Convolutional Neural Networks (CNN) for Digital Radio Frequency Memory (DRFM) Jammers

by

EG Friedel

A thesis submitted to The Johns Hopkins University in conformity with the requirements for the degree of Master of Science.

Baltimore, Maryland

May, 2023

Abstract

Radar electronic jammers are evolving from hostile nations thus becoming more complex and present serious issues when a radar system is trying to interrogate the actual targets of interest. Electronic jammers that present a serious challenge are in a class called Digital Radio Frequency Memory (DRFM). DRFM techniques work by generating coherent false targets to the radar receiver based on an intercepted pulse signal from the radar transmitter. This will position false targets either ahead or behind the actual radar target, thus masking the real target with false targets. The false targets can also be manipulated in amplitude, phase and frequency.

Traditional approaches to target detection and estimation for electronic jammers in general, rely on parametric modeling, that can fail because it violates the strict assumptions of classical signal processing algorithms. The result is substantial algorithm performance degradation. Furthermore parametric models to handle DRFM jammers are difficult to design and ineffective against an evolving DRFM technology. The key to identifying opportunities for improved electronic jammer protection and signal processing in radars is to use machine learning techniques to challenge the underlying assumptions of the standard parametric approach for the design and analysis of radar systems.

Convolutional Neural Networks (CNN) have gained popularity in the last few years with the advent of faster high performance computer systems which rely GPUs for the best computational performance. A CNN operates from a mathematical perspective and is used for non-trivial tasks such as image classification. CNN's have great performance while classifying images when they are very similar to the training dataset. However little work as been done in developing realistic radar models which are ignoring radar environmental and antenna effects thus providing inaccurate simulation training datasets and credibility. In addition, current public available research does not typically consider the five dimensions of a radar sensor, thus presenting an incomplete signal processing chain. From a first principles perspective the radar measures the following aspects of a signal target return: Range, azimuth, elevation, Doppler, and signal amplitude.

We propose to design a CNN that will use both temporal and spatial training datasets, where the radar signal processing with respect to DRFM jamming will be examined to identify and classify DRFM type jammers. The CNN will use raw uncompressed In-phase and Quadrature (IQ) time series data and range-doppler maps

ABSTRACT

as inputs to classify if our ground based phased array radar system is being jammed by DRFM false targets. A suitable network architecture and values for hyperparameters are found by iterative experimental studies. The final classification accuracy objective will be 95 percent or better. This achieved accuracy suggests that CNNs can be used to detect radar DRFM jamming with good results and thus allowing a radar system to decide on any further actions to mitigate a DRFM jammer attack.

Primary Reader and Advisor: Dr. David Shrug

Secondary Reader: Dr. Kurt Stein

List of Symbols

The following definitions of the symbols used throughout this thesis are listed here, some symbols such as θ have more than one meaning but should be clear from the context of its usage. Try and list these in some order that makes sense. List math operators first, vectors, then Greek letters, then alpha

$*$	Convolution operator
$H(f_1), H(f_2)$	Frequency response

List of Acronyms

CNN	Convolutional Neural Network
DRFM	Digital Radio Frequency Memory

Acknowledgments

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Dedication

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Overview

Ideally radar signal processing should be able to accurately and efficiently detect false or misleading targets of interest. The reality is that typical radar signal processing is ineffective against false targets produced by deceptiive electronic jammers. Traditional approaches to target detection and track estimation for electronic countermeasures (ECM) in general, rely on parametric modeling, that can fail because it violates the strict assumptions of classical signal processing algorithms. The result is substantial algorithm performance degradation. ECM in general attempts to interfere or deceive the radar system with misleading electronic signals. We propose to use machine learning techniques to challenge the underlying assumptions of the standard parametric approach for the design and analysis of radar systems. Convolutional Neural Networks (CNN) have gained popularity in the last few years with the advent of faster high performance Graphics Processing Unit (GPU) computers. Current research demonstrates CNN as a sound approach for radar signal classification with more work to be done. We will show a Convolutional Neural Network (CNN) that will use spatial training datasets in the form of range-doppler images to perform radar signal classification. The radar signal processing will then be examined to identify different ECM classes. The goal is to show that ECM can be mitigated and thus improve overall radar performance operation and target recognition. We focus on a particular type of active ECM called Digital Radio Frequency Memory (DRFM). We choose to focus on the DRFM jammer because it's the most problematic ECM for radars to date.

Success is defined by the identification of an ECM, in this case a DRFM jammer and its type via the CNN model. The new CNN model is proposed to be part of the radar's signal processing chain. The DRFM jammer will fail to effect radar detection and tracking operations in the radar. Identification is the first step in mitigating the effects of ECM.

The status quo is the failure to identify false targets or interference is what we face today since DRFM is problematic for typical signal processing. The failure effects are as follows:

- Promotes sub-optional signal processing operation
- Degrades detection performance

- Hides the real targets, deceiving the Tracker

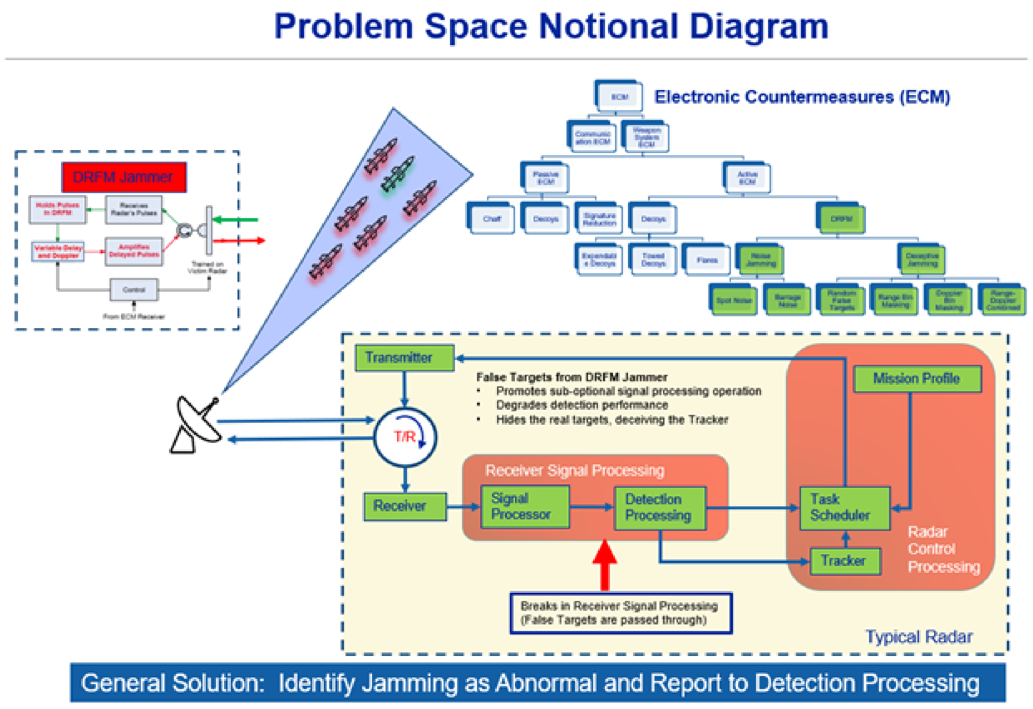


Figure 1.1: Radar Processing Chain with DRFM Jammer

This thesis proposes a novel CNN design exploring gradient descent approaches and cross-entropy loss functions in the application of image classification of radar targets. We put this in the categorically of machine learning. We cast this as an optimization problem where we are attempting to minimize (θ^*) over some loss function $L(\theta)$ to specifically solve an image classification problem.

1.2 Background

In recent years, the development of radar jamming technology is advancing and becoming more problematic for radar signal processing. Digital Radio Frequency Memory (DRFM) is one of the emerging technologies in the area of active deceptive jamming that is becoming problematic for existing radar capability to keep pace. [2].

An active radar jamming system using DRFM as the central subsystem in the generation and detection of specific signal characteristics was patented by Richard Wiegand in 1985. [3] DRFM works by intercepting the transmitting radar's pulse and then storing in memory its pulse characteristics, thus matching the transmitted pulse. The matching of the transmitted pulse is troublesome because typical signal processing uses a matched filter approach to separate the detection signal from the noise which is dependent on the transmitted pulse. The DRFM will then radiate a signal with a time or doppler delay that will present a false target to the receiving radar. The false target is directed towards the radar main-lobe which makes this a hard problem since this will get processed as a real target detection. We generally define this as deceptive jamming.

Deceptive jamming is a very active field of study in the radar community within the last decade. [4] Major efforts focus on the classification of the jamming signal which is where our interest lies. Various techniques have been studied such as the use of a likelihood generalized function to determine a false target produced by the jammer and a real target of interest. [5] [6] Likelihood methods are what is used in typical signal detection processing and rely on apriori information which needs to match the jammer. This approach falls short in the real world of jammer evolution. **What is the interesting thing here? What is the research and study? When does this start using CNN? What is your contribution?**

More recently, machine learning techniques have been proposed which have founded success in image recognition. [7] Machine learning was first conceived from the mathematical modeling of neural networks, which was published in 1943 by Walter Pitts and Warren McCulloch. Through the decades various applications and advancements have occurred. Most notably in 2006 with "deep learning" where Geoffrey Hinton developed algorithms that helped computers recognize different types of objects and text characters in pictures and videos. Also in 2010, ImageNet classification was conceived which dramatically improved the accuracy in machine learning and artificial intelligence image recognition. [8]In terms of radar jammer classification research since 2018 we note some of the relevant contributions which are as follows: NOT MY WORDS ... In [3], TDPAR, MKC and other characteristics which has low sensitivity to noise, low computational complexity, and is not easy to be disturbed are extracted, and then use decision tree algorithm to recognize jamming. In [4], a dense false target jamming recognition algorithm based on time-frequency atomic decomposition theory and support vector machine (SVM) is proposed. In [5], a feature fusion algorithm based on Bayesian decision theory is applied to the recognition of radar deception jamming. These algorithms are generally based on manual feature extraction, and rely more on manual judgment. They have strong ambiguity and subjectivity, leading to problems such as low accuracy and poor robustnessThis thesis will focus on image recognition. **Research for image recognition has what? What is the interest here? What is your contribution?**

1.3 Thesis Overview

The remainder of the thesis is as follows:

- Chapter 2 - A review of radar concepts that are essential to the test data generation and the problem space. We will consider a ground based phased array radar system and how we extract information from the radar signals which are then used to train our neural network.
- Chapter 3 - A review of machine learning concepts. An introduction to the theory and optimization techniques and how that applies to neural networks. We will focus specifically on Convolutional Networks (CNN) and justify how that applies to our problem space to then propose our algorithms and solution approach.
- Chapter 4 - An overview of the two simulation models, which are the Radar Model and the CNN Model will be presented in which we detail the theory, mathematics, and provide an overall model overview in terms of the software and hardware used.

- Chapter 5 - The thesis results will be provided with an overview of the scenarios and experiments performed. The data analysis approach is defined and the results are discussed.
- Chapter 6 - The findings and conclusions to the thesis are presented with the proposed future work.

Chapter 2

Radar Concepts

2.1 Overview

This chapter provides some basic radar concepts that help to support this thesis. We will consider for this thesis a phased array radar system and describe its basic signal processing capabilities and techniques. We will also introduce electronic jammers and the various techniques which are fundamental to the groundwork of this thesis. Most of the content is based on years of work-related experience in this field.

The chapter content is mostly supported by the standard references such as Skolnik and Richards which are well known and established throughout the radar community.

