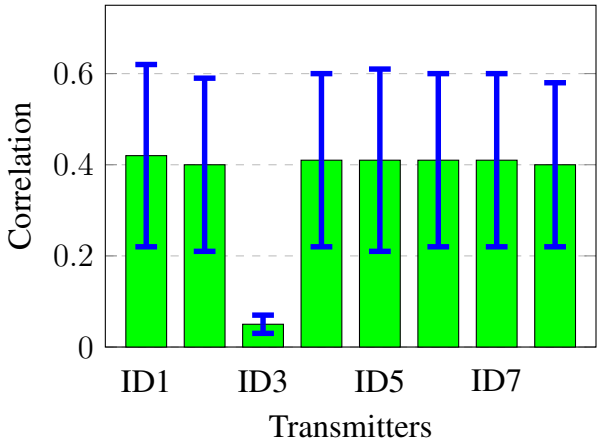


Figure 4.2: Spatial Correlation in the Dataset

4.2.3 Neural Network Libraries

There are many libraries available in *python* with support for different types of neural network and concurrent GPU architecture. We use *Keras* [18] as the frontend and *Tensorflow* [1] as the backend for our implementations. Keras is an overlay on the neural network primitives provided by Tensorflow [1] or Theano [2] and provides a customizable interface for quick deployment of complex neural networks. We also use *Numpy*, *Scipy*, and *Matplotlib* Python libraries.

4.2.4 Experimental Setup and Performance Metrics

We conducted the experiments on a Ryzen 8 Core system with 64 GB RAM, a GTX 1080 Ti GPU unit having 11 GB memory. During the training phase, we use data from each transmitter to train the neural network model. In order to test the resulting trained model, we use test data collected from one of the transmitters and present the same to the trained network. In general to measure the effectiveness of any learning algorithm, "accuracy" is used as the typical performance metric. However, accuracy can sometimes be misleading and incomplete when the data is skewed. For the

task of classification, a confusion matrix overcomes this problem by showing how confused the learned model is on its predictions. It provides more insights on the performance by identifying not only the number of errors, but more importantly the types of errors.

4.3 Implementations and Results

In this section we discuss the implementation of each of the proposed recurrent neural networks. We train each network for transmitter classification with K classes. For the sake of robustness and statistical significance, we present the results for each model after averaging over several runs.

4.3.1 Implementation with LSTM Cells

As discussed earlier, the recurrent structure of the neural network can be used to exploit the temporal correlation in the data. To that end, we first implemented a recurrent neural network with LSTM cells and trained it on the collected dataset using the paradigm as shown in Fig. 4.3. We used two LSTM layers with 1024 and 256 units sequentially. We also used a *dropout* rate of 0.5 in between these two LSTM layers. Next we used two fully connected (*Dense*) layers with 512 and 256 nodes respectively. We apply a *dropout* rate of 0.2, and add *batch normalization* [38] on the output, finally passing it through a *Dense* layer having 8 nodes. We use *ReLU* [58] as the activation function for the LSTM layers and *tanh* [11] for the *Dense* layers. Lastly, we use *stochastic gradient descent* [11] based optimization with categorical cross-entropy training. Note that the neural network architecture was finalized over several iterations of experimentation with the data and we are only reporting the final architecture here. We achieved 97.17% and 95.00% testing accuracy for 4 and 8 transmitters respectively. The accuracy plots and confusion matrices are shown in Figs. 4.4 and 4.5 respectively. Note that the number of nodes in the last layer is equal

to the number of classes in the dataset. It is also to be noted that during the process of designing the RNN architecture, we also fine tuned the hyper-parameters based generalization ability of the current network (as determined by comparing the training and validation errors). We also limited the number of recurrent layers and fully connected layers for each model for faster training [34], since no significant increase in the validation accuracy was observed after increasing the number of layers.

The rows and columns of the confusion matrix correspond to the number of transmitters (classes) and the cell values show the recall or sensitivity and false negative rate for each of the transmitters. Note that recall or sensitivity represents the true positive rates for each of the prediction classes.

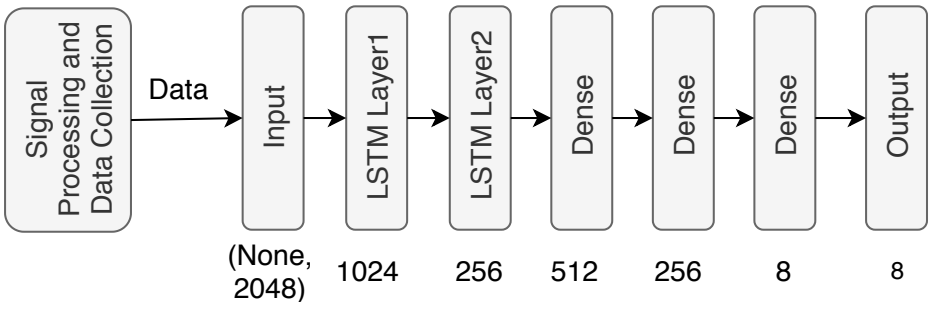
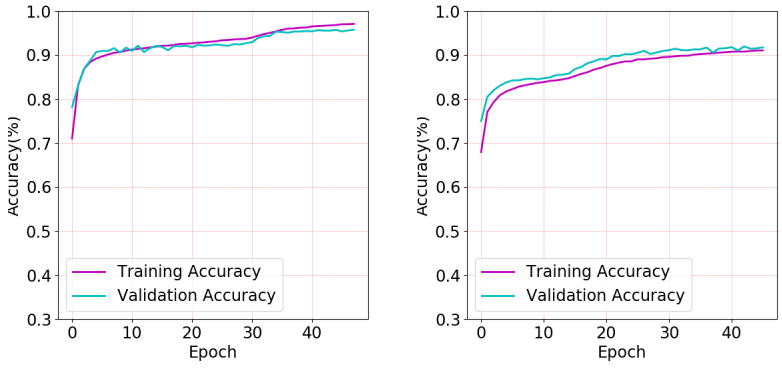
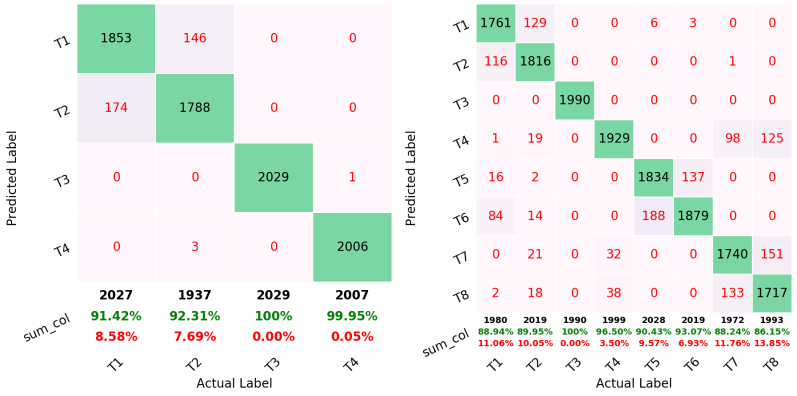


Figure 4.3: RNN Implementation with LSTM Cells for Transmitter Classification



(a) 4 Transmitters (b) 8 Transmitters

Figure 4.4: Accuracy Plots for Transmitter Classification using LSTM Cells



(a) 4 Transmitters (b) 8 Transmitters

Figure 4.5: Confusion Matrices for Transmitter Classification using LSTM Cells

4.3.2 Implementation with GRU Cells

Next we implemented another variation of the RNN model using GRU cells for leveraging temporal correlation. We used the same architecture as the LSTM implementation, presented in Fig. 4.6. The proposed GRU implementation needs fewer parameters than the LSTM model. A quantitative comparison is given in Section 4.3.5. The only difference is that we use two GRU layers with 1024 and 256 units instead of using LSTM cells. We achieved 97.76% and 97% testing accuracy for 4 and 8 transmitters respectively. The accuracy plots and confusion matrices are given in Figs. 4.7 and 4.8. The GRU implementation provided a slight improvement over the accuracy obtained using LSTM, for each run of the models, for both the datasets.

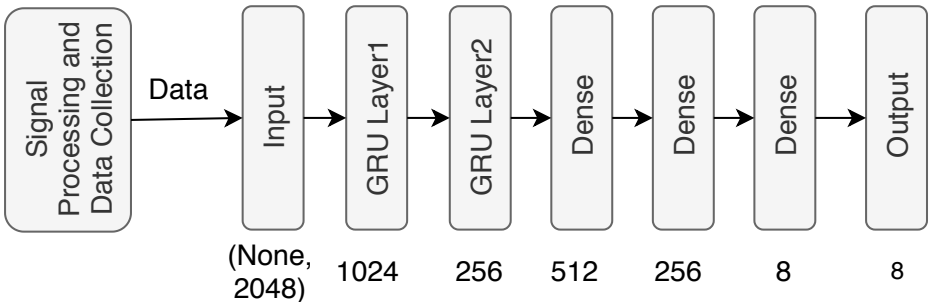
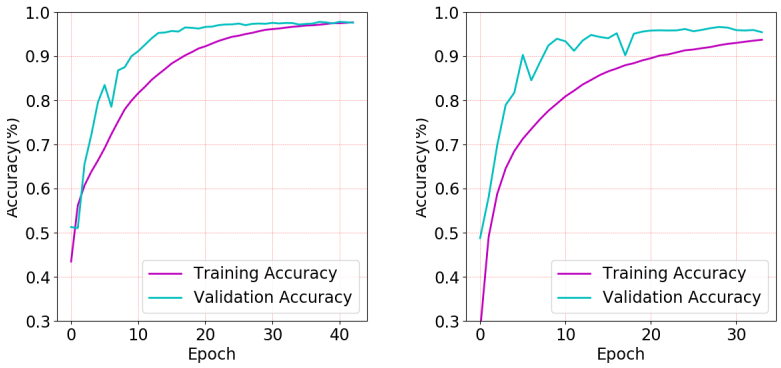
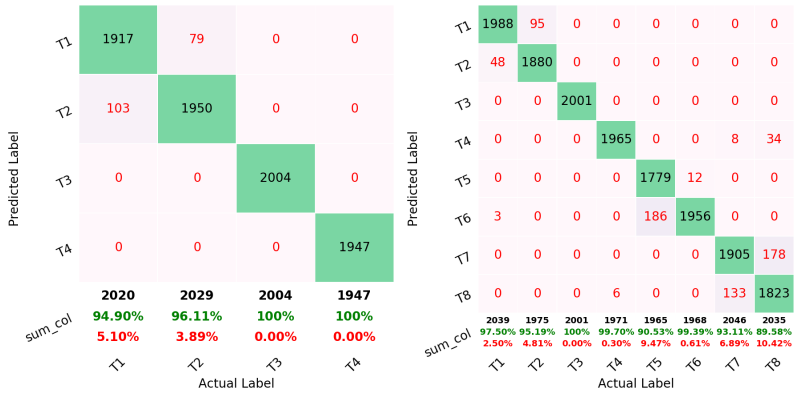


Figure 4.6: RNN Implementation with GRU Cells for Transmitter Classification



(a) 4 Transmitters (b) 8 Transmitters

Figure 4.7: Accuracy Plots for Transmitter Classification using GRU Cells



(a) 4 Transmitters (b) 8 Transmitters

Figure 4.8: Confusion Matrices for Transmitter Classification using GRU Cells

4.3.3 Implementation with ConvLSTM2D Cells

Finally, in order to exploit the spatio-temporal property of the signal data, we implemented another variation of the LSTM model with convolutional filters (transformations). The implemented architecture is shown in Fig. 4.9. *ConvLSTM2D* uses two dimensional convolutions for both input transformations and recurrent transformations. We first use two layers of *convLSTM2D* with 1024 and 256 filters respectively, and a *dropout* rate of 0.5 in between. We use kernel size of

(2,2) and stride of (2,2) at each *ConvLSTM2D* layer. Next we add two fully connected (*Dense*) layers having 512 and 256 nodes respectively after *flattening* the convolutional output. *ReLU* [58], and *tanh* [11] activation functions are used for the *convLSTM2D* and *Dense* layers respectively. ADADELTA [104] with a learning rate of 10^{-4} and a decay rate of 0.9, is used as the optimizer with categorical cross-entropy training. We achieved 98.9% and 98% testing accuracy for 4 and 8 transmitters respectively. The accuracy plots and confusion matrices are given in Figs. 4.10 and 4.11 respectively. Being able to exploit the spatio-temporal correlation, ConvLSTM implementation provides improvement over the accuracies obtained using the LSTM and GRU models, for both the datasets.

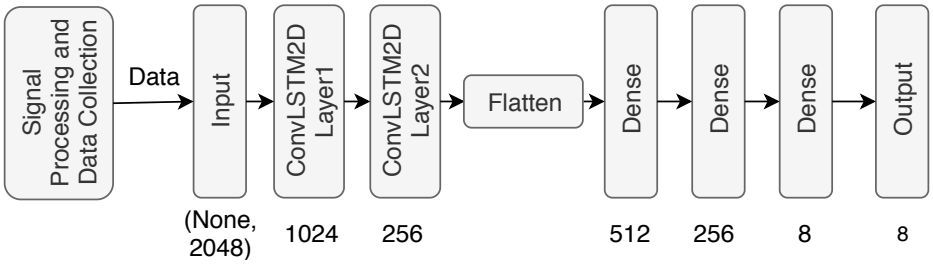
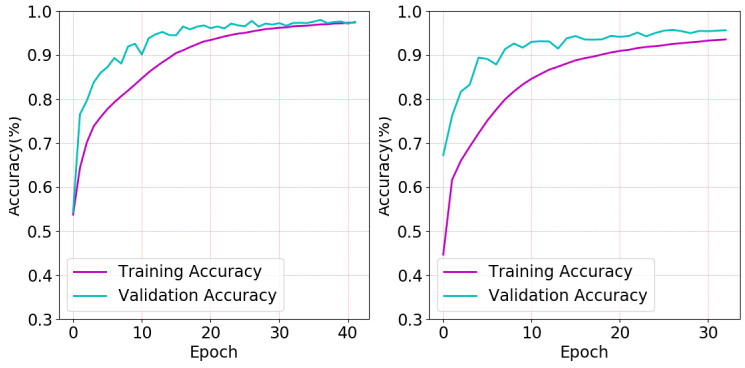
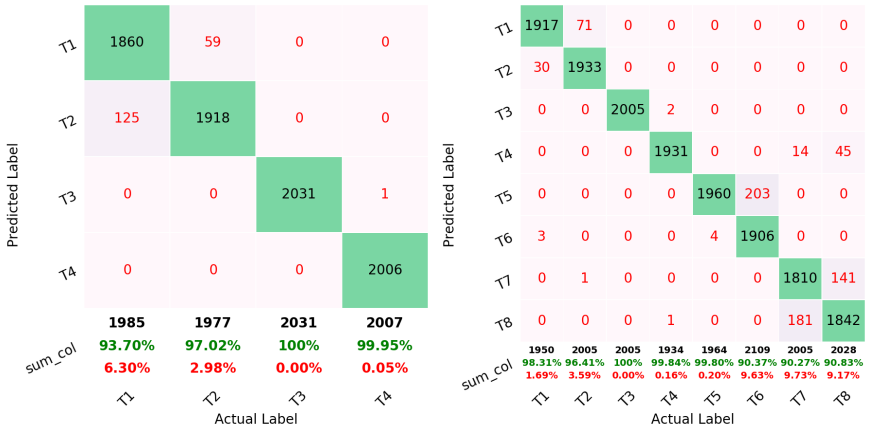


Figure 4.9: RNN Implementation with ConvLSTM Cells for Transmitter Classification



(a) 4 Transmitters (b) 8 Transmitters

Figure 4.10: Accuracy Plots for Transmitter Classification using ConvLSTM Cells



(a) 4 Transmitters (b) 8 Transmitters

Figure 4.11: Confusion Matrices for Transmitter Classification using ConvLSTM Cells

4.3.4 Comparisons of LSTM/GRU/ConvLSTM Implementations

We used 90%, 5%, and 5% of the data to train, validate, and test respectively. We ran each model for 50 epochs with early-stopping on the validation set. One epoch consists of a forward pass and a backward pass through the implemented architecture for the entire dataset. The overall accuracy of the different implementations is shown in Table 4.2. We find that the implementation of convolutional layers with recurrent structure (*ConvLSTM2D*) exhibit the best accuracy for transmitter classification, which clearly shows the advantage of using the spatio-temporal correlation present in the collected datasets.

