8-2022

Deep Learning Based Localization of Zigbee Interference Sources Using Channel State Information

Dylan Kensler
dkensle@g.clemson.edu

DEEP LEARNING BASED LOCALIZATION OF ZIGBEE INTERFERENCE SOURCES USING CHANNEL STATE INFORMATION

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Electrical Engineering

---

by
Dylan Zachary Kensler
August 2022

Accepted by:
Dr. Linke Guo, Committee Chair
Dr. Carl Baum
Dr. Harlan Russell

Abstract

As the field of Internet of Things (IoT) continues to grow, a variety of wireless signals fill the ambient wireless environment. These signals are used for communication, however, recently wireless sensing has been studied, in which these signals can be used to gather information about the surrounding space. With the development of 802.11n, a newer standard of WiFi, more complex information is available about the environment a signal propagates through. This information called Channel State Information (CSI) can be used in wireless sensing. With the help of Deep Learning, this work attempts to generate a fingerprinting technique for localizing a ZigBee interference source in the presence of 802.11.

Dedication

I'd like to dedicate this work to my family and my girlfriend, Megan. Without the support of you all, finishing this thesis would not have been possible and I would have quit 100 times over. You all pushed me to complete my Master's degree and I couldn't be more thankful to have a great support system.

Acknowledgments

I'd like to acknowledge the help and guidance of my advisor Dr. Linke Guo, and two of my colleagues, ChunChih Lin and Anthony Chen. ChunChih was always available to help whenever I needed it, and he could not have been more kind in helping me. His knowledge about communications far exceeds anything that I would expect out of PhD student and he was always willing to pass on some of the knowledge to me. He also contributed significantly in collecting the data for this project. Anthony and I spent, what feels like, an infinite number of days in the library together working on our respective projects. I guess you could say it was a sort of moral support since misery loves company.

I'd like to thank Dr. Harlan Russell and Dr. Carl Baum for serving on my committee and being two of the best professors I've had over the past six years at Clemson. They were always kind and willing to help with homework or whatever I needed help with.

Finally, I'd like to thank the rest of the ECE staff for guiding me through my master's degree. Mrs. Jeanine Hayes was immensely helpful in providing support and logistics of deadlines, paperwork, etc, and exceedingly kind when I would show up at her office door. I'd like to thank everyone involved in providing me with a Teaching Asisstantship. Trusting me to teach undergraduate students about senior design and microcontrollers really humbled me and it was a pleasure to be involved in their college experience.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Wireless Sensing is a growing topic in communications because it allows for a cheap and robust way to achieve various application goals. These applications include but are not limited to detection, navigation, imaging, and tracking. These applications of wireless sensing can be used in many commercial situations that help people in life threatening situations. Things such as locating firefighters in burned buildings, or navigating through disaster struck areas. Each of these tasks can be done utilizing the already commercially available WiFi systems.

As the Internet of Things (IoT) grows substantially with new devices and applications, interference between wireless standards, called cross technology interference (CTI), becomes increasingly common in a wireless environment. While many areas of academia are working to mitigate the effects of interference and develop a thriving coexistent environment, the thought occurs of using this interference for wireless sensing. Localizing interference sources could be extremely useful in certain situations, like in the case of a first responder's communication equipment failing. In the event this happens, their equipment might be sending signals into a wireless environment but the receiver fails to recognize it, or the signal is compromised by other wireless signals. However, a commercially available WiFi system could then localize their communication. Another application could be if an IoT device is being used in some adversarial way, the WiFi system could localize this device and send an administrator to remove it.

In the past fifteen years, new WiFi standards have allowed access to more complex information about signal propagation than previously thought. This, paired with deep learning techniques that are at the forefront of wireless sensing, can contribute heavily to more accurate and precise

localization ideas.

1.1 Statement of Work

The goal of this work is to introduce a new way to localize 802.15.4 ZigBee interference sources using deep learning and channel state information. The rest of this paper will be organized as follows. Chapter two will introduce background information, chapter three will talk about the methodology used in this work, chapter four will evaluate the system and chapter five will conclude the paper and discuss future work.

Chapter 2

Background

In this chapter, background information will be provided on current localization techniques, deep learning approaches, channel state information, and cross technology interference.

2.1 Indoor Localization

Localization is a technique in which the position is given of an object. Localization has been around for several decades, and was originally used in World War 2 to locate soldiers in extreme situations. Then a global positioning system (GPS) was introduced during the Vietnam war, and several decades later, GPS was introduced for commercial applications [11]. GPS is now used by virtually everyone in the United States in some way or another. This system can be used for navigation, mapping, locating, tracking, and many other applications. However, this localization system is primarily outdoors and does not work well indoors. This paper will focus on indoor localization rather than the previously developed and distributed outdoor localization techniques.

Indoor localization is the process of locating a user or device inside an enclosed area. This idea will have the potential to be used in commercial applications, like monitoring the elderly in hospitals or nursing homes, locating people in disaster struck areas, and mapping large indoor buildings for people to get around. It can also benefit novel technology such as wireless sensing and IoT systems [12], such as uncovering lost items or removing adversarial devices.

2.1.1 Localization Techniques

There are a few techniques that have been previously used and will be briefly described here. Most previous approaches localize a transmitter though triangulation, similar to how an earthquake's location might be found. This technique requires the use of the received signal strength (RSS), or alternatively, the received signal strength indicator (RSSI), and utilizes the strength of the incoming signal to determine the distance between a transmitter and receiver. The idea is that there are three receivers that will know how far a transmitter is from each of them, and their intersection point will define where the device is located. This can be seen in Figure 2.1 below. This solution is simple and cheap, however, it suffers from poor accuracy due to signal attenuation and the high volatility of signal propagation.

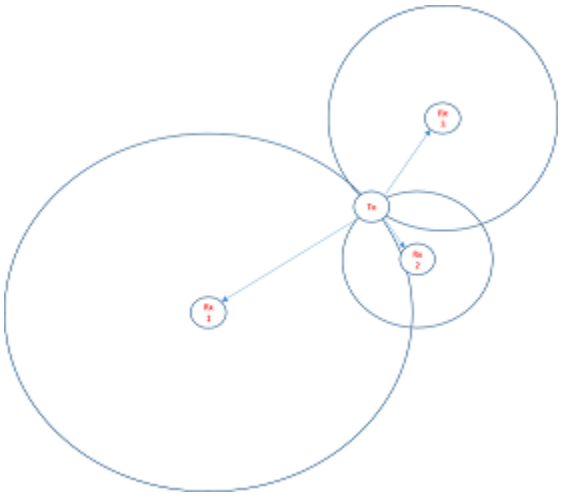


Figure 2.1: RSSI Localization

Newer approaches require the use of Channel State Information (CSI). The CSI provides the amplitude and phase of the channel in which a signal is transmitted. This information can be described by equation 2.1, where $|H(f)|$ is the amplitude and $\angle H(f)$ is the phase information of the propagation channel. CSI can be collected using many commercial network interface cards (NICs) and results in a higher localization accuracy and more stable measurements. This will be discussed further in a more detailed section about Channel State Information.

$$H(f) = |H(f)|e^{j\angle H(f)} \tag{2.1}$$

Fingerprinting is another common practice used in localization. In this technique, one of the

above data sets is collected and then analyzed. From here a map of discrete locations will be drawn. When deployed, the receiver will compare the received data (CSI or RSSI) to the previously analyzed data and match it to a location on the map. This technique is typically used in conjunction with artificial neural networks. A few less popular data sets being used include angle of arrival (AOA) and time of flight (TOF), in which this paper will not cover.

2.1.2 Localization Protocols and Systems

Most wireless protocols can be used for sensing, but the main protocol that is currently being researched for localization is 802.11 WiFi because it is widely commercially available and no additional infrastructure is needed to support it. Some other protocols that have been used are 802.15.1 Bluetooth, 802.15.4 ZigBee, and even acoustic and ultrasound signals. Bluetooth has recently seen the support of Apple using it for localization in their new feature, iBeacons, however ZigBee is less studied since its lower power feature makes computing complex values difficult.

There are two main types of localization systems, namely device based localization (DBL) and monitor based localization (MBL). DBL is a system in which a user or device will determine their own location, relative to some anchor point. It is used for things such as navigation, and uses reference nodes in the surrounding environment to determine where it is positioned. On the other hand, MBL is used for an administrative system to determine where a specific device or user is located. It is used for applications like tracking and uses its own reference nodes to determine the device's location.

2.1.3 Challenges

Traditionally, localization faces many challenges. One of the main problems using traditional techniques like RSSI is accuracy. The reason that RSSI struggles with accurate measurements is due to the multipath effect, in which the receiver will receive various attenuated and phase delayed sources of the same signal. To solve this issue using traditional approaches, there have been implementations of multipath suppressing algorithms to catch only the Line-Of-Sight (LOS) signal, however, these implementations are complex and infeasible on a wide scale.

Another issue with the traditional approaches, is that RSSI is unlikely to capture the interference sources. WiFi typically transmits at full power ($\sim$100MW) and ZigBee's low power

transmissions are about 1% of that. This comparatively low power will not affect the received signal strength enough to differentiate different interference locations, especially considering the effect of attenuation as the signal propagates. Thus, to solve these issues, more complex data is needed like channel state information.