2.2 Phased Array Antenna

When we think of radar its common to picture the antenna in our minds. The antenna selection is a unique and important part of any radar. There are various types of antennas used in radar application. This thesis will incorporate and model a ground-based phased array radar system. We focus on using the phased array antenna since this is an antenna type used for the Ballistic Missile Defense System (BMDS) [9] in defense of the United States against ballistic missile threats from hostile nations. **Mention something about going to a more digital radar opens up opportunities of digital processing below**

In analog beamforming, a single signal is fed to each antenna element in the array by passing through analog phase-shifters where the signal is amplified and directed to the desired receiver. The amplitude/phase variation is applied to the analog signal at transmit end where the signals from different antennas are added before the analog-to-digital (ADC) conversion. At present, analogue beamforming is the most cost-effective way to build a beamforming array but it can manage and generate only one signal beam.

In digital beamforming, the conversion of the radio frequency (RF) signal at each antenna element into two streams of binary baseband signals cos and sin, are used to recover both the amplitudes and phases of the signals received at each element of the array. The goal of this technology is the accurate translation of the analog signal into the digital realm. Matching receivers is a complex calibration process with each antenna having its own transceiver and data converters that generate multiple beams simultaneously from one array.

2.3 Radar Signals and Detections

Radar signals in the form of electromagnetic energy is both transmitted and received respectively by the transmitter and receiver devices of a radar. A radar typically transmits a waveform modeled by the complex function

$$\bar{s}(t) = A(t)e^{i2\pi f_0(t)+\theta(t)} \tag{2.1}$$

Where $A(t)$ is the amplitude over time, f_0 is the carrier frequency of the pulsed waveform, and $\theta(t)$ is the phase component from $[0, \pi]$. This is also considered to be the complex envelop of a linear frequency modulated (LFM) waveform. We apply a slight variation to $\bar{s}(t)$ when using a pulsed waveform since we have a pulse Doppler radar. Thus the transmitted waveform or commonly known as a radar beam can be represented by a rectangular envelop with a constant amplitude $A(t)$ where τ is the pulse duration.

$$A(t) = \begin{cases} 1 & : \ 0 \leq t \leq \tau \\ 0 & : \text{otherwise} \end{cases} \tag{2.2}$$

2.3.1 Basic Radar Measurements

Mathematically we start with a dataset that contains digital samples called Inphase and Quadrature (IQ) data. The IQ is the result of converting an analog signal from the radar received signals. We assume an N-samples data set $x[0], x[1], ...x[N-1]$. The IQ for each sample in the data set is a complex number defined by

$$e^{i2\pi ft} = cos(2\pi ft) + isin(2\pi ft) \tag{2.3}$$

where f is the frequency in Hz, $cos(2\pi ft)$ represents the magnitude of the vector and $i\sin(2\pi ft)$ represents the phase. We consider the IQ as a rotating vector in the complex plane. A radar is a five dimensional sensor and these dimensions are the origin of radar signal information that can be derived from the IQ data and the setup of the radar array. The five dimensions are as follows in no particular order: Doppler, range, azimuth, elevation, and amplitude.

This is a mess, need to organize the text below better

Doppler $F_D = \frac{2v}{c}F_t = \frac{2v}{\lambda_t}$ where v is the target velocity and λ_t is the transmitted wavelength

Range $R = \frac{cT_R}{2}$

Azimuth Azimuth is defined by the monopulse slope Δ/Σ where Δ represents the radar azimuth difference channels and Σ is the total summation of all the radar channels. Probably should derive all this for clarity.

Elevation Elevation is defined the monopulse slope Δ/Σ Same as above except the difference channels are in elevation.

Amplitude $A_{dB} = 10 * log_{10}\left[\frac{I^2+Q^2}{V_{FullScale}}\right]$

2.3.2 Radar Signal Processing

Standard radar signal processing starts with and uses a generalized matched filter receiver. The range-doppler maps are dependent on the matched filter output and serve as the primary inputs to our CNN. We will show how the matched filter transforms the output data. The figure below provides the primary components of a standard radar signal processing approach and where we will apply the CNN for image classification.

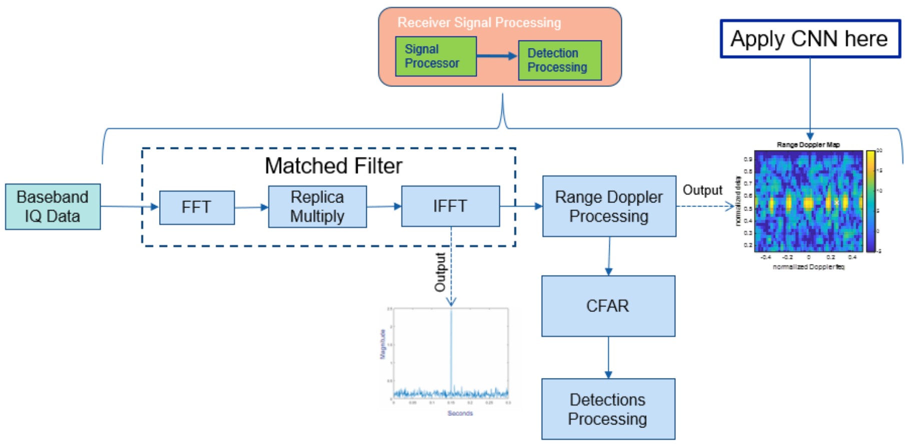


Figure 2.1: Standard Signal Processing Chain

2.3.3 Matched Filter

The matched filter is ubiquitous in radar signal processing and essential for detections processing, thus providing an essential signal processing step in the creation of the range-Doppler images. There is much documented about the matched filter, we cite Skolnik [10] as the reference which is preferred within the radar community. The matched filter attempts to optimize the signal-to-noise (SNR) of a detection in Gaussian noise. The replica or copy of the transmitted signal is used as a time-reversed and complex conjugate version of the transmitted signal to form the matched filter, which is then convolved with the received signal. We now provide the matched filter derivation to give mathematical reasoning to the range and frequency response of the received signal which makes up the range-Doppler images that are used as inputs to our CNN. We can express a matched filter in various forms as the Fourier transform of the finite received signal $s(t)$ where we define a measured signal with noise as $s_i(t) = s(t) + n_i(t)$. Where i depends on the number of samples. The noise $n_i(t)$ is defined as a random process where its mean and autocorrelation functions are time invariant, thus the function $n_i(t)$ is defined as wide-sense stationary (WSS). Now we consider an impulse response $h(t)$ with a Linear Time-Invariant (LTI) filter that takes our measure signal $s_i(t)$ and outputs an optimal filtered response $y(t) = s_0(t) + n_0(t)$.

We now develop the equations for the Fourier transform

$$n_0(t) = n_i(t) * h(t) \tag{2.4}$$

$$s_0(t) = s_i(t) * h(t) \tag{2.5}$$

The Fourier transform of Eq. (2.5) where t_0 is some instantaneous signal time and a function of frequency f.

$$S_0(f) = S(f)H(f)e^{i2\pi f t_0} \tag{2.6}$$

Integrating we get

$$s_0(t_0) = \int_{-\infty}^{\infty} S(f)H(f)e^{i2\pi f t_0}\, df \tag{2.7}$$

Using Parseval's theorem [11] the total energy of some signal E_x is the modulus squared of Eq. (2.7). So then at the output of the matched filter the total noise power using Pareseval's theorem is then defined by

$$N_0 = \int_{-\infty}^{\infty} N_i(f)|H(f)|^2\, df \tag{2.8}$$

Rewrite this later

$$SNR(t_0) = \frac{|\int_{-\infty}^{\infty} S(f)H(f)e^{i2\pi ft}\,df|^2}{\int_{-\infty}^{\infty} N_i(f)|H(f)|^2\,df} \tag{2.9}$$

$$= \frac{E_x}{\int_{-\infty}^{\infty} N_i(f)|H(f)|^2\,df} \tag{2.10}$$

The question to ask is what receiver frequency response $H(f)$ will maximize the SNR. We assert that the optimum SNR at the output of the receiver matched filter is expressed in Eq. (2.10). To prove this we can use the Cauchy-Schwarz inequality to establish that we have a maximized SNR dependent on the receiver frequency response $H(f)$. We generalize using X where $X_1(f) = H(f)\sqrt{N_i(f)}$ and $X_2(f) = \frac{X(f)e^{i2\pi ft}}{\sqrt{N_i(f)}}$

$$\left|\int X_1(f)X_2(f)\,df\right|^2 \leq \int |X_1(f)|^2\,df \int |X_2(f)|^2\,df \tag{2.11}$$

Equality is assured if and only if $X_2(f) = \alpha X_1^*(f)^1$, where $X_1^*(f) = S(f)e^{i2\pi ft}$ and α is some arbitrary constant. Rewriting in terms of Eq. (2.9) we get

$$SNR \leq \frac{|\int_{-\infty}^{\infty} X_1(f)X_2(f)e^{i2\pi ft}\,df|^2}{\int_{-\infty}^{\infty} |X_1(f)|^2\,df}, \tag{2.12}$$

[1]Starred transform which denotes a discrete time version of the Laplace transform

thus the SNR is maximized when

$$h(f) = \alpha s^*(T_i - t) \tag{2.13}$$

$$H(f) = \alpha S^*(f) e^{-i2\pi f T_i} \tag{2.14}$$

2.3.4 Detection Fundamentals

The detection itself is fairly straightforward, in which the power of the signal with noise is compared to some threshold value. A detection is positively identified if the signal power exceeds the threshold value. Thus the threshold value is critical to establishing detections. In general, the threshold value is a function of both the probability of false alarms and the probability of detections.

A primary function of any radar system is the function of generating and processing signal detections. The detection itself is binary, meaning it can be a real or false target of interest. A real target is the signal energy received by the radar receiver reflected from a real target of opportunity. A false target can be interference, random noise, or a jammer that is radiating noise or some deceptive signal towards the radar receiver. If its decided that we have a real target then further detection processing usually happens. A problem occurs when a false target can not determined and thus is considered a real target. The radar can be making up to a million of detection decisions per second. Since we are dealing with a large amount of data for detection decisions, a typical radar system targets can be best represented by statistical signal models. Consequently the process of deciding between a real or false target is a statistical hypothesis testing problem. This has limitations since it assumes that the interference level is known and constant. It also does not account for false targets that are deceptive, which thus appear as real targets when the statistical hypothesis testing is applied.

We propose a Range-Doppler processing step to help identify false targets. The Range-Doppler data is the input for our CNN that will be used to detect false targets to mitigate the failure effects as follows:

- Sub-optional signal processing operation
- Detection efficiency
- Tracker deception by hiding the real target(s)

2.3.4.1 Detection Hypothesis

The standard approach for target detection for any radar measurement depends on two hypothesis, in which one of the two hypothesis is assumed to be true. The radar measurement consists of either

1. The radar measurement contains only noise energy. No target is present.
2. The radar measurement is a combination of noise and target return energy. A target is present.

We regard this as a binary classification problem. Since we describe the signals statistically we consider the following probability density function (pdf) that describe the radar measurements to be tested under our two hypothesis (1.,2. from above).

$$p_x(\mathbf{x}|H_0) = \text{pdf of } \mathbf{x} \text{ given that the target was not present}$$

$$p_x(\mathbf{x}|H_1) = \text{pdf of } \mathbf{x} \text{ given that the target was present}$$

where $\mathbf{x}$ is a column vector of N samples based on some x_n data such that

$$\mathbf{x} \equiv [x_0, x_1, x_2, ...x_{N-1}]'$$

This gives N-dimensional joint pdfs $p_x(x|H_0)$ and $p_x(x|H_1)$, used to consider the following probabilities for use cases which are figures of merit when describing detection performance.

1. Probability of Detection (P_D): A target is present and detected.

2. Probability of False Alarm (P_{FA}): No target is present but a detection is identified.

3. Probability of a Missed Detection (P_M): A target is present but there is no detection identified where $P_M = 1 - P_D$.

