Spring 5-2022

Machine Learning Classification of Digitally Modulated Signals

James A. Latshaw
Old Dominion University, jlats001@odu.edu

MACHINE LEARNING CLASSIFICATION OF DIGITALLY MODULATED SIGNALS

by

James A. Latshaw
B.S. May 2015, Liberty University

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

ENGINEERING - ELECTRICAL AND COMPUTER

OLD DOMINION UNIVERSITY
May 2022

Approved by:

Dimitrie C. Popescu (Director)

Jiang Li (Member)

Otilia Popescu (Member)

ABSTRACT

MACHINE LEARNING CLASSIFICATION OF DIGITALLY MODULATED SIGNALS

James A. Latshaw
Old Dominion University, 2022
Director: Dr. Dimitrie C. Popescu

Automatic classification of digitally modulated signals is a challenging problem that has traditionally been approached using signal processing tools such as log-likelihood algorithms for signal classification or cyclostationary signal analysis. These approaches are computationally intensive and cumbersome in general, and in recent years alternative approaches that use machine learning have been presented in the literature for automatic classification of digitally modulated signals. This thesis studies deep learning approaches for classifying digitally modulated signals that use deep artificial neural networks in conjunction with the canonical representation of digitally modulated signals in terms of in-phase and quadrature components. Specifically, capsule networks are trained to recognize common types of PSK and QAM digital modulation schemes, and their classification performance is tested on two distinct datasets that are publicly available. Results show that capsule networks outperform convolutional neural networks and residual networks, which have been used previously to classify signals in the same datasets, and indicate that they are a meaningful alternative for machine learning approaches to digitally modulated signal classification. The thesis includes also a discussion of practical implementations of the proposed capsule networks in an FPGA-powered embedded system.

I would like to dedicate this work to my wife, Elizabeth, to my family and to my church family.

ACKNOWLEDGEMENTS

I would like to acknowledge Dr. Dimitrie Popescu, Dr. Chad Spooner, and soon to be Dr. John Snoap for all of their guidance and direction during my research. I would also like to acknowledge and thank Dr. Jiang Li and Dr. Otilia Popescu for being on my committee and for the valuable feedback that they provided. I would like to acknowledge and thank the entire WAHAB high performance cluster team for all of their help which was essential in training and testing the various networks. I would also like to thank Lori Barret for all of her administrative coordination and for the ODU Dean's office for their editing services.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

Information transmission bestows an intrinsic value on the ability to accurately determine a given transmitted message based on what has been received. In modern times, data is in a digital form; thus, modern communication systems focus on digital information transmission. Data is chosen by the transmitting user and is entirely unknown to the receiving user. The received data may be corrupted by noise or carry with it unwanted distortions all of which can adversely impact the receiver's ability to accurately determine what was transmitted based on the noisy received message. This ability to take an unknown and determine what was intended in spite of the presence of noise is of utmost importance; without it, there would be minimal confidence in that communication channel.

Similarly, there is great value in being able to correctly determine by which digital modulation scheme a given message is being transmitted. A digital modulation scheme takes this incoming digital information and encodes it as symbols which represent one or more binary bits. This stream of incoming symbols is then mapped to physical analog pulses which are transmitted over the channel of choice. The receiver is expected to detect the noisy and possibly distorted pulses and map each one back to a symbol which represents the digital information. There are a variety of digital modulation schemes which a transmitter may employ. If the digital modulation scheme is unknown then the receiver must determine the modulation scheme based on the noisy and distorted received pulses. Due to the variety of digital modulation schemes, and the possibility of noise, there is a need to be able to accurately determine the modulation scheme based on the received analog pulses.

There are many instances where a given digital modulation scheme may be unknown such as in electronic warfare applications, military or espionage applications where the receiver wishes to remain anonymous, or in bandwidth interference recognition, etc.

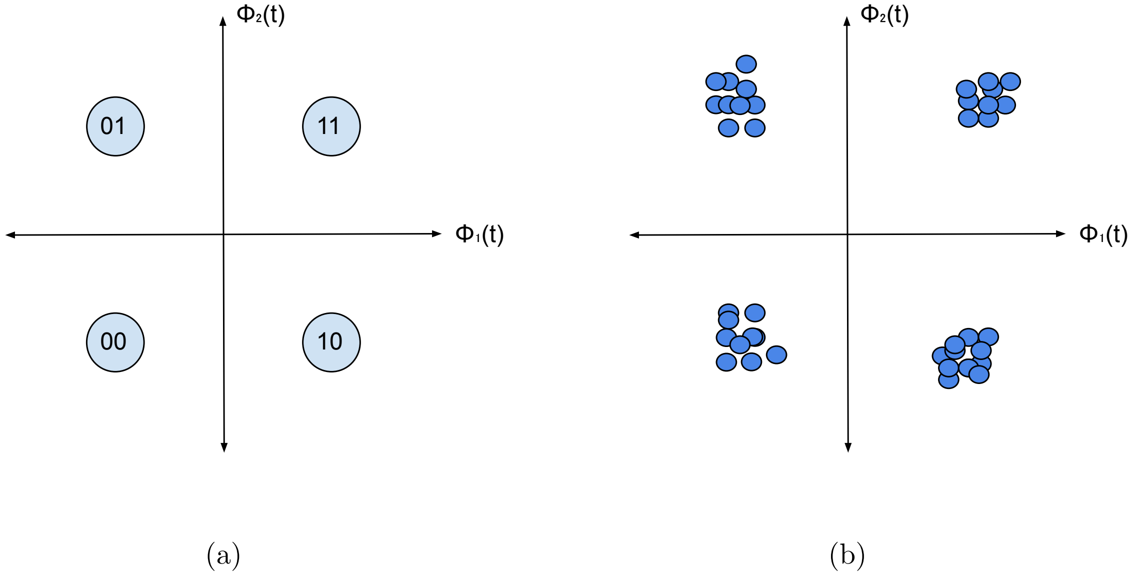
This ability to determine the unknown modulation type based solely on the received message poses an interesting problem. The message data being transmitted is unknown, and the modulation scheme being used is also unknown. If a receiver monitors a channel for a period of time, they may be able to accurately determine the parameters of the modulation scheme by means of signal processing techniques such as Cyclostationary Signal Processing (CSP). CSP approaches attempt to discover statistical properties that repeat on regular intervals in time and frequency. This signal processing is applied to the received analog pulses and results in a set of estimates of various features in the received data. These received features can be used for bandwidth estimation, in-band signal to noise estimation or even digital modulation scheme estimation.

This work applies machine learning to the problem of modulation classification in attempts to achieve high accuracy in modulation classification despite the message data being unknown and despite the presence of noise in the message data. The hope will be that machine learning techniques may be used to perform the same type of digital modulation recognition as CSP. Machine learning excels at recognizing patterns that are provided in the training data. By using sequences of received labeled data, an artificial neural network may be used to achieve excellent classification accuracy which will be the primary topic of this paper.

1.1 MODULATION CLASSIFICATION

A digital communication system has the goal of efficiently conveying a message signal from the transmitter to the receiver. The transmitter will employ a modulation scheme that is suited for its data. This data or message signal is deconstructed into symbols which can be thought of as groups of binary values. Simpler modulation schemes may have as few as two symbols, encoding a binary one or a binary zero. More complex modulation schemes may have 256 symbols (or more) with each symbol encoding a unique 8 bit value (or larger). More discussion of specific modulation schemes will be presented in Chapter 2, but for now it is sufficient to note that there are many modulation schemes. Once the transmitter decides on its modulation scheme it will begin to send its data over a channel that will reach the receiver. If the receiver knows the modulation scheme, then the receiver will detect the symbols and discover the data the transmitter has sent. If, however, the modulation scheme is unknown to the receiver then it will be unable to detect the symbols. In such a case, it is necessary for a receiver to be first capable of determining the modulation scheme.

The goal of modulation recognition is to accurately determine what specific modulation scheme is being used to transmit an arbitrary sequence of data. For this, the received signal is converted into its in-phase and quadrature signal components, which are orthogonal and can be used to completely reconstruct the received signal. These signal components form basis functions which can be used to recognize the time varying signal characteristics. For this Cartesian coordinates are used with the horizontal axis corresponding to the in-phase basis function and the vertical axis corresponding to the quadrature signal basis function.



(a)

(b)

FIG. 1: (a) An example of a modulation scheme with 4 symbols, each symbol having 2 bits. (b) An example of received data over a period of time (excursions are omitted for clarity).

Of the various modulation schemes, many symbols reside in similar locations. For this, additive white Gaussian noise (AWGN) will corrupt the received signal. With the corruption caused by the AWGN, the differences in these symbols becomes difficult to distinguish. Furthermore, the in-phase and quadrature signal components are time varying and capture the transition from one symbol to another which is called the excursion. Ideally these excursions will be a straight line moving directly from one symbol to another; however, they often are indirect wavy lines which only adds to the complexity. Regardless, these excursions may cross over coordinates that are a valid symbol for a given modulation scheme but not necessarily for the employed modulation scheme.

The receiver has no inclination as to what data the transmitter might be sending; if it did, there would be no point in transmitting it. Thus, the specific sequence of symbols is in no particular order and may transition from one symbol to another in a pseudo random fashion. However, if the receiver monitors the channel for a sufficient period of time, the majority of symbols may have been observed in this window of time. In Fig. 1b, an example of received data is shown with the excursions being omitted for clarity. In this case, the constellation can be clearly seen and the modulation scheme may be classified (there will be more discussion of specific modulation schemes in a later chapter).

There are various means of modulation classification in existence. One such approach is based on cyclostationary signal processing (CSP) [1] and involves using complicated algorithms that leverage statistical properties of the message signals power spectrum over an observed window of time in order to discover cyclic trends which may be used to classify the modulation type. Another method is the maximum likelihood (ML) or log likelihood options which use conditional statistical properties of the message signal and Bayes theorem to determine what the most likely modulation scheme is given the observed message data [2], [3], [4]. Currently, machine learning techniques involving artificial neural networks are considered to address this problem.

1.2 MACHINE LEARNING

Machine learning is a very broad topic encompassing many fields. In general machine learning is any algorithm or automated process wherein a computer sorts through data, acquires information from said data and then uses this information to make an informed decision. In many applications the objective is classification of data. For these models a set of predetermined classes are established, and the objective is to take an input value and map it to one of these classes. A simple two class example of this would be to take an unknown value and decide which class it belongs to. In Fig. 2, we have a two class example. The red dots belong to one class, and the blue dots belong to the other class. A certain number of examples of both classes would need to be available beforehand. When a new unknown dot is given, the coordinates of this dot will be compared against the coordinates of the other dots with respect to their class. Bayes Theorem will be applied and the conditional probabilities will be assessed to classify this new dot.

Such models work well when the data is linearly separable and each class is distinct. Linearly separable means that a line can be clearly drawn between the two classes. This requirement can be harder to maintain when there are many classes and for high dimensional inputs. An additional problem with the model described above is when the classes are not distinct, or when there is an overlap between classes. The data points along the border between the two classes will be more likely to be miss-classified. In both cases, these are nonlinearities introduced in the observed sample data.

In recent years a sub field of machine learning known as deep learning (DL) has gained great popularity due to its ability to overcome such nonlinearities in sample data. These models leverage multiple layers of neural networks with nonlinear activation functions to overcome nonlinearities present in the available data. A single example neuron is shown in Fig. 3. These networks will be discussed further in a later chapter, but as an introduction,

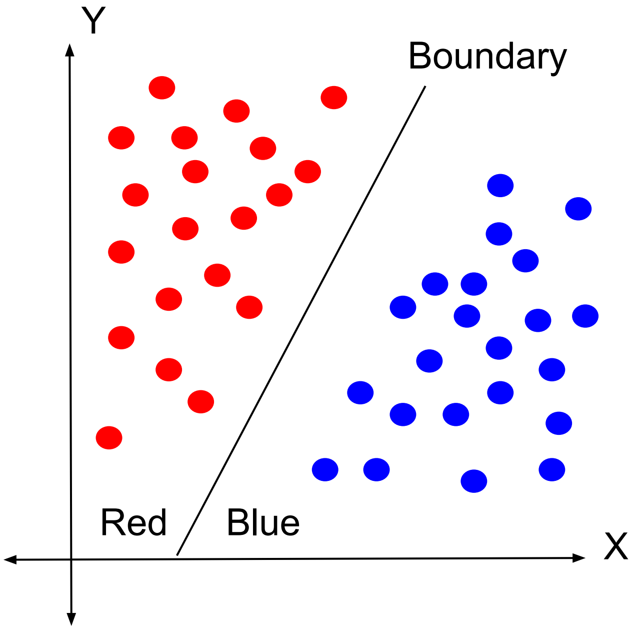


FIG. 2: An example of two class data for traditional machine learning.

these neural networks consist of weights and bias parameters which can be changed. These networks require a training process where the example data is referred to as the training data. This training data is used multiple times, and with each iteration the network will update its parameters to maximize the classification accuracy.