2.2 Channel State Information

Channel State Information (CSI) is a result of an update in wireless standards when 802.11n came about. This new standard allowed receivers to gather information about a channel in which a signal is transmitted. In a system that is using MIMO-OFDM (Multiple Input Multiple Output, Orthogonal Frequency Division Multiplexing) in a channel with M transmit antennas, N receive antennas, and K subcarriers, the returned three dimensional CSI matrix, H, will represent amplitude attenuation and phase shift of multi-path channels. Because H is a 3D matrix of M transmit antennas by N receive antennas by K subcarriers, it can be used in image recognition machine learning problems because the input is similar to an image that has pixels of height, h, by width, w, by depth, d, in which depth describes the three color channels (rgb).

2.2.1 Theory

As briefly stated above, wireless systems are able to acquire CSI which contains information about the wireless environment a received signal has propagated through. This information details how wireless signals behave during transmission from the transmitter to the receiver. Each entry in CSI shows the Channel Frequency Response (CFR) represented by Equation 2.2, where a is the amplitude attenuation factor, τ is the propagation delay, f is the carrier frequency [8], and N is the number of paths that the signal will take.

$$H(f;t) = \sum_{n}^{N} a_n(t) e^{-j2\pi f \tau_n(t)} \tag{2.2}$$

CSI is heavily influenced by multi-path channels, processing techniques, and the hardware and software being used to collect this information. The multi-path effect is a property of wireless signals in which the transmitted signal could take multiple different paths to the receiver, causing different time delays. Figure 2.2 shows an example of the multi-path effect.

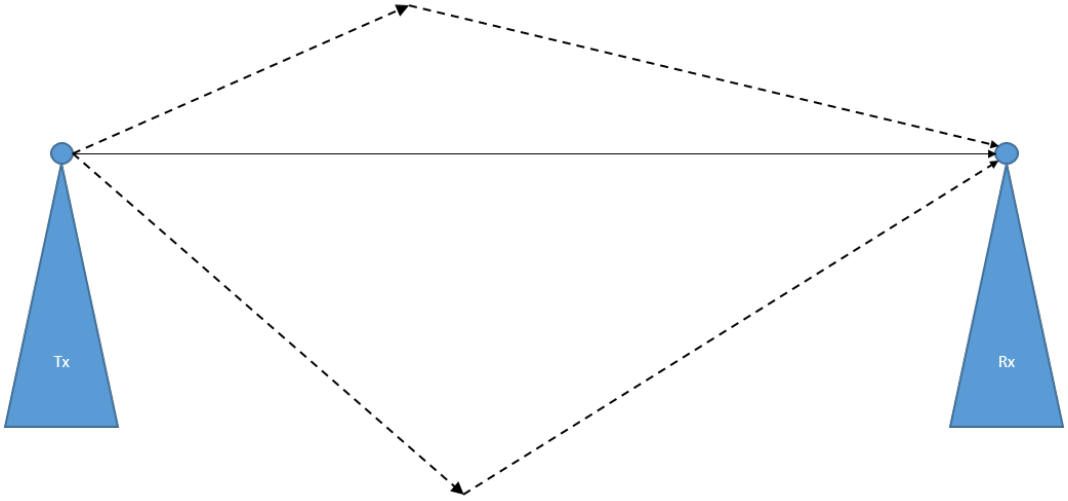


Figure 2.2: Multi-Path Effect

This effect has a significant impact on CSI. There are also other factors that affect the channel that are out of the scope of this paper. A model for the CSI on a single subcarrier can be seen in Equation 2.3, where the first term accounts for the multi-path effect on N paths and the subsequent terms account for Cyclic Shift Diversity, Sampling Time Offset, Sampling Frequency Offset, and beamforming, respectively. In Equation 2.3, $H_{i,j,k}$ is the CSI from the ith transmitting antenna to the jth receiving antenna on the kth subcarrier, d is the path length, f is the carrier frequency, τ is the time delay from Cyclic Shift Diversity, ρ is the Sampling Time Offset, η is the Sampling Frequency Offset, and q and ζ are the amplitude attenuation and phase shift of the beamforming matrix [8]. The multi-path channel is of the utmost importance in this model, thus signal processing is needed to remove the impact of the other factors.

$$H_{i,j,k} = (\sum_{n}^{N} a_n e^{-j2\pi d_{i,j,n} f_k / c}) e^{-j2\pi\tau_i f_k} e^{-j2\pi\rho f_k} e^{-2\pi\eta(f_k'/f_k - 1) f_k} q_{i,j} e^{-j2\pi\zeta_{i,j}} \tag{2.3}$$

2.2.2 Experimental

These equations can be very computationally expensive, so in reality there needs to be some simplification. The channel can be described by equation 2.4, where $\mathbf{y}$ is the received signal, $\mathbf{x}$ is the known transmitted signal, H is the channel, and $\mathbf{n}$ is additive white Gaussian noise (AWGN). In a WiFi preamble, there is a section called Long Training Symbols (LTFs), which are pre-defined. The

receiver uses these pre-defined symbols, and it's received symbols to estimate the channel matrix, H.

$$y = Hx + n \tag{2.4}$$

2.2.3 Collection

When the 802.11n WiFi standard came out, it released a new way to gather information about the channel between transmitter and receiver. Until this point, RSSI values were the only way to analyze information about wireless conditions, and as explained previously, RSSI only records the signal power received by the listener. However, because of this release, more complex information can be collected about the channel between the transmitter and receiver to learn about wireless conditions.

A network interface card (NIC) is what enables CSI to be collected. The one that is used most frequently and in this paper, is the 802.11n CSI Tool [3], which uses the Intel WiFi Link 5300 NIC. The way that it works is it reports the strongest 30 groups of subcarriers, of the 56 subcarriers available in a 20 MHz channel. There are also other tools available such as the OpenRF Tool and the Atheros CSI Tool both of which are modified versions of the 802.11n CSI Tool [8].

2.3 Machine Learning

Inspired by the human brain, researchers created neural networks, which are comprised of neurons, as a sub-field of artificial intelligence and machine learning. These networks try to learn features of data sets and predict a desired result. Society has turned to machine learning to solve some of the world's biggest problems. Problems in which a large amount of data has been (or could be) collected and applying that data to a problem. These problems include things such as how discovering how medical patients might respond to different treatments based on historical medical records or how to reduce traffic congestion by looking at prior traffic control methods [6].

Deep learning (DL) is one such subset of both artificial intelligence (AI) and machine learning (ML). Deep learning can be applied to almost any task to make it automated, and recently it was discovered that a deep learning model, ResNet-152 [4], has smaller error rate than humans when trying to recognize images. This is exciting for the future in machine learning, because it has finally

surpassed human capability.

There are four main types of deep learning, which include supervised learning, semi-supervised learning, unsupervised learning and deep reinforcement learning. These different techniques apply different ways for neurons to learn which features about a data set is important, and each technique can be applied to different applications.

Supervised learning is a type of deep learning in which labeled data is provided to a neural network in order for it validate or invalidate its predictions. The user will pass in some data to a machine learning model, and the model will make a prediction or output, $\hat{Y}$, based on some features that it currently thinks are important. Then, the user will tell the model what the correct output is, Y, and compute a loss function, $f_l(Y, \hat{Y})$. The model will then update its parameters to predict an output with more accuracy. There are many different types of networks that use supervised learning, such as deep neural networks (DNN), recurrent neural networks (RNN), and convolutional neural networks (CNN). Semi-supervised learning will briefly be discussed later in the paper, but unsupervised learning and deep reinforcement learning are out of the scope of this paper.

2.3.1 Convolutional Neural Network

Convolutional neural networks (CNNs) is one type of DNN that has several advantages over other types of networks including simplicity of tuning, similarity to human visual processing system, and optimality for processing 2-Dimensional and 3-Dimensional images [1]. CNNs consist of three main types of layers which are the convolutional layer, the sub-sampling or pooling layer, and the fully connected layer. A network will consist of any multiple combination of these three layers. An example of a network can be seen in Figure 2.3.

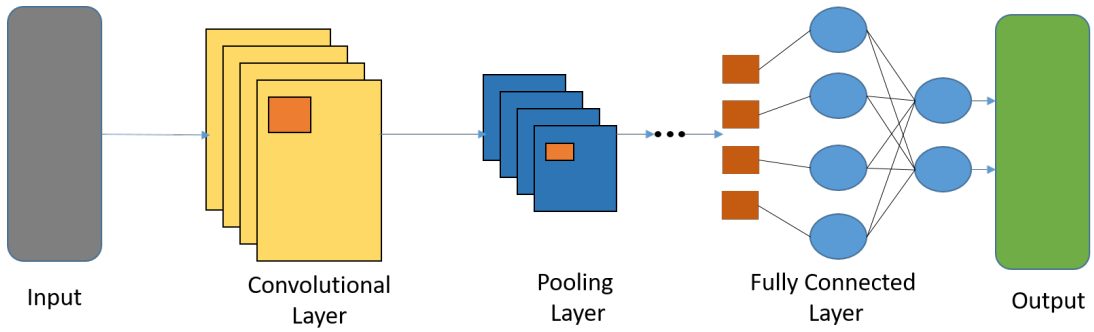


Figure 2.3: Example of a CNN

The convolutional layer consists of kernels that are capable of learning, which apply weights to a region of an image and compute the dot product. An example of a kernel and its convolution can be seen in Figure 2.4. The kernel will then stride over the entire image to form a new image, called a feature map. The output shown in Figure 2.4 is a single pixel of this feature map. These output maps will then go through a linear or non-linear function, the most common in deep learning being the rectified linear unit function (ReLU). The number of feature maps that are created, the stride length of the kernel, and the size of the kernel can all be considered hyperparameters that are tunable and have an impact on the output.

