4-30-2021

Radio frequency dataset collection system development for location and device fingerprinting

Nicholas G. Smith
ngsmith37@gmail.com

Radio frequency dataset collection system development for location and device fingerprinting

By

Nicholas G. Smith

Approved by:

Bo Tang (Major Professor)
John E. Ball
Maxwell Young
Qian Du (Graduate Coordinator)
James M. Keith (Dean, Bagley College of Engineering)

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical and Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

April 2021

Name: Nicholas G. Smith

Date of Degree: April 30, 2021

Institution: Mississippi State University

Major Field: Electrical and Computer Engineering

Major Professor: Bo Tang

Title of Study: Radio frequency dataset collection system development for location and device fingerprinting

Pages of Study: 40

Candidate for Degree of Master of Science

Radio-frequency (RF) fingerprinting is a process that uses the minute inconsistencies among manufactured radio transmitters to identify wireless devices. Coupled with location fingerprinting, which is a machine learning technique to locate devices based on their radio signals, it can uniquely identify and locate both trusted and rogue wireless devices transmitting over the air. This can have wide-ranging applications for the Internet of Things, security, and networking fields. To contribute to this effort, this research first builds a software-defined radio (SDR) testbed to collect an RF dataset over LTE and WiFi channels. The developed testbed consists of both hardware which are receivers with multiple antennas and software which performs signal preprocessing. Several features that can be used for RF device fingerprinting and location fingerprinting, including received signal strength indicator and channel state information, are also extracted from the signals. With the developed dataset, several data-driven machine learning algorithms have been implemented and tested for fingerprinting performance evaluation. Overall, experimental results show promising

performance with a radio fingerprinting accuracy above 90% and device localization within 1.1∂ meters.

Key words: radio fingerprinting, location fingerprinting, software-defined radio, machine learning, classifiers, wireless communication

DEDICATION

To Jennifer Hoang, Cathy Figueroa, and Dingo. I couldn't have done this without you all.

ACKNOWLEDGEMENTS

I thank Logan Smith, Ajaya Dahal, Surya Teja, and Daniel Rayborn for their input and expertise in the project. I thank Deepak Chapagain, Keith Hunter, and Mark McDonnell with their assistance in helping setup this project's hardware. I thank Keith Powell for dealing with my sporadic questioning at times. Finally, I thank Dr. Tang, Dr. Ball, and Dr. Young for their guidance and advice throughout this research.

This research is supported by the National Institute of Justice (NIJ) grant 2018-75-CX-K002. Permission to reproduce the MSU logo was given by Mississippi State University.

I thank my committee for their comments on this thesis, and I thank Bo Tang for directing this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In the past several years, the number of Internet of Things (IoT) and wireless devices have grown exponentially with no sign of stopping soon. With this in mind, it is vital for security applications that networks can determine and verify a device's identity. However, there are currently a number of adversarial attacks that can circumvent these efforts. One such method is media access control (MAC) address "spoofing." This technique requires the attacker to change the MAC address in their low layer packets to the MAC address of a trusted source. By doing this, a secured network can mistake the attacker as being trusted and allow them access to the network. To further complicate matters, IoT devices have limited power and computational resources that prevents them from performing intensive verification procedures. Moreover, the spectrum has become increasingly band-limited requiring spectrum efficient solutions to this troublesome security issue.

One such solution is known as radio frequency (RF) fingerprinting. When an RF transmitter sends a signal, the signal passes through a series of mixers, filters, amplifiers, and other RF circuitry before finally being transmitted through the antenna. These different RF circuit components are all manufactured with different tolerances making no two exactly alike. Together, the unique components contribute slight fluctuations on certain parts of the physical signal such as the rise-time signature and in-phase/quadrature (IQ) imbalances. As a whole, these fluctuations are known

as the device's RF fingerprint. Using these physical fingerprints, it has been shown that devices can be uniquely identified based on their physical signal alone as opposed to any information it may carry [10].

For practical reasons, once a device is determined to be rogue, it may also be useful to locate said device. This can be especially true for mobile cell phones which are presumed to be constantly moving and can also indicate its owner's location. To achieve this goal, one can couple RF fingerprinting with location fingerprinting. Location fingerprinting uses machine learning (ML) techniques to map the raw signals a device transmits back to their original locations. Typically, this requires a dataset collected over the target area where multiple devices' signals are paired with their respective locations. The ML algorithm takes these raw signal samples or certain properties of them such as Received Signal Strength Intensity (RSSI) and uses a regression network to determine the exact spot the signal was transmitted from.

The common factor for both of these approaches is a dataset of raw signals collected over the air from multiple devices over a fixed area. The labels for both of these problems would be the transmitter identities such as MAC addresses and the physical location of the transmitter. It is a known issue that collecting these datasets can be prohibitively difficult and time-consuming. This study attempts to provide this dataset using a testbed composed of software-defined radios (SDRs) as receivers and cell phones as transmitters. By using relatively low-cost SDRs and open-source algorithms, I prepared an accessible solution others can mirror to collect their own fingerprinting dataset.

SDRs are multi-purpose RF equipment that implements much of the functionality traditionally performed by hardware in software. This makes the development process much quicker and more

flexible by implementing different communication schemes using the same device. This has the benefit of communicating with devices using different protocols and evolving those protocols over time to agree with newer standards.

To prove the dataset's efficacy, it was tested with both location and device fingerprinting algorithms. One multi-layer perceptron (MLP) and one convolutional neural network were created for both of these tasks using Scikit-Learn and TensorFlow respectively [26] [5]. Each were trained and tested separately on the dataset and showed promising results comparable to the current state-of-the-art networks.

The rest of this paper is organized as follows. The background for location fingerprinting and the different protocols' demodulation parameters is provided in Chapter 2. Chapters 3 and 4 focus on the Wireless Fidelity (WiFi) and Long-Term Evolution (LTE) collection procedures and the fingerprinting algorithms to verify them. Finally, chapter 5 summarizes this work's conclusions and results.

Contributions from this thesis include the following:

- Modified open-source GNURadio module to collect IEEE802.11g frames and features
- Modified open-source srsLTE code to collect LTE uplink raw frames and features
- Code to process each dataset into an easy-to-use format for ML purposes
- A small LTE dataset for device/location fingerprinting with ML baselines
- A large WiFi dataset for device/location fingerprinting
- An SDR testbed for data collection purposes
- Supported multiple conference papers [35], [36], [27]

All code and datasets discussed are available within this document at available at https://github.com/nicksmith37/NIJ_Code.

CHAPTER II

BACKGROUND

2.1 RF Device Fingerprinting

Wireless devices operate in an environment where they are affected by multipath, temperature changes, movement, and other factors. These present challenges to device fingerprinting algorithms trying to distinguish different devices on minute signal fluctuations commonly characterized as noise. In order to ensure that these signal variations are preserved for the dataset, it is a requirement that the signal be captured with the least amount of modifications being applied to it as possible. This can prove to be challenging as it can be hard to receive signals without certain corrections and compensations. These are exacerbated by the fact that not only does the transmitter impart a fingerprint on the signal but the receiver does as well [29]. In the literature, there are several survey papers written discussing the topic of RF device fingerprinting [8] [48] [38].

There are many different traditional methods proposed for RF fingerprinting. Gerdes *et al.* used a matched filter and threshold approach to detect IEEE802.3 Ethernet preambles and classified them by model and manufacturer with very high accuracy [15]. Suski *et al.* attempted to classify IEEE802.11a frames by the power spectral density (PSD) of their preambles. Using only three devices, their results showed approximately 80% accuracy for moderately high signal-to-noise ratios (SNRs) [39]. Jana and Kasera proved it was possible to identify access points (APs) using their clock skews allowing users to determine fake APs [20]. Williams *et al.* performed classification

of Global System for Mobile Communication (GSM) signals from different manufacturers using both the transient and midamble of the signal [47]. They showed over 90% accuracy using the midamble at >= 12 dB and the transient at >= 16 dB suggesting that the midamble may outperform the transient in lower SNR regions.