Though powerful and able to overcome nonlinear features in training data, these networks require a large amount of training data that sufficiently and completely represent all underlying characteristics of the respective classes. Furthermore, these networks are sensitive to even small perturbations of the input data, such as rotation, scaling and inversion. They are also sensitive to being over trained, a situation which can occur when the network is able to memorize the training data. An over-trained network may achieve high classification accuracy during training data, but when new data is provided it will perform poorly. These topics and others will be discussed further in a later chapter.

Well trained neural networks are also able to filter out or ignore noisy features in training data. Here, noise can be thought of as minor perturbations in the sample data that do not represent any interesting feature. Due to their ability to excel at classification of nonlinear training data, neural networks appear to be an excellent candidate for the topic of digital

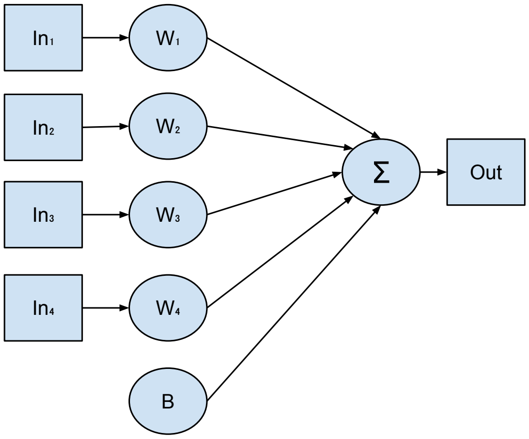


FIG. 3: An example of a single fully connected neuron without an activation function.

modulation classification. Specifically, their ability to recognize class features while ignoring noise in the digital signal will be very valuable.

1.3 PROBLEM STATEMENT

Ideally, a transmitter and receiver pair will always be cognizant of what digital modulation scheme is being used to transmit data. In reality, there are various situations where the receiver may not know which digital modulation scheme is being implemented. One prominent example of this is spectrum interference detection. In this case, the transmitter may unknowingly encroach on a neighboring spectrum. Identifying the digital modulation scheme would be helpful in determining where the interference originates. Thus, there is a need to identify the digital modulation scheme by means of the in-phase and quadrature components of the received digitized signal.

The main contribution of this thesis is the application of capsule networks for classification of digitally modulated signals and to compare the capsule network's performance to convolutional and residual networks for digital modulation classification. Other works employing CNN and residual network topologies can be found in [5, 6, 7, 8]. The work in this thesis has been submitted for presentation at IEEE Communication Conference COMM2022 [9].

1.4 THESIS OUTLINE

This section will provide an overall outline of the remainder of the text. The goal of these first few chapters will be to provide a reasonable background to pertinent information needed in order to properly understand the subject matter, and later chapters will present and assess performance.

- **Digital Modulation:**

  Chapter 2 will present the concept of a band-pass signal and what the quadrature and in-phase signal components are and how they may be obtained from a band-pass signal. This will be instrumental in understanding how the modulation schemes are represented as well as what the inputs to the neural network will look like. Additional topics such as signal to noise ratio (SNR) will also be discussed as these will be useful in evaluating the performance of the network.

- **Machine Learning for Feature Activation and Detection:** Chapter 3 will present neural networks in more detail, describe what their inputs look like, how they are trained and how classifications are performed. Then these concepts will be expanded to several prominent network topologies which are used for a wide variety of applications.

- **Deep Learning for Digital Modulation Classification:** Chapter 4 will discuss several network topologies for the purpose of digital modulation classification. Details on these network architectures will be provided.

- **Supervised Training:** Chapter 5 will present two training datasets which will be used for training and evaluating this network's performance. Details on the datasets as well as an assessment of the training data will be provided in this section. This section will also discuss the software tools and training setup that will be used to train the network and assess the results of the trained network.

- **Simulation Results and Analysis:** Chapter 6 will focus on the results for the trained network for each respective dataset. These results will be compared to each other and an analysis will be provided.

- **FPGA Implementation:** Chapter 7 will discuss considerations needed to implement the proposed network onto an embedded system.

- **Conclusions and Future Directions:** Chapter 8 will summarize and conclude the work of this thesis and provide thoughts for future work.

CHAPTER 2

DIGITAL MODULATION

The goal of this chapter will be to discuss digital modulation schemes and how these schemes are represented by in-phase and quadrature signal components. Furthermore, a description of the canonical bandpass signal representation will be provided as well as an explanation of how the in-phase and quadrature signal components may be extracted from a received bandpass signal. The focus will be on the digital modulation schemes that are commonly used in practical systems. The signal to noise ratio will also be discussed as this is an important metric in determining the performance of a digital modulation classifier. It is important to note that [10] and [11] were instrumental in the author's education on this topic.

2.1 CANONICAL BANDPASS SIGNALS

Modulation can be used to move a low frequency content message signal to a higher frequency where it can be broadcast via radio frequency (RF) electromagnetic waves using electric circuitry. This message signal can take many forms such as a high or low pulse, an audio waveform, etc. The simplest type of message signal would be a single frequency. This single frequency can be referred to as a tone and is represented in the time domain as a sinusoidal waveform. This message signal is a low frequency signal with a bandwidth equal to its highest frequency content. In the case of a tone, $f_{tone} = w$, where f_{tone} is the tone, a single frequency message signal, and w is the bandwidth. This bandwidth is the range of frequency occupied by the message signal which is also referred to as the spectrum of the signal. This is a lowpass/baseband signal because its spectrum is located around the zero frequency. An example of this is shown in Fig. 4. Note that this illustration includes an example of a tone message signal as well as a generalized arbitrary waveform.

Regardless of whether the message signal is a tone or an arbitrary waveform, the bandwidth is still represented as w. The carrier frequency, f_c is typically much larger than f_{tone}. Once the message signal is modulated with the carrier frequency, the signal is now considered a bandpass signal. Assessing this process from the frequency domain, the bandwidth of the message signal is shifted up so that the center of the bandwidth w resides at the center of the carrier frequency f_c. This is illustrated in Fig. 5.

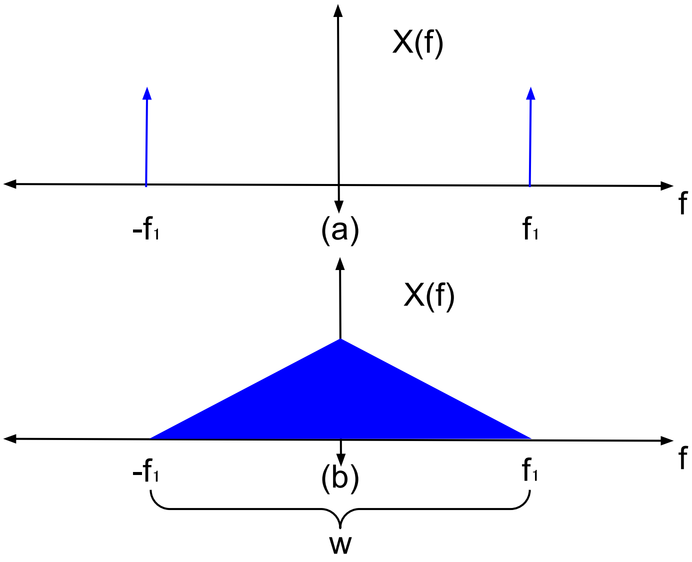


FIG. 4: (a) An example of the spectrum of a tone message signal. (b) An example of the spectrum of a lowpass/baseband signal with some arbitrary waveform. Note that its spectrum is centered around the zero frequency.

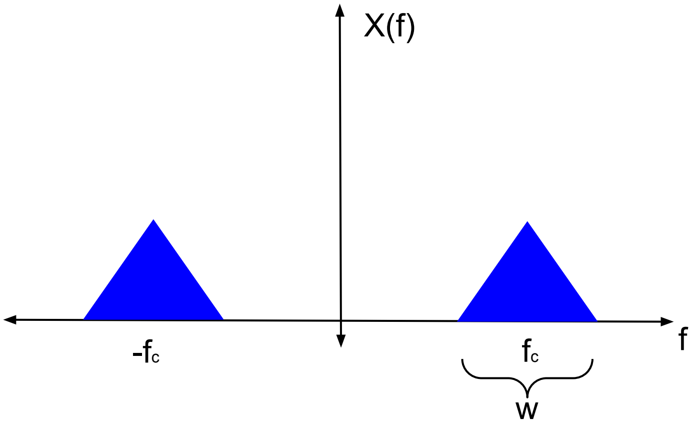


FIG. 5: An example of the spectrum of a bandpass signal with some arbitrary waveform.

The spectrum of the bandpass signal shown in Fig. 5 is a good representation of an ideal spectrum as it leaves the transmitter. This spectrum is represented as $X(f)$ and the inverse Fourier transform of this gives us $x(t) = \mathcal{F}^{-1}\{X(f)\}$. This is the time domain representation of the bandpass signal. If the message signal is some slowly varying signal, whose variations coincide with the information being transmitted, then this signal can be represented as $A(t)$. Due to the modulation with the carrier f_c, and assuming $-\pi \leq \theta(t) \leq \pi$ the bandpass signal may be represented as shown in equation (1).

$$x(t) = A(t)\cos(2\pi f_c t + \theta(t)) \tag{1}$$

Which, noting that $a\cos x = ae^{jx}$, can be represented in phasor form as shown in equation (2).

$$x(t) = A(t)e^{j2\pi f_c t\theta(t)} \tag{2}$$

The equation shown in (2) has real and imaginary components. To look at only the real components, equation (2) may be rewritten as shown in equation (3).

$$\begin{aligned} x(t) &= Re[A(t)e^{j2\pi f_c t\theta(t)}] \\ &= A(t)\cos(\theta(t))\cos(2\pi f_c t) - A(t)\sin(\theta(t))\sin(2\pi f_c t) \\ &= x_c(t)\cos(2\pi f_c t) - x_s(t)\sin(2\pi f_c t) \end{aligned} \tag{3}$$

In the above equation (3), the $x_c(t)$ term corresponds to the in-phase signal component and the $x_s(t)$ term corresponds to the quadrature signal component. Another method for achieving the results of (3) is by taking the time domain representation of the bandpass signal and summing it with the Hilbert transform of $jx(t)$. This is known as the analytic or pre-envelope signal. This pre-envelope signal is represented as x_+ and is given as shown in equation (4).

$$\begin{aligned} x_+(t) &= x(t) + \mathcal{H}\{jx(t)\} \\ &= x(t) + j\hat{x}(t) \end{aligned} \tag{4}$$

It is clear from the above equation that $x(t) = Re[x_+(t)]$. The end goal is to have a convenient signal for describing the in-phase and quadrature components of a baseband signal. This can be achieved by the complex lowpass equivalent of the base band signal which is denoted as $\tilde{x}(t)$ and is given in equation (5).