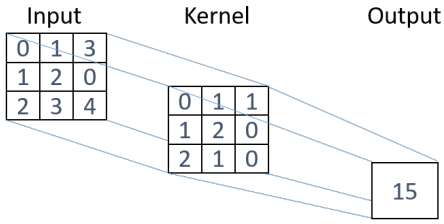


Figure 2.4: Kernel Convolution

Next, is the pooling layer, which consists of a kernel applying a function to a region of the feature map. The most commonly used functions are max pooling and average pooling. These functions will downsize a feature map. Max pooling will output the max of the kernel's input. An example of max pooling is shown in Figure 2.5. In contrast, average pooling outputs the mean of the kernel's input.

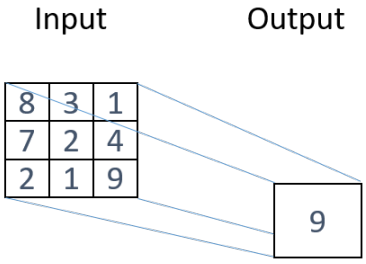


Figure 2.5: Max Pooling

Finally, there is the fully connected layer, which will typically flatten the output of the pooling layer and then implement a dense layer. This dense layer will apply some nonlinear function to the weights of all the neurons and try to output some sort of prediction. These predictions can be probabilities of classes in a classification problem, or, like in the work of this paper, can be an

(x, y) coordinate.

2.3.1.1 Activation Functions

One of the more important aspects of building a CNN is choosing an activation function. There are three main functions that currently are popular in building a CNN, though there are more being researched constantly. These three functions are the Rectified Linear Unit (ReLU), Sigmoid, and Hyperbolic tangent (tanh).

ReLU has become popular in the last decade, when AlexNet used it for their ImageNet classification. [7] states that CNNs that use ReLU for their activation function train several times faster than ones that use Sigmoid or tanh. ReLU relies on positive inputs, where it's output for all negative inputs is zero. For positive inputs, the output is equal to the input. The ReLU function can be described by Equation 2.5 and its graph can be shown in Figure 2.6

$$f(x) = max(0, x) \tag{2.5}$$

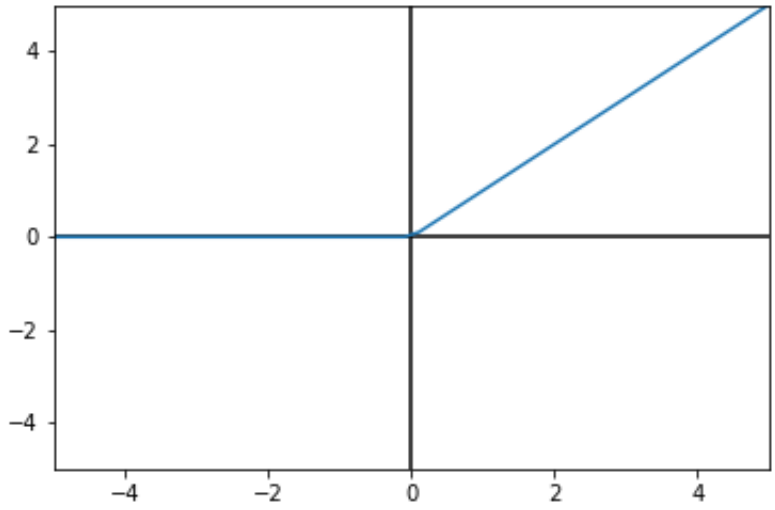
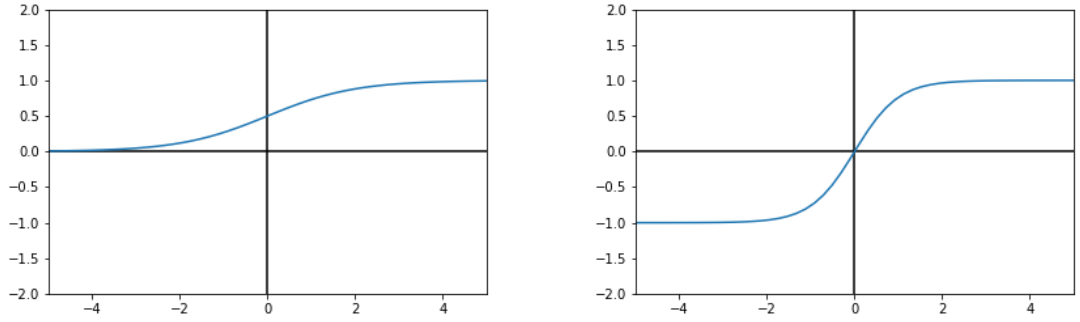


Figure 2.6: Rectified Linear Unit (ReLU)

The other two activation functions belong to a class of saturated functions. This means that if their input is either too high or too low, they will be snapped to a certain value. The sigmoid function will snap low inputs to zero and high inputs to one. While the tanh function will snap low



(a) Sigmoid Function

(b) Hyperbolic Tangent Function

Figure 2.7: Saturation Functions

inputs to negative 1 and high inputs to one. The equations of sigmoid and hyperbolic tangent can be seen in Equations 2.6 and 2.7. Their respective graphs can be seen in Figure 2.7.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

$$f(x) = tanh(x) \tag{2.7}$$

2.3.2 Generative Adversarial Network

Generative Networks are learning models that aim to augment data. This can include image translation, predicting future video frames, or addressing the lack of labeled data in a deep learning environment. Generative Adversarial Networks (GANs) are generative models, first developed by Goodfellow et. al. [2] that contain two networks that are considered adversaries of each other in order to generate data. The two networks are called the Generator, G, and the Discriminator, D. The goal of G is to take a latent variable, z, and produce fake samples that look like the real data, while the goal of D is to determine whether its input came from the real data space or the generator, signaling high values as real and low values as fake. GANs are a form of semi-supervised learning, where some labeled data is provided to help the discriminator understand what real data looks like by labeling data with a "real" tag. G and D then compete with each other to be successful in their tasks and their learning situation can be modeled with Equation 2.8, where p_{data} and p_z define the probability distribution of the real data space and the latent space respectively, x is an element of

the real data space, and $V(G, D)$ is a binary cross-entropy function [5].

$$\min_G \max_D V(G, D) = \min_G \max_D \mathbb{E}_{x \sim p_{data}}[logD(x)] + \mathbb{E}_{z \sim p_z}[log(1 - D(G(z))] \tag{2.8}$$

When looking at this equation from the point of view of D, which is trying to be maximized, the first term makes sense. A higher value given the real data will maximize the first term, since the logarithmic function is strictly increasing. At the same time, given a generated sample, a low value will maximize D in the second term, since the increasing log function is being subtracted. However, from the point of view of G, the second term will be minimized when G successfully deceives D and makes D output a large value given a generated sample. Thus, G and D have a minimax relationship and compete against each other. The goal of this model is to always have D output a 50% probability that a sample is real, thus meaning the discriminator cannot distinguish between real and fake, and having G generate data such that $p_g = p_{data}$, in which the probability distribution of the generated samples is equivalent to the probability distribution of the real data. Figure 2.8 shows the flow of a typical GAN.

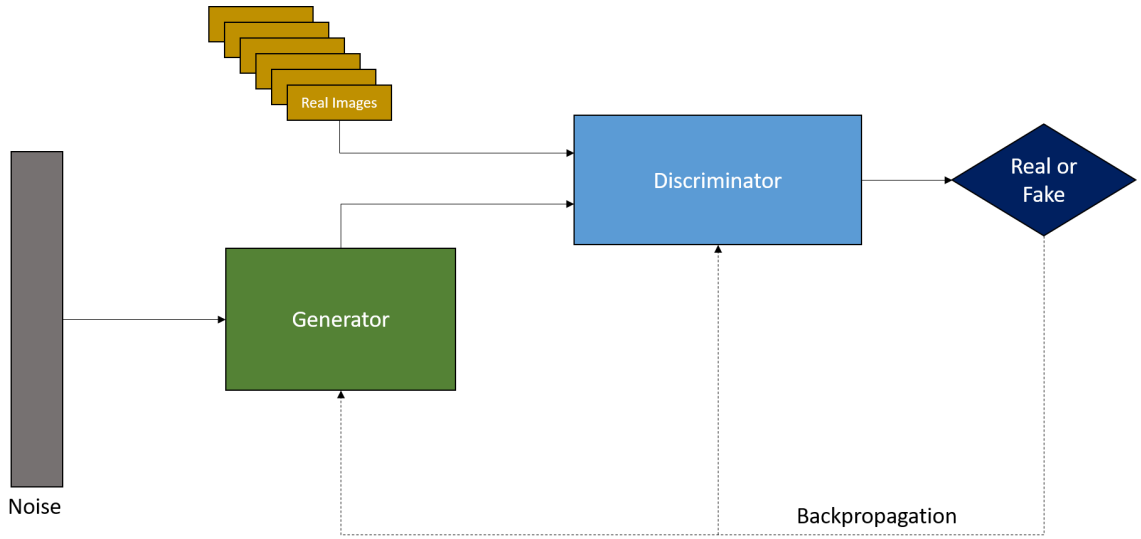


Figure 2.8: Generative Adversarial Network Process

There are several different types of GAN but one of the most popular is the Deep Convolutional Generative Adversarial Network (DCGAN). Up until Radford et. al. [10], there had been attempts to use CNNs in a GAN, all of which had been unsuccessful. When the DCGAN approach came out, there were noticeable changes to the typical CNN architecture. First, pooling functions

were replaced in both the generator and discriminator for strided convolutions. Second, they used a Batch Normalization layer, which standardizes inputs to have zero mean and a standard deviation of one. Third, they used the ReLU function in the generator and a Leaky Rectified Linear Unit Function (LeakyReLU) for the discriminator. The LeakyReLU function is similar to the ReLU function, except that it changes the negative inputs to have an output of αx, where typically $\alpha = 0.1$, instead of zero like in the ReLU function. The LeakyReLU function can be seen in Equation 2.9.