In addition to traditional techniques, ML techniques are often used for device fingerprinting with great success. Brik *et al.* used specific features of the modulated signal such as magnitude and phase errors, synchronization correlation errors, and IQ origin offset to develop a fingerprint profile for 138 IEEE802.11b devices [10]. These samples were collected between 3 to 15 meters from the antenna by a high-end vector network analyzer (VNA), which were used as a dataset by a support vector machine (SVM) and k nearest neighbors (k-NN) networks. This showed very high accuracy of 99% but requires costly (upwards of $100,000), well-calibrated equipment. Rehman *et. al* suggested that low-end SDRs could perform similarly to high-end SDRs and achieve strong classification results in 15 dB and higher SNR environments [28].

Deep learning has especially proven to be a fruitful method for device fingerprinting. Two of the most common types of deep learning networks are the deep neural network (DNN) and convolutional neural network (CNN). A DNN contains an input layer, multiple hidden layers, and an output layer all composed of neurons. Each neuron calculates the dot product of its inputs with its weights and adds a bias term before passing it to a nonlinear activation function. The output from the output layer is then passed to a loss function with the label to calculate the network's loss. This loss is used to update the weights of every neuron in the network using a backpropagation algorithm.

DNN is often used as a baseline to compare against in literature and thus can be useful to explore. Youssef *et al.* collected 12,000 samples from 12 transmitters using the IEEE802.11a/g protocol [50]. They then used a DNN to classify them and achieved an accuracy of 84.8%, notably despite the small dataset. Jafari *et al.* showed better performance with a much larger dataset at varying SNR levels[19]. They were able to classify six ZigBee devices with a DNN for an accuracy of 96.3%. Soltani*et al.* shows the limitations of DNNs for fingerprinting activities, especially in different channel environments and using different equipment [37]. They develop a scheme to augment the dataset to potentially increase accuracy by as much as 51%.

A common variant of the DNN is the CNN. A CNN uses filters to perform convolutions as it passes over the input. The output from these convolutions are then passed through nonlinear activation functions similar to the DNN before. These results are then stored in feature maps and often passed to pooling layers for feature reduction. After a series of these, the outputs are usually flattened and passed through a few fully-connected layers before being passed to the output. A major advantage of CNNs over DNNs is that they use fewer parameters which translates to less memory usage.

CNN have seen much use recently in the literature for device fingerprinting. Merchant *et al.* uses a CNN to classify seven ZigBee devices [24]. They use the time-domain complex baseband error signal and get 92.29% accuracy on for above 40 dB SNR. Wang *et al.* used the differential constellation trace figure to classify six mobile phones [41]. They also used a CNN and were able to achieve an accuracy of 99.77% for SNR of 50 dB or higher. Roy *et al.* compared the classification performance of a DNN, CNN, and recurrent neural network (RNN) for eight SDR transmitters [31]. Their dataset was composed of 1024 raw complex samples per one Quadrature Phase-Shift

Keying (QPSK) frame. The experiment yielded accuracy of 81.6%, 94.60%, and 97.06% for the CNN, DNN, and RNN respectively suggesting the CNN may have the worst performance on raw data samples.

2.2 Indoor Location Fingerprinting

Location fingerprinting is a very popular application of RF fingerprinting. Many early location fingerprinting algorithms focused on using RSSI for locating devices. RSSI is often available in many communication systems and is easy to calculate. The first location fingerprinting system used for WiFi is WiFi RADAR [7]. At every point in a grid, it stored the RSSI values that would later serve as the dataset for a k-NN network. The network overall was able to locate a user within 2-3 meters. On the other hand, the HORUS system used a probabilistic approach to determine the device's location[51]. It uses RSSI from different APs as input and infers the location using Bayesian inference and had an average error of less than 0.6 meters. However, it requires a large number of samples from the APs to construct an accurate representation of the data and can therefore be costly. A more recent work, CellinDeep, demonstrates the performance of RSSI from LTE signals collected by multiple cell phone towers [30]. They achieve an accuracy 0.78 meters in a floor of a building using a DNN. Going further, it has been shown that RSSI is especially vulnerable to multipath and attenuation, especially when not in line-of-sight [49].

The other metric commonly used for location fingerprinting is channel state information (CSI). CSI is already a part of most OFDM communication schemes and can provide a much more detailed observation of the channel. In the literature, CSI is often collected for WiFi via a commercial network interface card (NIC) with three antennae to provide the dataset [**?**] [**?**]. CSI-MIMO and

DeepFi were some of the first works to successfully pair deep learning with CSI for localization [12] [44]. They were able to provide sub-meter accuracy with mean distance errors of 0.95 and 0.94 meters respectively using a kNN and DNN. Another promising method using CSI is bi-modality deep learning. By using bi-modal data such as angle of arrival (AoA) and average amplitude over antenna pairs, BiLoc and ResLoc achieved considerable accuracy of 1.06 and 0.89 meters respectively using various deep learning techniques [43] [46]. Zhang *et al.* are able to achieve below a half meter accuracy with their proposed system in [53]. They use a single AP, a novel phase decomposition algorithm, and an SVM to achieve 0.46 meter accuracy for a single room. This is on par with Zhang *et al.* for LTE indoor positioning using CSI [52]. Their method used a time domain fusion approach by means of multiple MLPs to achieve an indoor mean distance error of 0.47 meters.

2.3 IEEE802.11g Properties and Structure

For WiFi, this work captures IEEE802.11g packets from over the air. IEEE802.11g was chosen due to being relatively simpler than other IEEE802.11 protocols while still holding widespread popularity. It is a Single-Input Single-Output (SISO) protocol meaning devices only communicate using one antenna pair for sending and receiving. It employs packet-based orthogonal frequency-division multiplexing (OFDM) with 52 subcarriers over a bandwidth of 20 MHz [1].

OFDM is a communication scheme dating back to 1966 that packs tightly spaced subcarriers in the frequency domain resulting in a longer symbol duration in order to achieve higher data rates [11]. This is in stark contrast to the other single carrier methods of the time that attempted to transmit at a much higher symbol rate using only a single carrier in order to achieve similar data

rates. OFDM also solves the problem of intersymbol interference (ISI) by appending a cyclical prefix (CP) to the beginning of each symbol. The CP is a direct copy of the last portion of the symbol and designed to be at least as long as the delay spread. This prevents adjacent symbols from "smearing" together due to multipath and allows for circular convolution, which makes the hardware design significantly simpler. When symbols are first being decoded, their CPs are discarded, which in effect is the signal being windowed with a Boxcar function. This results in a sinc wave in the frequency domain with zero crossings at the subcarrier spacings. This prevents intercarrier interference (ICI) in the frequency domain and makes the subcarriers orthogonal as seen in Fig. 2.1. For these reasons, OFDM was chosen as the communication scheme for IEEE802.11g.

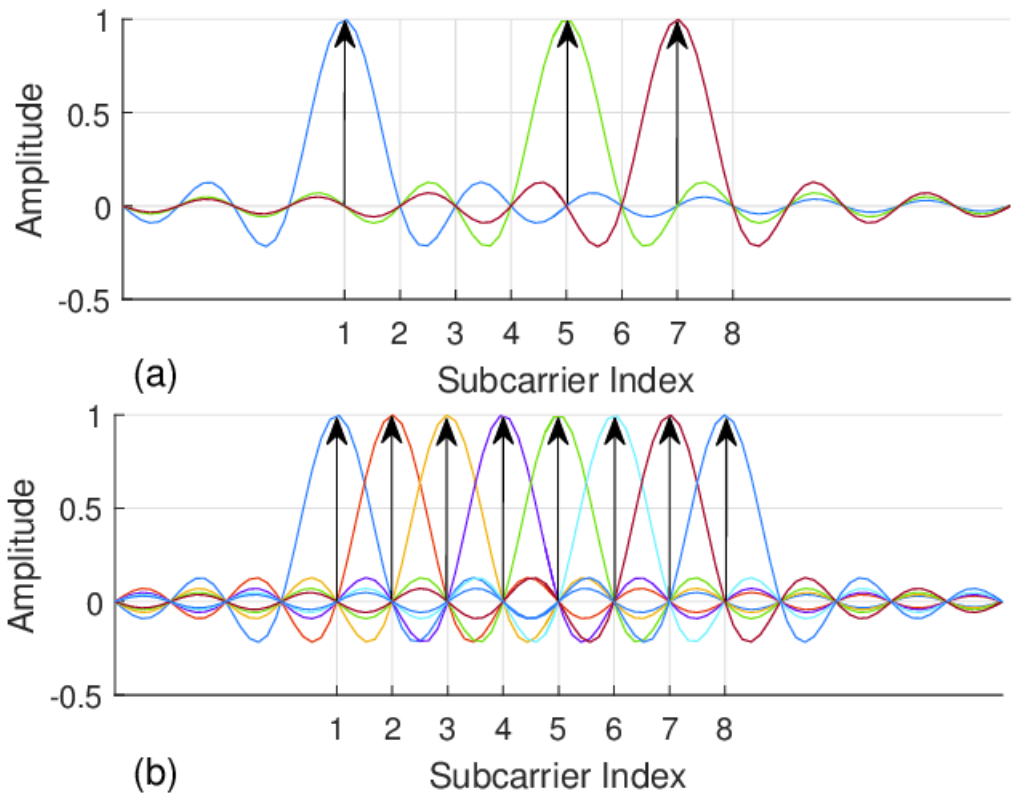

Figure 2.1: Subcarrier spacing in the frequency domain for OFDM [13]