Table 4.2: Accuracy for Different Implementations

#Trans	Models	#Parameters	Acc (%)
4	LSTM (6 layers)	14.2 M	97.17
4	GRU (6 layers)	10.7 M	97.76
4	ConvLSTM (6 layers)	14.2 M	98.90
8	LSTM (6 layers)	14.2 M	95.00
8	GRU (6 layers)	10.7 M	97.02
8	ConvLSTM (6 layers)	14.2 M	98

4.3.5 Comparisons of Proposed and Existing Approaches

Next we present two comparative studies of our proposed implementations with some existing techniques. We introduce a differential analysis of different RNN based implementations in the RF domain in Table 4.3. Another comparative study for different transmitter classification techniques is shown in Table 4.4.

Table 4.3: Comparison of Proposed Approach with the Existing RNN Implementations

Approaches	Model	SNR (dB)	Acc (%)	Inputs
Traffic Sequence Recognition [62]	LSTM	20	31.2	Hybrid Real-synthetic Dataset
Automatic Modulation Classification [73]	LSTM	20	90	Synthetic Dataset [72]
Transmitter Classification (Ours)	ConvLSTM	30	98	Raw Signal

Table 4.4: Comparison of the Our Implementation with the Existing Transmitter Classification Approaches

Approach	#Trans	SNR (dB)	Acc (%)	Inputs
Orthogonal Component Reconstruction (OCR) [101]	3	20	62 - 71	Spurious Modulation
Genetic Algorithm [94]	5	25	85-98	Transients
Multifractal Segmentation [86]	8	Not mentioned	92.5	Transients
k-NN [41]	8	30	97.2	Transients
Ours	8	30	98	Raw Signal

The "Inputs" column in both the tables refer to the type of inputs used for the methods under consideration. Table 4.3 shows a comparison of our ConvLSTM based RNN for transmitter classification with other RNN based implementations for separate tasks like modulation recognition and traffic sequence recognition. Table 4.4 establishes the efficacy of our ConvLSTM based RNN model for the task of transmitter classification by comparing the accuracy with that obtained using

other methods, for the same task. It is to be noted that all the other methods use expert crafted features as inputs ([41, 86, 94, 101]), or work with synthetic datasets ([62], [73]). Our method, on the other hand achieves superior accuracy (98%) using features automatically learned from the raw signal data, thereby paving the way for real-time deployment of large scale transmitter identification systems.

It must be pointed out that the proposed RNN models can be a trained using raw signal data from any type of radio transmitter operating both in indoor as well as outdoor environments. We would also like to point out that though our data was collected in a lab environment, we had no control over the environment, there were other transmissions in progress, people were moving in and out of the lab and there was a lot of multi-path due to the location and design of the lab. Furthermore the power of the transmitters was low and hence this compounded the problem further. Given this, though we say that the data was collected in a lab environment, in reality it was an uncontrolled daily use environment reflective of our surroundings. Thus we can safely say that these methods will work in any real world deployment of large scale radio network. In summary,

- Exploiting temporal correlation only, recurrent neural networks yield 95-97% accuracy for transmitter classification using LSTM or GRU cells. RNN implementation with GRU cells needs fewer parameters than LSTM cells as shown in Table 4.2.
- Exploiting spatio-temporal correlation, the implementation of RNN using ConvLSTM2D cells provides better accuracy (98-99%) for transmitter classification, thus providing a potential tool for building automatic real world transmitter identification systems.
- We present a comparative study of the proposed spatio-temporal property based fingerprinting with the existing traditional and neural network based models. This clearly shows that the proposed model achieves the best accuracy compared to any of the existing methods for the task.

4.4 Summary

In this chapter, we proposed a robust transmitter identification technique by exploiting both the inherent spatial and temporal properties of RF signal data. We designed and implemented three different types of neural network models for this purpose. We collected over-the-air signal data from USRP B210s and used the same to train and validate our system. The RNN model using LSTM cells yields 95% testing accuracy leveraging only temporal property of the signal data. The one using GRU cells does the same with lower space requirement and yields a better testing accuracy of 97.02%. Finally, the RNN model with ConvLSTM cell achieves 98% testing accuracy for the same dataset leveraging both the temporal and spatial correlations.

CHAPTER 5: DEFENSE AGAINST PUE ATTACK USING GAN BASED LEARNING

In this chapter, we present a solution towards the PUE attacks. We present two generative adversarial net (GAN) [32] based models to successfully emulate the primary users (PUs) in two ways. We propose a (i) dumb generator model without any “prior" knowledge of PU’s feature space, (ii) a smart generator model with some “prior" knowledge about PU’s transmission. Any GAN based model works on a synergistic training of two neural networks: (i) a generator, and (ii) a discriminator. Therefore, we use generated malicious entities to train the discriminator for “yet to be seen” real PUE attackers. We propose two deep neural network based discriminator models to discriminate between the PU and the emulated primary users (EPU) from the corresponding generators. Both the generator and discriminator of each GAN model gets smarter with iterative and sequential GAN training. Through a testbed evaluation, we show that discriminators are able to catch $\sim$50% of PUE attackers without the GAN training during the deployment phase. We also observe 100% accuracy for both the GAN models during training phase. Ultimately, after the GAN training, the discriminators achieved 98% and 99.5% accuracies, for dumb and smart generator models respectively, to distinguish “yet to be seen" PUE attacker.

The chapter is organized as follows: section 5.1 proposes a GAN based mechanism to train a centralized module to defend against the PUE attacker. In section 5.2, we present the implementation and experimental results of our proposed approach. The contents of this chapter appeared in [78].

5.1 Proposed GAN Approach

In this section we propose a GAN based approach to present a robust defense mechanism for PUE attack. In a GAN, we have one generator and one discriminator. We propose two GAN based models: (i) one model which will work without any prior knowledge about the PU or the CRN, (ii) the other with the assumption, that the attacker will have prior information about the PUs characteristics for that CRN. So, for the first case we take the SU to work as a dumb emulated primary user (EPU). However, in the second case the EPU is smart to gather or know all the helpful prior information about the PU transmission, so we call those as smart EPUs. Few examples of prior knowledge that an attacker could leverage to mimic the PUs characteristics are: (i) a set of modulation schemes which the PUs use, (ii) the used frequency band, (iii) channel bandwidth for PU's transmission, (iv) geographical location of the PU's transmitter, (v) sample space and sample size of the PU transmission, and so on.

5.1.1 GAN based Problem Formulation for PUEA Defense

A GAN framework comprises of two distinct models: the generator ($\mathcal{G}$) which learns the real data distribution and generates the "mimicked" data, and the discriminator ($\mathcal{D}$) which tries to distinguish the mimicked data from the "real" data by estimating the probability that the sample came from the real data rather than the $\mathcal{G}$. The overall idea is that, during the training phase the generators will pose as a selfish or malicious PUE attacker and make the discriminator stronger in order to fight against the real selfish and malicious PUE attackers during the deployment phase. We call the selfish and malicious PUE attackers as emulated primary users (*EPU*), and the legitimate primary users as *PUs*. We use a centralized spectrum allocator and train it using the GAN model and therefore work as a robust discriminator during deployment.

5.1.1.1 Generative Model

The generative model has two main inputs, (i) prior information about the PU's feature space ($s(t)$), (ii) a additive Gaussian white noise ($n(t)$). The noise is added with the prior information, $z(t) = s(t) + n(t)$. The output $z(t)$ is then fed to the generator which works as a unsupervised learning tool and learns the data distribution ($p_z(z)$) of PU's feature space. Note that we have indexed the prior information, noise and the generated data by time t. This has been done to acknowledge the fact that the signal characteristics can change over time. However for this work we do not consider the effects of variation of the signal characteristics (prior information) and noise with time and essentially assume that $s(t) = s \ \forall t$ and $n(t) = n \ \forall t$. Hence we also get $z(t) = z \ \forall t$. Recall, there are two types of EPUs, one is dumb and another smart. The generator for dumb EPU is deprived of any prior information about the PU's feature space, so $s(t)$ for the generator model of dumb EPU is random and does not reflect the actual signal characteristics of the PU. The cost function of generator is denoted by $V(\mathcal{G})$. The GAN model's target is to minimize this cost, i.e., to minimize the probability of $\mathcal{D}$ correctly identifying the data from $\mathcal{G}$.

5.1.1.2 Discriminative Model

The discriminator is fed with both real data (x) drawn from a data distribution $p_{data}(x)$, and generated data from generator ($z(t)$). The objective of the discriminator is to successfully distinguish between these two types, i.e., to learn the difference between $p_{data}(x)$, and $p_z(z)$. The cost function for discriminator is denoted as $V(\mathcal{D})$. The target of discriminator is to maximize its cost, i.e., to increase probability to correctly identify samples from training examples and data from $\mathcal{G}$. Target of the overall GAN model is to minimize the cost for generator and maximize the cost for discriminator. The overall cost $V(\mathcal{G}, \mathcal{D})$ is formulated as $V(\mathcal{G}, \mathcal{D})$ = $\mathbb{E}_{p_{data}(x)} \log \mathcal{D}(x)$ + $\mathbb{E}_{p_z(z)} \log(1 - \mathcal{D}(\mathcal{G}(z)))$, where $p_z(z)$ is the generator's distribution over generated data samples

z, $p_{data}(x)$ is the data distribution over real data samples x, $\mathcal{D}(x)$ is the probability that x came from $p_{data}(x)$ than $p_z(z)$. $\mathcal{D}(\mathcal{G}(z))$ represents the probability that x came from $p_z(z)$ than $p_{data}(x)$. Objective of the GAN training is:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{G}, \mathcal{D}) \tag{5.1}$$

Throughout the training phase, the GAN framework eventually converges to an unique optimal discriminator for a particular generator, $\mathcal{D}^*(\mathbf{x}) = \dfrac{p_{data}(\mathbf{x})}{p_{data}(x) + p_z(z)}$. It is intuitive that $\mathcal{D}$ is optimal when the discriminator can distinguish between each real sample (x) and generated sample (z). Similarly, it is also deduced that $\mathcal{G}$ is optimal when the $\mathcal{D}$ cannot distinguish between x and z, $\mathcal{G}$ is optimal when $p_z(z) = p_{data}(x)$.

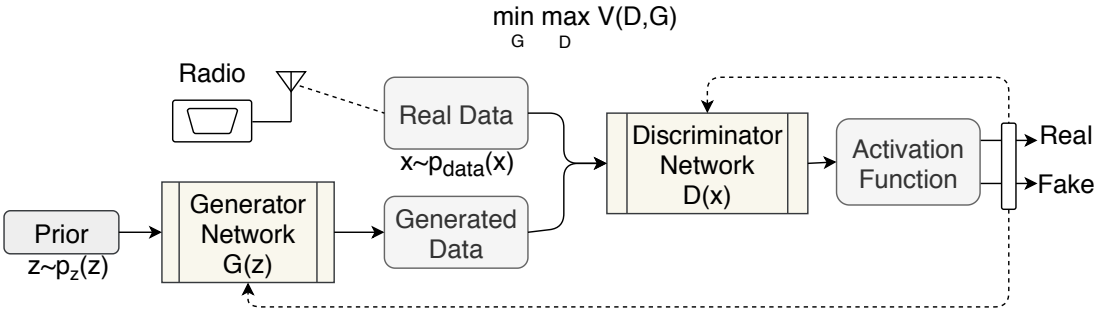


Figure 5.1: Implementation of GAN in RF Domain

5.1.1.3 GAN Architecture

The overall GAN architecture is shown in Fig. 5.1. Once the generator ($\mathcal{G}(z)$) is trained, it generates "mimicked" data from the data distribution ($p_z(z)$) with the prior information. The trained discriminator knows the difference between the mimicked data distribution ($p_z(z)$) and real data

distribution ($p_{data}(x)$), so it will try to distinguish the mimicked data (z) from the real data (x). The activation function at discriminator is *sigmoid*. The feedback from the output is fed to both $\mathcal{G}(z)$, and $\mathcal{D}(z)$. So ultimately the target of overall GAN model is to maximize the discriminator's cost and minimize the generators cost, mathematically formulated as equation (5.1).

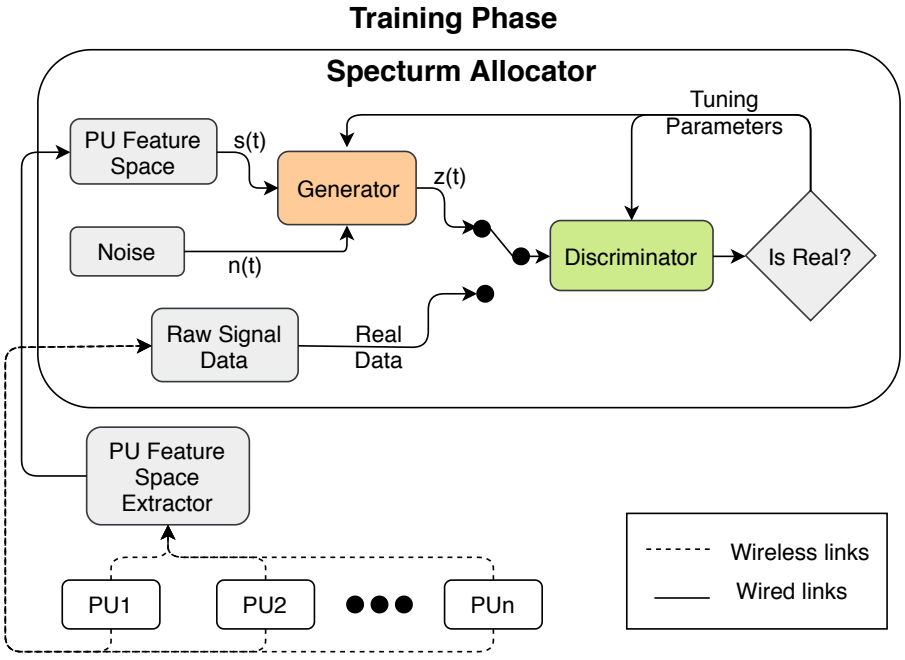


Figure 5.2: GAN Training in the Spectrum Allocator

5.1.2 GAN based Approach for PUEA Defense

The proposed method is compatible with the existing system of centralized spectrum allocation for dynamic spectrum access implementations. The idea is to install a generator and discriminator model inside the spectrum allocator. During the training phase, the discriminator is trained over the generated data and real signal data from the legitimate PUs. The overall proposed training approach is presented in Fig. 5.2. We train the discriminator over the generator model. We consider the generator to be either one of the two types: (i) a generator posing as a dumb EPU, (ii) a generator

posing as a smart EPU. The smart EPUs collect the prior information about the feature space of the PUs from a feature space extractor. The discriminator gets trained over both real (x) and generated data (z) over iterations and become robust enough to distinguish between the real and synthetic data. The output of the discriminator is fed to both the generator and discriminator models so as to tune-up the hyper-parameters of each model, depending on the result. In this way, the generator also gets smarter over each iteration and the discriminator gets smarter than the generator over the next iteration. Thus we design the training in such a way that over time the discriminator eventually over-powers the smartest possible generator model. Once the GAN training is complete, the trained discriminator is deployed in the spectrum allocator.