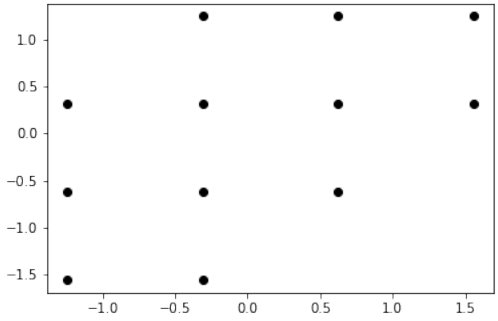
$$f(x) = max(\alpha x, x) \tag{2.9}$$

There are several issues regarding GANs that makes generating useful data difficult. One of the biggest challenges of GAN is called mode collapse. Mode collapse is a type of failure in GANs, in which a GAN will fail to generate different types (modes) of data that it is learning on. This lack of diversity can cause the GAN to be rendered useless, since in many applications, including localization, diversity of data samples is required. Figure 2.9 shows an example of a mode collapse problem. The overall goal of the generator is to learn to generate data across the whole data space, as shown in Figure 2.9a, however, the generator will try to learn to produce a single mode, as shown in Figure 2.9b, thinking that it can fool the discriminator into thinking it is looking at real samples. Once the discriminator catches on that the single mode generated samples are fake, the generator will switch to a different mode as shown in Figure 2.9c, and so on as seen in Figure 2.9d and Figure 2.9e. Overall, this failure in training results in a single mode being generated, no matter the input to the generator.

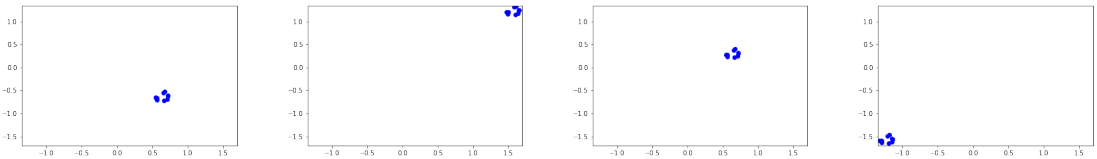
2.4 Cross Technology Interference

Cross technology interference (CTI) is a phenomenon when two or more different communication standards interfere with each other and their ability to communicate as thoroughly as they should be able to. It is becoming an issue in communications because as time goes on, more things become automated and need different standards. This holds especially true in the 2.4 GHz ISM (Industrial, Scientific, and Medical) band, since it is unlicensed and has the biggest mix of networks.

Two widely used (and growing) networks in this band are the 802.11 WiFi network and the 802.15.4 ZigBee network. WiFi is used in almost all commercial buildings in the United States and ZigBee is being used for low power wireless sensor networks, creating the perfect opportunity for



(a) Target Generation



(b) Step0 (c) Step1 (d) Step2 (e) Step3

Figure 2.9: A mode collapse problem

interference to occur. Many studies focus on enabling ZigBee in a harmful WiFi environment, since it is a much lower power network and is more vulnerable to packet loss and complete failure than WiFi is. However, ZigBee also degrades WiFi networks' performance and throughput.

2.4.1 Experiments

In [9], Pollin et. al. studies the impact of a heavily loaded 802.15.4 ZigBee network on an 802.11b network. They state that previously, the ZigBee network was thought to have little impact on a WiFi network because it has such a low power comparatively, and both standards operate with a listen before sending protocol, in which they will both analyze the channel they are about to send on, and if it is busy, they will wait to send their data. What these researchers find out is that both of these claims are false. First, they find out that 802.15.4 can decrease throughput in 802.11b by up to 80%. Second, they note that collisions occur at the beginning of 802.11b packets because the 802.15.4 network is too slow to recognize the start of an 802.11b transmission. Thus meaning, while ZigBee does not transmit (and interfere) with WiFi in the middle of a packet it can still interfere with the beginning of the packet, causing total packet loss.

In [13], Zhau et. al. evaluates the impact of 802.11n WiFi on 802.15.4 ZigBee and vice versa.

They state that with the advancement of 802.11 from legacy standards b/g to the new standard n, there are improvements in dealing with interference. First, the addition of Multiple Input Multiple Output (MIMO) allows for a much higher 802.11 throughput due to the ability of transmitters and receivers to transmit and receive multiple packets at once through multiple antenna. Second, the addition of frame aggregation (FA) allows transmitters to transmit multiple frames in a single transmission opportunity, thus also increasing the throughput. However, even with these additions there are still scenarios when 802.15.4 has a significant impact on 802.11 throughput. These scenarios are called symmetric scenarios and occur when the distance between a ZigBee transmitter and a WiFi transmitter are close enough that the ZigBee transmit power crosses the threshold for a WiFi transmitter to "hear" it and recognize that the channel is busy. In these scenarios, Zhao et. al. found that throughput can decrease by up to 25%, mainly due to backoff algorithms.

Chapter 3

Methodology

In this section, the design scheme for the work is discussed as well as the processes that are involved in each task.

3.1 Scheme Design

For this project there are three main steps involved. First, pre-processing is done on the data to make sure it is in the correct format. In this step, pre-processing techniques like normalization, or standardization will be applied to the data, or there may be a decision to leave the data in its raw form.

Second, a CNN will be built to learn patterns in the data. I chose a CNN because the CSI data is in the form of a 3D matrix, thus meaning it looks like an image. CNNs are designed to work well with image data.

Last, a GAN will be constructed. We need the GAN for this project to augment the data set. We can only physically collect data at so many discrete locations, so it is necessary to have the computer to generate more data at locations that we did not gather data from.

3.2 Data Processing

After data collection, pre-processing is necessary to extract the data in a useable format. Pre-processing was done in both Matlab 2021a and Python 3.7. First, the data had to be converted

from the ".dat" files that were given after data collection to files that could be more easily read. For this, a function provided by the 802.11 CSI Tool [3], specifically to read binary files was used since that is the format the data was in. Because all of the CSI data was in a complex form, in the model of $H = a + bi$, where a refers to the real part of the data and b refers to the imaginary part of the data, it needed to be separated into its amplitude and phase parts. In order to do this, I took the magnitude and angle of the data, described in Equation 3.1 and Equation 3.2, respectively.

$$|H| = \sqrt{a^2 + b^2} \tag{3.1}$$

$$\angle H = arctan(\frac{b}{a}) \tag{3.2}$$

After doing this, the pre-processing methods previously discussed were applied, and the data slightly rearranged. First, I gathered all the data from a single subcarrier on a single receive antenna from all the packets in a single transmission into three different arrays that correspond to the three receive antennas. Then, standardization or normalization techniques were applied to the data here if the experiment calls for pre-processing. That data will then be stored into an array. I will continue to do that for all 30 subcarriers and all 5 experiments of the same location. Next, all the data from all the experiments will be concatenated into an array that has shape $(30, 3, T_p)$, where T_p is the total number of packets across all five experiments. This will be done for all three transmitters and the 12 ZigBee locations.

From here, I switch to Python, and load in the three files corresponding to a single ZigBee location. The three files will be concatenated, forming a $(3, 30, 3, T_p)$, where the first 3 is the transmitter number, the 30 is the subcarrier number, and the last 3 is the antenna number. Finally, I will create T_p files that contain a $(3, 30, 3)$ matrix for each location as the correct input shape for the CNN.

After this, the data will be shuffled and split into three different types: Training, Testing, and Validation. The training set will make up 60% of the total data, and the Testing and Validation data sets will each make up 20% of the total data. The training data and validation data will be given to the CNN during training and the testing set will be used for evaluation.

3.3 Building the CNN

Building the CNN had different stages to it. First, I wanted to make a CNN that solved a classification problem. The problem consisted of 12 ZigBee locations shown in Figure 3.1, and the collected CSI was input to the CNN, which had to classify which interference location the CSI data belonged to. To do this, I implemented a slightly modified AlexNet [7] to fit my data size because it is a common architecture for a CNN. The architecture can be seen in Figure 3.2.