The ideal dataset for fingerprinting should contain no descriptive information about the signals other than the signals' physical properties. This descriptive information includes details like the MAC address, frame length, and information actually being transmitted. This makes it imperative to understand the structure of the frame and use only the parts not containing any information about the device, so the algorithm does not attempt to learn these features. The IEEE802.11g physical layer protocol data unit (PPDU) and training structure is shown below in Fig. 2.2.

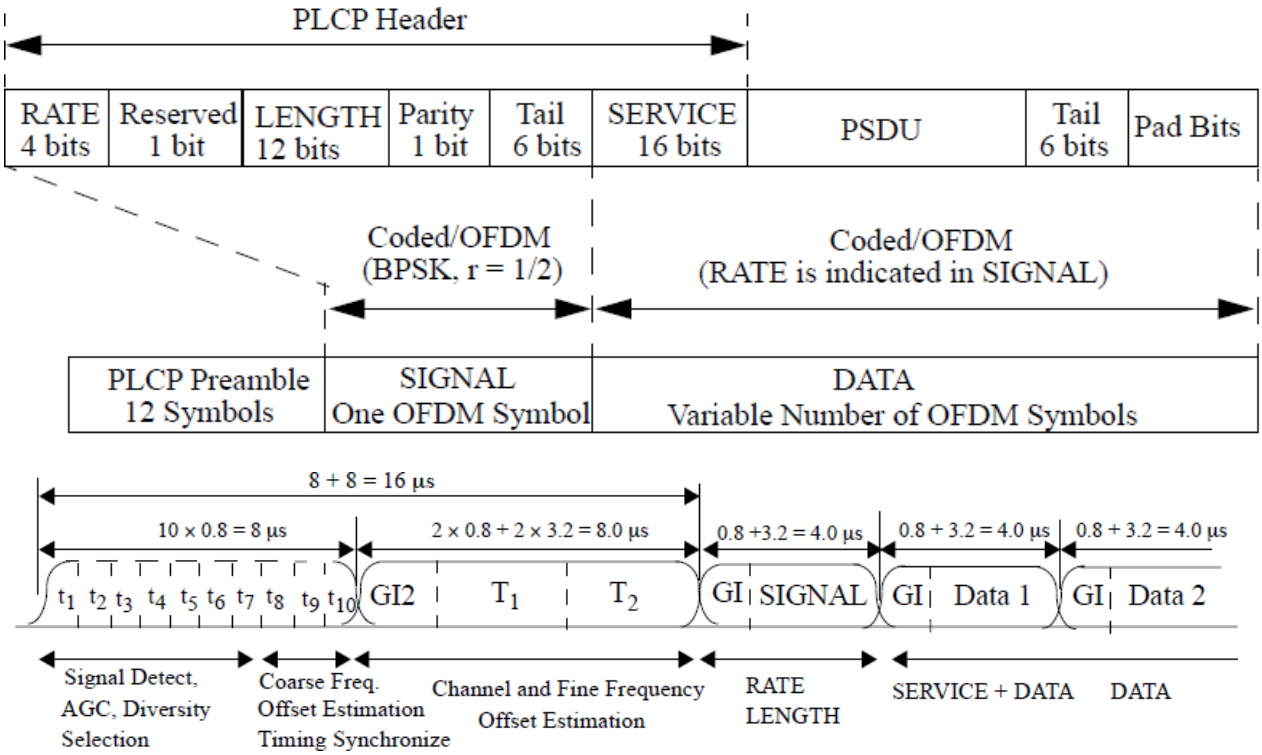

Figure 2.2: PPDU and training structure for IEEE802.11g [1]

As can be seen in the figure, the first field in the PPDU is known as the preamble. The preamble is made up of two components: the short training field (STF) and the long training field (LTF). Each occupies two OFDM symbol durations. The STF contains 10 iterations of a 0.8 microsecond

signal, which can be useful for OFDM synchronization [33]. The LTF is a predefined symbol repeated 2.5 times and is often used for fine frequency offset correction and channel equalization. After the preamble is processed, the next field is the signal field. The signal field contains one binary phase-shift keying (BPSK) OFDM symbol, which is the most robust OFDM modulation. It indicates to the receiver the length and encoding of the rest of the signal. The rest of the signal is a data payload of variable length with six trailing 0's and additional pad bits.

With this information, the preamble of the frame was chosen as the portion of the signal used in the dataset. This section of the signal contains no descriptive higher level information, is a set length, and includes the signal transient which has to proven to be useful for device fingerprinting [34].

2.4 LTE Uplink Frame Structure

Like WiFi, it is also important to understand the frame structure and modulation parameters of LTE in order to not contaminate the samples with any kind of noisy information. Cell phones transmit LTE signals over the air using single-carrier frequency-division multiple access (SC-FDMA) scheme, which is a variant of OFDM modulation. Since regular OFDM sends multiple subcarriers in parallel, it often results in a high peak-to-average power ratio (PAPR) compared to SC-FDMA. This results in more complex and costly architecture schemes for the transmitters which could problematic for the resource-constrained cell phones. Therefore, SC-FDMA was chosen as the modulation of choice for the LTE uplink channel. The SC-FDMA scheme can be seen in Fig. 2.3 in comparison to Orthogonal Frequency Division Multiple Access (OFDMA), an extension of OFDM used in the LTE downlink. SC-FDMA transmits “sub-symbols" instead of

subcarriers. These sub-symbols occupy the total bandwidth of the signal with a single carrier but are transmitted much faster allowing multiple sub-symbols to be transmitted per slot. This can carry the same amount of information as OFDM while occupying the same bandwidth and symbol duration. In addition, SC-FDMA retains the addition of the CP like OFDM, so it can also mitigate some of the effects of ISI from multipath. Even so, SC-FDMA still has drawbacks such as lower spectral efficiency and an increase in “noise enhancement" when used with linear amplifiers [40].

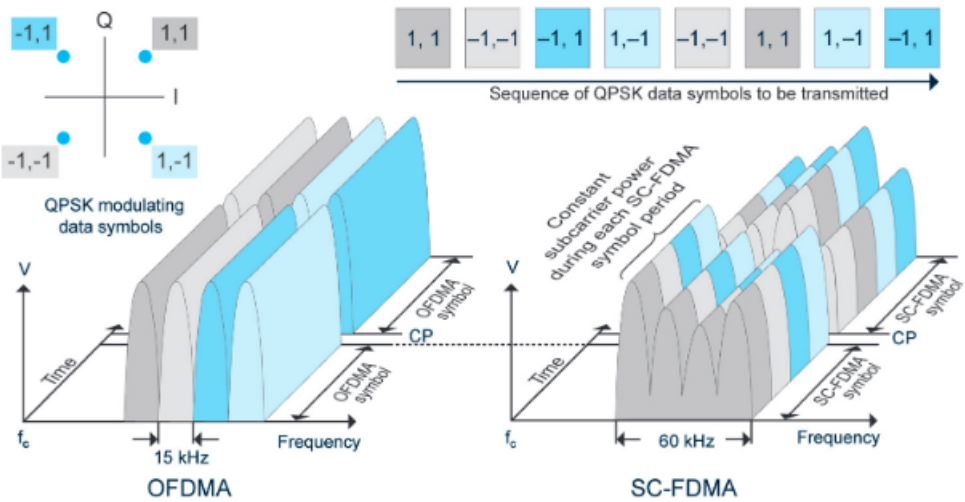

Figure 2.3-1. Comparison of OFDMA and SC-FDMA transmitting a series of QPSK symbols

Figure 2.3: Comparison between OFDMA and SC-FDMA [32]

Organizationally, the LTE uplink transmission structure is broken down into 10 ms frames [6]. These frames are divided into ten subframes that are 1 ms each. Each subframe has two slots of 0.5 ms each. These slots normally have seven 71.4 µs SC-FDMA symbols. The middle symbol of each slot is known as the Demodulating Reference Signal (DMRS), which is used for equalization and demodulation of the signal as the name implies. This system is illustrated in Fig. 2.4. In the

frequency domain, the signal is organized into resource blocks. These blocks span an entire slot in time and 180 kHz in frequency, or 12 subcarriers. The slot can have different numbers of Resource Blocks (RBs) based on the total bandwidth allocated to the LTE cell. The smallest unit in the LTE resource grid is called a Resource Element (RE).