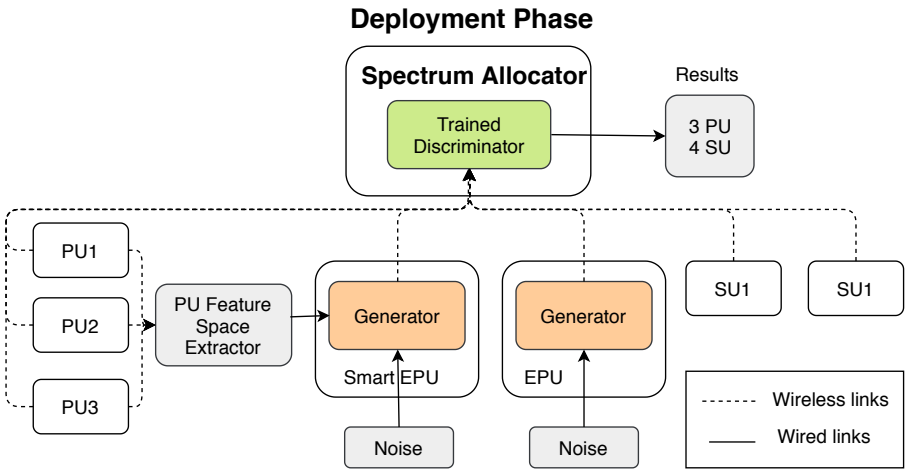


Figure 5.3: Deployment of Trained Discriminator in Spectrum Sensing Scenario

During the deployment phase, the centralized spectrum allocator gets signal information from spectrum sensors, and pass them through the trained discriminator before the spectrum is allocated. The overall proposed approach for deploying the trained discriminator is shown in Fig. 5.3. In the figure we consider 3 legitimate PUs, 1 dumb EPU, 1 smart EPU, and 2 SUs. The dumb EPU does not have any prior knowledge about PU's feature space, but the smart one has. The trained discriminator is able to recognize the 2 EPUs as SU, despite the effort of the malicious entities.

5.2 Implementation and Results

For implementing the GAN model, we use the proposed generator and discriminator models with data collected from over-the-air RF transmission. Next we describe the data collection, experimental setup, implementation details, and experimental results.

5.2.1 Used Dataset and Experimental Setup

We use a dataset of raw I/Q values from 8 software defined radios (universal software radio peripheral (USRP) B210 [26]). The details of I/Q data generation and data collection mechanism is similar to [79], and [76]. As described in earlier chapters, each I/Q dataset comprises T training samples (for T timestamps) and a sample size of N, where each sample is a vector $(I, Q) \in \mathcal{C}$ representing a number in the complex plane. Each vector at timestamp t is represented as: $x(t) = [[(I, Q)_i]^t; i = 1, 2, \cdots, N] \in \mathcal{C}^N$ for timestamp $t = 1, 2, \cdots, T$. We use these vectors as input during GAN training. Note that though we collect the data with respect to different time stamps, we do not treat the data thus collected as a time series for this work, that is we *ignore* the temporal aspect of the data.

We conducted the testbed evaluation on a Ryzen 8 Core system with 64 GB RAM, a GTX 1080 Ti GPU unit, and 11 GB memory. We used different machine learning libraries to design the proposed GAN models. We useu *Keras* [18] as the frontend and *Tensorflow* [1] as the backend for the neural network architectures used in GAN. We also use *Numpy*, *Scipy*, and *Matplotlib* python libraries for implementation of different operations.

5.2.2 GAN for Dumb PUE Attacker

In this case, the generator has no prior information. The generator starts by randomly generating data within the sample space $[X, Y]$, where X and Y are random numbers. For our experiments, we took X as 0 and Y as 1. This dumb generator picks a random sample size N. The randomly generated data is then used as input to three *dense* layers of sizes $N/2$, N, and $2N$ respectively with *tanh* activation function in the first two layers. The third neuron layer uses *sigmoid* activation function, and generates data of size $2N$.

The generator $\mathcal{G}(z)$ continues to learn the data distribution ($p_z(z)$) and generates fake samples of size $2N$ within a random sample space. The discriminator ($\mathcal{D}(x)$) for the dumb generator is simple. We first have one input layer of $2N$ nodes, which is followed by two hidden layers of N and $N/2$ nodes respectively. Ultimately, a *softmax* output layer of 2 nodes is used to identify whether the input is from a legitimate `SU` or malicious `EPU`. We use *tanh* as activation function at the hidden layers and add *Dropout* [89] of 0.5 in between those layers to avoid overfitting. We also use $l1$ regularization for the same purpose. The output from the last layer is fed to the layers of generator as well as discriminator in the next epoch for tuning the parameters of both. One epoch is actually a combination of a forward and a backward pass through the designed model over the entire dataset. The overall GAN implementation using dumb generator is shown in Fig. 5.4. In this case, the $\mathcal{G}(z)$, and $\mathcal{D}(x)$ are respectively 3-layered and 5-layered networks. In the figure, the solid lines represent the connection between two layers, whereas the dotted lines represent the feedback from output layer of the discriminator to the other layers of both generator and discriminator to be used in next epoch.

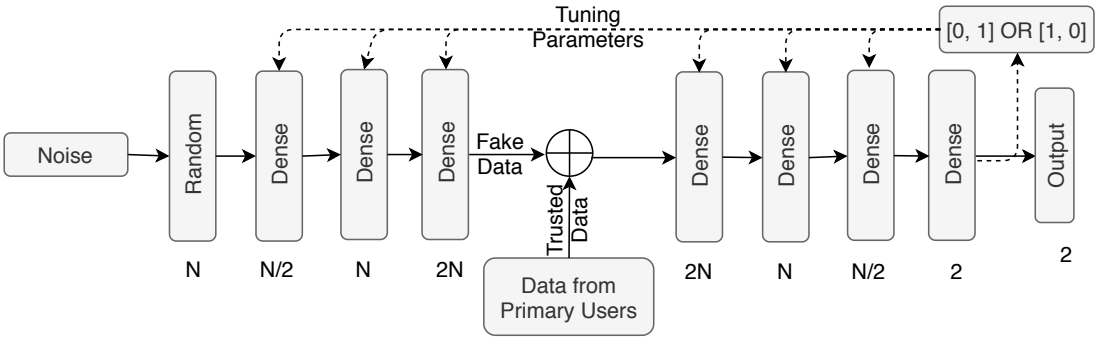


Figure 5.4: GAN Implementation for Dumb Emulated Primary User

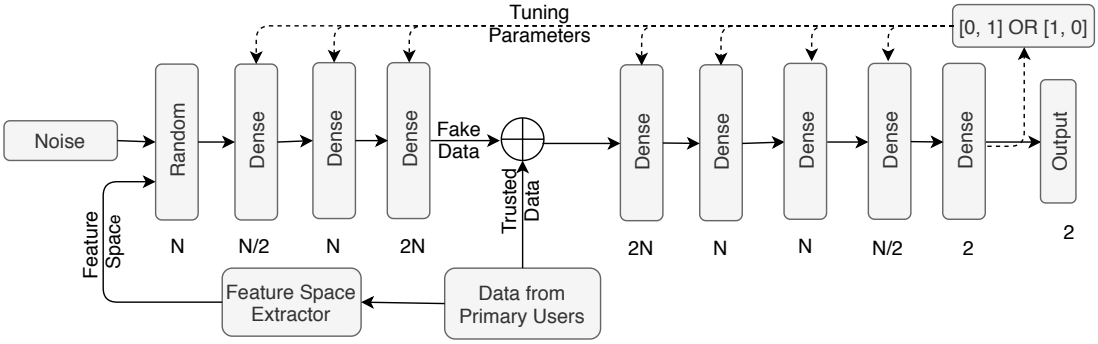


Figure 5.5: GAN Implementation for Smart Emulated Primary User

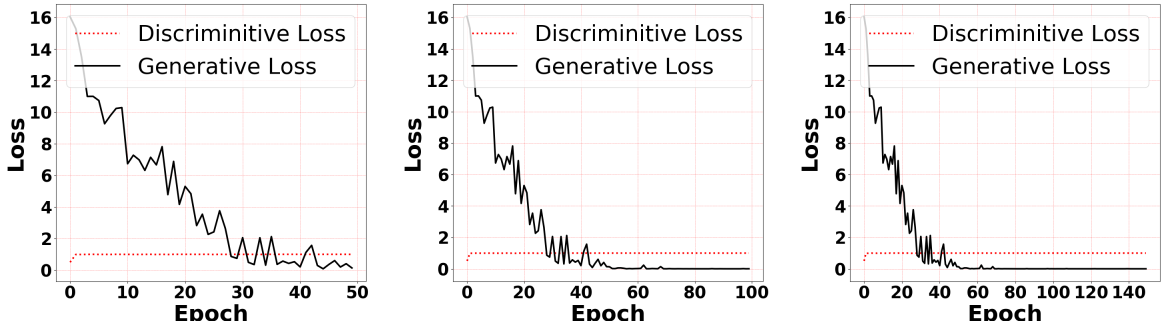
5.2.3 GAN for Smart PUE Attacker

In this case, generator $\mathcal{G}(z)$ has prior knowledge about some features of PUs. We get a sample of size 1024 and sample space of [-1, 1] from the PU feature extractor. These features eventually help the EPUs to better mimic legitimate PUs. In this set of experiments, we get the sample size (N) and the sample space from the "feature space extractor". Feature space extractor is a module in the spectrum allocator, which extracts features and builds a feature space for the PUs from the sensed signal data. The GAN implementation with a smart generator is shown in Fig. 5.5. For the sake of consistency and generality, we use the same number of layers as the dumb one, for modeling the generator of smart EPU. However, knowing that the generator is smart, we design the discriminator in a way that it is 1-layer more deep than the one for the dumb one. We use 2

dense layers with N nodes instead of 1. So now the $\mathcal{D}(x)$ becomes a 6-layered network. The rest of the generator and discriminator properties are similar as before.

5.2.4 Experimental Results

We perform two sets of experiments here: (i) first we train the GAN model on a dumb generator and use the discriminator as discussed in section 5.2.2, (ii) next we train our proposed model on the smart generator and use a discriminator as described in section 5.2.3. For both the cases we performed the training on the data collected from the authentic PUs. We notice that the discriminator trained on the dumb generator was able to detect the EPUs with 49.22% accuracy before the GAN training. Whereas, the discriminator trained on smart generator was able to do that with 50.15% accuracy. This slight improvement is due to the one extra layer of neurons in the smart discriminator. However, the overall achieved accuracy is far below expectations. The discriminator is naive and can distinguish the EPUs from the legitimate PUs only with $\sim 50\%$ of accuracy, which is similar to what one would obtain for random guessing.



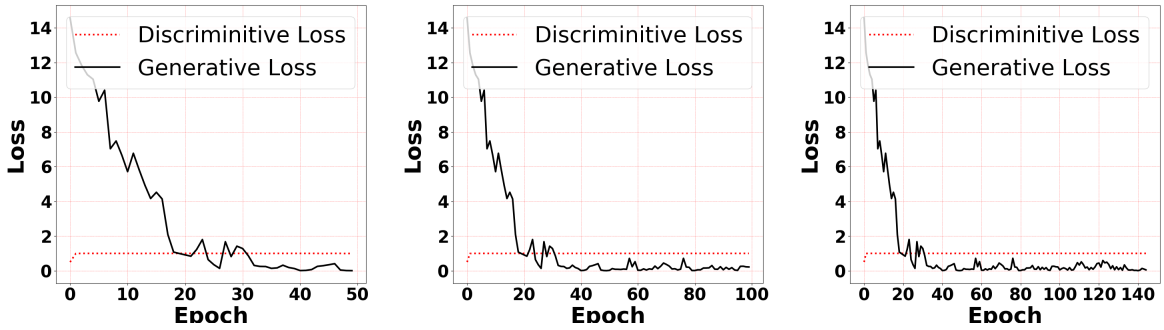
(a) 50 Epochs (b) 100 Epochs (c) 150 Epochs

Figure 5.6: Generator and Discriminator Loss of GAN Model with Dumb EPU

5.2.4.1 Training Phase

We train both the generator and discriminator through iterative sequential learning to strengthen the generative and discriminative model over time. We use *categorical cross-entropy* training on *Adam* [42] optimizer for *gradient based optimization*. We use 90%, 5%, and 5% of the total data for training, cross-validation and testing respectively.

We experimented with several training paradigms for training the GAN models in order to get the best performance from it. This leads us to train both the models in a 3-step approach. We first train the models for 50 epochs with learning rate of 10^{-4} and 10^{-3} for generator and discriminator respectively. Next, we train the models for another 50 epochs with lower learning rates of 10^{-5} and 10^{-4} for $\mathcal{G}(z)$ and $\mathcal{D}(x)$ respectively. In the last 50 epochs, we decrement both the learning rates by $1/10$ again. Notice that in each epoch, the learning rate of generator is lower than the discriminator, as we want the generator to learn precisely about the possible data distribution ($p_z(z)$) of PUs for better EPU emulation. However, we give the discriminator more layers to be able to accurately learn the decision boundary in order to eventually overpower the generator. It is clear from the Fig. 5.6, that the generator's loss starts to decrease and the discriminator's loss starts to increase at the beginning. However, after a certain number of epochs, both the losses saturate. The generator-loss, with the dumb generator fluctuates more and reaches saturation later (after 30 epochs); whereas the generator-loss with smart generator reaches saturation within less than 20 epochs, as shown in Fig.5.7. This result bolsters the intuition that knowing some of the PUs features will make the generators smart at emulating the PUs and this in turn will make the system converge faster. The discriminator's behavior is more or less stable for both the models. Once both the models are trained, we proceed to the deployment phase.



(a) 50 Epochs (b) 100 Epochs (c) 150 Epochs

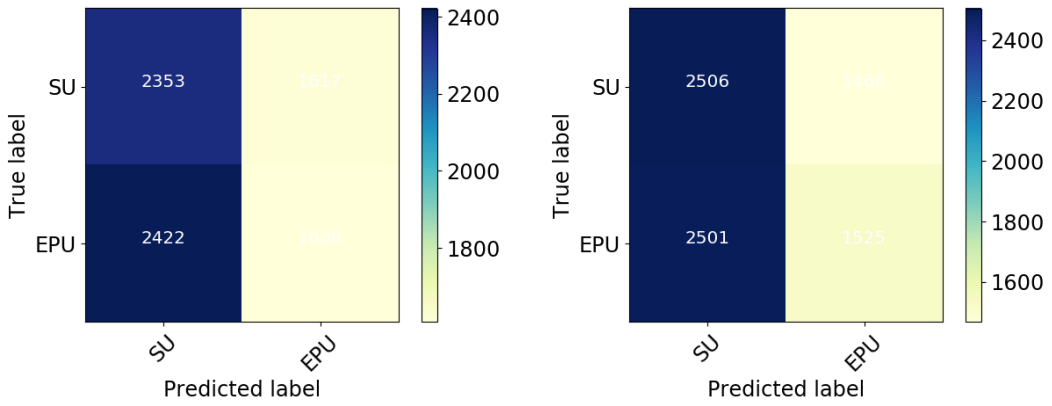
Figure 5.7: Generator and Discriminator Loss of GAN Model with Smart EPU

5.2.4.2 Deployment Phase

During the deployment phase, we randomly choose over-the-air signal data from EPUs and SUs. We collect 4000 signal data from legitimate SUs and 4000 signal data from EPUs. We programmed USRP B210s [26] to work as SUs and EPUs. These EPUs use same sample size and sample space as the PUs, making themselves indistinguishable to the normal discriminator (trained without GAN). We observe $\sim$ 50% EPU detection rate prior to GAN training. However, after the discriminator is trained and learned the data distribution ($p_z(z)$) of the EPUs, we get better detection rate. We achieved a 98.04% accuracy from the discriminator which was trained using the dumb generator and 99.5% accuracy from the discriminator which was trained using the smart generator. These results are presented in Table 5.1.