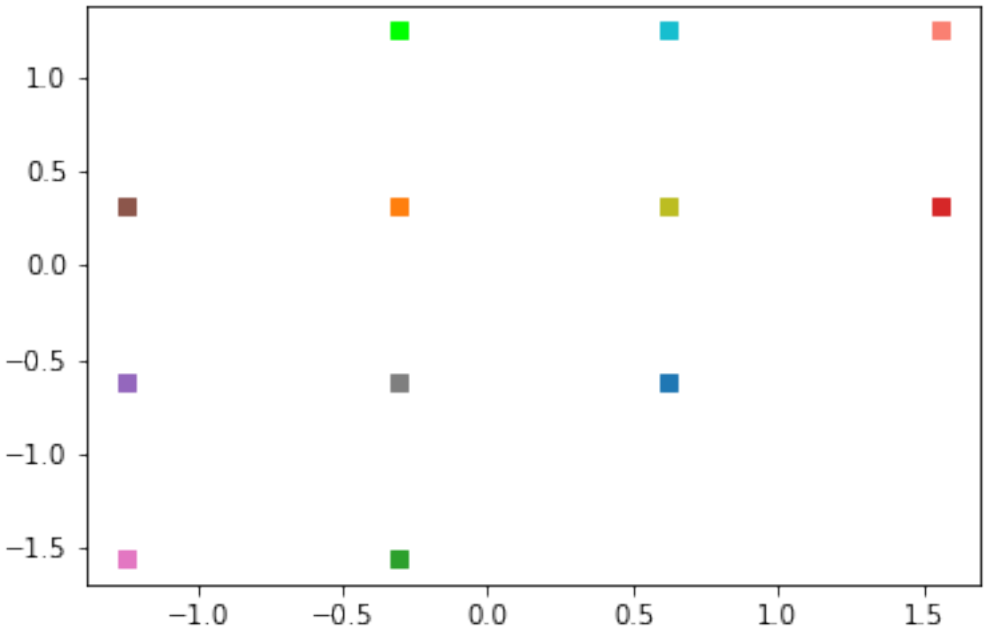


Figure 3.1: ZigBee Locations

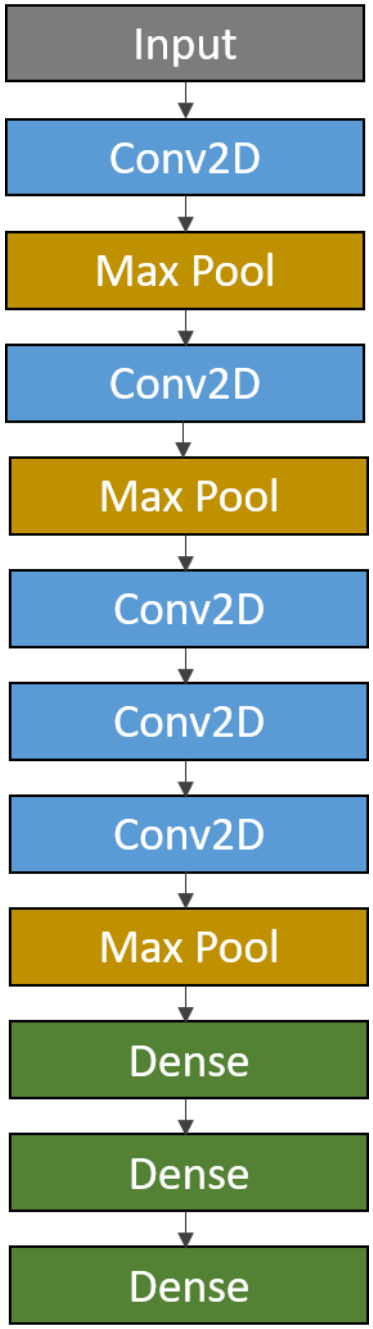


Figure 3.2: AlexNet Architecture

Once I saw that the predicted outputs in the classification problem had an accuracy over 95%, AlexNet was modified further to form a regression problem in the form of localization. In this scenario, the CNN would output an (x, y) coordinate of where it thought the interference was

coming from. The Euclidean distance was then calculated, shown in Equation 3.3, between the CNN's output and the actual location that the ZigBee interference was coming from, where a is the actual ZigBee (x, y) coordinate and p is the predicted ZigBee (x, y) coordinate. The updated architecture that I used can be shown in Table 3.1, where the parameters column signifies filters, defined by f in the convolutional layers, units, defined by u in the dense layers, rate, defined by r in the dropout column. This architecture was originally tuned for the data that was standardized during pre-processing.

$$d(a, p) = \sqrt{\sum_{i=1}^{2}(a_i - p_i)^2} \tag{3.3}$$

Layer Number	Layer Type	Parameters	Output Shape
0	Input		$(3, 30, 3)$
1	Conv2D	$f = 16$	$(3, 30, 16)$
2	Conv2D	$f = 32$	$(3, 30, 32)$
3	Batch Normalization		$(3, 30, 32)$
4	Conv2D	$f = 64$	$(3, 30, 64)$
5	Conv2D	$f = 128$	$(3, 30, 128)$
6	Batch Normalization		$(3, 30, 128)$
7	Conv2D	$f = 256$	$(3, 30, 256)$
8	Batch Normalization		$(3, 30, 256)$
9	Max Pooling		$(1, 30, 256)$
10	Flatten		$(1, 7680)$
11	Dense	$u = 2496$	$(1, 2496)$
12	Dropout	$r = 0.2$	$(1, 2496)$
13	Dense	$u = 1024$	$(1, 1024)$
14	Dropout	$r = 0.2$	$(1, 1024)$
15	Dense	$u = 2$	$(1, 2)$

Table 3.1: My CNN Architecture

3.4 Building the GAN

To build the GAN, I decided to use the approach laid out in [10], in which there are no pooling layers, and the only layers that were used were the allowed layers aforementioned in *Background*. The architecture used for the generator can be seen in Table 3.2, and the architecture for the discriminator can be seen in Table 3.3.

Layer Number	Layer Type	Parameters	Output Shape
0	Input	0	$(100, 1)$
1	Dense	$u = 2560$	$(1, 2560)$
2	Batch Normalization		$(1, 2560)$
3	ReLU		$(1, 2560)$
4	Reshape		$(1, 10, 256)$
5	Conv2DTranspose	$f = 256$	$(1, 10, 256)$
6	ReLU		$(1, 10, 256)$
7	Conv2DTranspose	$f = 128$	$(1, 10, 128)$
8	ReLU		$(1, 10, 128)$
9	Conv2DTranspose	$f = 64$	$(3, 30, 64)$
10	ReLU		$(3, 30, 64)$
11	Conv2DTranspose	$f = 3$	$(3, 30, 3)$
12	Tanh		$(3, 30, 3)$

Table 3.2: My GAN Generator Architecture

Layer Number	Layer Type	Parameters	Output Shape
0	Input		$(3, 30, 3)$
1	Conv2D	$f = 32$	$(3, 30, 32)$
2	Batch Normalization		$(3, 30, 32)$
3	LeakyReLU		$(3, 30, 32)$
4	Conv2D	$f = 64$	$(3, 30, 64)$
5	LeakyReLU		$(3, 30, 64)$
6	Conv2D	$f = 128$	$(1, 10, 128)$
7	LeakyReLU		$(1, 10, 128)$
8	Flatten		$(1, 1280)$
9	LeakyReLU		$(1, 1280)$

Table 3.3: My GAN Discriminator Architecture

Chapter 4

Results

In this chapter, the evaluation of the work will be discussed. During the evaluation, the CNN was discussed as well as the GAN.

4.1 Experimentation setup

Data was collected in a small 3m by 3m outdoor space in which there were no ambient WiFi sources present by two of my colleagues. It was collected outdoors so that it is clear there are no RF sources providing interference to the system. It is fine to collect the data outdoors when in application it will be indoors because the WiFi protocol used OFDM to cancel out any multipath interference. The WiFi devices that were used were two laptops running on an Ubuntu 14.04 operating system. The Intel Ultimate N Wifi Link 5300 Network Interface card was used, since it allows collection of CSI. The ZigBee device that is used is SimpleLink™ crystal-less BAW CC2652RB multiprotocol wireless MCU. The software used to control the ZigBee board is the SMARTRFTM-STUDIO designed by Texas Instruments.

The devices were laid out in the manner shown in Figure 4.1, with the blue triangle being the WiFi receiver, the green triangles being the WiFi transmitter, and the red circles being the locations in which the ZigBee device interfered with the transmitted signals. In order to make the ZigBee devices centered more evenly around the origin of the room $(x, y) = (0, 0)$, the ZigBee interference locations were standardized to have a mean of zero and a standard deviation of one. Table 4.1 shows the coordinates of each of the ZigBee devices before and after standardization. Table 4.2 shows the

coordinates of the WiFi receiver and WiFi transmitters.