The probability of detection and the probability of false alarm are of interest and can be expressed as integrals of joint probabilities over some region $\mathcal{R} = x : P(H_1|x) > l_0$, where l_0 is a threshold defined by $0 \leq l_0 \leq 1$.

$$P_D = \int_{\mathcal{R}} p_x(\mathbf{x}|H_1)dx \tag{2.15}$$

$$P_{FA} = \int_{\mathcal{R}} p_x(\mathbf{x}|H_0)dx \tag{2.16}$$

2.3.4.2 Constant False Alarm Rate (CFAR) Detection

An approach for typical threshold detection processing was discussed in Detection Fundamentals, we now expand on this to include an approach where the interference levels can be variable which is often the case with a real-world radar application. Quote from MATLAB example ... A CFAR detection occurs when the input signal level in a cell exceeds the threshold l_0. The l_0 for each cell depends on the threshold factor and the noise power in that derived from training cells. We desire to maintain a constant false alarm rate, the detection threshold will increase or decrease in proportion to the noise power in the training cells. The probability distribution in detection of signals with noise follows a Rayleigh distribution. There are various forms of CFAR we will focus on what works best against typical noise jammers. Cell-averaging (CA)-CFAR is proven to be superior. https:www.researchgate.net publication 221616765$_CFAR_detectors_in_presence_of_jammer_noise$**Show the CFAR math...**

2.3.4.3 Detection Optimization

Based on the detection hypothesis we need a method to optimize our choice. Most radar detection processing use a Bayes optimization criteria that assigns a cost to a target being present or not present. In most radar systems a special case of the Bayes theorem called the Newman-Pearson detection criterion is the standard approach. [12] **Show the NP mathematics ...**

2.4 Electronic Jamming Techniques

Electronic jamming covers various techniques and spans many areas. We will focus on the area of *Deceptive Jamming*. We will look at a class of electronic jammers called smart jammers, commonly referred to as a deceptive jammers or DRFM jammers. Jammers in general are specifically employed to radiate interference via noise or signal towards the opposing radar thus jamming the opposing radar receiver. There are two types of methods for electronic jammers. They are noise and repeaters. Today's electronic countermeasure (ECM) equipment can work both as a deceptive and a noise jammer, while in the past these were two different classes of equipment. The aim of any ECM is to reduce the operational capability of a system. In our case, the system is a ground-based phased array radar.

2.4.1 DRFM

The DRFM has the capability to intercept transmitting radar signals (see section 2.3 Radar Signals) through high speed digital sampling and then stores the samples in memory for later use. DRFM then uses a repeater technique that can modify and re-transmit the stored transmitted signal by adjusting the time or frequency. This results in a false target that is radiated to the opposing radar receiver thus deceiving the radar with a false target position. The deception or false target occurs because the radar echo radiation is matched to the opposing radar's receiver, thus resulting in a false target detection. A target is detected based on establishing a threshold at the output of the radar receiver. If the receiver output is large enough to exceed the threshold, a target is said to be present. [10]The process also then must decide if this detection measurement represents the influence of a target or only signal interference. Statistical signal models are typically used. This then becomes a problem of statistical hypothesis testing.

2.4.1.1 Deceptive Jamming

Explain from a mathematics perspective...

2.4.1.2 Noise Jamming

Explain from a mathematics perspective...

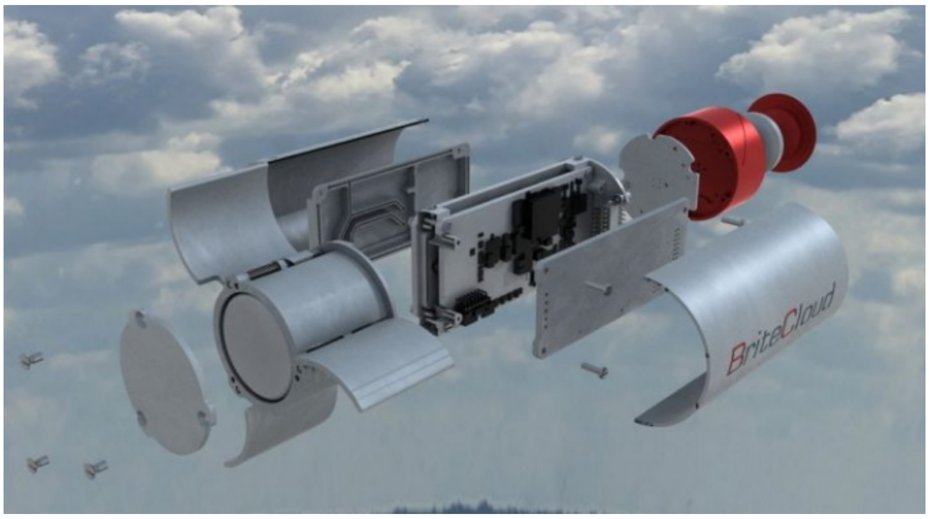


Figure 2.2: Hardware representation of an example DRFM object. The DRFM can be launched from a decoy dispenser or towed behind a missile threat since it is small (55 mm diameter) and light-weight (1.1 kg) [1]

Chapter 3

Machine Learning

3.1 Basic Concepts

We are still in the age of information with the advent of bigger and faster computers we have been able to store and process more and more big data. An understanding of basic concepts and terminology of machine learning will be addressed here and thus provides a foundation for the material that follows. Arthur Samuel in 1959 popularized the term "Machine Learning" (ML) which is the field of study that gives computers the ability to learn without explicitly being programmed. Since ML is considered an algorithm technique this falls under the larger category of Artificial Intelligence (AI). We define AI as any technique that gives a machine or computer the ability to perform tasks in a way like humans. In order to perform ML we need to first consider a framework for development and testing. We propose using modern application programming interfaces (APIs) such as Keras [13] and Tensorflow [14]. This will also support customized programming of any mathematical concepts that we develop. The training and execution will be done on graphics processing units (GPUs) which support parallel processing for faster code execution that we plan to benchmark for performance.

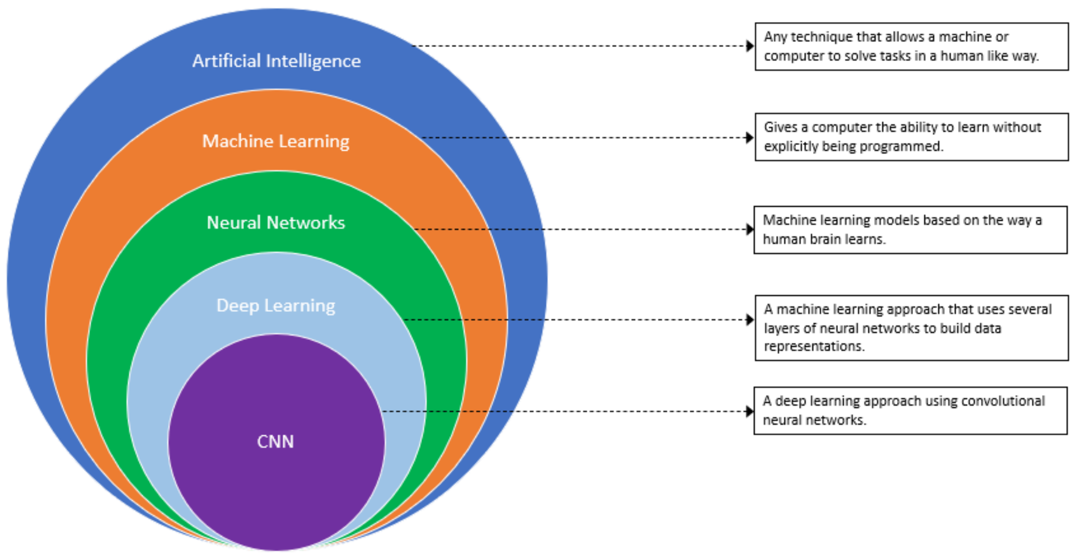

Artificial Intelligence
Machine Learning
Neural Networks
Deep Learning
CNN
Any technique that allows a machine or computer to solve tasks in a human like way.
Gives a computer the ability to learn without explicitly being programmed.
Machine learning models based on the way a human brain learns.
A machine learning approach that uses several layers of neural networks to build data representations.
A deep learning approach using convolutional neural networks.

Figure 3.1: Artificial Intelligence

Recall that we are using machine learning techniques because electronic countermeasures (ECM) in general that rely on parametric modeling can fail because it violates the strict assumptions of classical signal processing algorithms. We propose to use supervised ML to label our datasets or images to then train our algorithms that will learn and predict the correct jammer classification. The reasons that follow support the case for supervised ML:

- The problem of jammer classification is not a simple deterministic rules based solution solved by utilizing radar parameters. You can use supervised ML to effectively solve this problem.

- Radar range-Doppler images can manifest it many ways and thus is considered a large-scale problem. Supervised ML has been proven to be an effective approach for this type of problem. [15]

We propose stacking simple models in a creative fashion to realize a deep learning architecture such as CNN. Fundamentally we are seeking reduced dimensionality of our data to extract the image features which in our case are the target detections.

3.2 Hyperparameters

The selection and initial values of the parameters used in any machine learning problem is critical to understand to make the best use and choice for the problem space. We research different approaches and make recommendations that propose modifications to fit our problem space. We also propose to explore particular areas of interest so as to not limit or constrain ourselves to any one aspect of machine learning. The chosen areas of focus are detailed in the subsections that follow. The typical processing sequence used in machine learning for CNN motivates our areas of interest and exploration. The diagram below depicts the CNN processing chain.

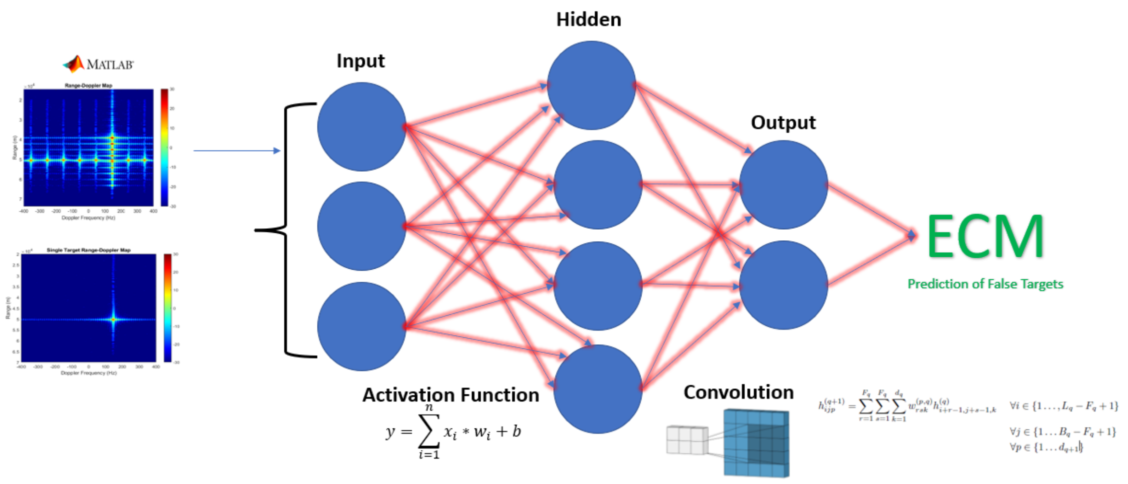

MATLAB
Input
Hidden
Output
ECM
Prediction of False Targets
Activation Function
$y = \sum_{i=1}^{n} x_i * w_i + b$
Convolution

Figure 3.2: High-Level CNN Processing Chain

3.2.1 Hyperparameters Tuning

Hyperparameter tuning and machine learning in general is often done manually by hand in a trail and error approach which relies on the empirical results to guide the experimenter in the optimization of the hyperparameters. This can be very time consuming to explore the hyperparameter space. There are two popular choices for a model and a brute force approach like Grid Search could be used by defining a search space over all the hyperparameters and going through every position in the grid which is not efficient. Random Search is the other approach where random values are chosen so there is the possibly of discovery but you could also miss important settings. So we must carefully define and constrain our model. We explore several key hyperparameters to optimize our tuning and results. A mathematical approach will be developed in this chapter so as to gain a more rigorous understanding for better informed choices when conducting our experiments.