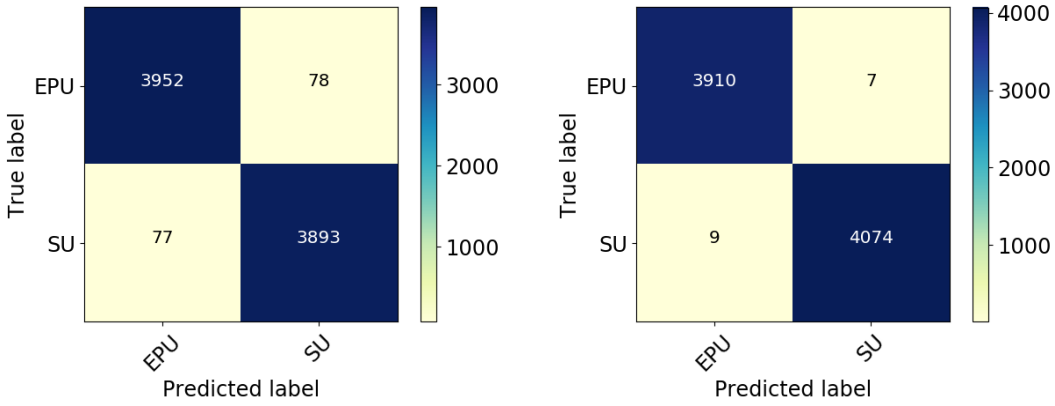
Table 5.1: Accuracies for Different Implementations

	Before GAN Training	After GAN Training
$\mathcal{D}$ with Dumb $\mathcal{G}$ Training	49.22%	98.04%
$\mathcal{D}$ with Smart $\mathcal{G}$ Training	50.15%	99.5%



(a) Dumb Generator (b) Smart Generator

Figure 5.8: Confusion Matrices with Dumb and Smart EPUs: before GAN Training



(a) Dumb Generator (b) Smart Generator

Figure 5.9: Confusion Matrices with Dumb and Smart EPUs at Deployment Phase: after GAN Training

5.2.4.3 Performance Analysis

Using accuracy as a measure of efficacy of an algorithm can sometimes be incomplete and misleading depending on the type of data, such as the case for skewed data. A confusion matrix overcomes those problems by showing a relative relation between false positives and false negatives and actual data labels. Hence we present the confusion matrices for both the experiments of deployment phase, in Fig. 5.8, and 5.9. It is evident from Fig. 5.8, that almost all of the signals are

predicted to be transmitted from SU, giving an accuracy of ∼50%. So, the discriminator is not able to distinguish between EPU and SU, thus predicting all as SU. However, the confusion matrices after the GAN training has lower false positive and false negative rates. We also notice that the false positives and false negatives are higher in case of the GAN model which was trained over dumb generator. In summary,

- We achieve ∼50% accuracy of EPU detection using the discriminator trained without the GAN for both type of proposed models, during deployment phase.
- During training, we observe 100% training accuracy for both dumb and smart GAN models.
- The trained discriminator with dumb and smart generator training gives testing accuracy of 98% and 99.5% respectively during the deployment phase.
- The proposed GAN based model can be applied for any type of cognitive radio transmitter irrespective of PU or SU's properties. We achieve 98-99% accuracy of EPU detection after the dumb generator training. So without any knowledge of PU properties, the proposed model is capable of detecting PUE attackers with 98% accuracy. Some "prior" information can boost up the accuracy to 99.5%.

5.3 Summary

In this chapter we present a robust defense mechanism against PUE attack in a cognitive radio network. We design GAN based models considering (a) no prior information and (b) prior information about the PUs. We call them as dumb and smart GAN models respectively. Through testbed evaluations, we show that the GAN training for both kind of generators give a competitive accuracy for EPU or PUE attacker detection during the deployment phase. However, GAN model with smart

generator training achieves better accuracies and faster saturation than the dumb one. Both models are able to detect the malicious and selfish PUE attacker with more than 98% of accuracy. Extending this concept towards providing security for other issues in wireless communication, could be one of the next steps to consider.

CHAPTER 6: PRIMARY USER'S ACTIVITY PREDICTION

In this chapter, we present recurrent neural network models which are able to accurately predict the primary users' activity in dynamic spectrum access (DSA) networks so that the secondary users can opportunistically access the unused spectrum. Using Universal Software Radio Peripheral (USRP) Software Defined Radios (SDRs), we collect over-the-air data from 8 primary users and train the learning models that we use in conjunction with a central spectrum sensor. We start by implementing two machine learning models: (i) traditional linear regression and (ii) neural network model using Long Short Term Memory (LSTM). These models are able to predict the primary users' activity with 75% and 97% accuracy respectively. To further improve the prediction accuracy, we exploit the spatio-temporal correlation in the collected data by implementing a Convolutional LSTM model– which achieves 99% accuracy for predicting the long-term activity of primary users. The experimental results demonstrate that the proposed models are able to successfully predict the primary users' activities, thereby reducing both the under-utilizations and interference violations in DSA networks. To the best of our knowledge, this dissertation is the first one to propose a strategy for long term prediction of the activity of the primary user [100].

This chapter is organized as follows: we present the system model and problem formulation in section 6.1 and different prediction models in section 6.2. The testbed setup and experimental results are presented in section 6.3. The contents of this chapter appeared in [80].

6.1 Problem Description

In this section, we lay out the assumptions, present the system model, and formulate the problem.

6.1.1 Primary User Activity Pattern

The primary users alternate between ON (busy) and OFF (idle) periods. Random variables r_{on} and r_{off} determine the duration of ON and OFF periods respectively. The probability distribution of r_{ON} and r_{off} depends on the specific activity pattern of the primary user. We assume that the primary user's ON and OFF times are independently and identically distributed and use a Poisson point process to model them. Thus we model the probability distributions for the ON and OFF times of the primary users as:

$$f_{on}(r_{on};\beta_{on}) = \begin{cases} \frac{1}{\beta_{on}} e^{-\frac{r_{on}}{\beta_{on}}} & r_{on} \geq 0 \\ 0 & r_{on} < 0 \end{cases} \tag{6.1}$$

$$f_{off}(r_{off};\beta_{off}) = \begin{cases} \frac{1}{\beta_{off}} e^{-\frac{r_{off}}{\beta_{off}}} & r_{off} \geq 0 \\ 0 & r_{off} < 0 \end{cases} \tag{6.2}$$

where, β_{on} and β_{off} are the mean ON and OFF times respectively.

We define the *activity factor* of the primary users as the ratio of the mean PU ON time to the sum of the mean of PU ON and OFF times and thus this is given by:

$$PU_{activity} = \frac{\beta_{on}}{\beta_{on} + \beta_{off}} \tag{6.3}$$

It must be noted that, it is not required to pre-define a model for learning; the proposed prediction models are able to learn any kind of PU activity pattern. We considered the Poisson process for

our testbed experiments.

6.1.2 System Model

We assume that PUs and SUs co-exist in a geographical area. The SUs can utilize the spectrum bands when they are not being used by the PUs. The PUs have priority and can transmit when they need to. On the other hand, SUs always have backlogged traffic to transmit and must yield to the PU as soon as they require the spectrum. The transmission times for the PUs obey the pattern (distribution) as discussed earlier. As a result of this system model, prior knowledge of PU activity is vital for SUs to effectively and efficiently share the spectrum. We assume that there is a centralized spectrum sensor that monitors the PU transmission activities and maintains records of all past observations. The trained models corresponding to the proposed neural network architectures are co-located with the spectrum sensor and use the past observations as inputs in order to predict the activity pattern of the PUs.

6.1.3 Problem Formulation

The trained neural network model predicts the primary user's activity at time τ for the next Γ timestamps and the SS sends that information to the secondary user. Depending on the prediction information received from the spectrum sensor, the secondary user schedules its transmission for the next Γ timestamps, thus minimizing the interference violation as well as the under-utilization.

We denote the states of the primary user's and secondary user's transmission as s_{PU} and s_{SU}. The ON state of each is represented as 1, and OFF state as 0. We also denote tolerable thresholds for interference violation and under-utilization as I_{thr}, and U_{thr} respectively.

At time τ, the trained model predicts the PU activity for the next Γ timestamps. In our system we

let the SS inform the secondary user when it sees continuous PU inactivity for more than 50% of the predicted Γ timestamps. Note that this is a *heuristic* for solving the underlying optimization problem as given by equation 6.4. However in the most general setting, at the time τ, depending on the information received from the SS, the secondary user sets its state s_{SU} to 0 or 1 for the next Γ timestamps. Let $I_{ch}(t)$ denote the indicator variable at time t ($\tau < t \leq \tau + \Gamma$), indicating whether the secondary user causes interference to the primary user on channel ch. Note that: $I_{ch}(t) = s_{PU}(t) \times s_{SU}(t)$. Similarly, let $U_{ch}(t)$ denote an indicator variable representing the under-utilization of ch channel at time t. Notice that: $U_{ch}(t) = \overline{s_{PU}(t) + s_{SU}(t)}$, where $+$ denotes the logical *OR* operation. $s_{PU}(t)$ and $s_{SU}(t)$ represent the PU's and SU's states respectively at time t. We note that a conservative strategy for dynamic spectrum allocation will result in under-utilization, whereas an aggressive strategy will lead to a considerable amount of interference violation, as discussed in connection to Fig. 1.2.

The secondary user can transmit on the channel when the primary user is in the OFF state (i.e., s_{PU} = 0). It should terminate its transmission as soon as the primary user starts to transmit (i.e., s_{PU} = 1). However, while the SU is transmitting ($\tau < t \leq \tau + \Gamma$), it will not be aware of this state change of the PU (that is it will not be able to detect the change in the $I_{ch}(t)$ states). Thus the possibility of having accurate knowledge of $I_{ch}(t)$ will depend on how well the neural network can model the underlying probability distribution of the activity of the primary user.

Finally, our objective is to maximally assign the secondary user to the ON state constrained over the thresholds of interference violation (I_{thr}) and under-utilization (U_{thr}). Thus, the general problem can be formulated as follows: let the number of timestamps where the SU is active be denoted by η where η can takes values between $\tau + 1$ and $\tau + \Gamma$. Then the problem can be formulated as that of maximizing η subject to the constraints on the interference and under-utilization being

bounded above by the respective thresholds. Thus we have:

$$\begin{aligned} &\max \eta \\ \text{subject to} &\sum_{t=1}^{\eta} I_{ch}(t) \leq I_{thr} \\ &\sum_{t=1}^{\eta} U_{ch}(t) \leq U_{thr} \end{aligned} \tag{6.4}$$

6.2 Proposed RNN Based Prediction Models

We present two types of neural network models that leverage the temporal and spatio-temporal correlation in the historical data of the PU activities, in order to predict future PU activities. We take the I/Q values of over-the-air signal data as raw features for future state prediction, as the I/Q values do not require further sophisticated signal processing. Hence they are capable of providing an end-to-end solution for our problem using machine learning techniques.

6.2.1 Recurrent Neural Network Model

Fully-connected and convolutional neural networks that are traditionally used for deep learning lack the capability to exploit the context available in temporal data. Additionally, there is the problem of *vanishing gradients*, when trying to use *back propagation* with temporal data. Both these problems are addressed by Recurrent Neural Networks (RNN) [45] which we now describe in brief in the context of our problem.

6.2.1.1 Temporal Property of I/Q data

Given T training samples (for T timestamps) where each sample is a vector of size M, and each component of the vector is a tuple $(I, Q) \in \mathcal{C}$ representing a number in the complex plane, we represent this vector for a given time stamp t as $x_t = [[(I, Q)_i]^t; i = 1, 2, \cdots, M] \in \mathcal{C}^M$ where $t = 1, 2, \cdots, T$, and use it as an input to the neural network. We use a sample size (M) of 1024 as a default for our experiments. We want to find the probability of $s_{PU}(t) = 0$ for the next input vector (x_{t+1}) for $t = t + 1$. The probability $P(s_{PU}(t) = 0|x_{t+1})$ can be written as:

$$P(s_{PU}(t) = 0|x_{t+1}) = \frac{P(x_t|s_{PU}(t) = 0)P(s_{PU}(t) = 0)}{P(x_t|s_{PU}(t) = 0)P(s_{PU}(t) = 0) + P(x_t|s_{PU}(t) = 1)P(s_{PU}(t) = 1)} \tag{6.5}$$

where $P(x_t|s_{PU}(t) = 0)$ and $P(x_t|s_{PU}(t) = 1)$ are the conditional probabilities of x_t given $s_{PU}(t)$ was set to 0 and 1 respectively. $P(s_{PU}(t) = 0)$ and $P(s_{PU}(t) = 1)$ are the marginal probabilities, values of which will depend on the activity factor of the primary user. The predicted value of vector x at the next timestamp ($t + 1$) will depend on the predicted value of the current one [87], which is given by:

$$\tilde{x}_{t+1} = \underset{x_{t+1}}{arg\,max}(P(x_{t+1}|\tilde{x}_t)) \tag{6.6}$$

The details of used LSTM cell models are described in Section 3.3.3.1 in Chapter 3. We next describe the overview of convolutional LSTM network.

6.2.2 Convolutional Recurrent Neural Network

The recurrent neural networks using LSTM cells do not consider the spatial information encoded in the input-to-state or state-to-state transitions [87]. However, there are many applications where spatio-temporal correlation exists within the dataset. To address this issue, we use a convolution within the recurrent structure of the RNN. We first discuss the spatio-temporal property of RF data and then model a convolutional LSTM cell to exploit the same.

6.2.2.1 Spatio-temporal Property of I/Q Data

Suppose that a radio signal is represented as a time varying series over a spatial region using R rows and C columns. Here R represents the time varying nature of the signal and as such in our case it represents the total number of time stamps at which the signal was sampled (T in our case). C on the other hand represents the total number of features sampled at each time stamp (in our case its 2048 since there are 1024 features sampled each of dimension 2). Note that each cell corresponding to one value of R and one value of C represents a particular feature (I or Q) at a given point in time. In order to capture the spatial variation of the signal we need to consider samples from the signal at consecutive time stamps. In our case we consider time intervals of length γ and hence we represent this by sub-matrices of the original $R \times C$ matrix representing the whole time varying signal. Each of these sub-matrices have γ rows corresponding to the selected timestamps and C columns. Thus in a nutshell: to capture the temporal property only, we made use of a sequence of vectors for timestamps $1, 2, \cdots, T$, namely, $x_1, x_2, \cdots, x_T$ whereas now, to capture both spatial and temporal properties, we introduce a new vector $\chi_{t,t+\gamma}$, which is formulated as: $\chi_{t,t+\gamma} = [x_t, x_{t+1}, \cdots, x_{t+\gamma-1}]$. So the vector $\chi_{t,t+\gamma}$ eventually preserves the spatial properties with an increment of γ in time. Thus, we get a sequence of new vectors $\chi_{1,\gamma}, \chi_{\gamma,2\gamma}, \cdots \chi_{t,t+\gamma}, \cdots, \chi_{t+(\psi-1)\gamma,t+\psi\gamma}$, where ψ is $\lfloor R/\gamma \rfloor$. We formulate the probability of pri-

mary user to be idle ($P(s_{PU}(t) = 0|\chi_{t,\,t+\gamma})$) for the next γ-length sequence as:

$$P(s_{PU}(t) = 0|\chi_{t,\,t+\gamma}) = \frac{P(\chi_{t-\gamma,\,t}|s_{PU}(t) = 0)P(s_{PU}(t) = 0)}{P(\chi_{t-\gamma,\,t}|s_{PU}(t) = 0)P(s_{PU}(t) = 0) + P(\chi_{t-\gamma,\,t}|s_{PU}(t) = 1)P(s_{PU}(t) = 1)} \tag{6.7}$$

The marginal probabilities for the primary user to be idle or busy are modeled as $P(s_{PU}(t) = 0)$ and $P(s_{PU}(t) = 1)$ respectively. $P(\chi_{t-\gamma,\,t}|s_{PU}(t) = 0)$, and $P(\chi_{t-\gamma,\,t}|s_{PU}(t) = 1)$ are the conditional probabilities of $\chi_{t-\gamma,\,t}$ given $s_{PU}(t)$ was set to 0 and 1 respectively. The predicted value of γ-length sequence vector χ at timestamp t will depend on the predicted value of the previously predicted γ length sequence [87], which is given by:

$$\tilde{\chi}_{t,\,t+\gamma} = \underset{\chi_{t+1}\cdots\chi_{t+\gamma}}{arg\,max}(P(\chi_{t,\,t+\gamma}|\tilde{\chi}_{t-\gamma,\,t})) \tag{6.8}$$

It must be noted that in our case $\gamma = \Gamma$ where Γ is as described in Section 6.1.3.