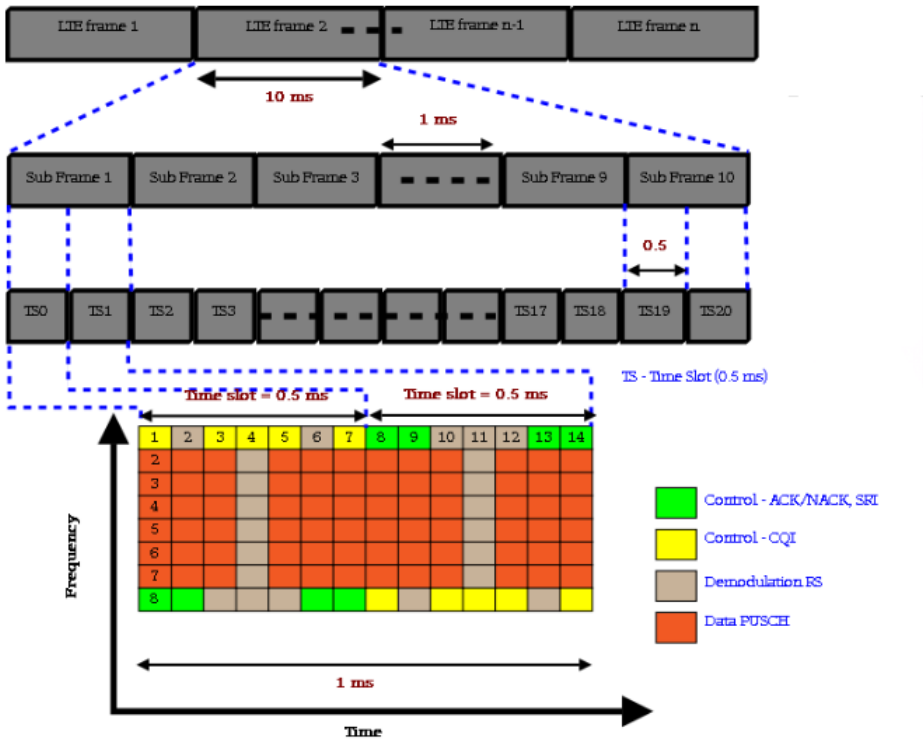

Figure 2.4: Timing structure of the LTE physical uplink layer [3]

For the goals of this work, it was decided that the DMRS would be the most appropriate section of the signal to use for the LTE dataset. It is also a set position for every uplink signal and contains no higher-order descriptive information about the device or location.

2.5 CSI and RSSI/RSRP Calculations

Besides the physical signals themselves, the dataset collects other features about the signals such as the information-rich CSI and RSSI. The methods to determine these quantities can greatly affect the values they take on, so it is important to note the steps to calculate each.

2.5.1 Channel State Information

CSI is determined in the equalizer of any communication scheme and is used to ensure reliable communications in any system, especially one with a rapidly changing electromagnetic environment. It contains information on many factors such as fading, scattering, decay, etc. and can be used by both the receiver and transmitter to improve the performance of physical communications.

For IEEE802.11g, the algorithm used in this paper to calculate CSI is the spectral temporal averaging (STA). This method has proven to be reliable and robust against noisy conditions [14]. This algorithm contains two steps and works as follows. The first symbol is demapped, and the CSI is calculated using the least-squares method shown in Eq. 2.1.

$$H_i(k) = \frac{S_{R,i}(k)}{X_i(k)} \tag{2.1}$$

where *i* is the index of the ith OFDM symbol, *H(k)* is the channel estimation, $S_R(k)$ is the received data symbol, and *X(k)* is the demapped transmitted symbol, which differs from the transmitted symbol due to imperfect channel estimation and could be incorrectly demapped. Then, the H_i is averaged in the frequency domain by the Eq. 2.2.

$$H_{update}(k) = \sum_{\lambda=-\beta}^{\beta} (\omega_\lambda H_i(k+\lambda)) \tag{2.2}$$

For the next step, the H_{update} is averaged in the time-domain by means of a low-pass filter with the following equation:

$$H_{STA,i}(k) = (1 - \frac{1}{\alpha})H_{\text{STA,i-1}}(k) + \frac{1}{\alpha}H_{update}(k) \tag{2.3}$$

$H_{STA,i}$ is the new, calculated CSI per symbol of the frame and stored for the dataset. The parameters α and β are channel dependent variables with α being related to the Doppler spread. For the purposes of this testbed, the values $\alpha = 0.5$ and $\beta = 2$ are used as optimized by [14]. Also, since most algorithms do not compute the CSI for the STF, this work opts to not as well, so the first CSI value corresponds to the first symbol of the LTF. In addition, there is 1 CSI value per symbol/64 samples (64 instead of 80 since the cyclical prefix is removed before equalization). For more information on STA, please refer to [42].

Like WiFi, LTE has its own methods for determining CSI. It obtains the initial CSI values from the DMRS and other pilot symbols using the least-squares method similar to Eq. 2.1. Once the initial values are obtained, they are averaged in order to reduce noise and interpolated for all the other carriers in the frame. A more rigorous explanation of this process as it pertains to LTE can be seen in [4].

2.5.2 RSSI/RSRP

RSSI is a loosely defined term in communications. Its formulation varies depending on the communication scheme related to the received radio signal power. This value can be used to make important optimization decisions regarding cell association, power usage, and packet scheduling. For IEEE802.11g, RSSI has no definition in the standard and is therefore decided by wireless

chipset manufacturers for their individual products. These manufacturers often provide formulas for tracing these RSSI back to a dBm value [23]. The one thing the standard does specify about RSSI is that it should be calculated only in the preamble portion of the frame [2].

Since the testbed uses SDRs and not wireless chipsets to collect signals, there is latitude on how to calculate the RSSI. A useful method in the literature that can be incorporated easily into the proposed system is described by Liu *et. al* in [22] Section 3:

$$y[n] = 10 * log_{10}(\frac{1}{N}\sum_{k=1}^{N}(I[k]^2 + Q[k]^2) \tag{2.4}$$

This is the average of the squared magnitude of samples in logarithmic scale for the preamble. I and Q represent the real and imaginary components of the sample k, and N is the length of the preamble

In LTE systems, RSSI comprises the linear average of the total received power in Watts in the measurement bandwidth over N number of resource blocks from all sources. The obvious problem with this is that including the power from all these sources can be counterproductive when only wanting to measure the power from the received signal's power. In addition, taking the average power over the entire resource grid with many empty resource elements results in a much lower value. For this reason, another metric is often used in LTE, Received Signal Received Power (RSRP). RSRP is the average value of the reference signals (pilot symbols) only measured in dBm, while RSSI is the average of the whole signal itself. The RSRP is determined from the energy received during the useful part of the pilot symbol, excluding the CP.

CHAPTER III

WIFI DATASET AND COLLECTION PROCESS

3.1 GNURadio

GNURadio 3.7 was used to implement and run the data collection algorithms for WiFi due to its open-source nature and flexibility [16]. GNURadio operates by connecting a series of functional blocks together and operating the blocks in parallel on a block per thread basis. The data collection algorithm described in this paper is based on a module in GNURadio called gr-ieee-80211 [9]. This module implements the functional blocks to build an IEEE802.11a/g/p transceiver and can be connected to internet if desired.