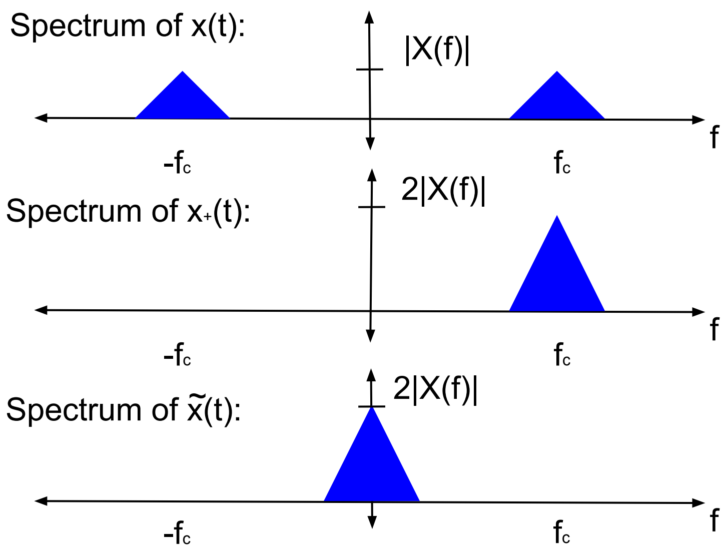


FIG. 6: Illustrations of the bandpass spectrum of $x(t)$, Pre-envelope spectrum of $x_+(t)$, and complex lowpass equivalent spectrum of $\tilde{x}(t)$

$$\tilde{x}(t) = x_c(t) + jx_s(t) \tag{5}$$

Now that (5) has been described, the pre-envelope signal may be rewritten in terms of the complex lowpass equivalent as shown in equation (6).

$$x_+(t) = \tilde{x}(t)e^{2\pi f_c t} \tag{6}$$

As established in (3), the baseband signal $x(t)$ may be written as the real part of the pre-envelope signal. Using equation (6), the baseband signal may be rewritten as shown in equation (7).

$$\begin{aligned} x(t) &= Re[x_+(t)] \\ &= Re[\tilde{x}(t)e^{j2\pi f_c t}] \\ &= x_c(t)\cos(2\pi f_c t) - x_s(t)\sin(2\pi f_c t) \end{aligned} \tag{7}$$

An important note about equation (7) is that it enables the bandpass signal to be represented in terms of the in-phase and quadrature components which are $x_c(t)$ and $x_s(t)$ respectively. Sometimes the in-phase component is also denoted as $x_I(t)$ and the quadrature

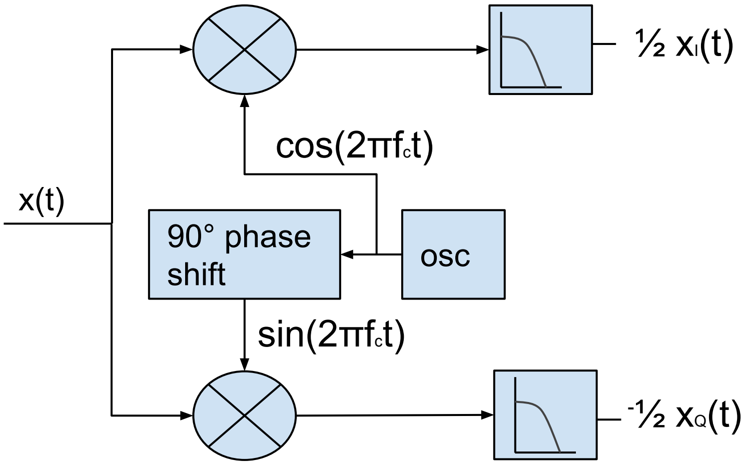


FIG. 7: A bandpass signal analyzer circuit. This is used to extract $x_I(t)$ and $x_Q(t)$ from $x(t)$.

component as $x_Q(t)$. Fig. 6 illustrates what the spectrum of $X(f)$, $x_+(t)$, and $\tilde{x}(t)$ are and how $\tilde{x}(t)$ represents a scaled version of desired message signal at baseband.

Inspired by equations (3) and (7), a signal analyzer circuit can be constructed in order to extract the in-phase and quadrature signal components from the bandpass signal. This circuit would appear at the receiver and would require that a local oscillator generate a sinusoidal signal. The baseband signal $x(t)$ will branch into two parts. The upper branch will be directly multiplied with the local oscillator. The lower branch will be multiplied by a $-90°$ phase shifted version of the oscillator. It can be noted that the Hilbert transform introduces a $-90°$ phase shift to sinusoidal signals. This illustration of the signal analyzer is shown in Fig. 7.

Working this out in equations (8) and (9), the result has the desired in-phase and quadrature components with some high frequency signal components. By using a low pass filter (LPF) at the output of the upper and lower branch, these high frequency components will be suppressed leaving only the desired in-phase and quadrature signal components. It is important to note that perfect synchronization between the transmitter and receiver is assumed in this ideal case. If perfect synchronization is not achieved, then this may introduce message signal attenuation or other undesirable effects.

$$\begin{aligned}
UpperBranch &= LPF[x(t)\cos(2\pi f_c t)] \\
&= LPF[(x_c(t)\cos(2\pi f_c t) - x_s(t)\sin(2\pi f_c t))\cos(2\pi f_c t)] \\
&= LPF[\frac{1}{2}x_c(t) + \frac{1}{2}x_c(t)\cos(4\pi f_c t) - \frac{1}{2}x_s(t)\sin(4\pi f_c t)] \\
&= \frac{1}{2}x_c(t) \\
&= \frac{1}{2}x_I(t)
\end{aligned} \tag{8}$$

Similarly for the lower branch, it can be shown that it results in the quadrature signal component.

$$\begin{aligned}
LowerBranch &= LPF[x(t)\sin(2\pi f_c t)] \\
&= LPF[(x_c(t)\cos(2\pi f_c t) - x_s(t)\sin(2\pi f_c t))\sin(2\pi f_c t)] \\
&= LPF[\frac{1}{2}x_s(t) - \frac{1}{2}x_s(t)\cos(4\pi f_c t) - \frac{1}{2}x_s(t)\sin(4\pi f_c t)] \\
&= -\frac{1}{2}x_s(t) \\
&= -\frac{1}{2}x_Q(t)
\end{aligned} \tag{9}$$

Now that the steps in equations (8) and (9) have been worked out, it is clear that the signal analyzer in Fig. 7 does produce scaled versions of the in-phase and quadrature components.

2.2 IN-PHASE AND QUADRATURE REPRESENTATION

Now that background has been provided as to how the in-phase and quadrature signal components can be extracted from a bandpass signal using a signal analyzer, specific focus can be given to these signal components and how they can be used to reconstruct the original message signal.

Equation (5) may be rewritten in polar coordinates. To do this, the envelope and argument must be determined. The envelope, or magnitude, is as shown in equation (10).

$$|\tilde{x}(t)| = \sqrt{x_I(t)^2 + x_Q(t)^2} \tag{10}$$

Furthermore, the argument or phase can be given as shown in equation (11).

$$\angle\tilde{x}(t) = \theta(t) = \arctan(\frac{x_Q(t)}{x_I(t)}) \tag{11}$$

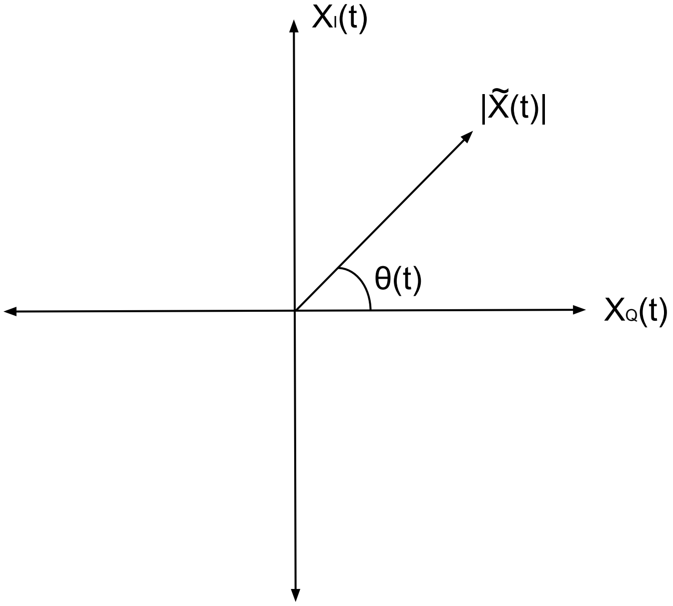


FIG. 8: An example of in-phase and quadrature coordinate system.

Combining the above, equation (5) is rewritten in polar coordinates as shown in equation (12).

$$\tilde{x}(t) = |\tilde{x}(t)|\angle\tilde{x}(t) \tag{12}$$

These components can be used to completely reconstruct the amplitude and phase of the original message signal at the receiver. The in-phase and quadrature components are orthogonal to each other and may be plotted using Cartesian coordinates. Conventionally, the in-phase component is denoted as the horizontal axis, and the quadrature component is denoted as the vertical axis. This is illustrated in Fig. 8.

Until now, the message signal has been arbitrarily described as some desirable, information carrying waveform. With regards to digital information transmission, a digital modulation scheme will be employed. These modulation schemes will focus on the transmission of symbols which correspond to specific binary value(s). These symbols will be represented on a two dimensional Cartesian coordinate plot in terms of basis functions chosen specifically for that modulation scheme. A subtle but important distinction is that the in-phase and quadrature components are reconstructing the instantaneous message signal waveform

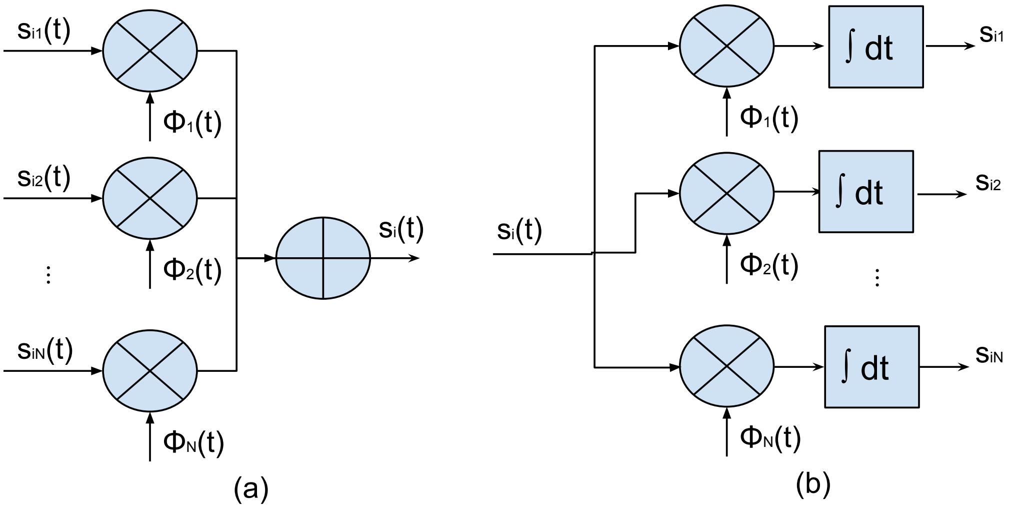


FIG. 9: (a) Illustration of a synthesizer for generating energy signals. (b) Illustration of an analyzer used for constructing a signal vector, $\boldsymbol{s_i}$.

whereas analysis with basis functions focus on symbol detection by assessing the signal energy of quadrature carriers. This distinction is provided as the machine learning networks will be utilizing the in-phase and quadrature signal components as inputs into the networks. These differences, as well as a description of the digital modulations schemes used in the datasets, will be elaborated upon in the next section.

2.3 SIGNAL CONSTRUCTION FOR DIGITAL MODULATION SCHEMES

Digital modulation uses symbols to represent strings of binary data that the transmitter desires to send to a receiver. It is said that M energy signals are defined for any given digital modulation scheme with each energy signal being denoted as $s_i(t)$. There are also N orthonormal basis functions represented as $\phi_i(t)$
$N \leq M$. Any symbol can be described as a linear combination of these basis functions. This can be viewed as a signal vector $\boldsymbol{s_i}$. To do this, the transmitter employs a signal synthesizer as shown in Fig. 9. Any transmitted symbol will be some linear combination of the basis functions.

At the receiver, a signal analyzer as illustrated in Fig. 9 may be used to reconstruct the signal vector $\boldsymbol{s_i}$. The signal analyzer will apply a correlation with each respective basis function which will result in an energy value for each signal component in the signal vector

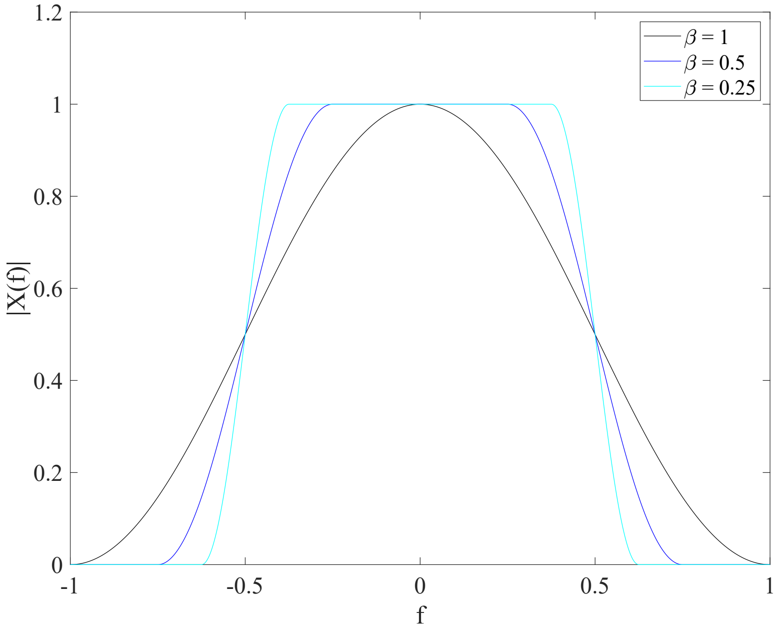


FIG. 10: Root-raise cosine pulse with various β rolloff factors and for $T = 1$.