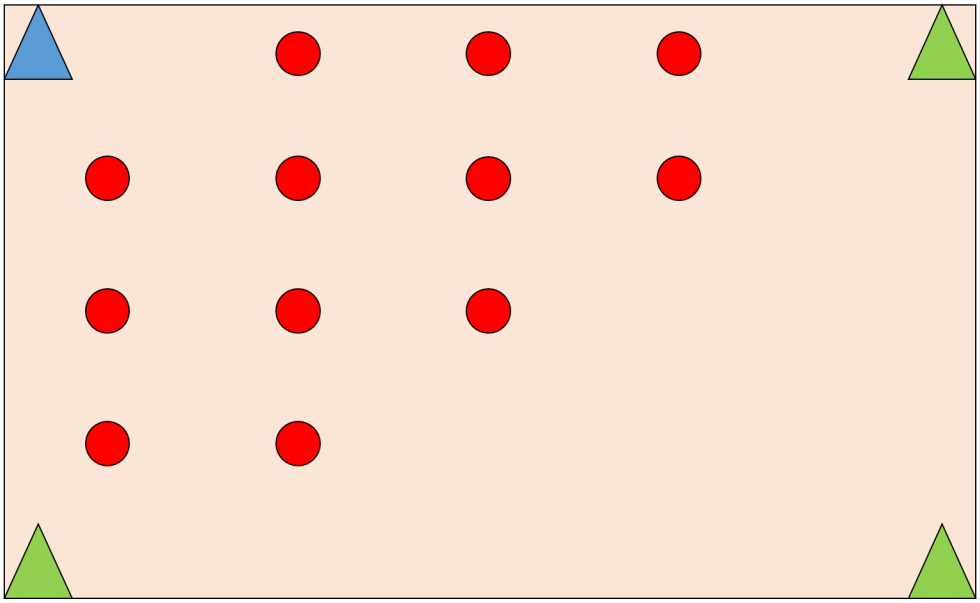


Figure 4.1: Layout of Data Collection Procedure

ZigBee Device Location Number	**Original Location** (x, y)	**Standardized Location** (x, y)
1	$(0, 0)$	$(0.6213, -0.6213)$
2	$(-0.75, 0.75)$	$(-0.3106, 0.3106)$
3	$(-0.75, -0.75)$	$(-0.3106, -1.5532)$
4	$(0.75, 0.75)$	$(1.5532, 0.3106)$
5	$(-1.5, 0)$	$(-1.2425, -0.6213)$
6	$(-1.5, 0.75)$	$(-1.2425, 0.3106)$
7	$(-1.5, -0.75)$	$(-1.2425, -1.5532)$
8	$(-0.75, 0)$	$(-0.3106, -0.6213)$
9	$(0, 0.75)$	$(0.6213, 0.3106)$
10	$(0, 1.5)$	$(0.6213, 1.2425)$
11	$(-0.75, 1.5)$	$(-0.3106, 1.2425)$
12	$(0.75, 1.5)$	$(1.5532, 1.2425)$

Table 4.1: ZigBee Interference Locations

WiFi Device	**Device Location** (x, y)
WiFi Receiver	$(-1.5, 1.5)$
WiFi Transmitter 1	$(-1.5, -1.5)$
WiFi Transmitter 2	$(1.5, -1.5)$
WiFi Transmitter 3	$(1.5, 1.5)$

Table 4.2: WiFi Device Locations

4.2 Convolutional Neural Network

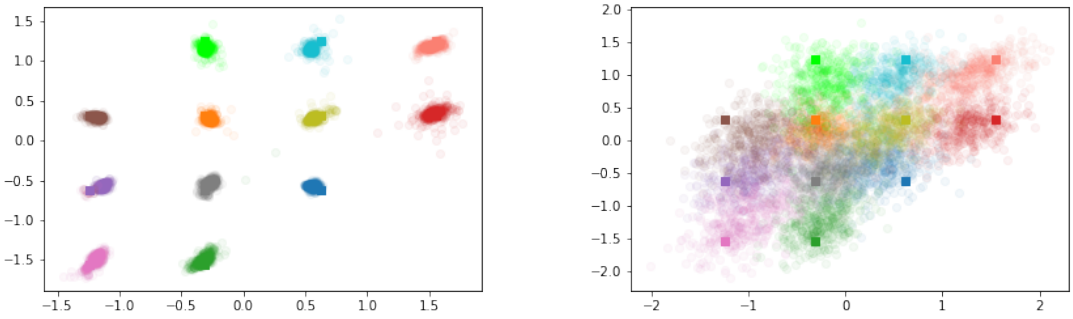
In this section the CNN will be evaluated. The evaluation metrics that are being used are time of convergence, time to train, and Euclidean distance. The hyperparameters that are being changed are batch size, loss functions, activation functions, and pre-processing techniques.

4.2.1 Pre-Processing

Before looking at different pre-processing techniques it is important to see how the CNN will perform given raw data. As can be seen in Figure 4.2, which shows a qualitative view of how the CNN performs given amplitudes and phase with no pre-processing done, the CNN performs quite badly when given phase as an input and performs quite well given amplitude. Because the CNN performed poorly given phase as input, it was omitted in the pre-processing stage. Normalization and standardization techniques were applied to the raw amplitude, and Figure 4.3 shows how the CNN performs when given data that had these two techniques applied to it. From this it can be concluded that the normalization technique performs quite well, similarly to the raw amplitude, and the standardization technique performs significantly worse.

One reason the CNN performs poorly given the standardized data could be due to the activation function used. The ReLU function discards all data that is below zero, and replaces it with zero. Because the standardization technique creates data that has a normal distribution with mean zero, half of the data in this scenario will be "thrown away". This then gives the CNN much less data to learn from. Normalized and raw data do not have datapoints that are less than zero, thus ensuring that the applied ReLU function will not throw any data away.

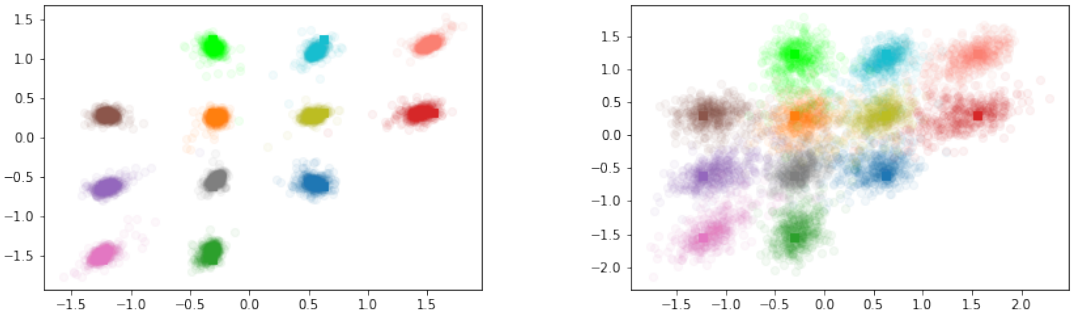
Another reason the CNN performs poorly given standardized data could be due to the



(a) Raw Amplitude

(b) Raw Phase

Figure 4.2: CNN Predicted Locations for Raw Data



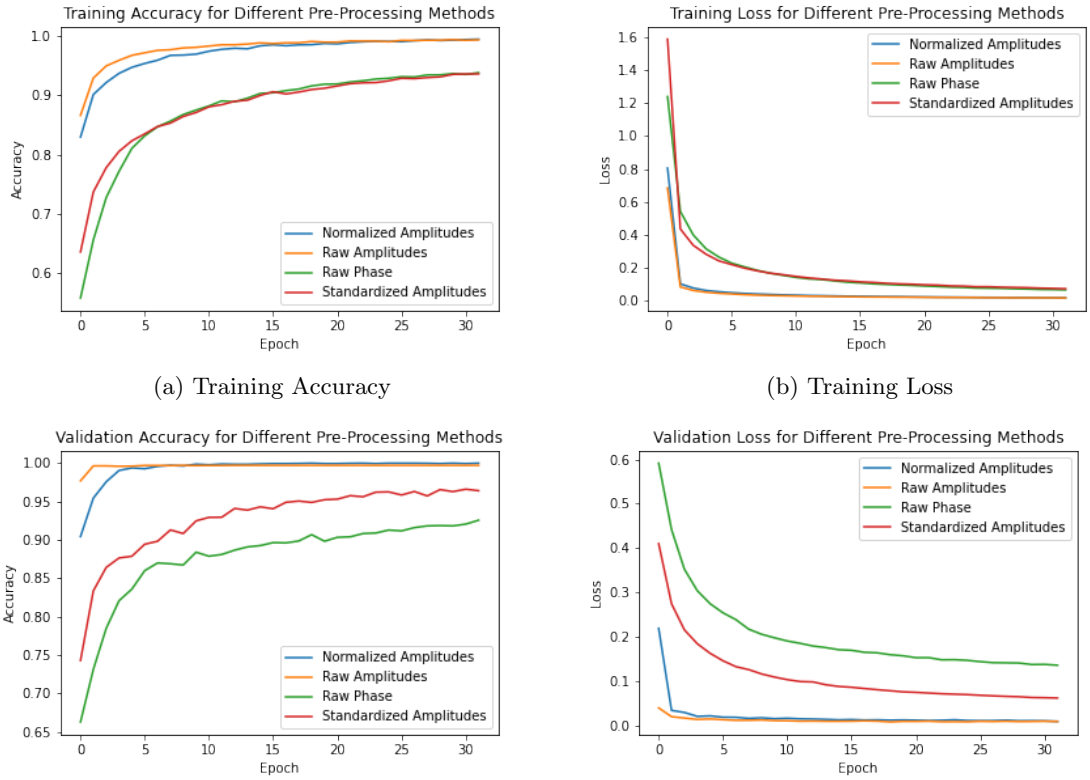
(a) Normalized Amplitude

(b) Standardized Amplitude

Figure 4.3: CNN Predicted Locations for Pre-Processed Data

fact that the distribution of a wireless channel looks more Rician than Gaussian. While these curves look relatively similar on a graph, a big difference is that Rician curves are asymmetric and Gaussian curves are symmetric, meaning half the data lies on the left and right of its peak. So standardization, which makes the data look Gaussian, applies a significant nonlinear shift in the data, causing its distribution to change completely. Though these figures show qualitative ideas of how the CNN performs, it is also important to look at more quantitative data shown in Figure 4.4 and in Table 4.3.