6.2.2.2 The ConvLSTM Model

We model the ConvLSTM cell as presented in Fig. 6.1. It is similar to an LSTM cell, but the input transformations and recurrent transformations are both convolutional in nature [87]. We formulate the input values, cell state and hidden states as 3-dimensional vectors, where the first dimension is the number of features ($C/2$) and varies over time, and the last two dimensions contain the spatial information (rows (R) and columns (C)). The details of used ConvLSTM cell models are described in Section 4.1.2.2 in Chapter 4.

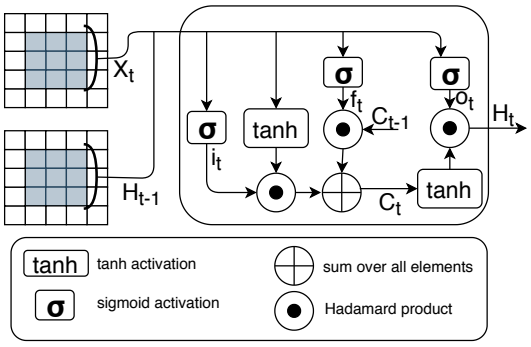


Figure 6.1: ConvLSTM Cell Architecture Used in the Proposed RNN Model

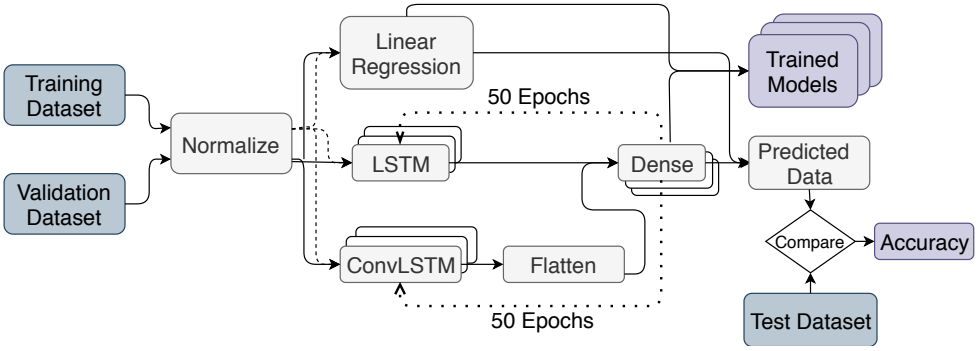


Figure 6.2: PU Activity Prediction Training of the Spectrum Sensor

6.2.3 Proposed PU Activity Prediction Model

We propose two models based on recurrent neural networks to train the spectrum sensor using the historical data of primary user activities. The objective is to learn every primary user's activity pattern and use it for scheduling SU activity. We also use the traditional linear regression model for prediction in order to test the performance of the neural network based models against classical techniques. We present the training strategy of each model in Fig. 6.2. First, we divide the dataset into subsets for training and validation. In each epoch, we validate the trained model on the validation data and depending on the validation error we tune the hyper-parameters of each model. A single epoch consists of a forward pass and a backward pass through the implemented architec-

ture for the entire dataset. We normalize the data for both training and validation to balance each feature. We design three different models using three different approaches on the normalized data.

The details of the RNN with ConvLSTM cells is presented in Fig. 6.3. The first two hidden layers consist of *ConvLSTM* cells with 1024 and 256 filers respectively. The last two hidden layers are *Dense* layers, applied after flattening the output from the last *ConvLSTM* layer. We also apply Dropout [89] of 0.5 between the different layers to avoid over-fitting of the trained model. The LSTM implementation is also similar, with the exception of not having the *Flatten* layer in between the *LSTM* and *Dense* layers. Increasing the number of filters or layers did not help in achieving higher accuracy for either of the models. The output layer consists of a fully connected *Dense* layer with 1 neuron to generate the predicted value. We run and validate the training data for 50 epochs for both the LSTM and ConvLSTM models, beyond which we did not notice any further improvement. During the testing phase, once we get the predicted values, we compare them with the known values for the test/validation dataset and estimate the accuracy of the trained models.

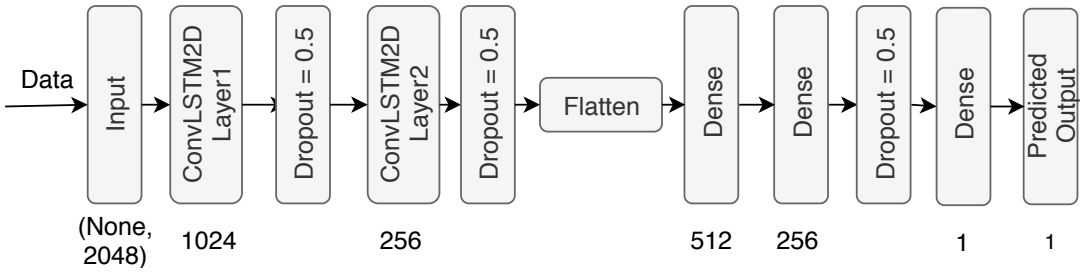


Figure 6.3: RNN Implementation with ConvLSTM Cells for PU Activity Prediction

Next we present the conceptual overview of the deployment phase of the proposed prediction models for a cognitive radio network as shown in Fig. 6.4. The trained models are deployed within the spectrum sensor. The SS also collects and stores the historical data of all the primary users for enough time to build a robust trained model. This trained model also gets updated on the newly learned primary user's activity after some specific duration of time. Determining the

data collection time and model updating time are application specific. Once the spectrum sensor can get a primary user's absence prediction from the trained model, it relays the information to the secondary users. The secondary users later access the idle channel in a cooperative manner, details of which is beyond the scope of this article. Now we are ready to describe our experimental results.

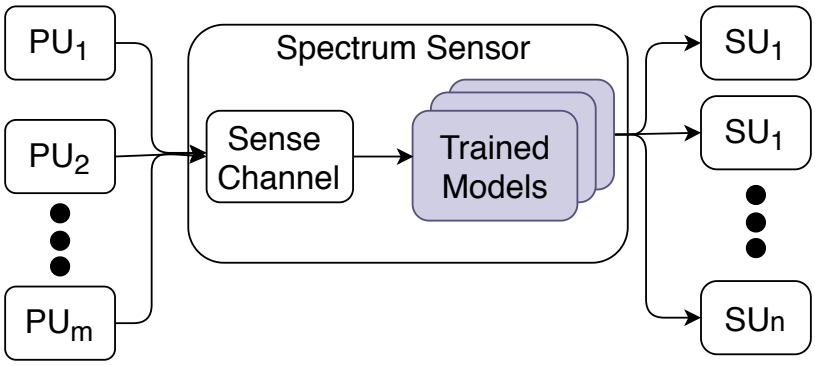


Figure 6.4: Deployment of the Proposed PU Prediction Model

6.3 Implementation and Results

In this section we present the details of our implementation and the experimental results.

6.3.1 Experimental Environment

In order to validate the proposed models, we collected over-the-air data in an indoor lab environment from 8 universal software radio peripheral (USRP) B210s [26] acting as primary users. We name these 8 radios as PU#1 to PU#8. We collected the dataset on an i7 machine with 16 GB RAM. We conducted the proposed model training and evaluation on a Ryzen 8 Core system with 64 GB RAM, a GTX 1080 Ti GPU unit with 11 GB graphics memory.

Table 6.1: Collected Dataset Sizes for Different Primary Users

Primary User	Size (GB)
PU#1	3.26
PU#2	3.5
PU#3	3.29
PU#4	3.2
PU#5	3.14
PU#6	3.16
PU#7	3.1
PU#8	3.22

6.3.2 Data Collection Environment

The data collection environment is presented in Fig. 6.5. A random signal is generated using GNURadio [29] and modulated with Quadrature Phase Shift Keying (QPSK). We programmed the USRP B210s to alternate between ON or OFF states such that the activity factor remained between 0.7 and 0.75. Thus, the collected dataset has more PU presence data than absence, which helps the learning process. The duration of ON and OFF times follows the model described in Section 6.1.1. We generated datasets of different sizes for different PUs. Collected dataset sizes are presented in Table 6.1.

The transmitted signal is received by a RTL-SDR [60] that used the *rtlsdr* python library. The AND gate (in Fig. 6.5) after the primary user block, is used to represent the fact that either the noise or the signal from primary user is transmitted at a particular timestamp. Primary user's OFF time is the absence of radio signal data, therefore it is represented as "Noise". We generate different datasets for all the primary users. The "over-the-air" transmission data was collected in an indoor lab environment where the B210 transmitters and the RTL-SDR receiver were at a distance of 10 feet with a direct line of sight. Thus, the underlying channel can be best modeled as a Rician fading channel. There was also multi-path effects due to the reflections from the walls.

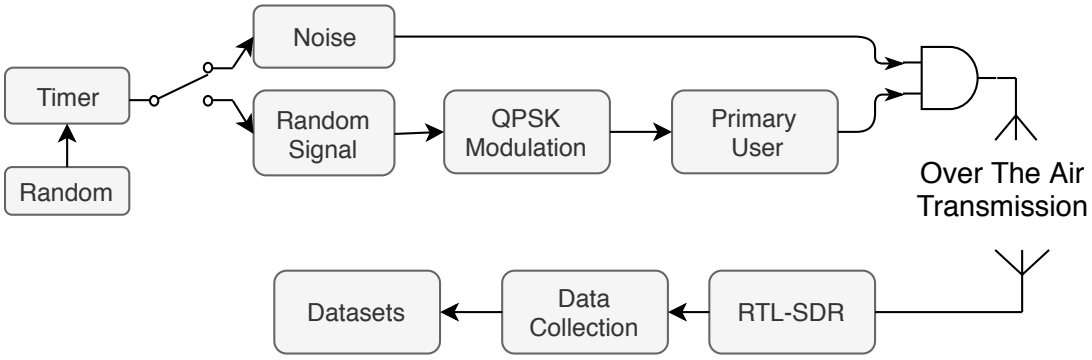


Figure 6.5: Data Collection Procedure for each Primary User

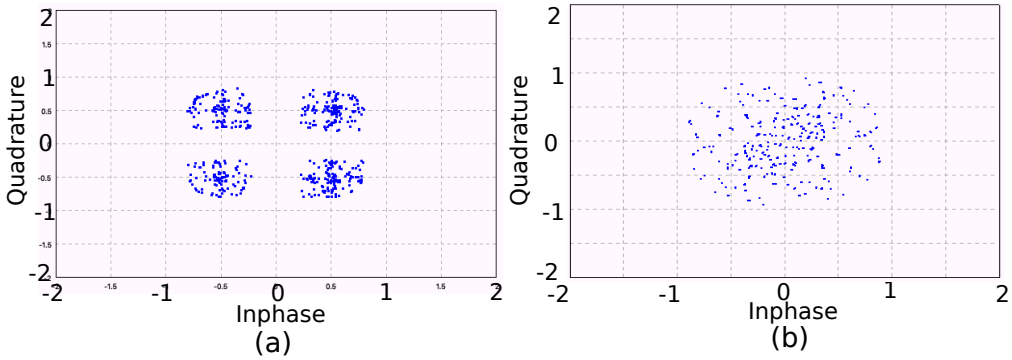


Figure 6.6: I/Q Values Representation: (a) Signal Present (b) Noise

During the primary user's ON time, the I/Q values were organized in a constellation (please refer to Fig. 6.6(a)). During the primary user's OFF time the I/Q values are random (please refer to Fig. 6.6(b)). Each data sample had 2048 entities consisting of 1024 I and 1024 Q values. We chose 1024 as sample size as it was sufficient to capture the spatial properties and at the same time the training was not computationally intensive. The configuration parameters for the radios are given in Table 6.2.

Table 6.2: Primary User Transmission Configuration Parameters

Parameters	Values
Transmitter Gain	45 dB
Transmitter Frequency	904 MHz (ISM)
Bandwidth	200 KHz
Sample Size	1024
Samples/Transmitter	Variable
Primary Users	USRP B210
Receiver	RTL-SDR
# Primary Users	8

6.3.3 Signal to Noise Ratio of Data Collection Environment

To measure the signal to noise ratio (SNR) of the testbed environment, we use a RTL-SDR [60] dongle and Spektrum [69] which is an open source spectrum analyzer available for both Windows and Linux. The screenshots of Spektrum are shown in Fig. 6.7 and 6.8 when the signal is absent and present respectively. We calibrate the SNR using the Spektrum software (rather than using a spectrum analyzer) in order to avoid the associated costs and also in order to show the robustness of our methods to imprecise measurements (as measurements in software are always inferior to actual hardware measurements). From Fig. 6.7, we found that the noise floor was between -20 dB and -30 dB. Fig. 6.8 shows that the signal strength for the 200 KHz (from 904.9 MHz to 904.1 MHz) channel was between 0 dB and 10 dB. We set the transmitter gain to 45 dB and calculated the SNR as the difference between the noise floor and the signal strength. Our calculated SNR was 5 dB - (-25 dB) = 30 dB, with a 45 dB transmitter gain. It is to be noted that the signal strengths (in dB) of noise and signal measured by Spektrum is relative, but the difference between them is absolute.

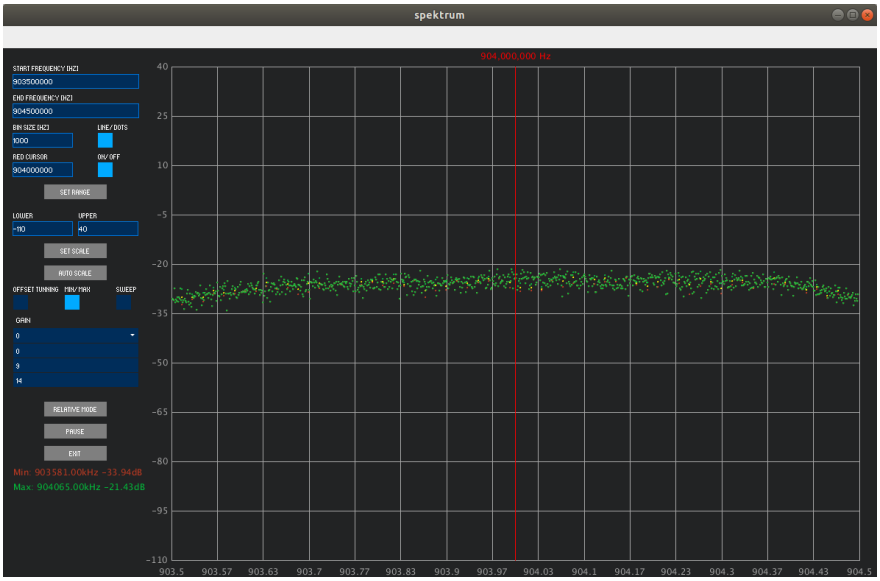


Figure 6.7: Noise Floor Plot using Spektrum [69] Software

6.3.4 Used Machine Learning Libraries and Performance Metrics

We used *Keras* [18] as the frontend and *Tensorflow* [1] as the backend for desiging the proposed neural network models. Keras is an overlay on neural network primitives with Tensorflow [1] or Theano [2] that provides a customizable interface for quick deployment of complex neural networks. We also use *Numpy*, *Scipy* and *Matplotlib* Python libraries for linear regression and other traditional methods.

"Accuracy" is used as the typical performance metric to measure the effectiveness of the proposed neural networks. However, accuracy can sometimes be misleading and incomplete when the data is skewed. In our dataset, the total PU ON time is greater than OFF times, making the dataset skewed. A confusion matrix overcomes this problem by showing how confused the classification model is on its predictions. It provides more insights into the performance by identifying not only the number of errors, but also the types of those, i.e., false positives and false negatives.