$\boldsymbol{s_i}$. The coordinates of $\boldsymbol{s_i}$ will be the symbol estimate. This symbol estimate can be plotted on a Cartesian coordinate system using the basis functions as horizontal and vertical axis.

These energy signals have a duration of T seconds. Ideally, these signals could be a simple on/off square wave in the time domain. In the frequency domain, this would result in poor bandwidth utilization as a time domain rectangular pulse in a sinc function in the frequency domain, which occupies a wide spectrum. In attempts to have better utilization of the spectrum a root-raised cosine pulse is used. An example of this is illustrated in Fig. 10.

Due to the fact that noise is random and is uncorrelated with every other signal, the noise will be suppressed by the correlators shown in Fig. 9. The result of the correlation for a time bound pulse is given by equation (13).

$$s_j(t) = \int_0^T s_i(t)\phi_j(t)d\tau \tag{13}$$

If a similar correlation occurs with some arbitrary impulse function as shown in equation (14), then a simplification can occur for a carefully chosen $h_j(t)$.

$$x_j(t) = \int_0^T s_i(t)h_j(T-t)d\tau \tag{14}$$

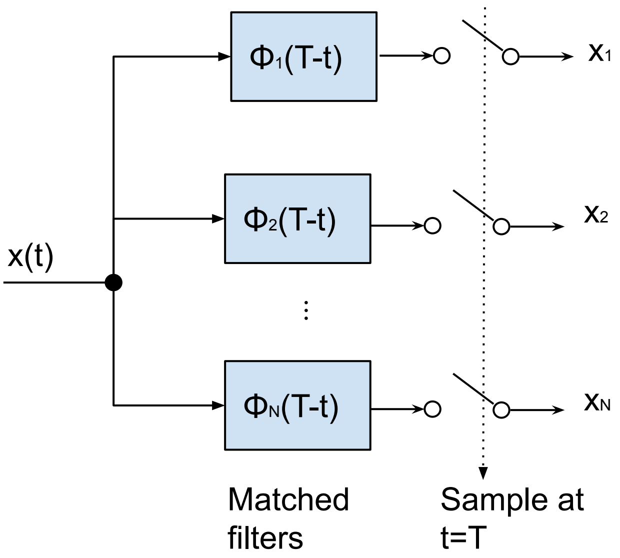


FIG. 11: Matched filter receiver.

The equation shown in (13) will be equivalent to (14) when the impulse response of $h_j(t)$ satisfies $h_j(t) = \phi_j(T-t)$. Thus, the matched filters may be used in place of banks of correlators. This is illustrated in Fig.11 and is referred to as a matched filter receiver.

Now that a sufficient background to the signal synthesizer and analyzers has been described, digital modulation constellations can be discussed. The first set of digital modulation constellations will be Phase-Shift Keying (PSK), the simplest of which is Binary PSK (BPSK). This uses $M = 2$ energy symbols with one basis function $N = 1$. This constellation is illustrated in Fig. 12. Here the pair of energy signals are described as shown in equation (15).

$$\begin{aligned} s_1(t) &= \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t) \\ s_2(t) &= \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t + \pi) = -\sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t) \end{aligned} \tag{15}$$

Here the basis function is given as shown in equation (16).

$$\phi_1(t) = \sqrt{\frac{2}{T_b}}\cos(2\pi f_c t) \tag{16}$$

The coefficients of the signal vector will ideally be as shown in equation (17).

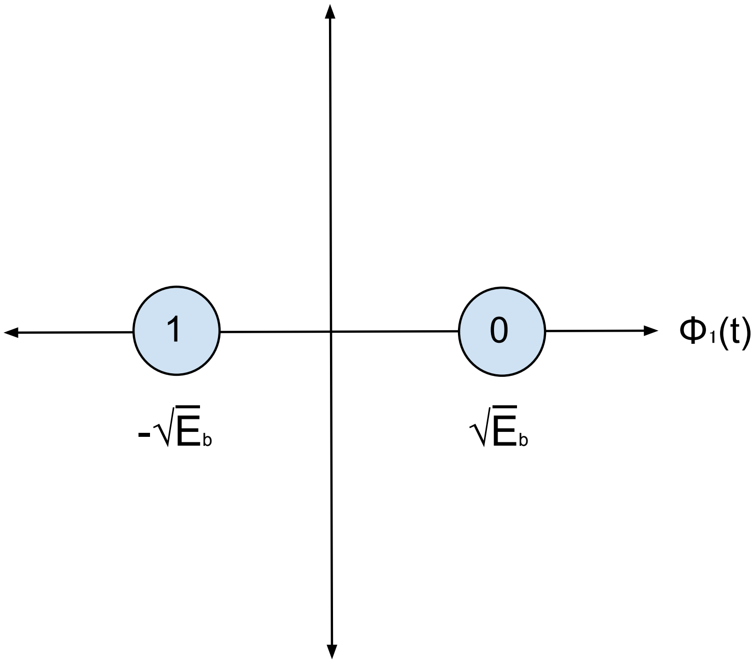


FIG. 12: A simple signal-space diagram of BPSK.

$$
\begin{aligned}
s_{11}(t) &= \int_0^{T_b} s_1 t \phi_1(t) dt = \sqrt{E_b} \\
s_{21}(t) &= \int_0^{T_b} s_2 t \phi_1(t) dt = -\sqrt{E_b}
\end{aligned}
\tag{17}
$$

Here, there are two symbol options with the decision boundary being the origin, where $\phi_1 = 0$. If the received energy is greater than 0, the received bit is assumed to correspond with symbol 1, otherwise symbol 2. Symbol 1 could correspond to a binary zero or low, and symbol 2 could correspond to a binary high or 1. For symbol 1, the received signal vector would be $\boldsymbol{s} = [\sqrt{E_b}, 0]^T$ and for symbol 2 it would be $\boldsymbol{s} = [-\sqrt{E_b}, 0]^T$.

By increasing N and M, additional constellations can be represented. For $N = 2$, $M = 4$, there is quadriphase PSK (QPSK) and for ($N = 2$, $M = 8$ there is octaphase PSK (8-PSK) constellations. These are as shown in Fig. 13.

For QPSK the transmitted signal energies are given by equation (18).

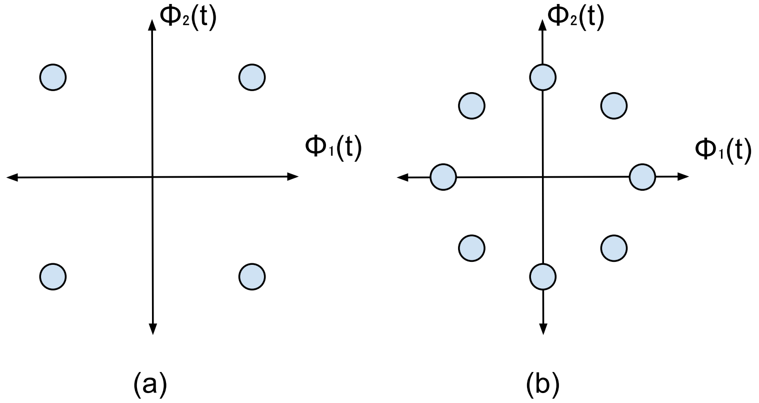


FIG. 13: (a) Signal space representation of QPSK. (b) Signal space representation of 8-PSK.

$$s_i(t) = \begin{cases} \sqrt{\frac{2E_b}{T_b}} \cos(2\pi f_c t + (2i-1)\frac{\pi}{4}), & \text{if } i = 1,2,3,4 \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

Unlike BPSK where only one carrier was used, for QPSK, quadrature carriers will be used which are given in equation (19).

$$\begin{aligned} \phi_1(t) &= \sqrt{\frac{2}{T_b}} \cos(2\pi f_c t) \\ \phi_2(t) &= \sqrt{\frac{2}{T_b}} \sin(2\pi f_c t) \end{aligned} \tag{19}$$

When the received message signal is passed through the signal analyzer it will ideally result in one of four possible message points. These message points can be defined by the two dimensional signal vector as shown in (20).

$$\boldsymbol{s_i} = \begin{bmatrix} \sqrt{E_b} \cos((2i-1)\frac{\pi}{4}) \\ -\sqrt{E_b} \cos((2i-1)\frac{\pi}{4}) \end{bmatrix}, i = 1,2,3,4 \tag{20}$$

This concept may be expanded for M-ary PSK (which encompasses 8-PSK. In this case, the same basis functions described in equation (19) are used; however, the M signal energies are described as shown in equation (21).

$$s_i(t) = \begin{cases} \sqrt{\frac{2E_b}{T_b}} \cos(2\pi f_c t + (2i-1)\frac{2\pi}{M}), & \text{if } i = 1,2,...,M \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

In addition to the PSK constellation shown above, there are also differential (DBPSK, DQPSK). For the differential PSK constellations, the same basis functions are used as shown for BPSK and QPSK respectively. However the way in which symbols are transmitted follows a different set of rules. For these differential schemes, the last bit and the phase of the current bit are used to inform what the intended transmit bit will be. Specifically for DBPSK, if the next bit to be transmitted is a 1, the symbol stays the same as what the last transmitted bit was, and if the new transmit bit is a 0, then the symbol bit changes. Furthermore, if the new symbol to be transmitted is a 1, the phase of the DPSK is unchanged and if the symbol to be transmitted is a 0, the phase is shifted 180^o from what it currently is. A similar set of rules is used for DQPSK except that the phase shift values are $0^o, 90^o, 180^o$, or -90^o. What is convenient about differential PSK is that synchronization between the receiver and the transmitter is not as important because the change in phase informs what the transmitted symbol is.

The next set of constellations to discuss are M-ary Quadrature Amplitude Modulation (QAM). These constellations use $N = 2$ quadrature carriers given as shown in equation (22) which are identical to those given in equation (19).

$$\begin{aligned} \phi_1(t) &= \sqrt{\frac{2}{T_b}} \cos(2\pi f_c t) \\ \phi_2(t) &= \sqrt{\frac{2}{T_b}} \sin(2\pi f_c t) \end{aligned} \tag{22}$$

QAM constellations are square constellations as shown in Fig. 14. QAM constellations experience both amplitude and phase modulation. The separation between any two adjacent QAM points is proportional to the square root of energy as shown in equation (23) with E being the message signal energy.

$$\frac{d_{min}}{2} = \sqrt{E} \tag{23}$$

From this, the M-ary message signal is as shown in equation (24). Here a_k and b_k are used to space adjacent points by a minimum of d_{min}.

$$s_k(t) = \sqrt{\frac{2E}{T}} a_k \cos(2\pi f_c t) - \sqrt{\frac{2E}{T}} b_k \sin(2\pi f_c t), k = 0, \pm1, \pm2, ... \tag{24}$$

For an M-ary constellation with an even number of bits per symbol, there will be $L = \sqrt{M}$. The value $\{a_i, b_i\}$, which is used to space the constellation points, is given as shown in equation (25).

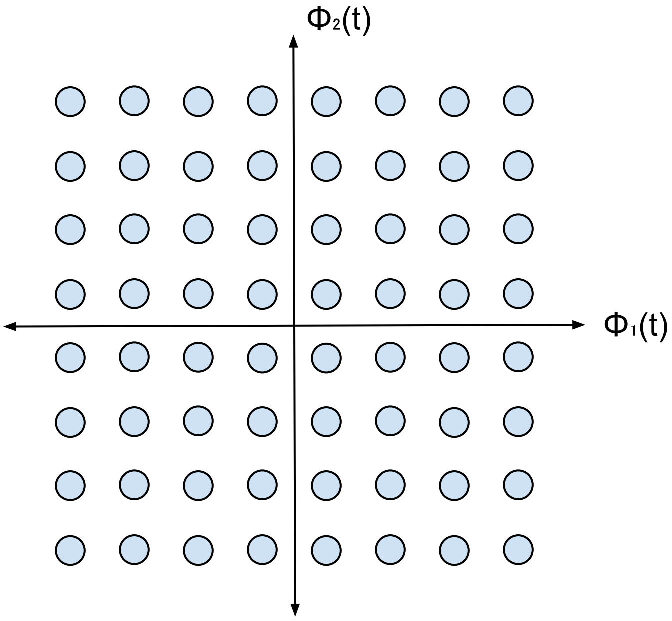


FIG. 14: Example of M-ary QAM, for M=64 constellation.

$$\{a_i, b_i\} = \begin{bmatrix} (-L+1, L-1) & (-L+3, L-1) & \dots & (-L-1, L-1) \\ (-L+1, L-3) & (-L+3, L-3) & \dots & (L-1, L-3) \\ \vdots & \vdots & \vdots & \vdots \\ (-L+1, -L+1) & (-L+3, -L+1) & \dots & (L-1, -L+1) \end{bmatrix} \tag{25}$$

These M-ary QAM constellations are easily extended to 16-QAM, 64-QAM, and 256-QAM which are used in the training data sets. These are very dense constellations, and it is expected that digital modulation classifiers may have difficulty distinguishing these three constellations.