First, it performs a normalized autocorrelation on the complex samples coming into the SDR to detect the STF of the IEEE802.11g based on its periodicity. The Sync Short block checks the autocorrelation against a user-defined threshold and triggers if it reaches that threshold. It then passes the next 43,200 samples/540 symbols (the approximate maximum size of a IEEE802.11g frame) to the next block after performing a coarse frequency correction. Next, the Sync Long block finds the start of the frame based on a cross-correlation with the known sequence for the LTF, performs a fine frequency correction, and removes the CP before the signal is passed to a Fast Fourier Transform (FFT). The FFT input size is 64 and uses a rectangular window. The subcarriers from the FFT's output are passed to the Frame Equalization block. Here the subcarriers are demapped to their constellation points, and the CSI is calculated using the STA method. They

are then deinterleaved and decoded using a Viterbi Decoder into bits before being passed to the Decode Mac block. The Decode Mac block simply decodes the bits of the PLCP header and MAC header into their respective values based on the rate and length field in the signal field and checks the checksum. Finally, these decoded values are sent to the Parse Mac block to be printed neatly to the output. The full flowgraph of this process is shown in Fig. 3.1.

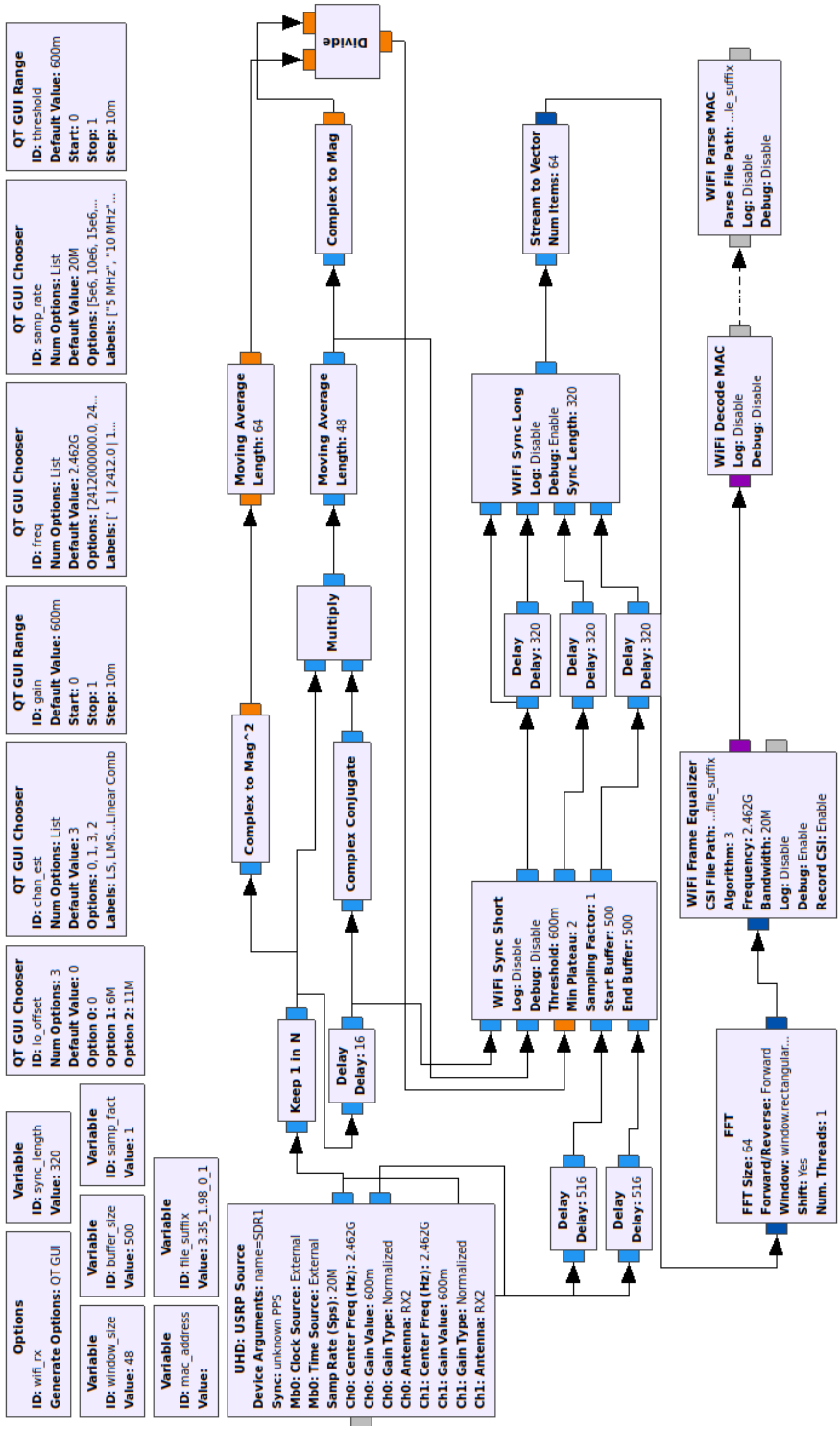

Figure 3.1: Flowgraph of WiFi receiver

3.2 Modified Collection Algorithm

In order to use this WiFi flowgraph for fingerprint collection, it needs to be modified to output the signals features such as the raw samples, CSI, and sequence number as well as labels to associate it with a device identity and location. To this end, the receiver was modified in the Sync Short and Sync Long blocks to the record the raw samples from both antennas before they were ever modified or frequency corrected. Then, the CSI calculated in the frame equalizer was recorded using a similar approach. These values were only saved if the MAC address and sequence number for the frame could be decoded and were not corrupted.

In post-processing, there are two binary files for each SDR: one containing the raw data for both of the antennas and MAC addresses and another containing the raw data for the CSI. The data in these two files are matched based on their received order. Next, the timestamps are also used along with the MAC header sequence numbers to correlate the frames among the different SDRs. At this point, every frame captured should contain the raw samples received at each SDR's two antennas and CSI. Next, the raw frames are trimmed to only their preambles and transients by first detecting the LTF using a cross-correlation. Then, the first 320 samples of the frame are saved as well as the 30 samples proceeding it. At this point, the RSSI can be calculated using the method discussed above in section 2.5.2 and attached. The final files are saved in .hdf5 format for easy loading into TensorFlow.

3.3 Testbed Layout

The space the dataset was collected in can be considered a challenging environment. It was collected in a 4.7 by 7.54 meter lab room that still contained furniture, electrical equipment, and

human beings. It was collected during daytime on campus and can therefore be reasonably expected to have other interferers on the same band. Four USRP B210 SDRs were tuned to channel 11 (2.426 GHz) with a sampling rate of 20 MHz and passively listened for incoming IEEE802.11g frames. The SDRs were positioned around the room close to the ceiling as shown in Fig. 3.2, and each were connected to their own individual computer. This positioning prevents symmetry, which should allow better performance for localization using RSSI. Each SDR uses two LP0965 directional log-periodic antennas for receiving. They are all connected to the Octoclock-G, a timing and synchronization device that should synchronize all the SDR's internal clocks.

Figure 3.2: Image of SDR setup

The testbed itself contains 55 training points and 13 testing points. The training points are arranged in a 5 by 11 grid spaced 0.5 m apart as seen in Fig. 3.3. This spacing has been shown to be beneficial for CSI fingerprinting while decreasing the performance of RSSI fingerprinting [45] [25] [18]. The testing points are randomly scattered throughout the grid sometimes occupying even the edges. This creates a more challenging dataset for the ML algorithms.

Four cell phones were used to collect this dataset. The phones were two Apple iPhone SEs, Samsung Galaxy J2 Prime, and Motorola G4 Plus. The cell phones were placed at each training and testing point on a 0.6 m stand. Once 1500 frames were collected for each collection point, the recording was stopped, and the device was moved to the next point where the recording started again. This process was continued for all four phones until there was data at all locations for each device.