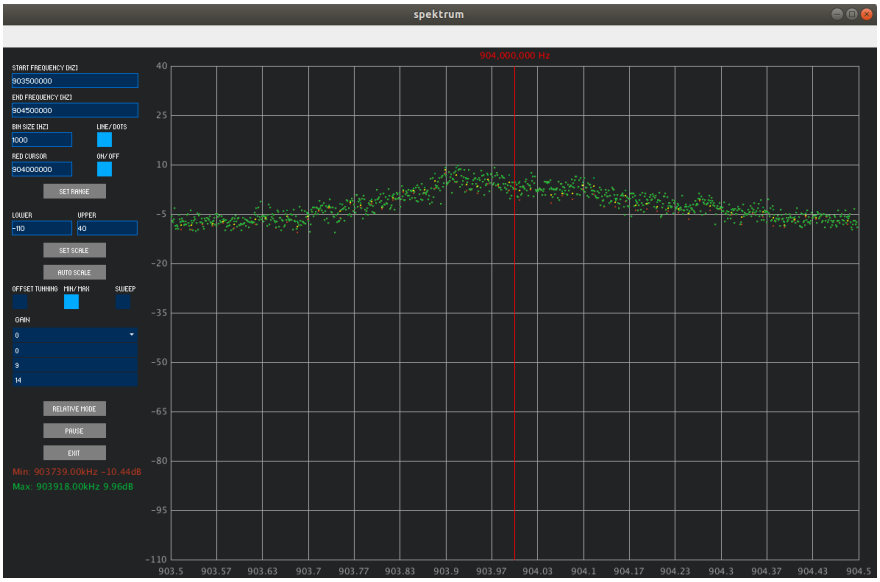


Figure 6.8: Signal Level Plot using Spektrum [69] Software

We use accuracy and confusion matrix to demonstrate the reliability of the proposed models. We show the interference violation and under-utilization to show the feasibility and applicability of the proposed models in cognitive radio networks.

6.3.5 Experimental Results

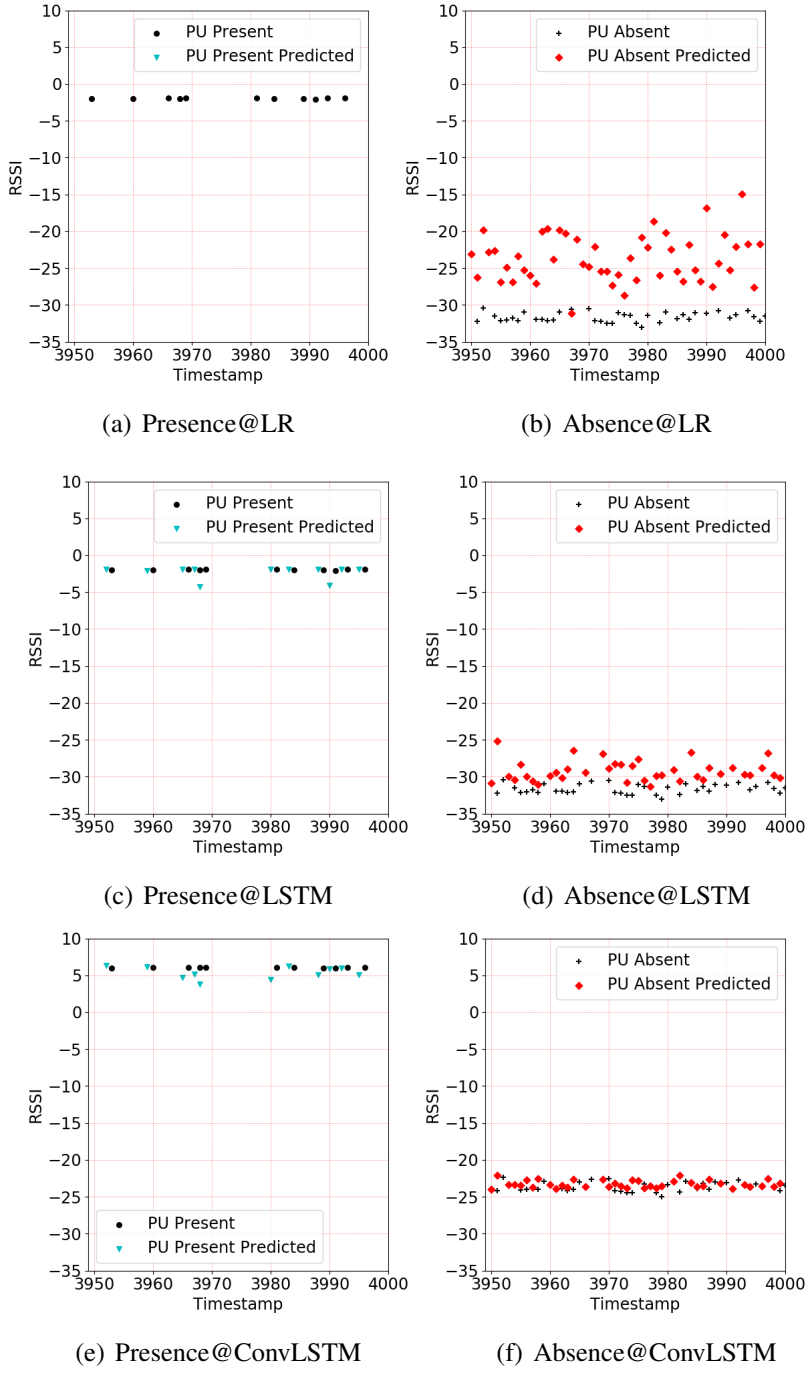
We train three different machine learning models on the collected dataset of each primary user. The linear regression model was straight forward and trained with 90% of each dataset. For both the LSTM and ConvLSTM based models, we use 90%, 5%, and 5% of data for training, validation, and testing respectively. During each training, we set the maximum epoch to 50 with an early stopping condition, such as, if there is no improvement of validation loss for consecutive 5 epochs, then the training is stopped. We choose 50 epochs because we observed through multiple runs of training, that each model reaches optimum accuracy within 50 epochs. We use Adam [42] based optimization with $10e-4$ learning rate and mean squared error loss for training both the LSTM and

ConvLSTM based models. We adjusted the hyper-parameter's values such that the model gives the best possible accuracy with training time trade-off.

Once each model is trained, we predict each primary user's activity for the next 4000 timestamps using the proposed models. We only present last 50 timestamps of those 4000 timestamps in Fig 6.9, for better display quality and understanding. The plot demonstrates the primary user's presence and absence using a threshold on the received signal strength indicator (RSSI) for PU#1. The threshold is application specific; it is set to -10 dB for our testbed. Although PU#1's activity factor was set between 0.7 to 0.75, we see more absence for the last 50 timestamps, which shows the robustness of the data collection procedure. It is evident from the plots that the predicted values for both primary user's presence and absence are getting closer to actual values for LSTM and ConvLSTM models, however, ConvLSTM yields the best results. We observe similar results for the other primary users (PU#2 - P#8) as well, plots are given in Appendix.

6.3.5.1 Analysis on Proposed Prediction Models

The prediction accuracy for all the primary users (PU#1 - PU#8), for all the proposed models are presented in Table 6.3. We notice that linear regression gives 73-76% accuracy, whereas the LSTM and ConvLSTM models manage to get 97%-99% accuracies respectively. This phenomenon can be justified by the presence of correlation within the recurrent structure of RF data. The linear regression based model does not leverage that property. The LSTM based model exploits only the temporal property of that recurrent structure giving a better accuracy ($\sim$97%). However, ConvLSTM based model exploits the spatio-temporal property within the data and hence gives the best accuracy($\sim$99% - 100%) among all the proposed prediction models for all the 8 primary users.



(a) Presence@LR (b) Absence@LR

(c) Presence@LSTM (d) Absence@LSTM

(e) Presence@ConvLSTM (f) Absence@ConvLSTM

Figure 6.9: Predictions of last few Timestamps for different Models for PU#1

6.3.5.2 Performance of ConvLSTM Model

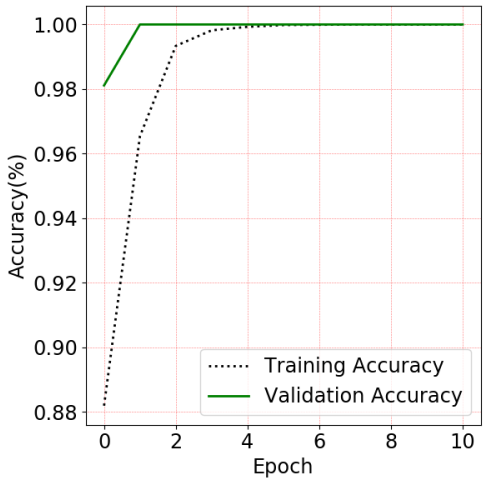
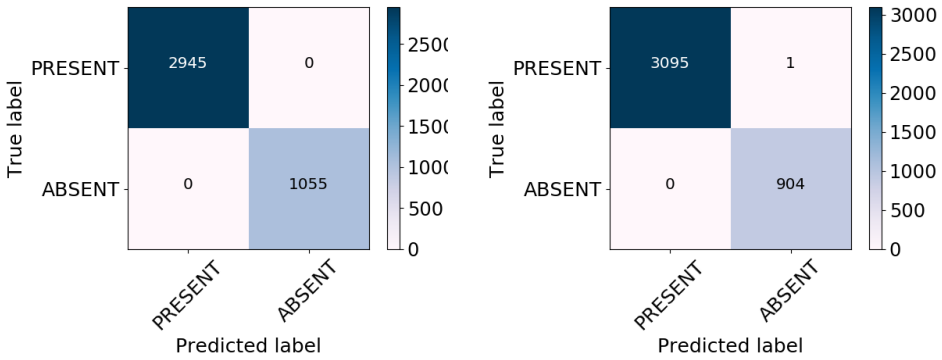


Figure 6.10: Prediction accuracy for ConvLSTM

The accuracies for training and validation are presented in Fig. 6.10 for PU#1. It is observed that training and validation accuracy saturates within a few epochs of the start of training. The model behaves in the same manner for the other PUs as well. Once the ConvLSTM model is trained, we find the confusion matrices for all the 8 PUs for the next 4000 timestamps, as presented in Fig. 6.11. Since the PU's activity factor was set between 0.7 and 0.75, we notice a skewed behavior for the total presence and absence times of the primary users respectively. It is clear from the confusion matrices that the ConvLSTM based model yields negligible numbers of false positives and false negatives during the deployment phase.



(a) PU#1 (b) PU#2

(c) PU#3 (d) PU#4

(e) PU#5 (f) PU#6

(g) PU#7 (h) PU#8

Figure 6.11: Confusion Matrices for Prediction using ConvLSTM for 8 PUs

Table 6.3: Accuracies of Implemented Models for different Primary Users

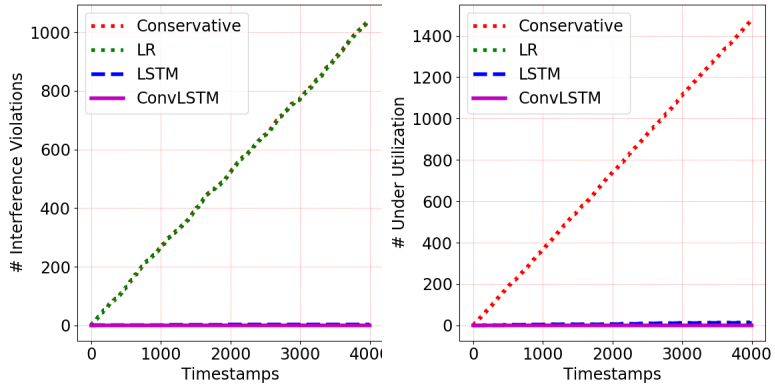
	Linear Regression	**RNN (LSTM)**	**RNN (ConvLSTM)**
PU #1	73.98%	97.62%	100%
PU #2	77.60%	97.83%	99.95%
PU #3	75.15%	97.92%	100%
PU #4	75.15%	97.60%	99.98%
PU #5	75.80%	97.88%	99.98%
PU #6	74.58%	97.62%	99.98%
PU #7	74.72%	97.75%	100%
PU #8	76.75%	97.72%	99.98%

Table 6.4: Comparison of Interference Violation and Under-utilization for the Proposed Models for PU#1

Techniques	**Interference Violation (IV)**	IV **Change**	**Under Utilization (UU)**	UU **Change**
Conservative	1043	-	1478	-
Linear Regression	1041	↓0.2%	15	↓98.9%
RNN-LSTM	7	↓99.3 %	6	↓99.5%
RNN-ConvLSTM	0	↓100 %	0	↓100%

6.3.5.3 Analysis on Interference Violations and Under-utilization

We calculate the total number of interference violations (IV) and under-utilization (UU) for each model. Table 6.4 presents the total numbers and improvement of IV and UU after using all the proposed learning models, over using the conservative allocation strategies. It is clear that the long-term prediction from ConvLSTM based model has no interference violation and under-utilization. Fig. 6.12 gives the graphical overview of how the IV and UU change as the time increases for all types of models. It is evident that the cumulative number of interference violation is significantly high for the conservative approach than the proposed learning based models. However, the cumulative under-utilization is high for both the conservative and linear regression models. The changes of LSTM and ConvLSTM based models are not quite visible in this figure, so we present the enhanced view of their changes in Fig. 6.13.

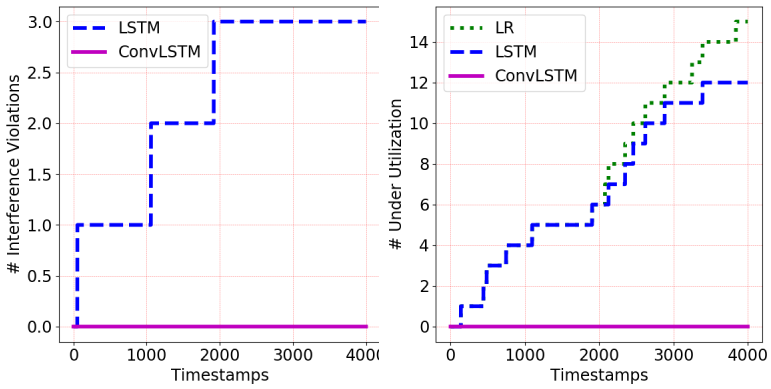


(a) Interference Violations (b) Under-utilization

Figure 6.12: Cumulative Interference Violations and Under-utilization for Conservative and all Proposed Models for PU#1

In summary we have shown that, (1) Linear regression based model gives 73-76% accuracy for primary user's activity prediction. (2) LSTM based recurrent neural network model increases that accuracy to 97% for the activity prediction of all the 8 primary users. (3) ConvLSTM based RNN

model gives the best accuracy (∼100%) for the long-term prediction of primary user's activity. (4) The ConvLSTM based model also decreased the number of interference violations and under-utilizations by 100% compared to the conservative one. (5) Trained ConvLSTM based RNN models can be deployed in a central spectrum sensor for a robust and efficient cognitive radio network.



(a) Interference Violations (b) Under-utilization

Figure 6.13: Enhanced Comparison of Cumulative Interference Violations and Under-utilization for the Proposed Models for PU#1

6.3.5.4 Computational Complexities

We focus on the computational time complexity for the training phase only, as the trained model gives the output within constant time ($O(1)$) during the deployment phase. Computing the time complexity for training a neural network is still evolving. In [51], the authors proved that a neural network of depth δ can be trained in $poly(s^{2^{\delta}})$, where s is the dimension of the input, and $poly(.)$ takes a polynomial time depending on the machine configuration. Here s depends on the dataset size.

Suppose each dataset has T samples. As mentioned earlier, we use 95% of data for training and

validation purpose. The complexity for RNN models with 6 layers using 95% of T data samples for training and validation, is $poly(0.95 \times Te3^{2^6})$. For linear regression model, the complexity is $O(p^2T + p^3)$, where p is the number of features. The prediction complexity is $O(p)$.

6.4 Summary

The *opportunistic usage* of spectrum by secondary users has the possibility of leading to a uniform and efficient usage of overly crowded radio frequency bands. In this regard we present the use of machine learning techniques to predict the possible *opportunities* for such spectrum usage by secondary users. We investigate the spatio-temporal aspect of over-the-air radio data for that purpose. The long-term pattern of primary user's ON and OFF times are learned by the proposed neural network models. The comparative analysis with linear regression shows that exploiting the recurrent structures with respect to temporal and spatial variations achieves the best possible accuracy, as seen from the proposed prediction models. Leveraging the memory of earlier transmission helps the sensing network to determine primary user activity pattern accurately over time. Successful implementation of the proposed models will improve spectrum utilization and lower interference violation for dynamic spectrum access networks.