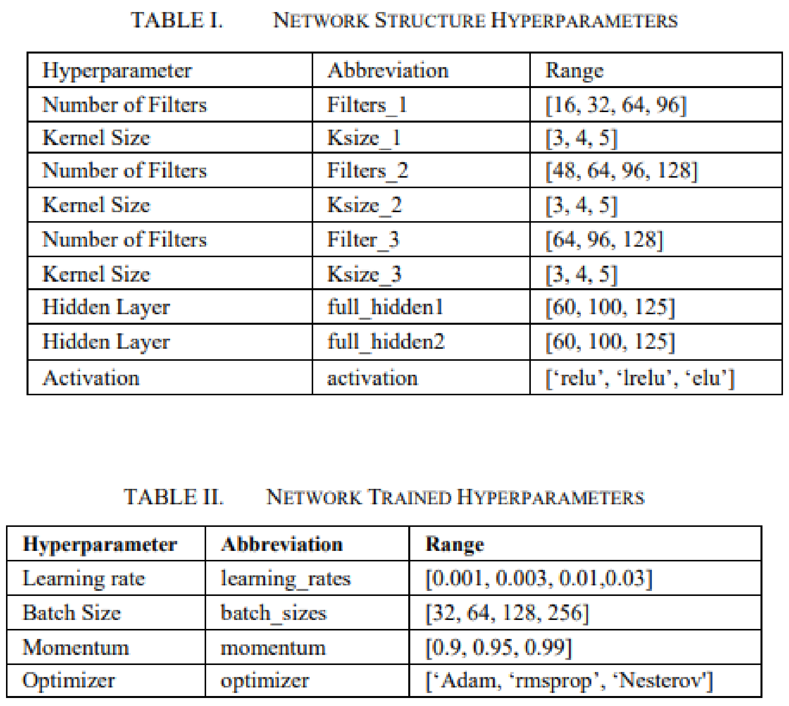


TABLE I. NETWORK STRUCTURE HYPERPARAMETERS

Hyperparameter	Abbreviation	Range
Number of Filters	Filters_1	[16, 32, 64, 96]
Kernel Size	Ksize_1	[3, 4, 5]
Number of Filters	Filters_2	[48, 64, 96, 128]
Kernel Size	Ksize_2	[3, 4, 5]
Number of Filters	Filter_3	[64, 96, 128]
Kernel Size	Ksize_3	[3, 4, 5]
Hidden Layer	full_hidden1	[60, 100, 125]
Hidden Layer	full_hidden2	[60, 100, 125]
Activation	activation	['relu', 'lrelu', 'elu']

TABLE II. NETWORK TRAINED HYPERPARAMETERS

Hyperparameter	**Abbreviation**	**Range**
Learning rate	learning_rates	[0.001, 0.003, 0.01,0.03]
Batch Size	batch_sizes	[32, 64, 128, 256]
Momentum	momentum	[0.9, 0.95, 0.99]
Optimizer	optimizer	['Adam, 'rmsprop', 'Nesterov']

Figure 3.3: Hyperparameters (Place Holder)

3.2.2 Activation Function

The activation function is responsible for the output of a node, in our case a node is each pixel for our Range-Doppler image. Each image is a 64x64 square matrix so that is 4096 input nodes in total. The question answered here "Is this pixel useful or not?", if useful then that pixel gets activated "fired like in the brain" and considered important, in a sense it's like a mathematical filter or gate. In this case we are seeking a non-linear activation function with a purpose to add non-linearity to the

NN, without the activation function it would be linear by design because all the hidden layers will behave in the same way because the addition of two or more linear functions is linear. In modern neural networks, the default recommendation is to use the rectified linear unit or ReLu. [16] That however does not prelude us from gaining a mathematical perspective and selecting the proper activation function for this problem.

A nonlinear activation function $f()$ sometimes called a transfer function [17] can be expressed as

$$y(x, w) = f\left(\sum_{k=1}^{N} w_k \phi_k(x)\right) = \mathbf{w}^T \phi(x) \tag{3.1}$$

Where N is the total number of parameters in the model. We now extend this function by making the basis functions $\phi_k(x)$ or features depend on N parameters, along with adjusting the weighing coefficients w_k during the training of our CNN model. This then gives us a general neural network model that can be expressed as

$$a_k = \sum_{l=1}^{M} w_{kl} x_l + b_l \tag{3.2}$$

where b_0 is the bias constant and a_k is our activations. We now can use a transformation assuming a differentiable non-linear activation function $h_j = g(a_k)$ to then give the output of each activation.

$$a_i^{(1)} = \sum_{j=1}^{M} w_{ij}^{(1)} h_j + b_j^{(1)} \tag{3.3}$$

We use the notation $a_i^{(1,2,...)}$ to denote the layers of our network. There can be multiple layers which is common for CNNs. Written in matrix and vector form we get **Update with multiple layers**

$$\sum_{j=1}^{M} w_{ij}^{(1)} h_j + b_j = \begin{bmatrix} w_{0,0}^{(1)} & w_{0,1}^{(1)} & ... & w_{0,j}^{(1)} \\ w_{1,0}^{(1)} & w_{1,1}^{(1)} & ... & w_{1,j}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0}^{(1)} & w_{i,1}^{(1)} & ... & w_{i,j}^{(1)} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_j \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_j \end{bmatrix} \tag{3.4}$$

Written in a compact form we get

$$\mathbf{a^{(1)}} = f(\mathbf{W}^{(1)}\mathbf{h} + \mathbf{b}) \tag{3.5}$$

This now makes it convenient for writing a code block as this simply can be written for example in Python as $a = relu(dot(w, h) + b)$ where $relu$ is the rectified linear unit activation function.

Now that we have some mathematical insight into the activation function, we can precede with an approach for selecting the best activation function for the problem at hand which we consider as a categorical image classification. Since our problem involves multiple classifications we can eliminate any activation function that is binary or is susceptible to vanish gradients. So for example the binary activation functions like binary step and sign functions would not be useful. Functions such as tanh and sigmiod can saturate or cause vanishing gradients which is not what we want when filtering through data. We now define requirements for a good activation function for our image classification.

- Nonlinear activation function since we have a multi-layer network
- Differentiable and monotonic to allow back propagation
- Speed and accuracy
- Solve vanishing gradient or saturation problem

For convenience we have plotted several activation functions to help in our selection process. After conducting research ReLU seems to be the best choice, although we need to consider the negative values mapping to zero (i.e. dead neurons). ReLU is based on a cortex-inspired silicon circuit. [18] It has been demonstrated to improve the training of deep neural networks for deep learning. [19] We define the ReLU function as follows:

$$f(\theta) = max(0, \theta) = \begin{cases} \theta & : \theta \geq 0 \\ -\theta & : \theta < 0 \end{cases} \tag{3.6}$$

$$f'(\theta) = \begin{cases} 1 & : \theta > 0 \\ 0 & : \theta < 0 \end{cases} \tag{3.7}$$

We note that ReLU is not differentiable at zero. We can arbitrarily assume when $\theta = 0$ that the differentiated value is zero. Also negative values result in a dead neuron since $f'(\theta) = 0$ for $\theta < 0$ such that no gradients flow back through the CNN processing chain. **Probably should illustrate this...** The consequence of pushing too many negative values through the processing chain is that our model will stop learning, since the activation is zero. To address this issue we can facilitate a slight curve or linear slope thus the negative values result in $f'(\theta) < 0$. Another approach is to lower the learning rate which we will explore in section 3.2.4.1.**Provide mathematical evidence or theory to support the choice of ReLU...**

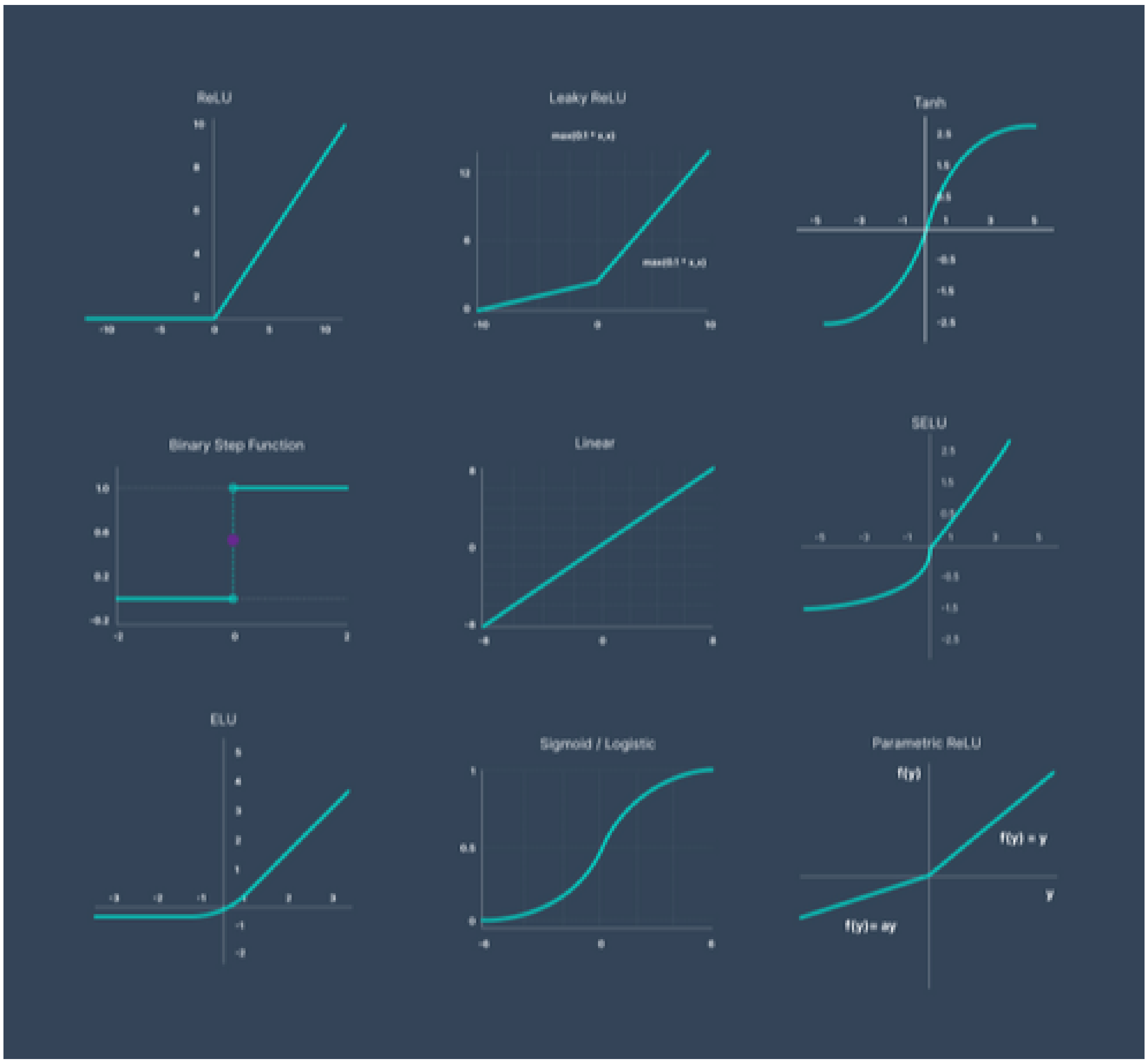


Figure 3.4: Activation Functions (Place Holder)

3.2.3 Loss Function

The Loss Function is used to evaluate the model predictions versus the truth or input data. The Loss Function acts as a guide to the optimizer which attempts to provide the correct direction towards the global minimum. The output is a scalar number that is used to minimize or maximize the model loss, which allows us to rank or compare candidate solutions. The question answered here is "How good or useful is my model at making predictions using a given set of parameters (i.e. weights and biases)?" Is it "To good to be true!" which is overfitting and not useful at all. Which objective function do we use? There are many different functions that can be used to provide error estimates for each set of parameter weights in a CNN. Cross-entropy is the default loss function to use for most multi-class classification problems. Cross-entropy measures the error between two probability distributions. We consider the predicted model and the true training probability distributions. We seek to minimize the cross-entropy between these two distributions. We also propose a constraint of a symmetric loss. The claim is that we do not always expect clean data and thus must account for noisy data in our classification. It is also highly desirable to learn from corrupted data since we expect this in real-world applications.