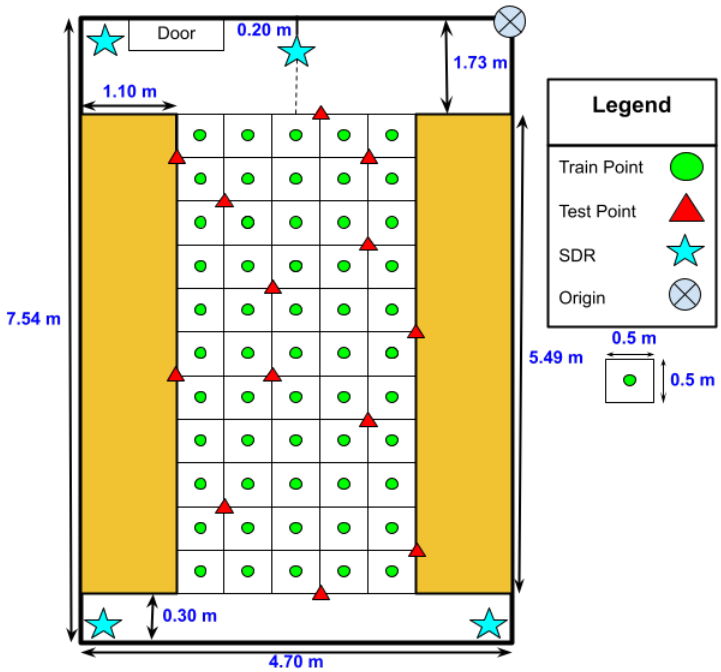

Figure 3.3: Layout of testbed for WiFi collection

Table 3.1 describes the final WiFi dataset's contents as well as the environment it was collected in.

Table 3.1: Properties and features of final WiFi dataset

MAC Format	IEEE802.11g
Frequency	2.462 GHz
Bandwidth	20 MHz
Sampling Rate	20 MHz
Number of locations	68 (55 training + 13 testing)
Number of transmitters	4
Number of receivers	4
Number of samples per frame	348 samples
Number of frames per location	24,000 = 1,500 frames * 4 TXs * 4 RXs
Number of frames per device	408,000 = 1,500 frames * 68 locations * 4 RXs
Dataset size	18.381 GB
Output file format	.hdf5 (hierachical compressed file)
CSI included	True
RSSI/RSRP included	True
Two antennae used	True

CHAPTER IV

LTE DATASET AND COLLECTION PROCESS

4.1 srsLTE

To be able to ascertain which device is transmitting and calculate their signals' CSI, one needs knowledge only privy to the Evolved Packet Core (EPC) of the LTE network that the device is connected. This presents a difficult problem as it eliminates any passive sniffing methods that can be used and requires a direct link to the cell phone. This would require simulating an LTE E-UTRAN Node B (eNodeB) to communicate with the cell phone. In LTE, the eNodeB acts as a basestation and handles the RF interface with cellular devices and many other cell functions. It was decided that the most appropriate software to use for this work would be srsLTE [17].

srsLTE 20.04.10 is a software suite that allows users to implement their own LTE EPC, eNodeB, and even User Equipment (UE), or cellular device. It can also connect to the internet through the terminal it is run on. The EPC and eNodeB are typically run on the same computer while the the UE must be run independently. Users can configure srsLTE through the use of configuration files that are read on startup. In these files, they can configure the network's bandwidth, operating frequency, RF gain, etc. With this software in hand, it is possible to connect normal cell phones to the eNodeB and EPC providing data service. This software provides us with the ability to directly connect with cell phones and obtain information privy only to the LTE network.

It is important to have a simplified idea of how srsLTE operates at the physical layer for the LTE uplink in order to collect data. srsLTE uses SC-FDMA modulation with its frames allowing it to still schedule multiple users per time slot to different RBs. This is notably different than IEEE802.11g which uses carrier-sense multiple access (CSMA) for users to know when to transmit. When a new cellular device wants to connect to an eNodeB, it must first undergo the attach process by communicating on the Physical Random Access CHannel (PRACH). Once the UE has been verified by the network and synchronized in time and frequency, it is scheduled in the MAC layer to transmit when needed on the Physical Uplink Shared CHannel (PUSCH). Once a UE transmits, the eNodeB records the slot it was assigned to. Then, it performs a FFT, calculates the CSI, and decodes the subcarriers into bits. The MAC layer is notified, and the bits from each phone are sent to different places.

4.2 Modified Collection Algorithm

The first modification made to srsLTE was to record the raw samples from both antennae of the SDR when the UE was scheduled to transmit. This was accomplished by modifying the eNodeB code to copy the sample buffers when the UE was scheduled to transmit. Next, the CSI was retrieved after it was calculated by the eNodeB using the least-square estimation and linear interpolation. Finally, the RSRP was calculated by taking the average of the reference symbols power and scaling it before going into the logarithmic domain. All these values were then saved to file along with the timestamp of the subframe.

It is important to note that in order to collect the signal emitted by a specific cell phone, one must filter out the subcarriers it was assigned from the rest of the signal. However, this work

attempts to only use the most raw data possible and therefore has decided to only collect signals from one phone at a time to ensure the best data quality.

4.3 Testbed

The testbed for LTE is considerably smaller than WiFi but occupies an equally challenging environment. It was collected in the 4.88 by 7.92 meter second floor of a fully furnished house. Only one USRP B210 SDR with four omnidirectional VERT2450 antennas was used to collect data, which was tuned to downlink EARFCN 3400 (2685 MHz for downlink, 2565 MHz for uplink). There were 25 RBs which translates to roughly 5 MHz of bandwidth used. The sampling rate was set to 5.76 MHz. The SDR was positioned as shown in Fig. 4.1 roughly one meter above the ground and was designated as the origin.

There are 17 collection points in this dataset spaced 1 by 1 meter apart from each other in a 5 by 2 meter grid. Five phones were set at each point roughly one meter above the ground. The phones were a Samsung Galaxy J2 Prime, Moto G4 Plus, and three IPhone SEs. At each location, 3000 LTE subframes of 5760 samples were collected for the dataset, resulting in 51,000 frames per device.

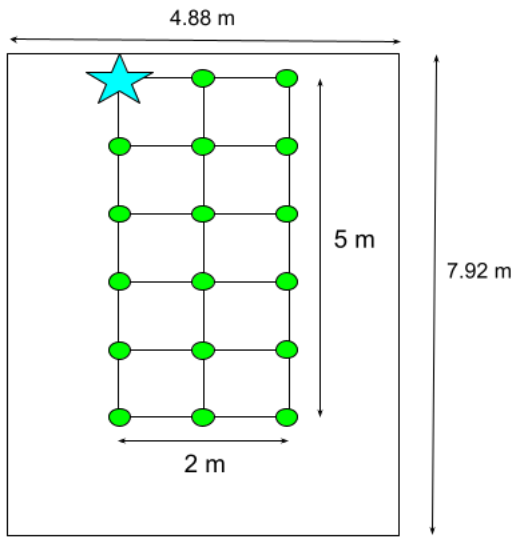

Figure 4.1: Layout of testbed for LTE collection showing SDR (blue) and collection points (green)

Table 4.1 describes the properties and features of the final dataset.

Table 4.1: Properties and features of final LTE dataset

Frequency	2.565 GHz
Bandwidth	5 MHz
Sampling Rate	5.76 MHz
Number of locations	17
Number of transmitters	4
Number of receivers	1
Number of samples per frame	1,500 samples
Number of frames per location	12,000 = 3,000 frames * 4 TXs
Number of frames per device	51,000 = 3,000 frames * 17 locations
Dataset size	6.125 GB
Output file format	.npz (Numpy compressed file)
Device fingerprinting accuracy	90.23%
Location fingerprinting accuracy	1.08 meters
CSI included	False
RSSI/RSRP included	False
Two antennae used	False

4.4 Fingerprinting Algorithms

No CSI, RSRP, or second antenna was needed for these algorithms. This shows the potential simplicity of the datasets constructed by these methods while still achieving good ML performance.

4.4.1 Device Fingerprinting

An MLP was used to validate this dataset for device fingerprinting. These are fairly standard networks and can serve as a baseline for any future models trained on the dataset. This MLP was built using scikit-learn on Google Colab using Python 3.6.9. Only one location (0.1) of frames was used for this network giving us a dataset of 3000 LTE frames at 27 dB SNR from each device. The frames are first trimmed to the first 1500 samples before the absolute value of the samples were taken. These absolute values were then standardized to a range of 0 to 1. Next, they are passed to the network shown in Table 4.2 below. As seen in the table, the MLP consists of four layers with 515 neurons and ReLU activations each before being passed to the output layer. The MLP uses a 5-fold cross-validation and has an L2 penalty term of 0.0001. It has a batch size of 200 and uses the Adam (adaptive moment estimation) method for training with a learning rate of 0.001 [21]. The training and testing split was 80% and 20%.