From the figure and the table, it is clear to see that raw amplitudes and normalization get the best performance out of the CNN. The reason they are so similar is because normalization is just a scaled down version of the raw data. Both of these techniques reach close to 99% training accuracy within the 32 epochs it was trained in, while using phase and a standardization technique



(a) Training Accuracy

(b) Training Loss

(c) Validation Accuracy

(d) Validation Loss

Figure 4.4: Pre-Processing Training Metrics

has significantly worse performance and a slower convergence time. Because the performance of the CNN is similar when given raw amplitudes and normalized amplitudes, and raw amplitudes seem to result in a slightly better performance, they will be used in the following evaluation of CNN hyperparameters.

4.2.2 Activation Functions

As talked about in *Methods*, there are three main activation functions that can be used in a CNN, ReLU, sigmoid, and tanh. The training metrics of these three functions can be seen in Figure 4.5.

From these graphs, it can be concluded that what AlexNet [7] proposed about ReLU being quicker to train than the original saturation functions apply to this data. ReLU converges quicker than the other two graphs, with sigmoid converging slower than tanh. The testing distances of the CNNs predicted output is comparable among the three activation functions, however it is shown that

Pre-Processing Technique	**Distance (cm)**
Raw Amplitude	9.7
Raw Phase	45.3
Normalized Amplitude	11.2
Standardized Amplitude	27.3

Table 4.3: Distances based on Pre-Processing Techniques

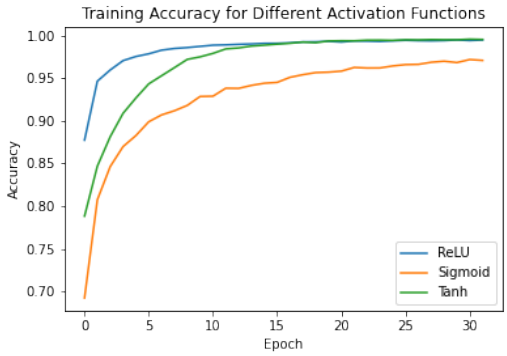
a CNN with ReLU outperforms a CNN with the other two activation functions after convergence slightly. This can be seen in Table 4.4.

Activation Function	Distance (cm)
ReLU	10.6
Sigmoid	11.4
Tanh	12.9

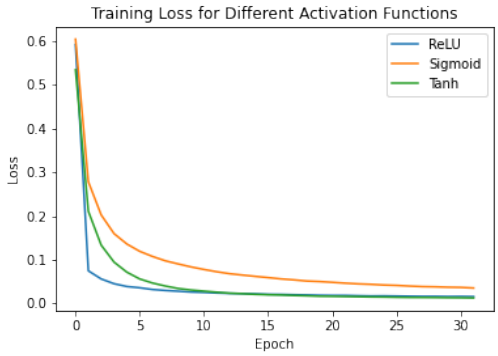
Table 4.4: Distances based on Activation Function

4.2.3 Loss Functions

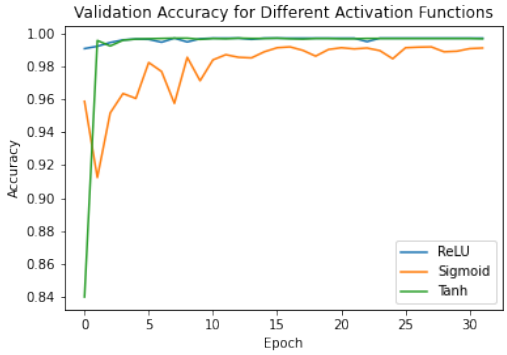
Choosing a loss function is an important part of training a model and there are two main loss functions that are used for this type of regression problem. These two loss functions are Mean Squared Error (MSE) and Mean Absolute Error (MAE). Figure 4.6 shows the different metrics of these two loss functions. First, it is noted that MSE performs slightly better than MAE. Typically, MSE will penalize large errors more than MAE, which could be the result of better performance since the channel is not necessarily stable and outliers could be a main predictor of where the ZigBee interference source is.



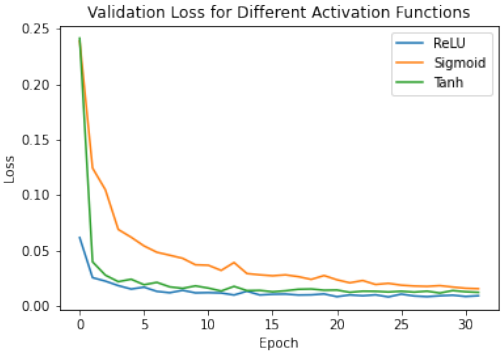
(a) Training Accuracy



(b) Training Loss

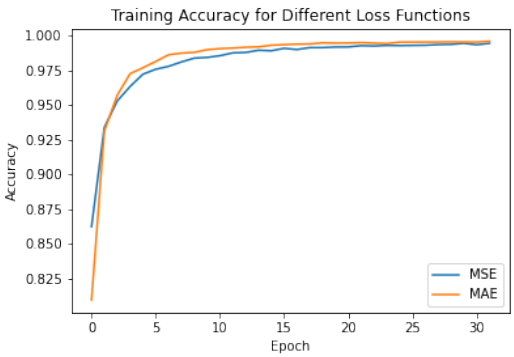


(c) Validation Accuracy

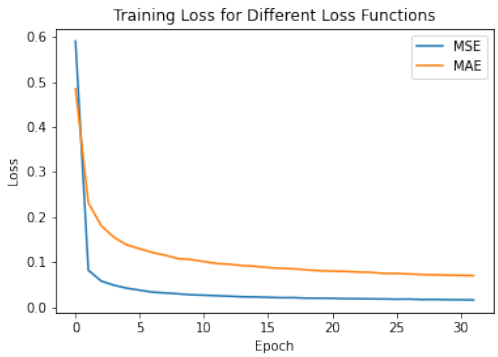


(d) Validation Loss

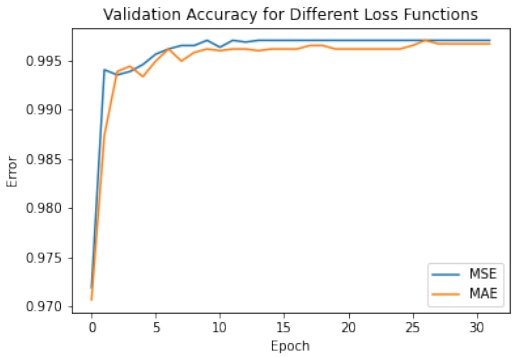
Figure 4.5: Activation Function Training Metrics



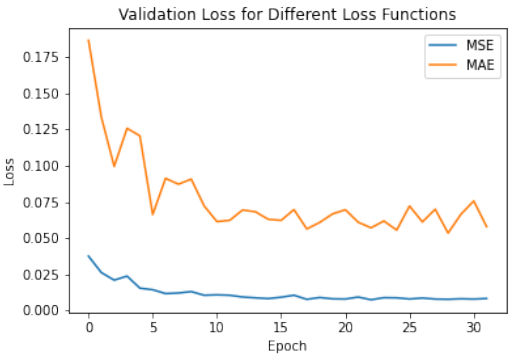
(a) Training Accuracy



(b) Training Loss



(c) Validation Accuracy

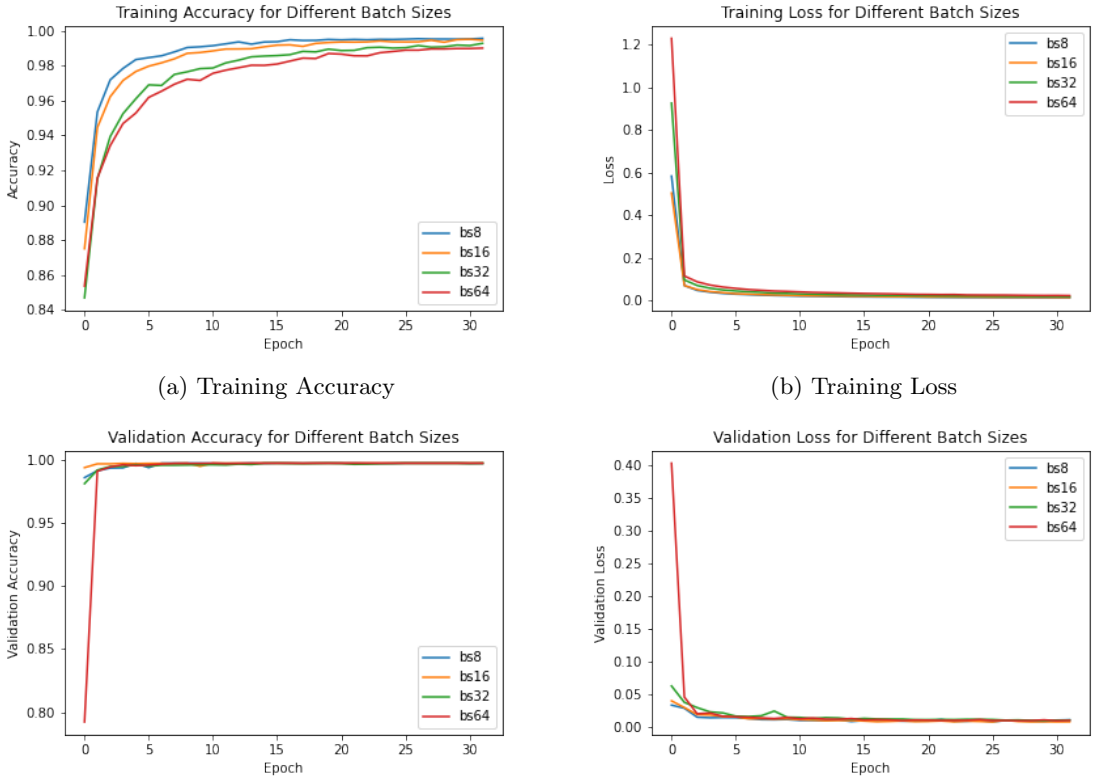


(d) Validation Loss

Figure 4.6: Loss Function Training Metrics

4.2.4 Batch Size

Batch size refers to how many images are being used for the CNN to learn before the neural network updates its parameters. The CNN will update its parameters n times where $n = T_s/BS$ and T_s is the size of the training data and BS is the size of the batch. Next, the number of epochs is how many times the CNN will filter through all of the training data. We are looking for how many epochs it takes for convergence on data of different batch sizes. The training accuracy can be shown in Fig 4.7. From these results, it is clear that a smaller batch size will converge quicker than a larger one. Smaller batch sizes also seem to have better accuracy, although the difference is negligible between a batch size of 8 and a batch size of 16. The distances for each batch size can be seen in Table 4.5.