We define our loss model as

$$\mathcal{L}(X,\theta) = -\sum_{j=1}^{N} X_j log(\theta_j) \tag{3.8}$$

where X is random variable whose domain is our dataset of range-Doppler samples and θ is the predicted variable of interest. Since we have multiple classes we need a a function to keep track of the probabilities of each class. The common approach is to use a softmax function which outputs a vector of sample values that sum up to a value of 1, which is generating the predicted probability distribution of the classes we define. Ideally the aim is to have the samples to be as close to unity as possible so then we have a high probability of the correct classification. We define our softmax function as $p : \mathbb{R}^N \rightarrow [0,1]^N$ where $N \geq 1$ such that

$$p_i = \frac{e^{s_i}}{\sum_{j=1}^{N} e^{s_j}} \tag{3.9}$$

where $i = 1, 2, ...N$ and $\mathbf{s} = [s_1, s_2, ...s_N] \in \mathbb{R}^N$

Using our logarithmic measure from Eq.(3.8) we now have our general equation for the loss function.

$$-\sum_{i=1}^{N} log\left(\frac{e^{s_i}}{\sum_{j=1}^{K} e^{s_j}}\right) \tag{3.10}$$

3.2.4 Optimization

We define our problem as non-convex which fits for a stochastic gradient descent (SGD) based optimization approach. This implies that we seek an optimization that is not based on a direct calculation of the gradient. The SGD approach aligns with real world types of problems where analytical solutions are very few. We also confine our focus to mini-batch SGD which is the popular choice for training a CNN. Online training was considered but seems impractical to adjust the gradient after each forward and backwards pass of the data thus affecting overall performance. Using mini-batch approach allows each subset of data to be trained with the gradient estimated at each step. The motivation is twofold. First we are not loading the data all at one time thus reducing the memory footprint. In our problem space that amounts to over ($64x64x4000$) 16 million data samples. Second we are updating the parameters more frequently with mini-batch by running numerous batches compared to a single batch. Therefore we reduce the probability of getting stuck in a local minima and increase the probability of finding a global minima. Compare this to a full batch that tends to get stuck in local minima thus not optimal for global solutions. [20] An additional consideration is the speeding up of the convergence. Learning rate and momentum methods can be used to accelerate the solution convergence compared to default SGD. For optimization we will establish a customized SGD approach that fits our problem space of range-Doppler imaging classification.

3.2.4.1 Learning Rate and Momentum

Learning rates matter and its regard among practitioners to be the most important hyperparameter to tune. [21] [16] Too large a step and we diverge, too small of a step and we do not make progress. So if I'm really confident about the gradient then for example we can set the learning rate = 1.0 and take a confident large step forward. If its uncertain then a smaller step or learning rate is required but now it takes longer to converge to a global minima or worse get stuck in less desirable local minima. How do we make the optimal choice for the learning rate? There are two popular choices to consider they are learning rate schedulers and adaptive learning rate methods. We seek an adaptive method. Using an adaptive approach we hope to optimize the step size to avoid small local minima and thus converge faster and have more accurate learning models that seek a global minima. We propose to combine an adaptive learning rate and momentum so as not to concern the user with having to experiment with setting these hyperparameters to obtain the best results. Recent activity in this area has motivated research in finding optimal solutions for these hyperparameters [22] [23] and thus is the foundation for a new algorithm that we name the Adaptive Learning Rate and Momentum (ALRM) method. Our high-level goals for this method are to reduce the computational complexity, faster convergence to a global minimum with auto tuning parameters that provide numerical stability, and easy implementation of the algorithms.

3.3 ALRM Method

We start with a default expression for SGD where θ_t is our model variable parameter of interest, X_t is a random variable and η_t is the adaptive learning rate we evaluate at each iteration:

$$\theta_{t+1} = \theta_t - \eta_t \nabla \mathcal{L}(X_t, \theta_t) \tag{3.11}$$

Research points out a sensitivity to the learning rate for default SGD. [24] [25] So we propose an implicit SGD approach that reduces learning rate sensitivity and provides numerical stability. Implicit updates are evaluated at the next iteration $(t+1)$ rather than the current iteration (t). We define this as implicit because we have θ_{t+1} on both sides of the equation. So then we have

$$\theta_{t+1} = \theta_t - \eta_t \nabla \mathcal{L}(X_t, \theta_{t+1}) \tag{3.12}$$

Expanding on this implicit Eq.(3.11) to gain more intuition and motivate numerical stability [26], lets suppose a limit that approaches infinity for our model vector parameter θ such that

$$\lim_{n\to\infty} \theta_{t+1}^{(n)} \quad \text{where } n \in \mathbb{Z}^{0+} \tag{3.13}$$

Now lets write out the sequence

$$\begin{aligned}
\theta_{t+1}^{(1)} &= \theta_t - \eta_t \nabla \mathcal{L}(X_t, \theta_t^{(0)}) \\
\theta_{t+1}^{(2)} &= \theta_t - \eta_t \nabla \mathcal{L}(X_t, \theta_{t+1}^{(1)}) \\
&\ldots \\
\theta_{t+1}^{(\infty)} &= \theta_t - \eta_t \nabla \mathcal{L}(X_t, \theta_{t+1}^{(\infty)}).
\end{aligned}$$

We can observe that the last term of the sequence above can be re-written as $\theta_{t+1} = \theta_t - \eta_t \nabla \mathcal{L}(X_t, \theta_{t+1})$ where we assume that the limit converges to a fixed point. We then have the implicit SGD we defined in Eq. (3.11). Therefore the implicit SGD is shown to be a repeated sequence or variant of the default expression for SGD. Its easy to see that we are just updating the θ_t term until a fixed point or optimal solution is reached. This concept is related to the *self-consistency principle* in statistics. The term "self-consistency" was introduced in 1989 by Hastie and Stuetzle, where given a smooth curve or surface that each point is considered the mean of all points that project orthogonally onto it. We now provide mathematical argument for numerical stability by re-writing our implicit Eq.(3.11).

$$\begin{aligned}
\theta_{t+1} &= \theta_t - \eta_t \nabla \mathcal{L}(X_t, \theta_t), \\
\theta_t &= \theta_{t+1} + \eta_t \nabla \mathcal{L}(X_t, \theta_t), \\
||\theta_t - \theta^*||^2 &\geq ||\theta_{t+1} - \theta^*||^2 + 2\eta_t(\theta_{t+1} - \theta^*)^T \nabla \mathcal{L}(X_t, \theta_t), \\
||\theta_t - \theta^*||^2 &\geq (1 + \eta_t \mu)||\theta_{t+1} - \theta^*||^2 \quad \text{where } \mu > 0, \\
||\theta_{t+1} - \theta^*||^2 &\leq \frac{1}{(1 + \eta_t \mu)}||\theta_t - \theta^*||^2.
\end{aligned}$$

This shows that $||\theta_{t+1} - \theta^*||^2$ has a high probability of contracting and thus providing the numerical stability that we desire.

A critical and important operation is finding the weighted averages of our function. So we define a finite-sum structure by minimizing over our loss function $\mathcal{L}(X, \theta)$ that is continuously differentiable over a dataset space called S where $\mathcal{L}(X, \theta)$ is defined by

$$\mathcal{L}(X, \theta) : [S \subset \mathbb{R}^n \to \mathbb{R}] \quad \text{where } n \text{ is the number of sample data points} \tag{3.14}$$

We consider an unconstrained optimization in which we seek the minimization of

$$\theta^* = \min_{\theta \in \mathbb{R}^n} \mathcal{L}(X, \theta) = \frac{1}{n} \sum_{t=1}^{n} \ell_t(X, \theta) \tag{3.15}$$

$$\approx \mathbb{E}[\ell(X, \theta)] \tag{3.16}$$

such that a sample data point θ^* is a local minimum if $\mathcal{L}(\theta^*) \leq \mathcal{L}(X, \theta) \forall \in S$. Furthermore we make the following assumptions for non-convex stochastic optimization which are typical for this type of problem.

- Assumption 1: The stochastic gradient is uniformly bounded, for example, $sup_t(||g^t||) \leq R$ with $R > 0$.
- Assumption 2: The SGD is an unbiased estimate, for example, $\mathbb{E}[\ell(X, \theta)] = \nabla \mathcal{L}(X, \theta_{t+1})$.
- Assumption 3: The gradient of $\mathcal{L}$ is $L - Lipschitz$, for example, $||\nabla \mathcal{L}(x) - \nabla \mathcal{L}(y) \leq L||x - y||$ where $L > 0$.

3.3.1 Learning Rate Dynamic Adjustment

We propose using a Quasi-Newton method which is a well established approach to solve a non-convex optimization problem. This is based on Newton's method which is an alternative that assumes that the Hessian matrix is not available or practical since its computational complexity is $\mathcal{O}(n^3)$ to compute the Hessian matrix and its inverse. So then the idea is to then approximate the Hessian (and it's inverse) matrix. We will start with the BFGS method which is named after the authors Broyden, Fletcher, Goldfarb, and Shanno and is considered a very effective Quasi-Newton method and its computational complexity $\mathcal{O}(n^2)$ is less expensive to compute and given by [27]:

$$B_{k+1} = B_k - \frac{1}{\mathbf{s}_k^T B_k \mathbf{s}_k} B_k \mathbf{s}_k \mathbf{s}_k^T B_k + \frac{1}{\mathbf{y}_k^T \mathbf{s_k}} \mathbf{y_k} \mathbf{y}_k^T \tag{3.17}$$

where

$$y_k = \nabla\mathcal{L}(\theta_{k+1}) - \nabla\mathcal{L}(\theta_k) \text{ and } s_k = \theta_{k+1} - \theta_k.$$

The BFGS method provides positive-definite solutions so then we define B_k as the Hessian estimate which is a positive definite symmetric nxn matrix, furthermore the matrix B_{k+1} is guaranteed to be positive definite given B_k is also positive definite and we satisfy the curvature condition of $\mathbf{s}_k^T \mathbf{y}_k > 0$. There is still a performance concern if the rank of the matrix B_k is high. The optimization problem we are solving assumes to be unconstrained since we are required to solve for $\nabla\mathcal{L}(\theta_k)$ at each iteration. There are methods that we recommend to construct a sequence of lower rank matrices that have been proven to do a good job of approximating the Hessian matrix and thus provide the improved computational performance we seek. The limited-memory BFGS (L-BFGS) is that method [28]. So then we continue with Eq. (3.16) and let us define S_k and Y_k by $(n \times k)$ matrices where

$$S_k \triangleq [\mathbf{s}_0, \mathbf{s}_1, ..., \mathbf{s}_{k-1}] \tag{3.18}$$

$$Y_k \triangleq [\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_{k-1}] \tag{3.19}$$