Table 4.2: Architecture of LTE device fingerprinting MLP

Layer	**Size**	**Options**
Input	1500	Size of signals
Dense	515	relu activation
Dense	515	relu activation
Dense	515	relu activation
Dense	515	relu activation
Output	1	predicts class label

4.4.2 Location Fingerprinting

For location fingerprinting, this work uses a 1D CNN. CNN exploit the spatial correlations in data and require much less memory than normal DNNs. The CNN built for this task uses TensorFlow built on Google Colab using Python 3.6.9. The network optimizer was the Adam method for training with a learning rate of 0.0001 [21]. The batch sizes were 64, and the training, validation, and testing splits were 60%, 20%, and 20% respectively. The structure for the network is seen below in Table 4.3. The absolute value of the first 1500 complex samples were used for this network to decrease network size. Next, the frames are passed to a conv1D with 25 filters of size 20 and a max pooling layer of size 2. Next, they were sent to a Batch Normalization layer to reduce overfitting, where afterwards is was sent again to a conv1D layer with 40 filters of size 13 and a max pooling layer of 2. Finally, the conv1D layer is used with 56 filters of size 7 and a max pooling layer of size 2. They are followed by a dropout layer of 0.5 and then flattened. This is sent to a 3 fully-connected layers of size 64, 32, 24, and 12 before going through another dropout layer of 0.7. This is connected to the output layer with two nodes representing the x and y coordinate of the device.

Table 4.3: Architecture of LTE location fingerprinting CNN

Layer	**Size**	**Options**
Input	(None, 1500, 1)	
Conv1D	25	kernel size 20 relu activation
MaxPool1D	2	
BatchNormalization		
Conv1D	40	kernel size 13 relu activation
MaxPool1D	2	
Conv1D	56	kernel size 7 relu activation
MaxPool1D	2	
Dropout		rate 0.5
Flatten		
Dense	64	tanh activation
Dense	32	tanh activation
Dense	24	tanh activation
Dense	12	tanh activation
Dropout		rate 0.7
Dense	2	(for x and y coordinate)

4.5 Fingerprinting Results

4.5.1 Device Fingerprinting Results

The MLP for device fingerprinting ran for 59 epochs before meeting its early stopping condition of not improving the validation score by 0.001. This took in total 3.4 minutes. The confusion matrices for the training and testing sets are shown below in Fig. 4.2. It can be seen that the misclassifications were fairly evenly spread out among the devices. The only notable exceptions would be between two of the IPhone SEs which is to be expected since they are both of the same make and model. However, overall the algorithm shows good performance for the dataset with an accuracy of 90.23% and proves to be a valuable baseline for future work.

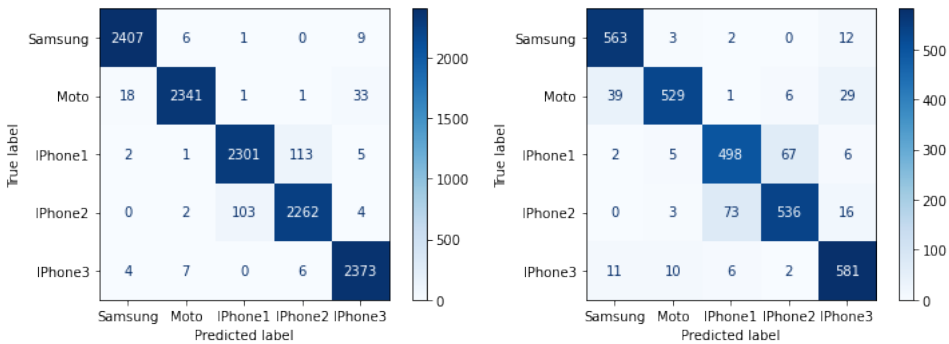

Figure 4.2: Training(left) and testing(right) confusion matrices for the MLP

4.5.2 Location Fingerprinting Results

The CNN ran for 25 epochs and took approximately 7.1 minutes. The training and validation losses for the model is shown below in Fig. 4.3. The final root mean squared error (RMSE), or Euclidean distance, for the test set was 1.08 meters. This shows a good approximation given the spacing between the points should allow the user to differentiate between where in the room the

phone emitted the signal. This is simply a preliminary work and could most likely be improved given other features such as CSI or RSRP.

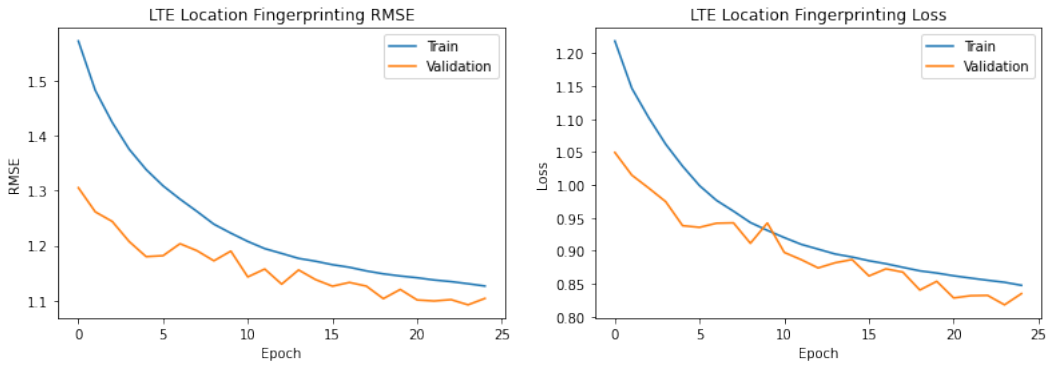

Figure 4.3: The training and validation RMSE (left) and loss (right)

CHAPTER V

CONCLUSIONS

To conclude, this work details methods for one to build their own dataset of WiFi or LTE devices for device/location fingerprinting. This includes the ability to use multiple low-end receivers and open-source software to collect the data. Different features are gathered including raw IQ samples, CSI, and RSSI/RSRP.

In addition, two datasets are provided using the above methodology for both WiFi and LTE. These datasets contain many signals from multiple devices from different locations in a crowded room. This provides extra difficulty to the datasets and the problems they attempt to solve. Finally, baseline ML algorithms were provided for each that had performances similar to state-of-the-art algorithms verifying their authenticity. Others can use these datasets and baseline algorithms to test their fingerprinting algorithm's performance.

Future work regarding this thesis could involve calculating different features from the data collected. In addition, additional methods to collect more modern WiFi protocols such as IEEE80211ac or LoRA could be designed with open-source and low-cost in mind.

REFERENCES

[1] "IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: High Speed Physical Layer in the 5 GHz band," *IEEE Std 802.11a-1999*, 1999, pp. 1–102.

[2] "IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, 2016, pp. 1–3534.

[3] "4G LTE Uplink frame common to FDD and TDD access modes.,", May 2019.

[4] 3rd Generation Partnership Project, "Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) Radio Transmission and Reception, document TS 36.101," *Technical Specification Group Radio Access Network*.

[5] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,", 2015, Software available from tensorflow.org.

[6] E. U. T. R. Access, "Physical Channels and Modulation, document TS 36.211 v. 13.2. 0," *Technical Specification (TS), 3GPP*, 2017.

[7] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, 2000, vol. 2, pp. 775–784 vol.2.

[8] G. Baldini and G. Steri, "A survey of techniques for the identification of mobile phones using the physical fingerprints of the built-in components," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, 2017, pp. 1761–1789.

[9] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "Performance Assessment of IEEE 802.11p with an Open Source SDR-based Prototype," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, May 2018, pp. 1162–1175.

[10] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," *Proceedings of the 14th ACM international conference on Mobile computing and networking*, 2008, pp. 116–127.

[11] R. W. Chang, "Synthesis of band-limited orthogonal signals for multichannel data transmission," *The Bell System Technical Journal*, vol. 45, no. 10, 1966, pp. 1775–1796.