(a) Training Accuracy

(b) Training Loss

(c) Validation Accuracy

(d) Validation Loss

Figure 4.7: Batch Size Training Metrics

Another important aspect of batch size is how quickly the CNN will actually finish training. Typically, a smaller batch size will lead to a longer training time because the CNN has to learn on

Batch Size	**Distance (cm)**
8	12.1
16	9.8
32	11.5
64	11.2

Table 4.5: Distances based on Batch Size

more batches per epoch. The data in Table 4.6 shows this result. The table confirms this result as a batch size of 8 takes the longest amount of time to train, while a batch size of 64 takes the shortest amount of time to train. While these numbers aren't super inconvenient, in the real-world there will be a lot more data to collect, and the training times will increase. A batch size of 8 has an 80% longer training time than a batch size of 16, a 190% longer training time than a batch size of 32, and a 309% longer training time than a batch size of 64. As the amount of data that is used to train the CNN is increased in a real world environment, the training times will result in a significant increase as well, and having the option for slightly worse performance and a significantly shorter training time becomes an important decision to make.

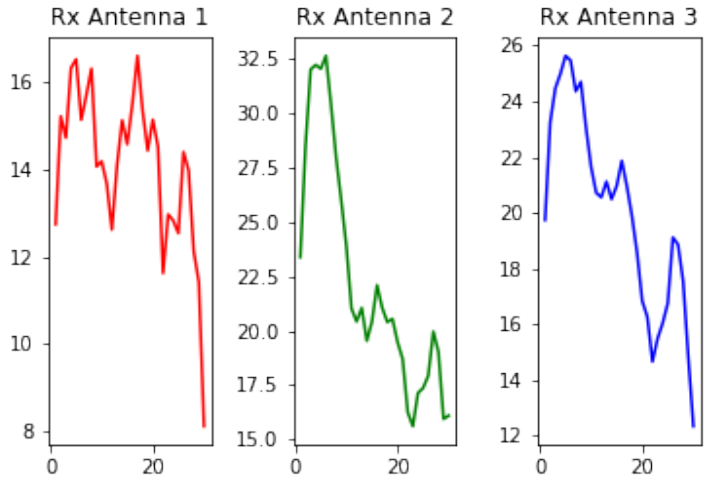
Batch Size	Time (s)
8	380
16	212
32	131
64	93

Table 4.6: CNN Training Time Based on Batch Size

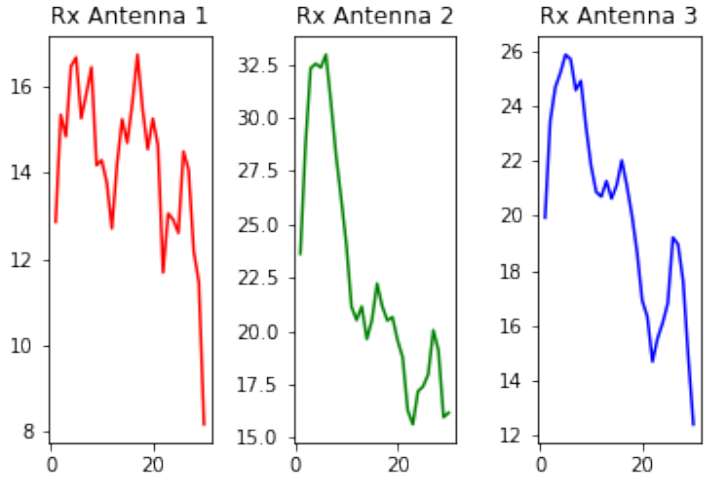
4.3 Exploratory Results

GAN is needed for this project to augment the data set and produce CSI that imitates what CSI would look like for locations not measured. However, this is difficult as generators have trouble with overfitting their discriminators. In Figure 4.8, a single CSI measurement is shown across thirty subcarriers and three receiver antennas. It is clear that the two generated samples, which have completely different noise vectors as the generators input, generate nearly identical samples. These generated samples cause the CNN to predict interference locations that are extremely close to each other. Figure 4.9a shows the CNNs prediction of one thousand different generated images. It is clear to see that these images are not *exactly* the same because there is *some* disparity between the samples. However, upon zooming out of the picture, as shown in Figure 4.9b, one can see that all of

the predictions are concentrated in a single area. This is a clear example of a mode collapse problem as previously mentioned.

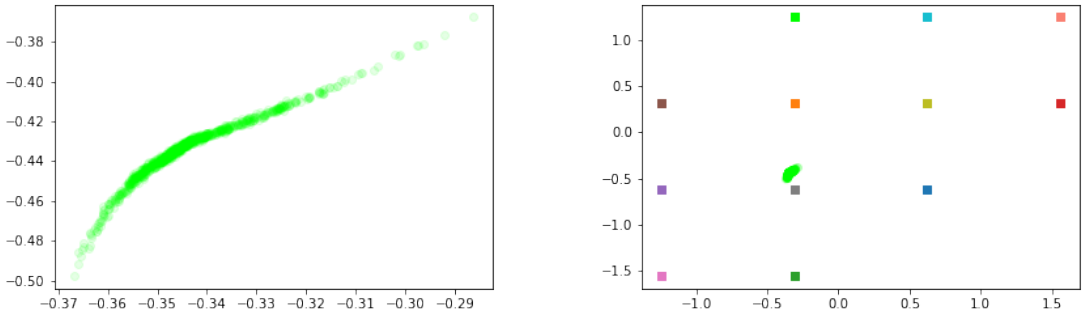


(a) First Generated Sample



(b) Second Generated Sample

Figure 4.8: Generated samples



(a) First Real Sample

(b) Second Real Sample

Figure 4.9: CNN Prediction of GAN

Chapter 5

Conclusions and Discussion

There is a promising future in wireless sensing, including all of the localization applications that come with it. This work explored utilizing Channel State Information to localize ZigBee interference sources. These interference sources can play a role in a slower WiFi throughput, harming the most prevalent wireless system on Earth. However, it is important to recognize that these ZigBee interference sources will play a big role in the future of the Internet of Things. Thus, research is needed to further the idea of coexistence between these two radio frequency sources.

5.1 Future Work

There will still need to be future work done in this project of localizing interference sources. The biggest issue that was faced, was the mode collapse problem in the GAN. It is important to be able to have a useful GAN because data will not be able to be collected in every part of a room. Another thing that could be worked on is NLOS systems in which interference sources can be localized even if they are not in the same room. These systems would be significantly more useful since they could be deployed in large buildings with rooms and walls. However, these systems would also be more difficult because the ZigBee devices would have less of an impact on the channel since their signals would attenuate more in a NLOS experiment. Finally, there could be approaches that not only localize ZigBee interference sources but many different wireless standards, especially up and coming standards such as LoRa, that transmit in a long range setting. Once everything works in simulation, it will need to be deployed in some way on a commodity off the shelf WiFi system

and evaluated.

Bibliography

[1] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[3] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Tool release: Gathering 802.11 n traces with channel state information. *ACM SIGCOMM computer communication review*, 41(1):53–53, 2011.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Yongjun Hong, Uiwon Hwang, Jaeyoon Yoo, and Sungroh Yoon. How generative adversarial networks and their variants work: An overview. *ACM Computing Surveys (CSUR)*, 52(1):1–43, 2019.

[6] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[8] Yongsen Ma, Gang Zhou, and Shuangquan Wang. Wifi sensing with channel state information: A survey. *ACM Computing Surveys (CSUR)*, 52(3):1–36, 2019.

[9] Sofie Pollin, Ian Tan, Bill Hodge, Carl Chun, and Ahmad Bahai. Harmful coexistence between 802.15. 4 and 802.11: A measurement-based study. In *2008 3rd International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CrownCom 2008)*, pages 1–6. IEEE, 2008.

[10] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[11] Maja Stella, Mladen Russo, and Dinko Begušić. Rf localization in indoor environment. *Radioengineering*, 21(2), 2012.

[12] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. A survey of indoor localization systems and technologies. *IEEE Communications Surveys & Tutorials*, 21(3):2568–2599, 2019.

[13] Zenghua Zhao, Xuanxuan Wu, Xin Zhang, Jing Zhao, and Xiang-Yang Li. Zigbee vs wifi: Understanding issues and measuring performances of their coexistence. In *2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, pages 1–8. IEEE, 2014.