The final constellation that will be considered is Minimum Shift Keying (MSK). MSK is a type of frequency shift keying. Unlike the previously discussed constellations which modulate amplitude and phase to represent different symbols, MSK uses energy signals that are given by equation (26).

$$s(t) = \sqrt{\frac{2E_b}{T_b}}\theta(t)\cos(2\pi f_c t) - \sqrt{\frac{2E_b}{T_b}}\theta(t)\sin(2\pi f_c t) \tag{26}$$

The basis functions are given as shown in equation (27).

$$\begin{aligned}\phi_1(t) =& \sqrt{\frac{2}{T_b}}\cos(\frac{\pi}{2T_b}t)\cos(2\pi f_c t)\\ \phi_2(t) =& \sqrt{\frac{2}{T_b}}\sin(\frac{\pi}{2T_b}t)\sin(2\pi f_c t)\end{aligned} \tag{27}$$

To evaluate the signal vector coefficients, different time intervals are used for MSK. For $s_1(t)$ the interval range is over twice T_b. To find $s_1(t)$, the correlation between the energy signal and the basis function are taken as shown in equation (28).

$$s_1(t) = \int_{-T_b}^{T_b} s(t)\phi_1(t)dt = \sqrt{E_b}\cos(\theta(0)), -T_b \leq t \leq T_b \tag{28}$$

In a similar fashion, to find $s_2(t)$ the correlation between the energy signal and the respective basis function is taken; however, the interval of integration is changed as shown in equation (29).

$$s_2(t) = \int_{0}^{2T_b} s(t)\phi_2(t)dt = \sqrt{E_b}\sin(\theta(T_b)), 0 \leq t \leq T_b \tag{29}$$

These results can be represented in a signal space constellation which appears identical to QPSK. As is clear from the above basis functions and signal energy definitions, the MSK is quite different form QPSK in that MSK works by sending binary symbols with a phase-pair (by changing $\theta(t)$).

Ultimately, the goal of presenting these constellations is to show the similarities and differences of the various constellations. This will help in interpreting the results of the digital modulation classifier. Though emphasis for detecting the various symbols was given, the classifier is not interested in the actual data being transmitted; rather, it is interested in recognizing the constellation and classifying it.

2.4 SIGNAL TRANSMISSION

Transmitted signals are distorted by the transmission channel and are corrupted by noise. In all real world applications there is some amount of noise accompanying the message signal. This noise is random and as such is not correlated with any message signal. Noise is modeled as a Additive White Gaussian Noise (AWGN) channel and is represented mathematically as shown in equation (30). The AWGN assumption is valid in most practical settings [1].

$$r(t) = s_i(t) + w(t) \tag{30}$$

In equation (30), $r(t)$ is the received signal, $s_i(t)$ is the estimate of the i^{th} symbol and $w(t)$ is the channel noise. As long as the power contained within the desired signal is significantly larger than the power of the AWGN channel noise, the transmitted symbol can be accurately detected.

AWGN is broadband, meaning it encompasses the entire spectrum $-\infty \leq f \leq \infty$ and has infinite average power. The ratio of the desired signal power to the noise power is referred to as the Signal to Noise Ratio (SNR) as shown in equation (31) and considers noise power only in the measured band.

$$SNR = \frac{\text{Signal Power}}{\text{Noise Power}} \tag{31}$$

However, the message signal only occupies a limited amount of bandwidth, usually smaller than the measured band. Only the noise corrupting the band of interest will impact the message signal. The band of interest, as defined by [12], is an interval in the spectrum with an elevated power spectral density (PSD) and with either side of this interval being flat. The power spectrum density is the average signal power at various frequencies. Fig. 5 contains an example of the spectrum of an arbitrary waveform. The noise that exists within the band of interest is used to define the in-band SNR. Only the in-band noise impacts the message signal; thus, equation (31) can be further refined to be as shown in equation (32) where N_0 is the noise density (often attributed to thermal noise), w is the band of interest and P_R is the received power. The N_0w term is achieved by filtering the noise PSD with a bandpass filter (BPF). An illustration in Fig. 15 is provided to help show what the noise equivalent bandwidth looks like graphically.

$$\textit{In-band SNR} = \frac{P_R}{N_0 w} \tag{32}$$

It can be expected that lower SNR values will result in inefficient data transmission due to errors in detection. As such, even the best digital modulation scheme will perform poorly for extreme in-band SNR values. This is due to the fact that, for low in-band SNR values, the actual energy symbol is indistinguishable from the noise. It is expected that digital modulation classifiers will have decreased performance for low in-band SNR values.

Some researchers pursuing the topic of digital modulation classification by means of deep learning methods such as [5, 6] use training datasets that characterize performance by the total SNR which corresponds to the average noise power in the observed bandwidth rather than the actual signal bandwidth. Only the in-band noise will impact symbol detection and as such the in-band SNR is considered a more accurate representation of the noise corrupting

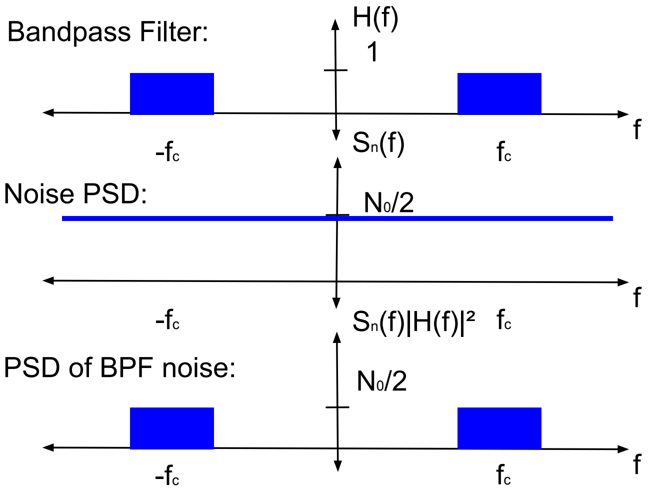


FIG. 15: An example of a bandpass filter with bandwidth w centered at f_c, the broadband noise power spectral density (PSD), and the noise equivalent bandwidth.

the message signal. The datasets presented in chapter 5 characterize performance based on the in-band SNR.

CHAPTER 3

MACHINE LEARNING FOR FEATURE ACTIVATION AND DETECTION

Often learning by example is one of the more effective ways of knowledge transfer. Evaluating the differences between items can be the best way to learn how to distinguish two distinct items from one another. Subtle indicators, that are unconsciously developed, inform a human's ability to mature their classification process. This classification development occurs over a long period of time after seeing many examples. Within the field of machine learning, there is a topic known as Deep Learning (DL) which uses Artificial Neural Networks (ANN) in various configurations to mimic a human's learning process. Deep learning is a subset of machine learning.

ANNs are very powerful and can be applied to complex classification problems. These types of networks are especially effective at adapting to nonlinear features in data as well as for working with noisy data. This chapter will focus on machine learning approaches in general as well as some helpful network typologies that may be applied to far more than just digital modulation classification problems. It is important to note that [13] and [14] were instrumental in the author's education with regards to this topic.

At its simplest component, an artificial neural network has a single neuron. This neuron is modeled as a summation of inputs multiplied by weights corresponding to each input, which are then summed together with some biasing term. This is shown in equation (33) with the a_j being the activation of the jth neuron. In this configuration w_{ji} corresponds to the jth weight for the ith input and w_{j0} is the bias term.

$$a_j = \sum_{i=1}^{n} w_{ji} x_i + w_{j0} \tag{33}$$

These activations are transformed by a non-linear, differentiable activation function. Many activation functions exist with new ones constantly being developed within the machine learning community. For now, this activation function will be denoted as $h(\cdot)$ and the activation equation for a single jth neuron with n inputs is given by (34). These are referred to as the hidden units.

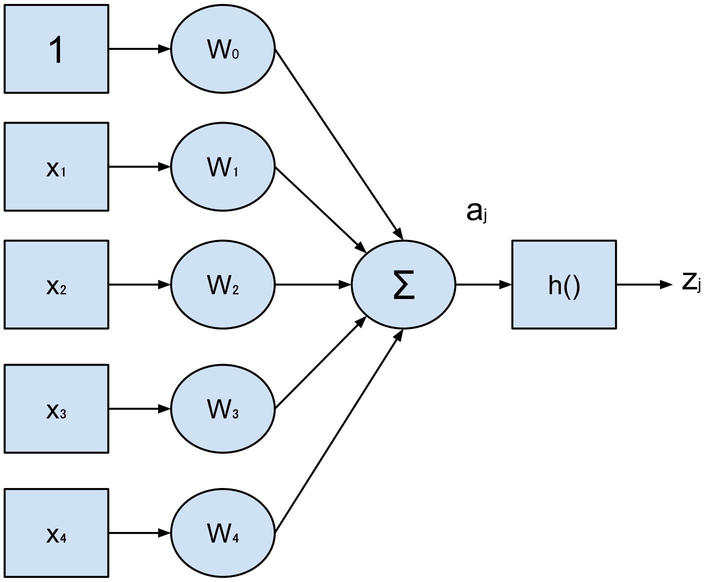


FIG. 16: An example of a single neuron with an activation function.

$$z_j = h(a_j) \tag{34}$$

For the sake of notation, it is often convenient to absorb the bias term into the weights by adding an additional constant driver (say 1) to each input layer and adding an additional weight which corresponds to this constant input. Mathematically, this is equivalent to having the bias term as shown in equation (33). This simplified notation is shown in equation (35).

$$a_j = \sum_{i=0}^{n} w_{ji} x_i \tag{35}$$

An illustration of a single neuron described above is given in Fig 16. The weights and bias are both values referred to as 'learnable parameters' or 'adaptive parameters.' These values can be updated so that for specific inputs a large activation will be returned. This is typically referred to as the neuron firing. This result is passed through the non-linear activation function which typically clips the output so that it is within a fixed range depending on the activation function used. Normally this is a value between 0 and 1.

Multiple neurons are grouped together in layers. Typically a fully connected topology is used, where all of the inputs of every neuron are connected to every input in the previous layer. These fully connected layers are very dense and require many weights.

Multiple layers of neurons can be cascaded together. Artificial neural networks are highly customizable, but usually at least two layers are needed in order to be considered deep learning. There is no limit to how many layers may exist within an ANN; however, large networks can be hard to implement as they require many weights and encounter training difficulties.

A major use of ANN is for classification problems. A classification problem involves taking an input value and grouping it into a predefined class. An example would be having a group of images of cats, dogs, cows and horses. The goal of the ANN would be to take a new picture of one of these creatures and group it with the class it most likely belongs to. The output of the network will inform which class the input most likely corresponds to.

3.1 ARTIFICIAL NEURAL NETWORK

The output layer for classification topologies is referred to as the classification layer. There are several classification layers available, but the most common is the softmax shown in equation (36), [15]. The goal of the classification layer is to help the user clearly see which class the input most likely corresponds to. The last layer in the ANN must have the same number of neurons as there are classes. For the previous example this would be four. The outputs of the last layer of neurons in the ANN can be represented as a vector $\boldsymbol{z}$. The output of the softmax will also be a vector represented as $\boldsymbol{\sigma}$. The softmax vector will have values between 0 and 1 that correspond to the probability that the input image matches the respective class. For example, if an input image of a cat was provided to a trained ANN, the element of $\boldsymbol{\sigma}$ that corresponds with cat would be the highest (ideally in the .99 to 1 range), and the elements that correspond to other animals would be low. The actual output of the network is typically denoted as a vector $\boldsymbol{y}$.

$$\boldsymbol{y} = \boldsymbol{\sigma}(\boldsymbol{z}) = \frac{e^{z_k}}{\sum_j e^{z_j}} \tag{36}$$

The inputs to the ANN can be anything, an image, a sequence of data, etc. They can be in any shape/format, but they are normally flattened to match the shape of the first layer in the ANN. Fig. 17 illustrates what a simple ANN might look like. Now that an overall description of the ANN has been provided, the process by which they are trained can be introduced.