Then let B_0 be a positive definite symmetric nxn matrix such that $s_i^T y_i > 0$ then using Eq. (3.16) we get a more compact form

$$B_k = B_0 - [B_0 S_k \; Y_k] \begin{bmatrix} S_k^T B_0 S_k & L_k \\ L_k^T & -D_k \end{bmatrix}^{-1} \begin{bmatrix} S_k^T B_0 \\ Y_k^T \end{bmatrix} \tag{3.20}$$

where D_k is the diagonal part and L_k the lower triangular part of the matrix we define as a symmetric $k \times k$ matrix

$$(L_k)_{i,j} = \begin{cases} s_{i-1}^T y_{j-1} & if\ i > j \\ 0 & otherwise. \end{cases} \tag{3.21}$$

We now take the compact form of the BFGS Eq. (3.19) and define as L-BFGS. Let

$B_0 = \sigma I$ where I is an identity matrix and σ is a positive scale value. Substituting into Eq. (3.19) we then get

$$B_k = \sigma I - [\sigma S_k \; Y_k] \begin{bmatrix} S_k^T \sigma S_k & L_k \\ L_k^T & -D_k \end{bmatrix}^{-1} \begin{bmatrix} S_k^T \sigma \\ Y_k^T \end{bmatrix} \tag{3.22}$$

We now provide the inverse of the Hessian estimate H_k where $H_k = B_k^{-1}$, so then we simply get

$$H_{k+1} = \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) H_k \left(I - \frac{s_k y_k^T}{y_k s_k^T}\right) + \frac{y_k y_k^T}{y_k s_k^T} \tag{3.23}$$

Its important to point out that the main weakness we discovered of the L-BFGS method. If we have a Hessian matrix that contains a wide variance of eigenvalues, the solution tends to convergence more slowly due to the ill-conditioned problem. This approach is still favorable over a basic gradient descent since it has the property of always monotonically decreasing at each iteration unless the model parameter θ has arrived at a local or global minimum. Also keeping in mind one of our goals for ALRM is for easy implementation. Code libraries exist for the L-BFGS method through Tensorflow and SciPy, which aligns with our coding implementation strategy.

3.3.2 Momentum Dynamic Adjustment

The use of stochastic momentum with SGD is a popular choice and widely accepted which we will explore. In particular, we consider a stochastic heavy ball momentum with Polyak averaging or Polyak's momentum. [29] It is also the default choice of momentum in Tensorflow which is an API we use for implementation of our CNN model. The motivation is to escape the local minima or saddle points which can inhibit the path to a global minimum. We did consider Nesterov momentum which is also a popular choice and is an option in the popular Adam optimizer. Since we are using a finite sum structure recent papers have proved a possible divergence with Nesterov's approach. [30] [31]

First we assert that a weighted average of stochastic gradients will be preserved at each iteration through our sample data. We also will concentrate on locating a second-order point assuming a smooth non-convex optimization by implicit SGD using a *stochastic heavy ball method* which we first establish by the *heavy ball method* which is defined by the following state transitions:

$$p_t = \nabla \mathcal{L}(X, \theta_{t+1}) \mid \nu_t p_{t+1} \tag{3.24}$$

$$\theta_{t+1} = \theta_t + \eta_t p_t \tag{3.25}$$

We can rewrite this as the iteration using our general Eq.(3.11) that we derived earlier.

$$\theta_{t+1} = \theta_t - \eta_t \nabla \ell_t(X, \theta_{t+1}) + \nu_t(\theta_{t+1} - \theta_t) \tag{3.26}$$

where the term $\theta_{t+1} - \theta_t$ is referred to as *momentum.* Note that we replaced the true gradient $\nabla \mathcal{L}(\theta_{t+1})$ with our unbiased estimate gradient from *assumption 2* where $\nabla \ell_t = \mathbb{E}[\ell(X, \theta)]$ to simply make this a *stochastic heavy ball method.* We now have a general Eq.(3.25) for the learning rate η_t combined with momentum ν_t. We still need a more rigorous understanding of the *stochastic heavy ball method.* So we establish the following auxiliary theorems.

Lemma 1.

The non-convex unconstrained optimization problem is defined as before where

$$\theta^* = \min_{\theta \in \mathbb{R}^n} \mathcal{L}(\theta) = \frac{1}{n} \sum_{t=1}^{n} \ell_t(\theta) \tag{3.27}$$

$$= \boldsymbol{E}[\mathcal{L}(\theta)] \tag{3.28}$$

3.3.3 Algorithms

Algorithm 1 *ALRM*, our new algorithm for SGD that incorporates adaptive methods for learning rate and momentum. Vector operation is element-by-element. **Still in work...**

```
Require: $\nabla \ell_t(X, \theta_{t+1})$ : SGD loss function
Require: $X$ : Random variable
Require: $\theta$ : Model parameter weights                ▷ Variable column vector of interest
Require: $\epsilon$ : Convergence tolerance
Require: $\eta$ : Learning rate
Require: $\nu$ : Momentum
Require: $m$ : Mini-batch size
Ensure: $\epsilon > 0$
 1: Input: A set of training data that consists of training vectors $\{x_n\}$ where $n = \{1, 2, ..., N\}$.
 2: function ALRM($a, b$)
 3:     for $t = 0, ..., T$ do
 4:         Choose $m$ integers $k_1, k_2, ..., k_m$ uniformly and independent from {1,2,3,...,N}
 5:         $s_t \leftarrow$ sample a mini-batch of set of training data using $m$ integers without replaceme
 6:         while $||\nabla \ell(X, \theta_{t+1}) < \epsilon||$ do                ▷ Quasi-Newton condition
 7:             $\eta \leftarrow \beta \cdot \eta$                ▷ $\eta$ will be smaller by at most a factor of $\beta$
 8:             perform Quasi-Newton method                ▷ See Algorithm 2
 9:             perform stochastic heavy-ball method                ▷ See Algorithm 3
10:             $\theta_{t+1} \leftarrow \theta_t - \eta_t \nabla \ell_t(X, \theta_{t+1}) + \nu_t(\theta_{t+1} - \theta_t)$ ▷ Implicit SGD with $\eta$ and $\nu$
11:         end while
12:         $\mathbf{\theta_{t+1}} \leftarrow \hat{\theta}_\mathbf{t}$
13:     end for
14:     return $\mathbf{\theta_{t+1}}$
15: end function
```

Algorithm 2 *Quasi-Newton Method based on L-BFGS*, solve a non-linear optimization problem with an approximation of the Hessian matrix and its inverse. **Still in work...**

```
Require: $\epsilon$ : Convergence tolerance
Require: $\nabla\ell(\theta)$ : Gradient loss function
Ensure: $\epsilon > 0$
 1: function L-BFGS($a, b$)
 2:     Choose an initial Hessian inverse $H_t^0$                        ▷ cite
 3:     $p_k \leftarrow -H_t\nabla\ell(X, \theta_{t+1})$        ▷ compute a search direction $p_t$
 4:     $x_{t+1} \leftarrow x_t + \eta_t p_t$          ▷ $\eta$ satisfies the Wolfe conditions(cite)
 5:     if t>m then
 6:         $\{s_{t-m}, y_{t-m}\}$                                  ▷ delete from memory
 7:     end if
 8:     $s_t \leftarrow x_{t+1} - x_t$                                   ▷ save in memory
 9:     $y_t \leftarrow \nabla\ell(\theta_{t+1}) - \nabla\ell(\theta_t)$                        ▷ save in memory
10:     $t \leftarrow t + 1$
11:     if $\sim$ (converged || failed) then
12:         return inverse Hessain updated state
13:     end if
14: end function
```

Algorithm 3 *Momentum Method*, a stochastic heavy ball method based on Polyak momentum. **Still in work...**

```
Require: $0 \leq \nu \leq 1$ : Momentum
Require: $0 \leq \eta \leq 1$ : Learning rate (i.e. Step size)
 1: function SHB(a,b)
 2:     for $t = 0, ..., T$ do
 3:         $\theta_{t+1} = \theta_t - \eta_t\nabla\ell_t(X, \theta_{t+1}) + \nu_t(\theta_{t+1} - \theta_t)$
 4:     end for
 5:     return $\nu$
 6: end function
```

Chapter 4

Simulation Modeling

4.1 Modeling Approach

The approach is to use synthetic models rooted in the MATLAB and Python code base.

4.2 Radar Model

The radar model is implemented in MATLAB. The selection of MATLAB is based on several reasons. First, many professional radar engineers rely on MATLAB because of its rich and vetted radar functions along with the toolboxes which model radar systems and handle radar signal processing. Secondly, I have used MATLAB for many years and I trust the software for modeling radar systems. This trust of course relies on data metrics and evaluation which we will expand upon in Chapter 5.

4.2.1 Requirements

1. The radar testbed shall simulate the radar transmit and receiver operations.
2. The radar testbed shall model environmental effects.
3. The radar testbed shall model antenna effects.
4. The radar testbed shall provide a waveform model.
5. The radar testbed shall generate DRFM false targets.
6. The radar testbed shall generate real targets.
7. The radar testbed shall generate noise interference.
8. The radar testbed shall generate raw uncompressed IQ data.
9. The radar testbed shall perform signal processing on the IQ signal.
10. The radar testbed shall perform detection processing of the targets (i.e. search and track).
11. The radar testbed shall perform target DRFM classification (i.e. real vs false targets).
12. The radar testbed shall output range Doppler maps.

4.2.2 Training Data Generation

To define our data we first introduce the meta data or parameters which are used in our MATLAB radar simulation model.

4.3 CNN Model

4.3.1 Architecture

A CNN is made up of ...

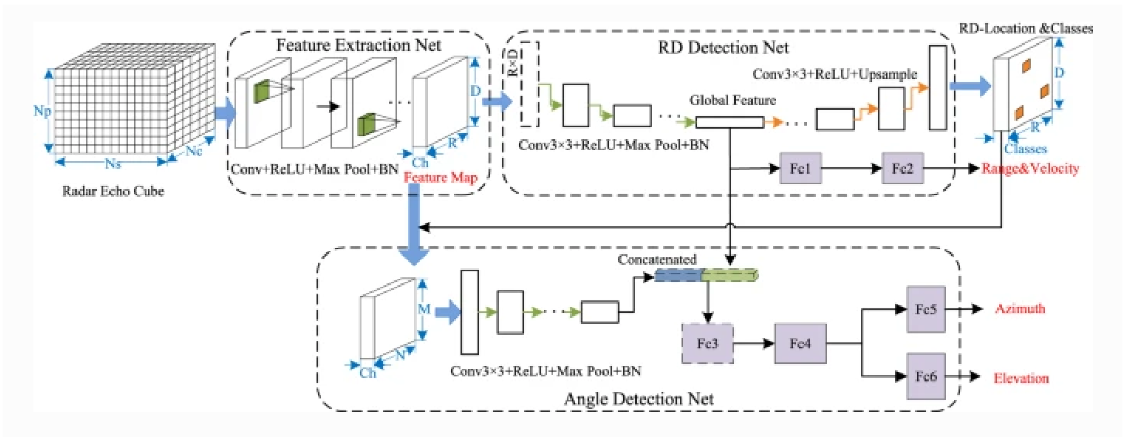


Figure 4.1: CNN (Place Holder)

4.3.2 Training Strategy

A training strategy is applied to our CNN to establish the minimum error loss that is possible. To accomplish the minimum error loss we seek the most optimal hyperparameter settings for the CNN to the Range-Doppler inputs that are used as training data.