CHAPTER 7: CONCLUSIONS

In this dissertation, we addressed some of the fundamental challenges on how to effectively apply different learning techniques in the RF domain. In the presence of adversaries, malicious activities such as jamming, and spoofing are inevitable which render most machine learning techniques ineffective. To facilitate learning in such settings, we proposed an adversarial learning-based approach to detect unauthorized exploitation of RF spectrum. First, we showed the applicability of existing machine learning algorithms in the RF domain. We designed and implement three recurrent neural networks using different types of cell models for fingerprinting RF transmitters. Next, we focused on securing transmissions on dynamic spectrum access network where PUE attacks can pose a significant threat. We presented a GAN based solution to counter such PUE attacks. Ultimately, we proposed recurrent neural network models which are able to accurately predict the primary users' activities in DSA networks so that the secondary users can opportunistically access the shared spectrum. We implemented the proposed learning models on testbeds consisting of USRPs working as Software Defined Radios (SDRs). Results revealed significant accuracy gains in accurately characterizing RF transmitters- thereby demonstrating the potential of our models for real world deployments.

We argued that most machine learning techniques would not be effective in adversarial settings and that breakthroughs in GAN can be instrumental in detection of rogue transmitters and accurate identification of known ones in adversarial settings. We proposed and implemented a generative model and a discriminative model for the GAN. We collected over-the-air raw I/Q data using USRP B210 and used that to train the GAN. The discriminator was able to detect rogue transmitters with an accuracy of $\sim$99.9%. As for transmitter classification, we first implemented a convolution neural network (accuracy $\sim$89%) for exploiting the correlation between I/Q data. Then we designed and implemented deep neural network and recurrent neural networks that showed accu-

racies around 97% for trusted transmitter identification. We also showed how the proposed neural network models (especially CNN) faired when radios from different manufacturers were used.

As we got promising results for “transmitter fingerprinting" using convolutional and recurrent neural network both, we wanted to explore the spatial and as well as temporal properties of RF data. So we appraoched towards exploiting both the inherent spatial and temporal properties for a bulding a robust “transmitter fingerprinting" approach. We modeled the temporal and spatio-temporal correlations in RF data. We explored that using temporal correlation only, the RNNs can achieve upto 95-97% of classification accuracy by using LSTM or GRU cell models. Whereas, exploitation of spatio-temporal properties boosted up the accuracy to 98-99% by using ConvLSTM cell model.

The presence of adversaries can potentially cause some attack on the wireless network system. To that end, we considered one significant attack, PUE attack, in a CRN. We presented a robust defense meachnishm against PUE attack by proposing two GAN based models. We catergorized the proposed GAN model based on the availability of prior information. The dumb generator-discriminator model did not have any prior knowledge about the attacker, then also acheived 98% accuracy of PUE attacker detection. Whereas the smart generator-discriminator model leveraged the prior informations to acheive around 99.5% accuracy to catch the PUE attackers.

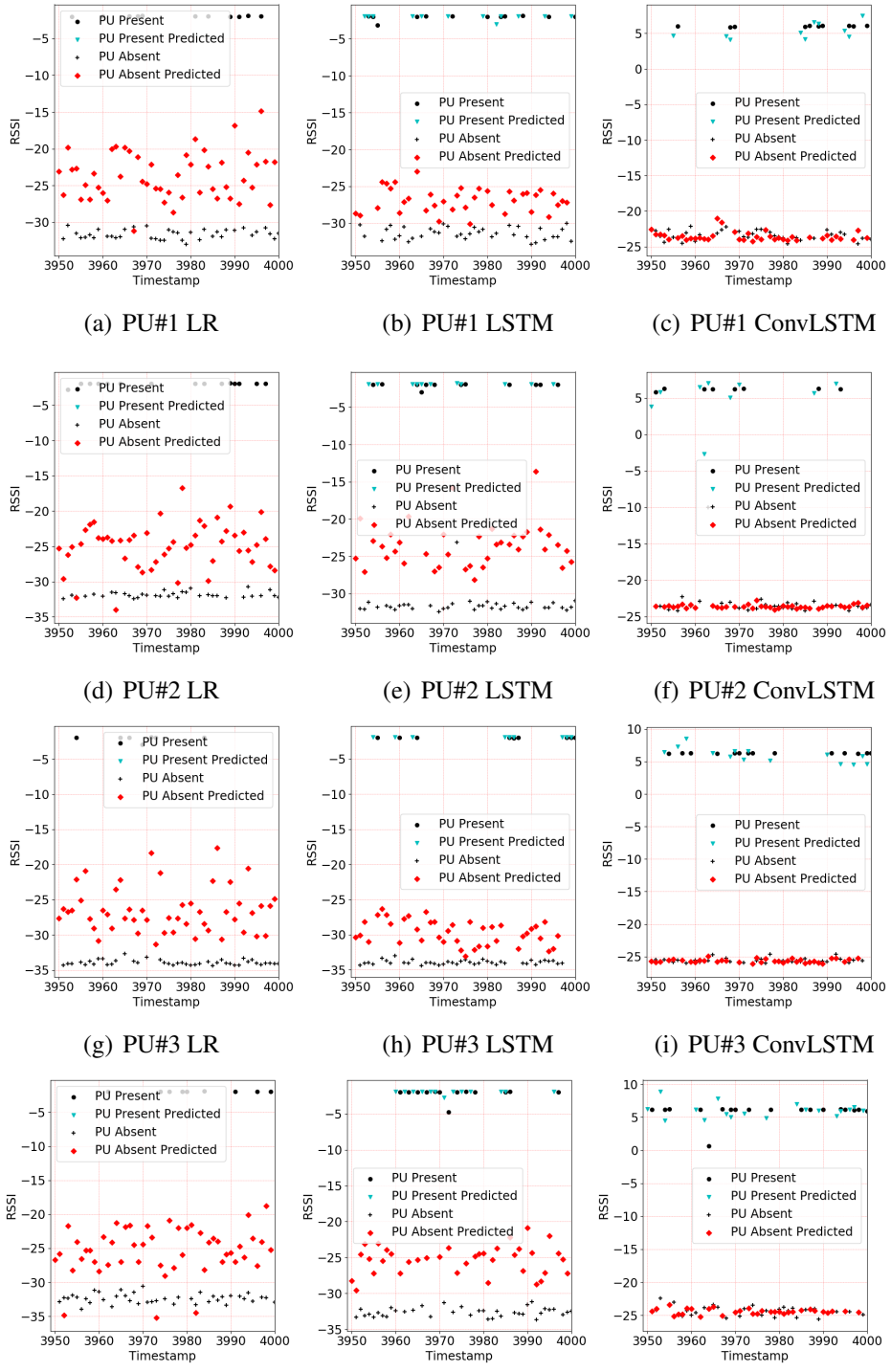
As of the PU activity prediction is concerned, we presented different ML techniques to predict the “long-term" acitivities of the PUs. We leveraged both the temporal, and spatio-temporal aspect of RF data in proposed a recurrent neural networks consisting of LSTM and ConvLSTM cells respectively. We found that the long-term pattern of primary user's ON and OFF times are learned by the proposed neural network models. We noticed the proposed RNN models outperform the traditional linear regression method for predictions. The proposed ConvLSTM model gives 99.9-100% accuracy of long term prediction by leveraging the memory of earlier transmissions. Such accuracte predictions eventually improved spectrum utilization and lower interference violation for

DSA networks.

7.1 Goal of the Dissertation

With more and more autonomous deployments of wireless networks, accurate knowledge of the radio frequency (RF) environment is becoming indispensable. For example, a wireless sensor network relies on trustworthy signals; however, malicious transmitters can contaminate the signals and jeopardize the utility of the sensor network. Existence of such threats underscore the need for techniques that recognize and authenticate the identity of transmitters, irrespective of the network protocols and communication technologies being used. In recent years, there has been a proliferation of autonomous systems that use machine learning algorithms on large scale historical data. When using ML techniques for communication networks, malicious entities, such as rogue transmitters can alter the signal, hence the data. Such threats and their ease of implementation in communication networks necessitates the use of robust learning algorithms, that are agnostic to the network and radio parameters. In this dissertation, we demonstrated the use of adversarial machine learning for the task of robust RF transmitter characterization.

APPENDIX : PU ACTIVITY



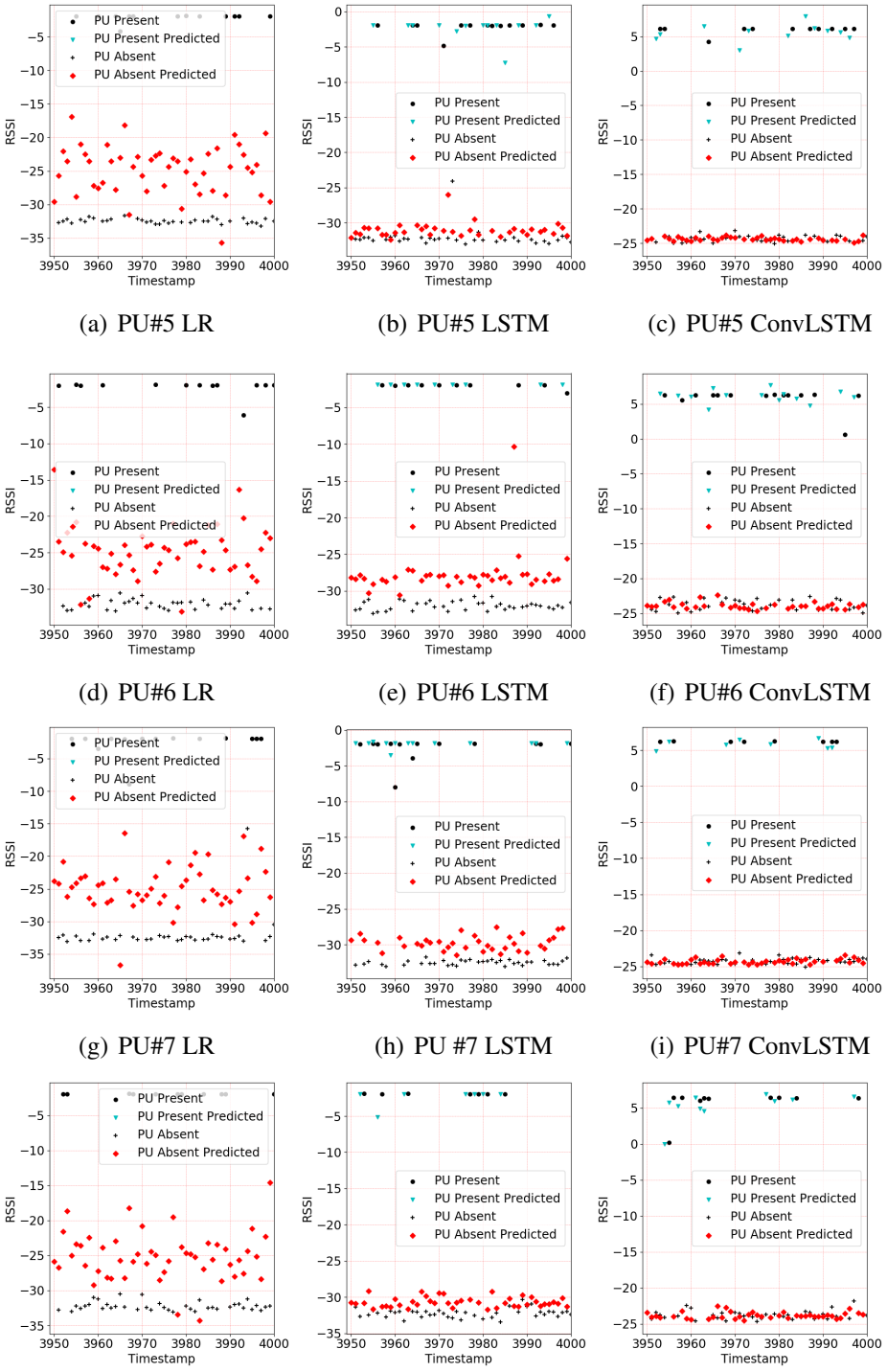
(a) PU#1 LR (b) PU#1 LSTM (c) PU#1 ConvLSTM

(d) PU#2 LR (e) PU#2 LSTM (f) PU#2 ConvLSTM

(g) PU#3 LR (h) PU#3 LSTM (i) PU#3 ConvLSTM

(j) PU#4 LR (k) PU#4 LSTM (l) PU#4 ConvLSTM

Figure .1: Predictions of last few Timestamps for different Models for PU#1 - PU#4



(a) PU#5 LR (b) PU#5 LSTM (c) PU#5 ConvLSTM

(d) PU#6 LR (e) PU#6 LSTM (f) PU#6 ConvLSTM

(g) PU#7 LR (h) PU #7 LSTM (i) PU#7 ConvLSTM

(j) PU#8 LR (k) PU#8 LSTM (l) PU#8 ConvLSTM

Figure .2: Predictions of last few Timestamps for different Models for PU#5 - PU#8

LIST OF REFERENCES

[1] Martín Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, 2016.

[2] Rami Al-Rfou et al. Theano: A python framework for fast computation of mathematical expressions. *CoRR*, 2016.

[3] L. Anttila, M. Valkama, and M. Renfors. Blind Moment Estimation Techniques for I/Q Imbalance Compensation in Quadrature Receivers. In *IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–5, 2006.

[4] ARINC618-5. air/ground character-oriented protocol, specification. Aeronautical Radio, Inc., August 2000.

[5] ARINC618-5. Automatic Dependent Surveillance-Broadcast (ADS-B). `https://www.faa.gov/nextgen/programs/adsb/`, August 2018.

[6] Siqi Bai, Mingjiang Yan, Yongjie Luo, and Qun Wan. RFedRNN: An End-to-End Recurrent Neural Network for Radio Frequency Path Fingerprinting. In *Recent Trends and Future Technology in Applied Intelligence*, pages 560–571, 2018.

[7] S. A. Bassam, S. Boumaiza, and F. M. Ghannouchi. Block-Wise Estimation of and Compensation for I/Q Imbalance in Direct-Conversion Transmitters. *IEEE Transactions on Signal Processing*, 57(12):4970–4973, 2009.

[8] David Berthelot, Tom Schumm, and Luke Metz. BEGAN:Boundary Equilibrium Generative Adversarial Networks. *CoRR*, abs/1703.10717, 2017.

[9] Shameek Bhattacharjee, Shamik Sengupta, and Mainak Chatterjee. Review: Vulnerabilities in Cognitive Radio Networks: A Survey. *ELSEVIER COMCOM*, 36(13):1387–1398, 2013.

[10] Shameek Bhattacharjee, Shamik Sengupta, and Mainak Chatterjee. Vulnerabilities in cognitive radio networks: A survey. *Elsevier Computer Communications*, 36(13):1387–1398, 2013.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[12] S. Burglechner, G. Hueber, and A. Springer. On the Estimation and Compensation of IQ Impairments in Direct Conversion Transmitters. In *European Conference on Wireless Technology*, pages 69–72, 2008.

[13] D. Cabric, S. M. Mishra, and R. W. Brodersen. Implementation Issues in Spectrum Sensing for Cognitive Radios. In *Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 772–776, 2004.

[14] R. Chen, J. . Park, and K. Bian. Robust Distributed Spectrum Sensing in Cognitive Radio Networks. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 1876–1884, 2008.

[15] R. Chen, J. Park, and J. H. Reed. Defense against Primary User Emulation Attacks in Cognitive Radio Networks. *IEEE Journal on Selected Areas in Communications*, 26(1):25–37, 2008.

[16] S. Chen, K. Zeng, and P. Mohapatra. Hearing is believing: Detecting mobile primary user emulation attack in white space. In *Proceedings IEEE INFOCOM*, pages 36–40, 2011.

[17] Shichuan Chen, Shilian Zheng, Lifeng Yang, and Xiaoniu Yang. Deep learning for large-scale real-world ACARS and ADS-B radio signal classification. *Computing Research Repository (CoRR)*, abs/1904.09425, 2019.

[18] F. Chollet et al. Keras: The Python Deep Learning library. `https://keras.io`, 2015.

[19] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*, abs/1412.3555, 2014.

[20] F. E. Churchill, G. W. Ogar, and B. J. Thompson. The Correction of I and Q Errors in a Coherent Processor. *IEEE Transactions on Aerospace and Electronic Systems*, AES-17(1):131–137, 1981.