When an input is fed into the network, the raw pixel or data values will be multiplied by the weights and passed through the activation function. This is known as forward propagation. This process will repeat for each layer. For the last layer, the neurons output into the classification layer where classification occurs. The weights of the network are typically

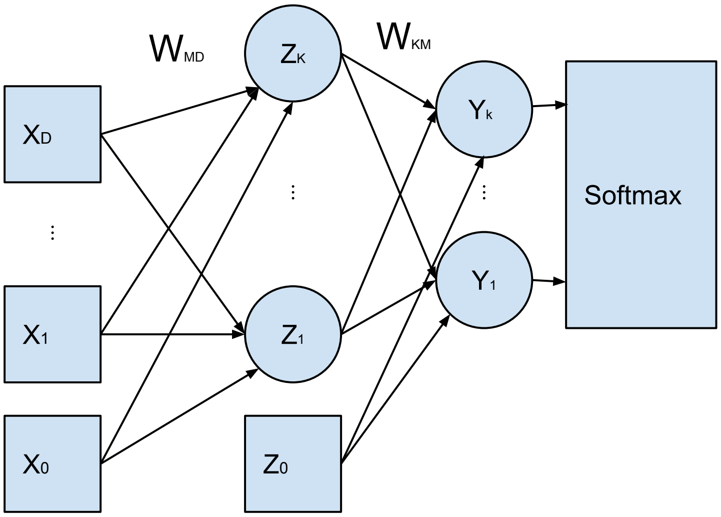


FIG. 17: Illustration of a simple ANN with D inputs, M hidden units and K outputs.

randomly initialized when the network is first simulated. The classification accuracy will be very poor for randomly initialized weights. In order to improve accuracy, the network is trained using labeled data.

Labeled data is when specific data inputs are associated with a known label. This typically involves a human manually classifying the image/data. These labeled images are passed through the network and the result of the classification layer is compared with the label. The difference between these can be thought of as the error of the network which can be used to adjust the weights to improve performance.

Using the previous example, if the input image was labeled a cat which is represented as vector $[1, 0, 0, 0]$ and the output of the softmax was $[0.26, 0.25, 0.24, 0.25]$ then the difference between the labeled data and the estimate of the ANN is the error of the network. A cost function can be established to quantify this error. A simple error function is the sum-of-squares error given by equation (37). Here x_n is the nth training input and y_n is the output which is the feed-forward result of the the nth input for a given set of weights. Since this is labeled data, the target will be denoted as t_n as shown below.

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}(y_n(X_n, w) - t_n)^2 \tag{37}$$

The goal is to minimize this error function by changing the weight values as scaled by some learning rate η. The weights are updated as shown in equation (38). Here, $\eta > 0$ and τ corresponds to the current weight values and $\tau + 1$ corresponds to the updated weights by some small step.

$$w^{\tau+1} = w^{\tau} - \eta \nabla E(w^{\tau}) \tag{38}$$

Each small step is chosen to help minimize the error gradient. This process is referred to as gradient decent. Stochastic gradient decent is a very similar process except that updates to the weight vector are based on one data point per update, as shown in equation (39).

$$w^{\tau+1} = w^{\tau} - \eta \nabla E_n(w^{\tau}) \tag{39}$$

To evaluate the gradient of the error function, the errors of the feed forward network are used in a process referred to as error backpropagation. Specifically, the gradient of the error function is evaluated with respect to each weight in the network. The error seen by a specific weight is represented as δ_j and is given by equation (40) where δ_k are errors in later nodes in the network.

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k \tag{40}$$

These equations allow the error to pass back to earlier layers, and then have that error, coupled with the learning rate, influence by what amount that specific weight is updated. This allows the weights to adapt to the training data to minimize error.

Fully connected ANNs perform well as universal approximation functions. However, the inputs and, thus, their associated weights are fixed to specific locations. An FC-ANN, can learn to classify images if the object of the image always appears in the same location in the image. As the network is trained, the weights associated with the center of the image will learn features in the input data to help it classify the image. However, if the object of focus shifts from the center, then the accuracy of the ANN is compromised because the weights on the edge of the network will not have been exposed to these image features and thus will not have learned to recognize these features. To counter this problem, there is a different topology of ANN known as convolutions neural networks (CNN).

3.2 CONVOLUTIONAL NEURAL NETWORK

The previous section described fully connected layers and how these may be cascaded to form a multi layer ANN. A convolutional layer uses a different paradigm to map inputs

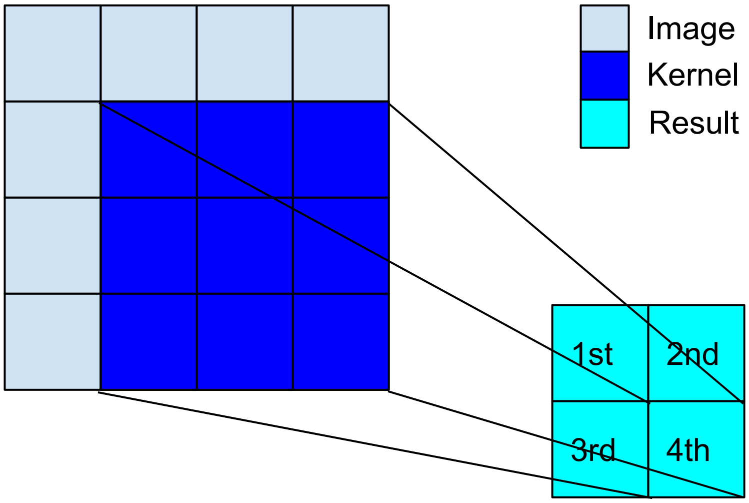


FIG. 18: Illustration of the CNN operation.

to their corresponding neurons. Unlike the fully connected layer equation in (35) where all inputs are passed to each neuron, the CNN restricts the observed input values to a subset of the entire input image using kernels of a fixed size. An example would be a 3x3 pixel kernel. Each element in this kernel can be thought of as a weight and is an adaptive parameter. This kernel starts at an edge of the image (normally the top left hand corner) and is placed over top of the input image. The value of each pixel will be multiplied by the corresponding weight in the corner in a one to one fashion. Then all of these values will be summed and will be the activation for this kernel location. An illustration of this is shown in Fig. 18.

The activation of a CNN is as shown in equation (41). It is important to note that the activation output can be a three dimensional matrix. It is also important to note that, despite the name, the sliding operation is better described as a correlation function as opposed to a convolution (note, there is no rotation). The below symbol, $*$, is representative of this correlation operation.

$$a_{ji} = w_{ji} * x + w_{ji} \tag{41}$$

The kernel will then move to its next location, and the process will be repeated until the entire image has been passed over by this kernel. The activations that result from this will form a matrix. If there is one kernel, this will be a two dimensional matrix with a_{ji} being the activation that results from the $\{j, i\}$ operation. If there are K kernels, this will be a K dimensional matrix. This matrix will be smaller than the original image depending on the kernel size and the stride used. After the activation matrix is complete, it will be passed through a non-linear, differentiable, transformation function. This will be the output of the CNN layer.

These kernels will learn to recognize features such as edges or shapes and the weights will be updated to achieve a high activation value once these features are observed. The fact that these kernels move across the image helps the ANN learn spatial invariance. Unlike the fully connected layers that require the object of focus to always appear in the center of the image, CNN can learn to recognize objects that are not in the center. CNN are trained with the same concept of backpropogation. Multiple CNN layers can be cascaded, or a mix of CNN with fully connected layers can be used. CNN are a powerful tool to apply to classification problems due to their ability to parse through an image and activate on the features of interest. CNNs are typically used in image processing and image classification problems. CNN are a powerful tool for classification and will be used for the application of digital modulation classification in Chapter 6.

One common problem for both fully connected layers and CNN layers is that for very deep networks, with many layers, the error that is backpropogated to earlier neurons will be very small. In such a case, the weights in earlier layers will not adapt; thus, the network will be unable to learn further, regardless of training time. This is due to the problem known as vanishing gradient. To counter the problem of vanishing gradient, residual networks are used.

3.3 RESIDUAL NEURAL NETWORK

Residual neural networks (RESNET) are very similar to convolutional neural networks with the primary difference being that these networks employ skip connections. During training, error is used to modify adaptive parameters to improve accuracy as discussed in previous sections. This error is used to update all weights in the network. Equation (40) shows that the error of a current layer is dependent on the error observed in later layers. As error is passed to previous layers, the weights in that layer will absorb some of this error as they adapt their values. As this process repeats, the error values that are backpropogated to earlier layers become small if not zero. This is known as vanishing gradient, due to the fact that the error gradient approaches zero for very deep networks. The motivating effort behind residual networks is to create an additional path which allows error to pass back to earlier layers. To do this skip connections are introduced.

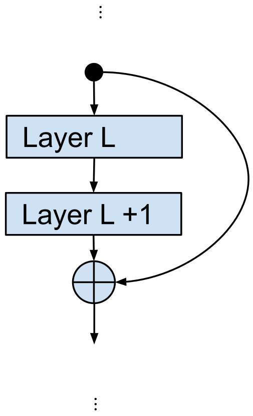


FIG. 19: Illustration of a skip connection in a Residual Network.

Residual networks will have skip connections which connect an earlier layer to a later layer. An example of a skip connection is shown in Fig. 19. Here, several layers are skipped, thus allowing the error to have an additional path so that it may backpropogate through the skip connection.

Many such skip connection blocks are connected in series allowing for a large number of CNN to be used. In [16], Residual networks containing 50 and 101 layers have been implemented and shown to have great benchmark testing. The increased number of layers allows the networks to learn complex features in input data.

In simulation, residual networks work very well in that they are excellent at classifying complex datasets. One drawback to such networks is their increased complexity. Because these networks are so large and the adaptive parameters are so great, physical implementation in real time devices may suffer. However, for offline or post processing applications, residual networks are very desirable.

3.4 CAPSULE NETWORK

ANN are very customizable and include many more types of layers than the ones mentioned above. Another common type of layer is pooling layers which can be a linear operation that sum or average the outputs of the previous layer. These can be useful for reducing the feature input size to the following layer. This reduces the number of adaptive weights and can reduce training time with no loss in accuracy.

One particular caveat to pooling layers was presented by [17]. The claim is that pooling

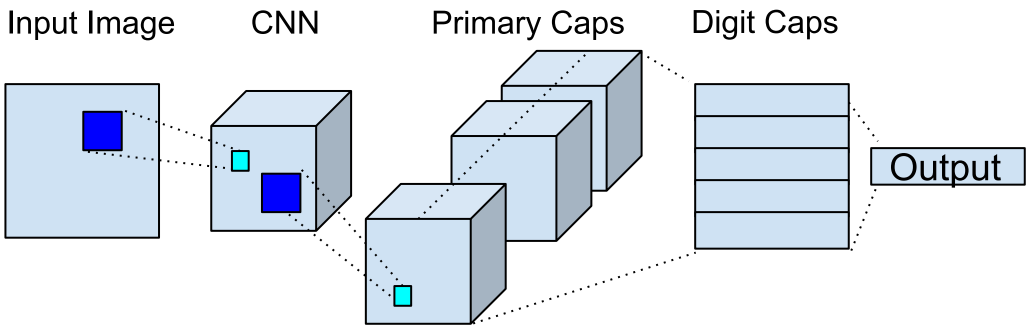


FIG. 20: High level diagram of a capsule network.

layers cause CNN layers to learn features based on proximity to other features. An example of this would be for facial recognition. A human's eyes are located near the nose, and the nose is located near the mouth, etc. If a CNN, with pooling layers, is trained with normal facial composition it can learn to recognize a human face. However, if the image has the location of the nose and eyes swapped and the mouth and ears swapped, the CNN will not be able to recognize the image as a face. This is known as the Picasso problem and is often attributed to pooling layers.

A different type of architecture referred to as capsule networks has been presented in [17]. These networks come in various forms and can have special classification layers, but the primary goal of the network is to model hierarchical relations between input features based on the way neurons are organized in nature. These networks will be described in detail in the next chapter, but as an introduction, these networks are comprised of CNN layers whose feature output is passed to several identical branches, each consisting of additional CNN layers and possibly fully connected layers. For multi-class classification problems, there can be one capsule per class. A high level figure of a capsule network is provided in Fig. 21.

These networks typically avoid pooling operations in the capsules and have outputs as vectors. For an M class problem, there will be M output capsule vectors. As originally presented by [17] the capsule networks employ a squashing layer and a specific routing algorithm referred to as routing by agreement which updates the weights between the output vector and the squashed layer by evaluating the conditional probability that the input image belongs to class M given the values in the last layer.

The magnitude of this output vector is used to determine how likely the input image corresponds to its respective class. As mentioned in [17], there could be many different approaches to routing the last layer to the outputs. Furthermore, there are other recent papers exploring different routing algorithms or modifications to the capsule network such as in [18].