4.3.3 Test Data

The training data inputs are Range-Doppler images that are 64x64 in size, which represents the spatial location and 256 grayscale values that range from (0:black to 255:white) for each pixel in the image. While color images with a size of 656x875x3 could have been used and visually look more appealing it introduces another dimension. The extra dimensional increases computational complexity, along with the higher resolution. This is not needed since we are interested in the intensity of each pixel value, therefore grayscale works just fine. We down-sampled the images to a size of 256x256 in grayscale. Several different image sizes were visually inspected. The sizes ranged from 16x16 up to 256x256. The selection of 64x64 with 4096 numerical values had enough resolution to by used the CNN and produce reasonable results. Too much resolution just introduces computational overhead and does not really improve results. We also constrain our range-Doppler images to square matrices which allows for useful linear algebra operations such as calculating the determinant, eigenvalues, eigenvectors, and singular value decomposition (SVD). Since the range-Doppler images are dominated mostly with Gaussian noise the matrices are also dense and thus mostly non-zero.

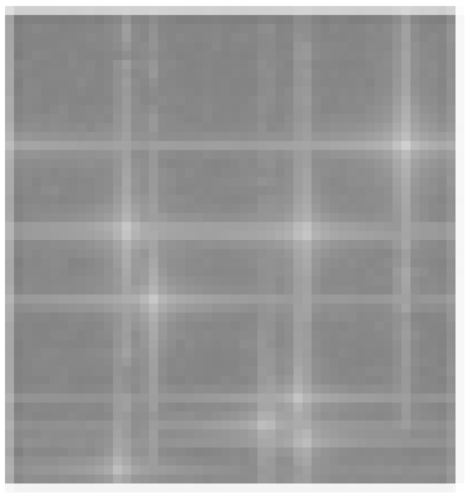


Figure 4.2: Example jammer image with random false targets at grayscale (64x64)

Each range-Doppler image is assigned a particular category which maps to the DRFM type. The following DRFM types are as follows:

- Range bin masking
- Doppler bin masking
- Combined range and Doppler bin masking
- Random false targets

Chapter 5

Results

5.1 Methodology

There are fundamental principles of statistics that need to be understood when developing methods for model building and analysis. A baseline for detection measurement is established by the Radar model using the detection process covered in the Radar Signal processing 2.3.3 section and will be used as comparison to the CNN model. The diagram below represents a process developed to support EN.625.661.83.SP21 Statistical Models and Regression JHU course project work, which we will incorporate to evaluate the goodness of our Radar and CNN models.

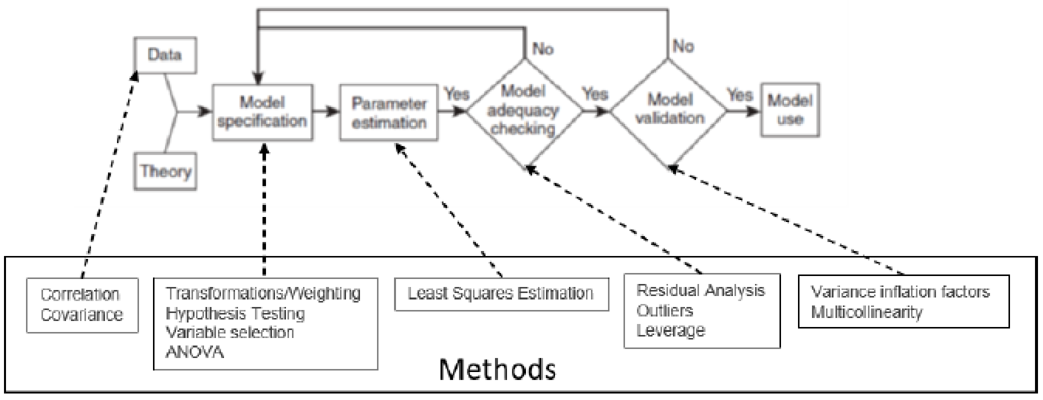


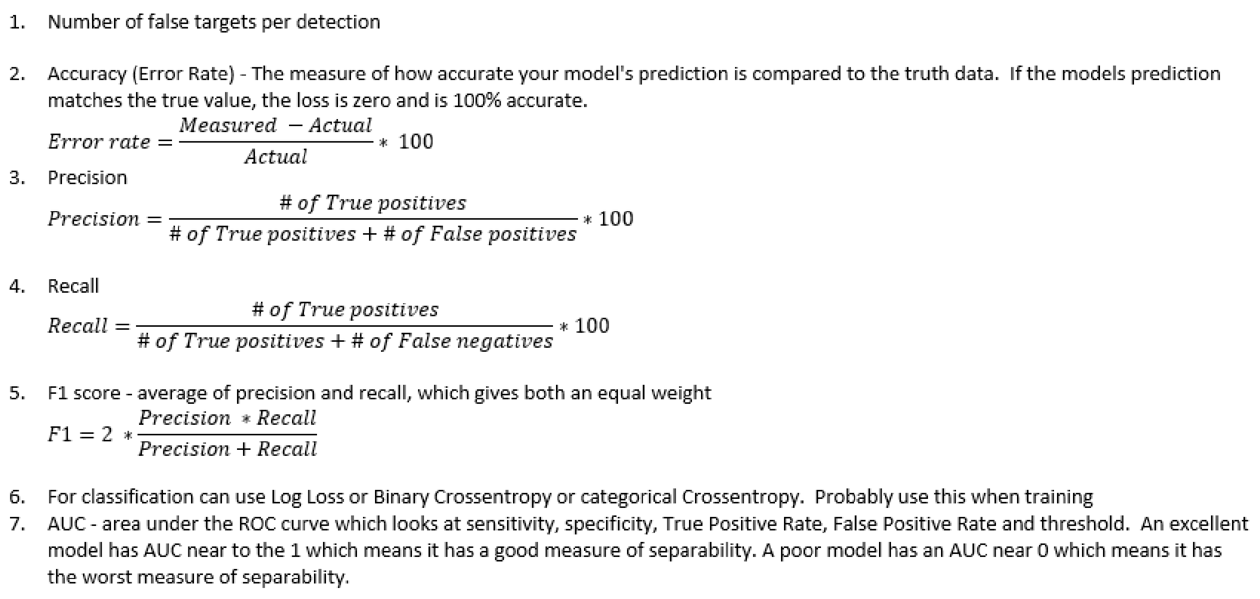
Figure 5.1: Model Building Process (Place Holder)

5.2 Scenarios

- Mainlobe beam contains both a mix of real and false targets
- Mainlobe beam contains false targets
- Mainlobe beam contains real targets
- Mainlobe beam contains NO targets (i.e. just noise)

5.3 Data Analysis

5.3.1 Metrics and Evaluation



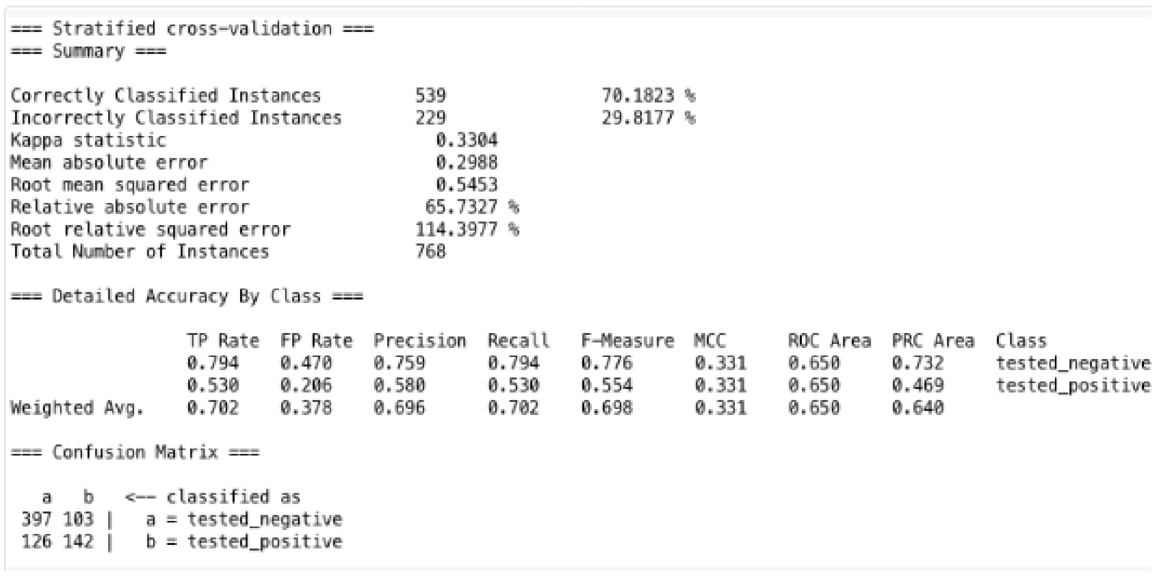
1. Number of false targets per detection
2. Accuracy (Error Rate) - The measure of how accurate your model's prediction is compared to the truth data. If the models prediction matches the true value, the loss is zero and is 100% accurate.
   $Error\ rate = \frac{Measured\ - Actual}{Actual} * 100$
3. Precision
   $Precision = \frac{\#\ of\ True\ positives}{\#\ of\ True\ positives + \#\ of\ False\ positives} * 100$
4. Recall
   $Recall = \frac{\#\ of\ True\ positives}{\#\ of\ True\ positives + \#\ of\ False\ negatives} * 100$
5. F1 score - average of precision and recall, which gives both an equal weight
   $F1 = 2 * \frac{Precision\ * Recall}{Precision + Recall}$
6. For classification can use Log Loss or Binary Crossentropy or categorical Crossentropy. Probably use this when training
7. AUC - area under the ROC curve which looks at sensitivity, specificity, True Positive Rate, False Positive Rate and threshold. An excellent model has AUC near to the 1 which means it has a good measure of separability. A poor model has an AUC near 0 which means it has the worst measure of separability.

Figure 5.2: Metrics (Place Holder)

5.3.2 Statistical Methods

For this thesis we will limit ourselves to standard statistical concepts and not attempt to develop any novel statistical techniques or measurements.

5.3.3 Algorithm Performance



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         539               70.1823 %
Incorrectly Classified Instances       229               29.8177 %
Kappa statistic                          0.3304
Mean absolute error                      0.2988
Root mean squared error                  0.5453
Relative absolute error                 65.7327 %
Root relative squared error            114.3977 %
Total Number of Instances              768

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.794    0.470    0.759      0.794    0.776      0.331    0.650     0.732     tested_negative
                 0.530    0.206    0.580      0.530    0.554      0.331    0.650     0.469     tested_positive
Weighted Avg.    0.702    0.378    0.696      0.702    0.698      0.331    0.650     0.640