[21] B. Danev and S. Capkun. Transient-based identification of wireless sensor nodes. In *International Conference on Information Processing in Sensor Networks*, pages 25–36, April 2009.

[22] Emily L Denton et al. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. In *Advances in Neural Information Processing Systems*, pages 1486–1494. 2015.

[23] Analog Devices. ADALM-PLUTO Overview. https://wiki.analog.com/university/tools/pluto, 2018.

[24] R. Dey and F. M. Salemt. Gate-variants of Gated Recurrent Unit (GRU) neural networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1597–1600, 2017.

[25] M. C. Du Plessis and J. C. Olivier. Radar Transmitter Classification using a Non-stationary Signal Classifier. In *International Conference on Wavelet Analysis and Pattern Recognition*, pages 482–485, 2009.

[26] Ettus Research. USRP B210. https://www.ettus.com/product/details/ UB210-KIT, 2018.

[27] FCC ET Docket No. 08-260. Second Report and Order and Memorandum Opinion and Order – Unlicensed Operation in the TV Broadcast Bands / Additional Spectrum for Un-

licensed Devices Below 900 MHz and in the 3GHz Band. US Federal Communications Commission, Washington DC, FCC 08-260, 2008.

[28] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. Technical report, IDSIA, 1999.

[29] GNURadio. GNU Radio. https://www.gnuradio.org, 2018.

[30] Yong Gong, Guyu Hu, and Zhisong Pan. Structured Sparsity Preserving Projections for Radio Transmitter Recognition. In *International Conference on Mobile IT Convergence*, pages 68–73, 2011.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

[32] Ian Goodfellow et al. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680. 2014.

[33] P. Hao, X. Wang, and A. Behnad. Performance Enhancement of I/Q Imbalance based Wireless Device Authentication through Collaboration of Multiple Receivers. In *IEEE International Conference on Communications*, pages 939–944, 2014.

[34] Kaiming He and Jian Sun. Convolutional Neural Networks at Constrained Time Cost. In *IEEE CVPR*, June 2015.

[35] S Hochreiter and J Schmidhuber. Long Short-term Memory. *Neural computation*, 9(8):1735—1780, November 1997.

[36] C. Hsu and W. Sheen. Joint Calibration of Transmitter and Receiver Impairments in Direct-Conversion Radio Architecture. *IEEE Transactions on Wireless Communications*, 11(2):832–841, 2012.

[37] Yuanling Huang and Hui Zheng. Radio Frequency Fingerprinting based on the Constellation Errors. In *IEEE Asia-Pacific Conference on Communications*, pages 900–905, 2012.

[38] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*, abs/1502.03167, 2015.

[39] G. Jakimoski and K. P. Subbalakshmi. Denial-of-Service Attacks on Dynamic Spectrum Access Networks. In *IEEE International Conference on Communications Workshops*, pages 524–528, 2008.

[40] K. Youssef, Louis-S. Bouchard, K.Z. Haigh, H. Krovi, J. Silovsky, and C.P. Vander Valk. Machine Learning Approach to RF Transmitter Identification. *CoRR*, arXiv:1711.01559, 2017.

[41] I. O. Kennedy, P. Scanlon, F. J. Mullany, M. M. Buddhikot, K. E. Nolan, and T. W. Rondeau. Radio Transmitter Fingerprinting: A Steady State Frequency Domain Approach. In *IEEE Vehicular Technology Conference*, pages 1–5, 2008.

[42] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014.

[43] Barnard Kroon, Susan Bergin, Irwin O. Kennedy, and Georgina O'Mahony Zamora. Steady State RF Fingerprinting for Identity Verification: One Class Classifier versus Customized Ensemble. In *AICS*, pages 198–206, 2010.

[44] M. D'Arco L. Angrisani and M. Vadursi. Clustering-based method for detecting and evaluating I/Q impairments in radio-frequency digital transmitters. *IEEE Transactions on Instrumentation and Measurement*, 56(6):2139–2146, 2007.

[45] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[46] W. Lee, M. Kim, and D. Cho. Deep Cooperative Sensing: Cooperative Spectrum Sensing Based on Convolutional Neural Networks. *IEEE Transactions on Vehicular Technology*, 68(3):3005–3009, 2019.

[47] Lime Microsystems. A Pocket Digital TV Transmitter with Raspberry Pi Zero and LimeSDR Mini. https://www.crowdsupply.com/lime-micro/limesdr-mini/updates/a-pocket-digital-tv-transmitter-with-raspberry-pi-zero-and-limesdr-mini, 2019.

[48] Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223–1247, 2017.

[49] Chia-Ling Liu. Impacts of I/Q Imbalance on QPSK-OFDM-QAM Detection. *IEEE Transactions on Consumer Electronics*, 44(3):984–989, 1998.

[50] Y. Liu, P. Ning, and H. Dai. Authenticating Primary Users' Signals in Cognitive Radio Networks via Integrated Cryptographic and Wireless Link Signatures. In *IEEE Symposium on Security and Privacy*, pages 286–301, 2010.

[51] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the Computational Efficiency of Training Neural Networks. In *Advances in Neural Information Processing Systems*, pages 855–863. 2014.

[52] M. J. Marcus. Unlicensed Cognitive sharing of TV Spectrum: The Controversy at the Federal Communications Commission. *IEEE Communications Magazine*, 43(5):24–25, 2005.

[53] A. Massouri, L. Cardoso, B. Guillon, F. Hutu, G. Villemaud, T. Risset, and J. Gorce. CorteXlab: An open FPGA-based facility for testing SDR cognitive radio networks in a reproducible environment. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 103–104, 2014.

[54] M. McHenry. Spectrum white space measurements. New America Foundation Broadband Forum, 2003.

[55] K. Merchant, S. Revay, G. Stantchev, and B. Nousain. Deep learning for rf device fingerprinting in cognitive communication networks. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):160–167, 2018.

[56] Cyrille Morin, Leonardo Cardoso, Jakob Hoydis, Jean-Marie Gorce, and Thibaud Vial. Transmitter Classification With Supervised Deep Learning. *Computing Research Repository (CoRR)*, arXiv:1905.07923, 2019.

[57] Tathagata Mukherjee, Michael Duckett, Piyush Kumar, Jared Devin Paquet, Daniel Rodriguez, Mallory Haulcomb, Kevin George, and Eduardo Pasiliao. RSSI-Based Supervised Learning for Uncooperative Direction-Finding. In *Proceedings of ECML-PKDD 2017*. ECML-PKDD, Springer, 10 2017.

[58] Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of International Conference on International Conference on Machine Learning*, pages 807–814, 2010.

[59] N. T. Nguyen, G. Zheng, Z. Han, and R. Zheng. Device Fingerprinting to Enhance Wireless Security using Nonparametric Bayesian Method. In *IEEE INFOCOM*, pages 1404–1412, 2011.

[60] NooElec. USRP B210. http://www.nooelec.com/store/sdr/sdr-receivers/nesdr-mini-rtl2832-r820t.html, 2018.

[61] M. P. Olivieri, G. Barnett, A. Lackpour, and A. Davis and. A Scalable Dynamic Spectrum Allocation System with Interference Mitigation for Teams of Spectrally Agile Software De-

fined Radios. In *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 170–179, 2005.

[62] T. J. O'Shea, S. Hitefield, and J. Corgan. End-to-end Radio Traffic Sequence Recognition with Recurrent Neural Networks. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 277–281, 2016.

[63] Timothy O'Shea and Nathan West. Radio Machine Learning Dataset Generation with GNU Radio. *Proceedings of the GNU Radio Conference*, 1(1), 2016.

[64] Timothy J. O'Shea, T. Charles Clancy, and Robert W. McGwier. Recurrent neural radio anomaly detection. *CoRR*, abs/1611.00301, 2016.

[65] Timothy J. O'Shea, Johnathan Corgan, and T. Charles Clancy. Convolutional Radio Modulation Recognition Networks. In *Engineering Applications of Neural Networks*, pages 213–226, 2016.

[66] Timothy J. O'Shea et al. Physical Layer Communications System Design Over-the-Air Using Adversarial Networks. *CoRR*, abs/1803.03145, 2018.

[67] T. O'Shea and J. Hoydis. An Introduction to Deep Learning for the Physical Layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, 2017.

[68] T. J. O'Shea, T. Roy, and T. C. Clancy. Over-the-Air Deep Learning Based Radio Signal Classification. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):168–179, 2018.

[69] Pavel Sorejs. Spektrum. https://github.com/pavels/spektrum, 2015.

[70] M. Pospíšil, R. Marsalek, and J. Pomenkova. Wireless Device Authentication through Transmitter Imperfections — Measurement and Classification. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 497–501, 2013.

[71] D. Pu, Y. Shi, A. V. Ilyashenko, and A. M. Wyglinski. Detecting Primary User Emulation Attack in Cognitive Radio Networks. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–5, 2011.

[72] radioML. RFML 2016. https://github.com/radioML/dataset, 2018.

[73] S. Rajendran et al. Deep Learning Models for Wireless Signal Classification With Distributed Low-Cost Spectrum Sensors. *IEEE Transactions on Cognitive Communications and Networking*, 4(3):433–445, 2018.

[74] Jimmy SJ Ren, Yongtao Hu, Yu-Wing Tai, Chuan Wang, Li Xu, Wenxiu Sun, and Qiong Yan. Look, Listen and Learn-A Multimodal LSTM for Speaker Identification. In *AAAI*, pages 3581–3587, 2016.

[75] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury. Deep Learning Convolutional Neural Networks for Radio Identification. *IEEE Communications Magazine*, 56(9):146–152, 2018.

[76] D. Roy, T. Mukherjee, and M. Chatterjee. Machine Learning in Adversarial RF Environments. *IEEE Communications Magazine*, 57(5):82–87, 2019.

[77] Debashri Roy, Tathagata Mukherjee, Mainak Chatterjee, Erik Blasch, and Eduardo Pasiliao. RFAL: Adversarial Learning for RF Transmitter Identification and Classification. *IEEE Transactions on Cognitive Communications and Networking*, 2019. [Early Access].

[78] Debashri Roy, Tathagata Mukherjee, Mainak Chatterjee, and Eduardo Pasiliao. Defense against PUE Attacks in DSA Networks using GAN based Learning. In *IEEE Global Communications Conference (GLOBECOM)*, 2019.

[79] Debashri Roy, Tathagata Mukherjee, Mainak Chatterjee, and Eduardo Pasiliao. Detection of Rogue RF Transmitters using Generative Adversarial Nets. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–7, 2019.

[80] Debashri Roy, Tathagata Mukherjee, Mainak Chatterjee, and Eduardo Pasiliao. Primary User Activity Prediction in DSA Networks Using Recurrent Structures. In *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 1–10, 2019.

[81] Debashri Roy, Tathagata Mukherjee, Mainak Chatterjee, and Eduardo Pasiliao. RF Transmitter Fingerprinting Exploiting Spatio-temporal Properties in Raw Signal Data. In *IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 89–96, 2019.

[82] Rulemakings 12-354 and 17-258. 3.5 GHz Band / Citizens Broadband Radio Service. `https://www.fcc.gov/wireless/bureau-divisions/mobility-division/35-ghz-band/35-ghz-band-citizens-broadband-radio-service`, 2015.

[83] H. Rutagemwa, A. Ghasemi, and S. Liu. Dynamic Spectrum Assignment for Land Mobile Radio with Deep Recurrent Neural Networks. In *IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, 2018.

[84] Yosra Ben Saied and Alexis Olivereau. D-HIP: A distributed key exchange scheme for HIP-based Internet of Things. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–7, 2012.

[85] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury. ORACLE: Optimized Radio clAssification through Convolutional neuraL nEtworks. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 370–378, 2019.

[86] D. Shaw and W. Kinsner. Multifractal Modelling of Radio Transmitter Transients for Classification. In *IEEE WESCANEX*, pages 306–312, 1997.

[87] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, pages 802–810, 2015.

[88] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, San Diego, CA, USA, 1997.

[89] Nitish Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[90] Mark Stanislav and Tod Beardsley. Hacking IoT: A case study on baby monitor exposures and vulnerabilities. *Rapid 7*, 2015.

[91] K. W. Sung, S. Kim, and J. Zander. Temporal Spectrum Sharing Based on Primary User Activity Prediction. *IEEE Transactions on Wireless Communications*, 9(12):3848–3855, 2010.

[92] T. J. O'Shea, J. Corgan and T. C. Clancy. Unsupervised Representation Learning of Structured Radio Communication Signals. In *First International Workshop on Sensing, Processing and Learning for Intelligent Machines (SPLINE)*, pages 1–5, 2016.

[93] Y. Tang, Q. Zhang, and W. Lin. Artificial Neural Network Based Spectrum Sensing Method for Cognitive Radio. In *Proceedings IEEE WiCOM*, pages 1–4, 2010.

[94] J. Toonstra and W. Kinsner. A radio transmitter fingerprinting system ODO-1. In *Canadian Conference on Electrical and Computer Engineering*, volume 1, pages 60–63, 1996.

[95] J. Tubbax et al. Compensation of IQ imbalance in OFDM systems. In *IEEE International Conference on Communications*, pages 3403–3407, 2003.

[96] V. K. Tumuluru, P. Wang, and D. Niyato. A Neural Network Based Spectrum Prediction Scheme for Cognitive Radio. In *IEEE International Conference on Communications*, pages 1–5, 2010.

[97] M. Valkama, M. Renfors, and V. Koivunen. Advanced Methods for I/Q Imbalance Compensation in Communication Receivers. *IEEE Transactions on Signal Processing*, 49(10):2335–2344, 2001.

[98] N. Wagle and E. Frew. Spatio-temporal Characterization of Airborne Radio Frequency Environments. In *IEEE GLOBECOM Workshops (GC Wkshps)*, pages 1269–1273, 2011.

[99] Rolf H Weber and Romana Weber. *Internet of Things*, volume 12. Springer, 2010.

[100] X. Xing, T. Jing, W. Cheng, Y. Huo, and X. Cheng. Spectrum prediction in cognitive radio networks. *IEEE Wireless Communications*, 20(2):90–96, 2013.

[101] S. Xu, L. Xu, Z. Xu, and B. Huang. Individual Radio Transmitter Identification based on Spurious Modulation Characteristics of Signal Envelop. In *IEEE MILCOM*, pages 1–5, 2008.

[102] K. Youssef, L. Bouchard, K. Haigh, J. Silovsky, B. Thapa, and C. V. Valk. Machine Learning Approach to RF Transmitter Identification. *IEEE Journal of Radio Frequency Identification*, 2(4):197–205, 2018.

[103] T. Yucek and H. Arslan. A Survey of Spectrum Sensing Algorithms for Cognitive Radio Applications. *IEEE Communications Surveys Tutorials*, 11(1):116–130, 2009.

[104] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[105] C. Zhao, W. Wang, L. Huang, and Y. Yao. Anti-PUE Attack Base on the Transmitter Fingerprint Identification in Cognitive Radio. In *International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–5, 2009.

[106] C. Zhao, L. Xie, X. Jiang, L. Huang, and Y. Yao. A PHY-layer Authentication Approach for Transmitter Identification in Cognitive Radio Networks. In *International Conference on Communications and Mobile Computing*, volume 2, pages 154–158, 2010.

[107] Q. Zhao and B. M. Sadler. A Survey of Dynamic Spectrum Access. *IEEE Signal Processing Magazine*, 24(3):79–89, 2007.

[108] Y. Zhao, L. Morales, J. Gaeddert, K. K. Bae, J. Um, and J. H. Reed. Applying Radio Environment Maps to Cognitive Wireless Regional Area Networks. In *IEEE DySPAN*, pages 115–118, 2007.

[109] S. Zheng, S. Chen, L. Yang, J. Zhu, Z. Luo, J. Hu, and X. Yang. Big Data Processing Architecture for Radio Signals Empowered by Deep Learning: Concept, Experiment, Applications and Challenges. *IEEE Access*, 6:55907–55922, 2018.

[110] Jun-Yan Zhu et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *CoRR*, abs/1703.10593, 2017.