This chapter provided a background on deep learning, what the various layers of the networks look like, how they function and how these networks learn. Now the focus will be on how to use ANN for digital modulation classification.

CHAPTER 4

DEEP LEARNING FOR DIGITAL MODULATION CLASSIFICATION

The previous chapter discussed various machine learning layers and concepts that are used in image and data classification. Most of the examples provided were of image classification; however, these machine learning tools are well suited for classification of various forms of data including digital modulation schemes.

Blind modulation classification has historically been achieved by means of signal processing techniques such as likelihood based approaches [2, 3] or CSP approaches [19] which focus on extracting features and performing classifications based on these features whereas machine learning techniques, like those mentioned in the previous chapter, rely on extensive training to automatically learn these features and perform classifications. There have been a wide range of recent approaches that use convolutional or residual neural networks to perform digital modulation classification based on the raw in-phase and quadrature (I and Q) components [6, 5, 8]. Other networks use signal amplitude, phase or spectrum representation as inputs into the network with goal of digital modulation classification [20, 21, 22, 23, 24].

The benefit of using in-phase and quadrature data is that these require minimal preprocessing and can be directly fed into a ANN for classification. Typically a window of time must be observed; the longer this window, the more excursions should be observed and thus the network should be better able to classify the modulation scheme.

This chapter will begin by reviewing some topologies using CNN and RESNET for digital modulation classification and provide some commentary on these approaches. Then, this chapter will focus on a very specific digital modulation classification method by means of capsule like networks. Others have had success in doing this in [25, 19]. Capsule networks initially presented by [17] promise to learn hierarchical relationships in the underlying data. This can be especially useful for digital modulation schemes which may have excursions (or features) that are independent from other excursions. In this way, the individual excursions may be learned as opposed to the correlation of multiple excursions appearing in the same sample window. To use the Picasso face analogy, the excursion from one symbol to another symbol may occur in various sequences, just like the Picasso picture may have the facial components appear in various sequences. In both cases, if the hierarchical components can be isolated (i.e. the facial features and symbol excursions) then accurate classification can occur. This chapter will discuss several deep learning approaches with a focus on the specific application of capsule like networks for digital modulation classification.

4.1 CNN FOR DIGITAL MODULATION CLASSIFICATION

As established in Chapter 3, CNNs can be used for classification problems. One of the primary strengths of a CNN is that their kernels will learn to recognize features in the input data and activate on these features. Because the kernels slide over the input data, they will be well suited for learning features while maintaining spatial invariance in the signal data and will be capable of detecting features throughout the entire signal duration. This aspect makes the CNN desirable for digital modulation classification problems. The authors in [8, 6, 5, 7] employed CNN based machine learning approaches for digital modulation classification. The network in [8] was of particular interest and the architecture of this network was replicated with similar results. This network has an architecture as described in Table 1. This architecture consists of six CNN layers with nonlinear ReLU activations. The multiple CNN layers allow for highly nonlinear features to be extracted by the following layer. This furnishes the network with sufficient complexity to learn to classify the various excursions and features present in the various digital modulation schemes. Followed by each activation function is either a max pooling or an average pooling layer. These pooling layers allow only the strongest activations from the previous layer to survive to the next layer. Notionally, this helps the network learn only the most prominent features but some authors suggest that this creates dependence on specific features occurring over a given duration [17]. In either case, the replicated CNN produces reasonable results which will be assessed in Chapter 6.

Due to the few layers in this network, the problem of vanishing gradient is not manifested. If many more layers were added, the problem of vanishing gradient would begin to arise. Additional layers may increase performance, but to avoid the problem of vanishing gradient, a residual network topology must be employed which will be elaborated upon in the next section.

4.2 RESNET FOR DIGITAL MODULATION CLASSIFICATION

The authors in [16] designed a RESNET for digital modulation classification which avoids the problem of vanishing gradient by means of skip connections. These skip connections

TABLE 1: CNN Network Layout

Layer	Filter	Stride	Size/Weights
Input			$2 \times 32,768$
Conv	[1,23]	[1,2]	$23 \times 2 \times 16$
Batch Normalization			
ReLU			
Max Pool	[1,2]	[1,2]	
Conv	[1,23]	[1,2]	$23 \times 16 \times 24$
Batch Normalization			
ReLU			
Max Pool	[1,2]	[1,2]	
Conv	[1,23]	[1,2]	$24 \times 23 \times 22$
Batch Normalization			
ReLU			
Max Pool	[1,2]	[1,2]	
Conv	[1,23]	[1,2]	$23 \times 32 \times 48$
Batch Normalization			
ReLU			
Max Pool	[1,2]	[1,2]	
Conv	[1,23]	[1,2]	$23 \times 48 \times 64$
Batch Normalization			
ReLU			
Max Pool	[1,2]	[1,2]	
Conv	[1,23]	[1,2]	$23 \times 64 \times 96$
Batch Normalization			
ReLU			
Average Pool	[1,32]	[1,1]	
Fully Connected			8
SoftMax			

allow the error from later layers to propagate to earlier layers which enables all adaptive parameters in the network to be updated during training. The additional convolutional layers will theoretically assist the network in learning complex features and characteristics.

An RESNET was designed with parameters as described in Table 2 and its results will be displayed in Chapter 6. This network has more convolutional layers than the CNN described in the previous section. Additional skip connections can be included, but they increase training time without any noticeable improvement in performance. Furthermore, it was found that using strides to reduce the feature map had better performance than pooling layers which is why they were avoided in this topology.

In Table 2, the skip connections are a direct connection from one part of the network to another. For example, the Tanh with the note, *'branch to ADD-1'*, has a direct connection from that activation function to the ADD-1 layer. The ADD-1 layer sums the outputs of the prior activation function and the branched activation functions skip connection. The skipped portion of the network is called a residual block. The branch that is connecting the activation function to the ADD layer at the end of the residual block is the skip connection. In this network there are three of these residual blocks with a skip connection on each block.

Many variations of the CNN and RESNET were attempted, but the networks described in Table 1 and 2 had the best performance for their respective topology. Other variations using multiple branches were also pursued which eventually lead to research into capsule networks. These capsule networks will be discussed in the following section.

TABLE 2: RESNET Network Layout

Layer	Filter	Stride	Size/Weights
Input			$2 \times 32,768$
Conv	[1,22]	[1,9]	$22 \times 2 \times 64$
Batch Normalization			
Tanh			
Conv	[1,23]	[1,8]	$23 \times 64 \times 48$
Batch Normalization			
Tanh (branch to ADD-1)			
Conv	[1,24]	[1,1]	$22 \times 48 \times 64$
Batch Normalization			
Tanh			
Conv	[1,24]	[1,1]	$22 \times 64 \times 48$
Batch Normalization			
Tanh			
Add-1 (branch to ADD-2)			
Conv	[1,24]	[1,1]	$22 \times 48 \times 64$
Batch Normalization			
Tanh			
Conv	[1,24]	[1,1]	$22 \times 64 \times 48$
Batch Normalization			
Tanh			
Add-2 (branch to ADD-3)			
Conv	[1,24]	[1,1]	$22 \times 48 \times 64$
Batch Normalization			
Tanh			
Conv	[1,24]	[1,1]	$22 \times 64 \times 48$
Batch Normalization			
Tanh			
Add-3			
Fully Connected			8
SoftMax			

4.3 PROPOSED CAPSULE NETWORK FOR DIGITAL MODULATION CLASSIFICATION

The aim of capsule networks is to better emulate how humans learn to recognize desirable characteristics in their vision. As elaborated upon in [17], when the human eye receives an input image, the eye does not focus on the entire image. Instead, points of fixation are established which are used to reconstruct a mental image in the person's mind. In other words, when a human looks at an object, they aren't actually looking at the entire object; rather, their eyes are picking out the important characteristics of the object. These important characteristics inform how the person classifies the object. With respect to digital modulation classification, the capsule network will discover characteristics that are intrinsic to that modulation scheme which will be used to classify it.

To emulate the points of fixation, several capsules will be established. These individual capsules are expected to learn the characteristics unique to each modulation scheme so that it may be accurately classified. The inputs to the proposed capsule network will be raw in-phase and quadrature data.

A capsule network is a CNN style network comprised of several layers (or sections). The first section is a feature extraction layer. This is a standard CNN layer with an activation function. The following layer is referred to as the primary caps section which consists of several CNN layers. The output of each primary caps layer is referred to as the capsule vector or digit caps and has dimensions $1xN$ with N being the number of neurons in that vector which correspond to the number of class specific attributes that the capsule network will learn during training. There is one digit caps per class. It is important to note that this is different from standard CNN approaches which rely on a single output neuron per class. Ideally, the magnitude of this capsule vector will correspond with the probability that the input data matches this class. The neurons in the digit caps section have connections with the neurons in the primary caps layer. The weights for these connections are determined iteratively using the dynamic routing by agreement algorithm [17].

However, in this text an alternative to the dynamic routing by agreement algorithm will be applied. Instead of using the method in [17] to update the weights between a given capsule vector and higher layer neurons, all higher layer neurons will be fully connected to each neuron in a $1 \times N$ neuron vector. These neurons are expected to discover class specific attributes and activate on these recognized characteristics. This topology is a simpler approach to [17] and allows for an easier real world implementation and efficient training as matrix operations are very efficiently computed with a graphical processing unit (GPU) whereas

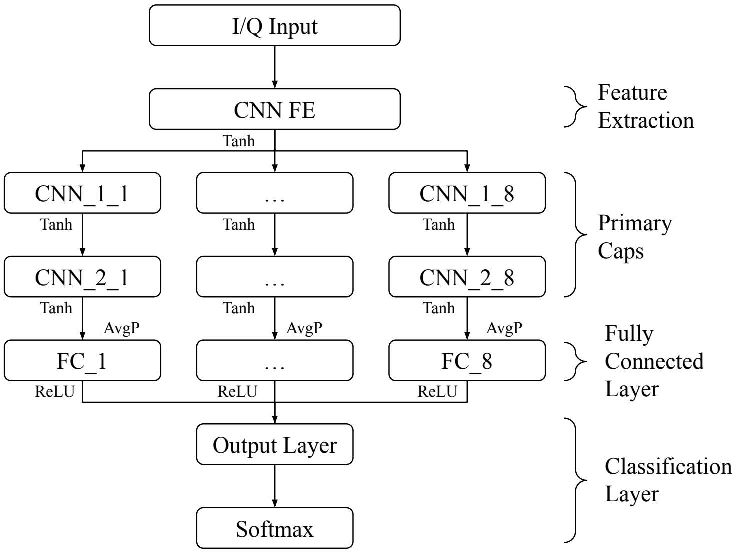


FIG. 21: The simplified topology of the capsule network considered for digital modulation classification.

iterative learning by means of dynamic routing by agreement is not. The remainder of this chapter will focus on the specifics of this capsule-like network. Other recent modifications to capsule networks can be found in [18].

4.3.1 NETWORK TOPOLOGY

The proposed capsule network is shown in Fig. 21. This network is designed to classify the following eight digital modulation schemes: BPSK, QPSK, 8-PSK, DQPSK, MSK, 16-QAM, 64-QAM, and 256-QAM. As such, there are eight branches/capsules in this topology. Each of the highlighted sections in Fig. 21 are elaborated upon in the below subsections. The specific parameters of each layer are captured in Table 3.

TABLE 3: Capsule Neural Network Layout

Layer	Filter	Stride	Size/Weights
Input			$2 \times 32,768$
Conv	[1,22]	[1,9]	$22 \times 2 \times 64$
Batch Normalization			
Tanh			
Conv-1-(i)	[1 23]	[1,7]	$23 \times 64 \times 48$
Batch Normalization-1-(i)			
Tanh-1-(i)			
Conv-2-(i)	[1 22]	[1,8]	$22 \times 48 \times 64$
Batch Normalization-2-(i)			
Tanh-2-(i)			
Average Pool (i)	[1,8]	[1,1]	
FC-(i)			32
Batch Normalization-3-(i)			
ReLu-1-(i)			
Point FC-(i)			1
Depth Concatenation(i=1:8)			8
SoftMax			

Feature Extraction Layer

The very first layer of this network is the feature extraction layer. Its purpose is to use a CNN layer to perform general feature mapping of the input in-phase and quadrature signal data. Specific kernel seizes, strides and CNN parameters were inspired specifically by [6, 7, 8]. This input is passed to each of the primary capsules (primary caps) found in the following layer. It is important to note that these layers branch out, like a parse tree, with a number of branches equaling the desired number of classes. All of the primary caps receive the same input.

Primary Caps

There are eight primary caps layers, one for each modulation class. All primary caps layers have an identical structure and all receive the same output from the previous feature extraction layer. This layer consists of two CNN layers with similar filter and stride parameters as the previous layer, and an activation function. The primary caps layers is then fed into a fully connected layer.