=== Confusion Matrix ===

   a   b   <-- classified as
 397 103 |   a = tested_negative
 126 142 |   b = tested_positive
```

Figure 5.3: Example Results

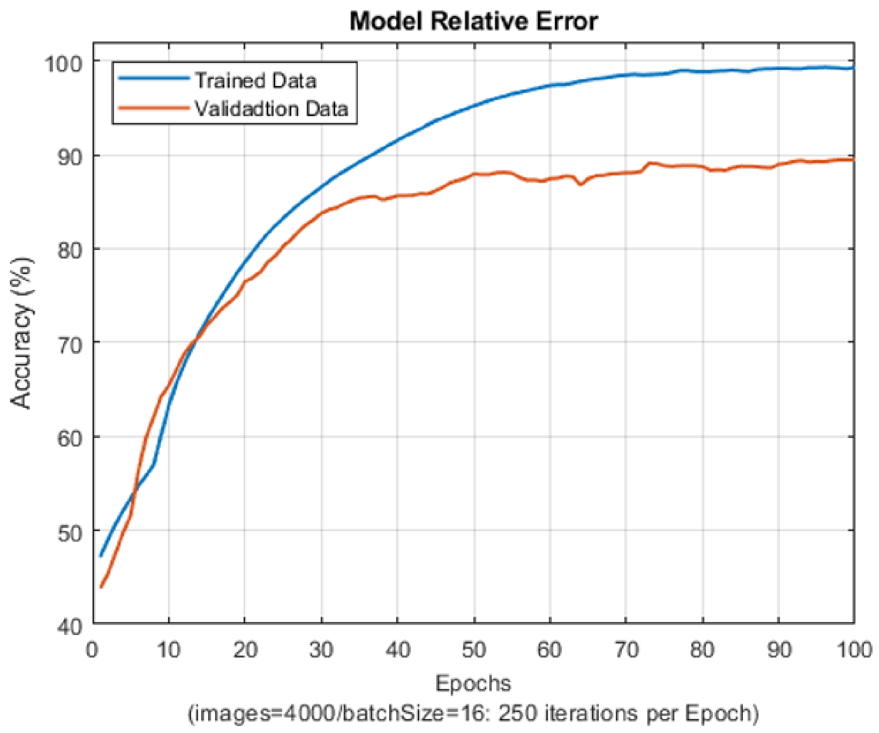


Figure 5.4: Model Relative Error

Define performance parameters

5.4 Conclusion

We summarize our main conclusions based on the research and body of work produced in this thesis. Our findings are based on evidence to support any claims made. The contributions are both to the fields of radar and machine learning.

5.4.1 Findings and Contributions

Findings

Where is your evidence to support findings that you made in this thesis?

What are some interesting findings or things that were surprising? What challenges did you encounter? Where you able to overcome? What did this thesis highlight? Any trades performed? Any limitations or design constraints?

Contributions

What are the important and main contributions to the fields of radar and machine learning? Include performance....

- Radar simulation
- Data Collection
- CNN Model
- Data analysis and Interpretation
- Reviewer contributions

5.4.2 Future Work

- Apply to an actual real-world system. IRAD funding has been secured by my work place to continue development of these concepts and methods for ECM identification.

- Identification of false targets is proven successful. Move towards methods that prohibit false targets from detections processing in the receiver signal processing chain. Methods such as null steering, adaptive beamforming, waveform diversity, frequency hopping are techniques used by the radar community that could be expanded and exploited by using ML.

- Separating the real targets from the false targets is a real challenge since real along with false targets are appearing in the main-lobe of the beam and thus should be considered. The main-lobe of a radar beam is like a sinc function where $y(t) = \frac{sin(\pi t)}{\pi t}$ and $-\infty < t < \infty$. The detection decision is typically done by hypothesis testing using a Bayes optimization which determines the optimal choice between our hypotheses. In this case, I'm considering a multi-hypothesis approach for TBD features. We could try a particle filter approach which is a recursive, Bayesian state estimator that uses discrete particles to approximate the posterior distribution of the estimated state.

- Dive further into the mathematical theory behind ML to open up the black box.

Chapter 6

Appendix A - Source Code

6.1 MATLAB Source Code

The MATLAB source code that follows is used for the radar simulation that generates the IQ data and Range-Doppler images which serve as inputs into the CNN model.

```
function [opts,M] = getParameters(filename,type)
% DESCRIPTION: This function defines the parameters for the phasedArrayRadar
% simulation.  There are two types of structures defined which are for the
% target and the radar.
% INPUTS:  filename:  string  (Excel spreadsheet file)
%          type:  string  (types - 'radar' or 'targets')
% EXAMPLE: [~,tgt] = getParameters(tgtFile,"targets");
%          [~,radar] = getParameters(radarFile,"radar");
% See Also:  phasedArrayRadar.m

opts = detectImportOptions(filename);
preview(filename,opts)
M = readmatrix(filename,opts);

switch type
    case "targets"
        target = struct;
        target.id = M(:,1);
        target.pos = M(:,2);
        target.vel = M(:,3);
        target.rcs = M(:,4);
        target.type = M(:,5);
        target.pw = M(:,6);
        M = target;
    case "radar"
        radar = struct;
        radar.pwEnum = M(:,1);
        radar.pw = M(:,2);
        radar.duty = M(:,3);
        radar.fc = M(:,4);
        radar.pri = M(:,5);
        radar.prf = M(:,6);
        radar.fs = M(:,7);
        radar.wb = M(:,8);
        radar.nPulses = M(:,9);
        radar.nDoppler = M(:,10);
        radar.peakPwr = M(:,11);
        radar.txGain = M(:,12);
        radar.rxGain = M(:,13);
        radar.noiseFigure = M(:,14);
        radar.minTgtRng = M(:,15);
        radar.maxTgtRng = M(:,16);
        radar.minVel = M(:,17);
        radar.maxVel = M(:,18);
        radar.deltaVel = M(:,19);
        radar.deltaRange = M(:,20);
        M = radar;
    otherwise
        disp("Invalid type use 'radar' or 'targets'.")
end
end
```

Figure 6.1: getParameters.m

Chapter 7

Appendix B - Random Notes (Delete this later)

- Matrix multiplication across layers adds a certain amount of instability
- The correct choice of an algorithm can save time and money by efficiently providing a better solution and thru ease of implementation.
- Latin Hypercube Sampling (LHS) is a way of generating random samples of parameter values. It is widely used in Monte Carlo simulation, because it can drastically reduce the number of runs necessary to achieve a reasonably accurate result.

Mathematical Assumptions

- CNN: Linear independence of the input features concerning the image data

- CNN: Lower dimensions will provide accurate resultsCNN.

- Assume that an optimal θ exists with no closed form solution. It important that we are optimizing only the loss function measurements, not the gradient measurements. Once optimized the gradient should equal zero.

- The solution corresponds to a vector of parameters where the gradient of the loss function $f(\theta^*) = \nabla L(\theta) =$ such that $\theta^* = \theta$ (values converge) with respect to the problem parameters being optimized. More than one point can converge (i.e. local minimum) we should allow a continued search for a global minimum solution.

- The data is i.i.d. Independent and identically distributed random variables

- $y(\theta) = L(\theta) + noise$, where L is the loss function and θ is a continuous p-dimensional vector of value parameters. If $noise = 0$ then exact loss function measurements are available and thus discrete. Otherwise we do not have exact function measurements due to noise but measurements are available for any θ value in either case.

- $L(\theta)$ is a differentiable function of θ. Gradient descent fails for non-differentiable functions. $L(\theta)$ is a differentiable function of θ. Gradient descent fails for non-differentiable function

- No direct measurements (with or without noise) for the gradient exist or hard to compute (requires knowledge of the relationship between the optimized parameters and the loss function), thus a reason why we can use gradient descent to approximate a numerical solution value. This is what makes a gradient descent algorithm useful! If the function is differentiable, $\nabla L(\theta) = 0$ can be typically solved if the formula is known, in this case we do not have that information available. We are finding the way as we go, so where we start is of great importance.

- Constrain or not to constrain the parameters?
- Implementing algorithms from scratch can be time consuming but if we use libraries without understanding then it still remains a black box for us.

Bibliography

[1] "BriteCloud DRFM (Digital RF Memory) countermeasure." [Online]. Available: https://electronics.leonardo.com/en/products/britecloud-3

[2] B. Luo and L. Liu, "Development of radar active jamming recognition technology," in *2019 2nd International Conference on Mechanical Engineering (MEIMIE*, 2019, p. 462.

[3] R. J. Wiegand, "Drfm patent." [Online]. Available: https://patents.google.com/patent/US4743905A/en

[4] Y. Zhang, B. Jiu, P. Wang, H. Liu, and S. Liang, "An end-to-end anti-jamming target detection method based on cnn," *IEEE Sensors Journal*, vol. 21, no. 19, pp. 21 817–21 828, 2021.

[5] S. Zhao, Y. Zhou, L. Zhang, Y. Guo, and S. Tang, "Discrimination between radar targets and deception jamming in distributed multiple-radar architectures," *IET Radar, Sonar & Navigation*, vol. 11, no. 7, pp. 1124–1131, 2017.

[6] M. Greco, F. Gini, and A. Farina, "Radar detection and classification of jamming signals belonging to a cone class," *IEEE transactions on signal processing*, vol. 56, no. 5, pp. 1984–1993, 2008.

[7] M. Pak and S. Kim, "A review of deep learning in image recognition," in *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*, 2017, pp. 1–3.

[8] W. M. Walter Pitts, "A timeline of machine learning history." [Online]. Available: https://www.techtarget.com/whatis/A-Timeline-of-Machine-Learning-History

[9] "The missile defense system." [Online]. Available: https://www.mda.mil/system/system.html

[10] M. I. Skolnik, *Introduction to Radar Systems.* McGraw-Hill, 2001, ch. 2.2, pp. 31–32.

[11]

[12] M. A. Richards, *Fundamentals of Radar Signal Processing.* McGraw-Hill, 2005, ch. 6.1.1, pp. 297–298.

[13] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[14] 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. [Online]. Available: https://www.tensorflow.org/

[15] L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, and H. H. Olsson, "Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions," *Information and Software Technology*, vol. 127, p. 106368, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584920301373

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016, p. 174.

[17] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning.* Springer, 2006, vol. 4, no. 4, p. 227.

[18] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *nature*, vol. 405, no. 6789, pp. 947–951, 2000.

[19] A. F. Agarap, "Deep learning using rectified linear units (relu)," *CoRR*, vol. abs/1803.08375, 2018. [Online]. Available: http://arxiv.org/abs/1803.08375

[20] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” *CoRR*, vol. abs/1609.04836, 2016. [Online]. Available: http://arxiv.org/abs/1609.04836

[21] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” *Neural Networks: Tricks of the Trade: Second Edition*, pp. 437–478, 2012.

[22] Z. Hao, Y. Jiang, H. Yu, and H. Chiang, “Adaptive learning rate and momentum for training deep neural networks,” *CoRR*, vol. abs/2106.11548, 2021. [Online]. Available: https://arxiv.org/abs/2106.11548

[23] S. Vaswani, A. Mishkin, I. H. Laradji, M. Schmidt, G. Gidel, and S. Lacoste-Julien, “Painless stochastic gradient: Interpolation, line-search, and convergence rates,” *CoRR*, vol. abs/1905.09997, 2019. [Online]. Available: http://arxiv.org/abs/1905.09997

[24] F. Bach and E. Moulines, “Non-Asymptotic Analysis of Stochastic Approximation Algorithms for Machine Learning,” in *Neural Information Processing Systems (NIPS)*, Spain, 2011, pp. –. [Online]. Available: https://hal.science/hal-00608041

[25] E. K. Ryu and S. Boyd, “Stochastic proximal iteration: a non-asymptotic improvement upon stochastic gradient descent,” *Author website, early draft*, 2014.

[26] P. Toulis, D. Tran, and E. Airoldi, “Towards stability and optimality in stochastic gradient descent,” in *Artificial Intelligence and Statistics.* PMLR, 2016, pp. 1290–1298.

[27] “Quasi-Newton Methods.” [Online]. Available: https://optimization.cbe.cornell.edu/index.php?title=Quasi-Newton_methods

[28] R. H. Byrd, J. Nocedal, and R. B. Schnabel, “Representations of quasi-newton matrices and their use in limited memory methods,” *Mathematical Programming*, vol. 63, no. 1-3, pp. 129–156, 1994.

[29] B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” *Ussr computational mathematics and mathematical physics*, vol. 4, no. 5, pp. 1–17, 1964.

[30] M. Assran and M. Rabbat, “On the convergence of nesterov’s accelerated gradient method in stochastic settings,” *arXiv preprint arXiv:2002.12414*, 2020.

[31] C. Liu and M. Belkin, “Accelerating sgd with momentum for over-parameterized learning,” *arXiv preprint arXiv:1810.13395*, 2018.

Vita

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.