[12] Y. Chapre, A. Ignjatovic, A. Seneviratne, and S. Jha, "CSI-MIMO: Indoor Wi-Fi fingerprinting system," *39th Annual IEEE Conference on Local Computer Networks*, 2014, pp. 202–209.

[13] S. Dogan Tusha, A. Tusha, and H. Arslan, "OFDM with index modulation for asynchronous mMTC networks," *Sensors*, vol. 18, 04 2018, p. 1280.

[14] J. A. Fernandez, D. D. Stancil, and F. Bai, "Dynamic channel equalization for IEEE 802.11p waveforms in the vehicle-to-vehicle channel," *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2010, pp. 542–551.

[15] R. M. Gerdes, T. E. Daniels, M. Mina, and S. Russell, "Device Identification via Analog Signal Fingerprinting: A Matched Filter Approach.," *NDSS*, 2006.

[16] GNU Radio Website, ,", accessed February 2021.

[17] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "srsLTE: an open-source platform for LTE evolution and experimentation," *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*. ACM, 2016, pp. 25–32.

[18] O. Hashem, M. Youssef, and K. A. Harras, "WiNar: RTT-based Sub-meter Indoor Localization using Commercial Devices," *2020 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2020, pp. 1–10.

[19] H. Jafari, O. Omotere, D. Adesina, H. Wu, and L. Qian, "IoT Devices Fingerprinting Using Deep Learning," *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, 2018, pp. 1–9.

[20] S. Jana and S. K. Kasera, "On Fast and Accurate Detection of Unauthorized Wireless Access Points Using Clock Skews," *IEEE Transactions on Mobile Computing*, vol. 9, no. 3, 2010, pp. 449–462.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] W. Liu, M. Kulin, T. Kazaz, A. Shahid, I. Moerman, and E. De Poorter, "Wireless Technology Recognition Based on RSSI Distribution at Sub-Nyquist Sampling Rate for Constrained Devices," *Sensors*, vol. 17, 09 2017.

[23] G. Lui, T. Gallagher, B. Li, A. G. Dempster, and C. Rizos, "Differences in RSSI readings made by different Wi-Fi chipsets: A limitation of WLAN localization," *2011 International Conference on Localization and GNSS (ICL-GNSS)*, 2011, pp. 53–57.

[24] 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*, vol. 12, no. 1, 2018, pp. 160–167.

[25] G. Pecoraro, S. Di Domenico, E. Cianca, and M. De Sanctis, "CSI-based fingerprinting for indoor localization using LTE signals," *EURASIP Journal on Advances in Signal Processing*, vol. 2018, no. 1, 2018, p. 49.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, 2011, pp. 2825–2830.

[27] D. Rayborn, L. Smith, S. Kodipaka, N. Smith, B. Tang, J. E. Ball, and M. Young, "Towards simulating multipath interference at detectors: a tool for validating location fingerprinting methods," *Signal Processing, Sensor/Information Fusion, and Target Recognition XXIX*, I. Kadar, E. P. Blasch, and L. L. Grewe, eds. International Society for Optics and Photonics, 2020, vol. 11423, pp. 203 – 210, SPIE.

[28] S. U. Rehman, K. Sowerby, and C. Coghill, "Analysis of receiver front end on the performance of RF fingerprinting," *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, 2012, pp. 2494–2499.

[29] S. U. Rehman, K. W. Sowerby, S. Alam, and I. Ardekani, "Portability of an RF fingerprint of a wireless transmitter," *2014 IEEE Conference on Communications and Network Security*, 2014, pp. 151–156.

[30] H. Rizk, M. Torki, and M. Youssef, "CellinDeep: Robust and Accurate Cellular-Based Indoor Localization via Deep Learning," *IEEE Sensors Journal*, vol. 19, no. 6, 2019, pp. 2305–2312.

[31] D. Roy, T. Mukherjee, M. Chatterjee, E. Blasch, and E. Pasiliao, "RFAL: Adversarial Learning for RF Transmitter Identification and Classification," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 2, 2020, pp. 783–801.

[32] M. Rumney, *Air Interface Concepts*, chapter 2, John Wiley Sons, Ltd, 2013, pp. 11–89.

[33] T. M. Schmidl and D. C. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Transactions on Communications*, vol. 45, no. 12, 1997, pp. 1613–1621.

[34] D. Shaw and W. Kinsner, "Multifractal modelling of radio transmitter transients for classification," *IEEE WESCANEX 97 Communications, Power and Computing. Conference Proceedings*, 1997, pp. 306–312.

[35] L. Smith, N. Smith, J. Hopkins, D. Rayborn, J. E. Ball, B. Tang, and M. Young, "Classifying WiFi "physical fingerprints" using complex deep learning," *Automatic Target Recognition XXX*, R. I. Hammoud, T. L. Overman, and A. Mahalanobis, eds. International Society for Optics and Photonics, 2020, vol. 11394, pp. 82 – 96, SPIE.

[36] L. Smith, N. Smith, D. Rayborn, B. Tang, J. E. Ball, and M. Young, "Identifying unlabeled WiFi devices with zero-shot learning," *Automatic Target Recognition XXX*, R. I. Hammoud, T. L. Overman, and A. Mahalanobis, eds. International Society for Optics and Photonics, 2020, vol. 11394, pp. 129 – 137, SPIE.

[37] N. Soltani, K. Sankhe, J. Dy, S. Ioannidis, and K. Chowdhury, "More Is Better: Data Augmentation for Channel-Resilient RF Fingerprinting," *IEEE Communications Magazine*, vol. 58, no. 10, 2020, pp. 66–72.

[38] N. Soltanieh, Y. Norouzi, Y. Yang, and N. C. Karmakar, "A Review of Radio Frequency Fingerprinting Techniques," *IEEE Journal of Radio Frequency Identification*, vol. 4, no. 3, 2020, pp. 222–233.

[39] W. C. Suski II, M. A. Temple, M. J. Mendenhall, and R. F. Mills, "Using spectral fingerprints to improve wireless network security," *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*. IEEE, 2008, pp. 1–5.

[40] L. Temino, G. Berardinelli, S. Frattasi, K. Pajukoski, and P. Mogensen, "Single-user MIMO for LTE-A Uplink: Performance evaluation of OFDMA vs. SC-FDMA," 02 2009, pp. 304 – 307.

[41] S. Wang, L. Peng, H. Fu, A. Hu, and X. Zhou, "A Convolutional Neural Network-Based RF Fingerprinting Identification Scheme for Mobile Phones," *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 115–120.

[42] T. Wang, A. Hussain, Y. Cao, and S. Gulomjon, "An improved channel estimation technique for IEEE 802.11 p standard in vehicular communications," *Sensors*, vol. 19, no. 1, 2019, p. 98.

[43] X. Wang, L. Gao, and S. Mao, "BiLoc: Bi-modal Deep Learning for Indoor Localization with Commodity 5GHz WiFi," *IEEE Access*, 01 2017.

[44] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, 2016, pp. 763–776.

[45] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, 2016, pp. 763–776.

[46] X. Wang, X. Wang, and S. Mao, "Indoor Fingerprinting With Bimodal CSI Tensors: A Deep Residual Sharing Learning Approach," *IEEE Internet of Things Journal*, vol. 8, no. 6, 2021, pp. 4498–4513.

[47] M. D. Williams, M. A. Temple, and D. R. Reising, "Augmenting bit-level network security using physical layer RF-DNA fingerprinting," *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*. IEEE, 2010, pp. 1–6.

[48] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device Fingerprinting in Wireless Networks: Challenges and Opportunities," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, 2016, pp. 94–104.

[49] Z. Yang, Z. Zhou, and Y. Liu, "From RSSI to CSI: Indoor Localization via Channel Response," *ACM Comput. Surv.*, vol. 46, no. 2, Dec. 2013.

[50] 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*, vol. 2, no. 4, 2018, pp. 197–205.

[51] M. Youssef and A. Agrawala, "The Horus location determination system," *Wireless Networks*, vol. 14, no. 3, 2008, pp. 357–374.

[52] H. Zhang, Z. Zhang, S. Zhang, S. Xu, and S. Cao, "Fingerprint-Based Localization Using Commercial LTE Signals: A Field-Trial Study," *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–5.

[53] L. Zhang, E. Ding, Y. Hu, and Y. Liu, "A novel CSI-based fingerprinting for localization with a single AP," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, 2019, p. 51.