Fully Connected Layer

The fully connected layer is comprised of $1 \times N$ neurons, which can be thought of as a neuron vector, with the weights being fully connect to each neuron in the last layer of the primary caps section. Ideally, the neurons in this layer will discover characteristics specific to the digital modulation scheme associated with this layer. This approach is chosen in lieu of the dynamic routing by agreement algorithm. To make the output of this layer compatible with a softmax classification layer, each of the neurons within this layer is fully connected to a single neuron, or point neuron. Thus, the final output of the capsule branch is a single neuron. Because there are eight branches there will be eight neurons. All eight of these output neurons will be combined depth wise to produce an 8-dimensional vector $\mathbf{a}$, which is passed to the classification layer. The elements in $\mathbf{a}$ are representative of the likelihood that the characteristics corresponding to a specific digital modulation scheme were present in the input in-phase and quadrature signal data.

Classification Layer

The classification layer is a standard softmax layer. The equation for the softmax was shown in equation (36). This output vector from the previous layers, $\mathbf{a}$, is fed into the softmax layer which will result in an eight dimensional vector σ. The element with the highest value corresponds to the modulation scheme that is most likely represented by the input in-phase and quadrature data.

4.3.2 BENEFITS OF NETWORK TOPOLOGY

As shown in Fig. 21, the capsule network consists of only a few layers in depth. These types of networks are often referred to as shallow CNNs because of their small layer depth. As such, capsule networks are not as susceptible as deeper neural networks to the problem of vanishing gradient and thus do not encounter this type of training difficulty.

Due to the fact that the capsule network emulates a parse tree structure, each branch can focus on learning characteristics of a specific class. All of the kernels in the primary caps will focus on learning to activate on features which correspond to that branch's class. In other networks, such as CNN and RESNET, the kernels in each layer could correspond to features of any class. This distinct difference allows the branches to excel at activating on class specific features.

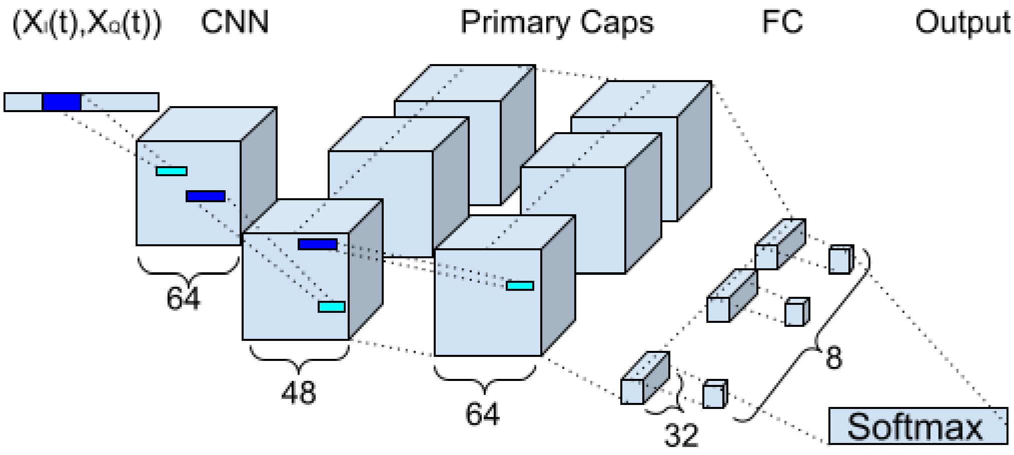


FIG. 22: Illustration of parameter space for the proposed capsule network.

Another advantage to this specific capsule like network is the simplicity of its architecture. The shallow layers coupled with the similarity between branches make it well suited for implementation into a field programmable gate array (FPGA). The illustration in Fig. 22 help shows the learnable parameter space needed for the proposed network. There are $2,076,872$ overall parameters needed for the proposed capsule network which is reasonable for real world implementation.

CHAPTER 5

SUPERVISED TRAINING

As mentioned in Chapter 3, artificial neural networks are comprised of many adaptive parameters which can be tuned arbitrarily. The process by which these adaptive parameters are tuned is referred to as training. The goal of training is to adjust the adaptive parameters such that they activate upon features present in the input data which are then used to predict the classification. Of the various training approaches available, the most common is supervised learning. Supervised learning uses input data which belongs to a known class. When a given input is associated with a known class, this data is considered to be labeled. The data itself can be anything from pixel intensities of an image, a list of statistical values, or a sequence of a digital signal, etc. The label is a finite category that the input data belongs to. Labels are often generated by humans manually assigning the data to one of several categories. Once an arbitrarily sufficient amount of labeled data is achieved, and a ANN topology is chosen, the process of supervised training may commence.

The goal of supervised training is to adapt the weights of the ANN so that the ANN may successfully map any given input data to one of the known categories. In other words, the goal is for the ANN to perform accurate classification of the input data. Typically, for supervised learning, the entire labeled dataset is separated into several disjoint subsets. These subsets are referred to as the training, validation and testing subsets. All of these subsets should have similar distributions of classes and similar class variations. The training dataset is exclusively used to tune the adaptive parameters of the ANN by means of backpropagation which was introduced in Chapter 3.

The input data is supplied to the ANN which returns a prediction. The prediction of the ANN is compared with the labeled value (often referred to as the truth). The error between the prediction and the truth is used to adapt the weights to improve performance. Once the entire training dataset has been used to tune the adaptive parameters, it is then shuffled and used again to continue training the network. The number of times the training dataset is used is referred to as the number of Epochs. Often many epochs are used to train the network. If too many epochs are used, the ANN may memorize or over-fit the training dataset. When a network over-fits it will have excellent performance for the training data but poor performance for any other input data. To prevent over-fitting, the ANN is evaluated at scheduled intervals with the validation dataset.

The validation dataset is separate from the training dataset and is no way used for adjusting the weights of the ANN. Instead, the validation dataset is used to evaluate the performance of the ANN with data that was not used for training. It is impossible for the ANN to over-fit the validation dataset thus making its predictions a useful tool for evaluating over-fitting. At scheduled intervals, normally at the end of each epoch, training is paused and the entire validation dataset is input into the ANN. The predictions of the ANN are compared to the truth values. If the performance of the ANN is similar to the performance of the training data, then the ANN is likely not over-fitting. The validation dataset is often the smallest of the three subsets but it is an important tool for evaluating the unbiased performance of the ANN during training. Because it is a small subset it may not carry with it sufficient statistical diversity to truly evaluate the network's performance. Thus, the third subset, the testing dataset, is used to evaluate performance.

After training has concluded, the testing dataset is input into the ANN which makes predictions. The predictions of the ANN are compared to the truth values resulting in the overall network performance. The training dataset is in no way used to adjust the weights of the network and is only used for performance evaluation. It is important that all three subsets have a similar distribution of labeled data with similar characteristics. Specific examples of this will be highlighted for the datasets used for training. However, the overall goal is that there should not be any appreciable differences in the training data and the testing data which give the testing data an advantage.

The training, validation and testing subsets of the labeled dataset are often described in terms of a percentage of the overall dataset. A common ratio is 75%/5%/20% which means that 75% of the dataset is used for training, 5% is used for validation and 20% is used for testing. This ratio is arbitrary and often requires a delicate balance between having a sufficient amount of training data to train the network while maintaining enough testing samples to accurately evaluate the network's performance.

This chapter will now explore two datasets which will be used for training the proposed network. These datasets contain signal samples for the following digital modulation schemes: BPSK, QPSK, 8-PSK, DQPSK, MSK, 16-QAM, 64-QAM, and 256-QAM. All of these digital modulation schemes and their symbol representation have been introduced in a previous chapter. There are two datasets that will be described, both containing the same digital modulation schemes but with some specific parameter variations which will be discussed in detail.

This chapter will also introduce the simulation setup, specifically the tools used and the process by which the networks are trained. Analysis of the pros and cons of certain simulation approaches will be provided.

5.1 DATASETS

The labeled datasets used for simulations are of the in-phase and quadrature signal components representing various digital modulation schemes. The signal sequence is the input data of the dataset. Each input data is labeled with its corresponding digital modulation schemes which are the labeled categories for the dataset. Each dataset contains an equal distribution of labeled data among the various digital modulation schemes. For any digital modulation scheme there are several varying characteristics that may be present.

As discussed in Chapter 2, the various roll off factors may be used for the root raised pulse shaping functions. The pulse shaping function will vary over a specified interval. The various pulse shaping functions are not expected to impact classification but they are a characteristic that varies and thus are listed. Another characteristic that varies is the number of in-phase and quadrature signal samples per symbol. This will influence the duration of the symbol pulses which may cause a stretching effect from the perspective of the ANN. Another characteristic that varies is the carrier frequency offset. Variations to the carrier frequency offset may introduce inter-carrier interface and cause difficulty in symbol detection. Finally, the range of in-band SNR values will determine the ratio of AWGN power to signal power that is present. The AWGN corrupts the symbol and makes its detection difficult. The specifics for the various datasets used are provided below.

5.1.1 DATASET1

The first dataset that will be presented is publicly available on [26]. This dataset is comprised of BPSK, QPSK, 8-PSK, DQPSK, MSK, 16-QAM, 64-QAM, and 256-QAM with there being $14,000$ of each digital modulation scheme for a total of $112,000$ available signals. Each signal is $32,768$ samples in length and contains the unfiltered in-phase and quadrature signal components. Prior to training, these signal components are normalized. This dataset will be referred to as *DataSet1*.

DataSet1 employs square-root raised-cosine pulse shaping that has a roll off factor within the interval of $[0.1, 1.0]$. The symbol rates for the *DataSet1* vary between $2\frac{samples}{symbol}$ and $50\frac{samples}{symbol}$. The carrier frequency offsets (CFO), represented as Δf_c, are uniformly distributed within the interval of $[-0.001, 0.001]$. For the *DataSet1* the in-band signal to noise ratio (SNR) varies between the values of -2 dB and $+12$ dB, where the SNR is represented in decibels. The specific in-band SNR range for the entire dataset is represented in Fig. 23.

The in-phase and quadrature signal sample data has an assumed sample rate of $f_s = 1$. If the time domain or frequency domain spectrum of these signals is plotted, an arbitrary sample rate may be used. This is a common convention especially in plotting the spectrum of Fast Fourier Transforms (FFT).

The similarities in the Cartesian symbol plots between some of these modulation schemes can make it difficult to distinguish one from another. *DataSet1* is designed to be challenging to perform digital modulation classification.

5.1.2 DATASET2

Especially in the field of machine learning, sometimes a minor variance in some of the parameters can result in undesirable classification accuracy. As a simple example, imagine if a facial recognition ANN only were given upright faces as the training images. Naturally, it would learn to recognize upright faces accurately. If this same network were tested on images that were flipped or rotated by 90^o then the network would likely have less than desirable classification accuracy. This is the motivating reason behind use a second dataset. The second dataset will be referred to as *DataSet2* and is available upon request from [26].

DataSet2 consists of the same digital modulation schemes with the same number of signals as *DataSet1*. The signal characteristics of *DataSet2* are comparable with the exception of the symbols rates and the CFOs Δf_c interval.

The symbol ranges for this dataset vary between $1\frac{samples}{symbol}$ and $30\frac{samples}{symbol}$. This interval

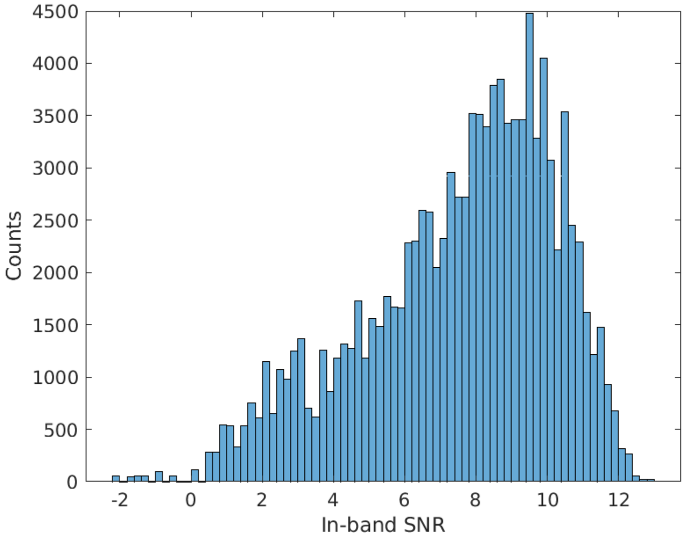


FIG. 23: *DataSet1* In-band SNR range for all signals.

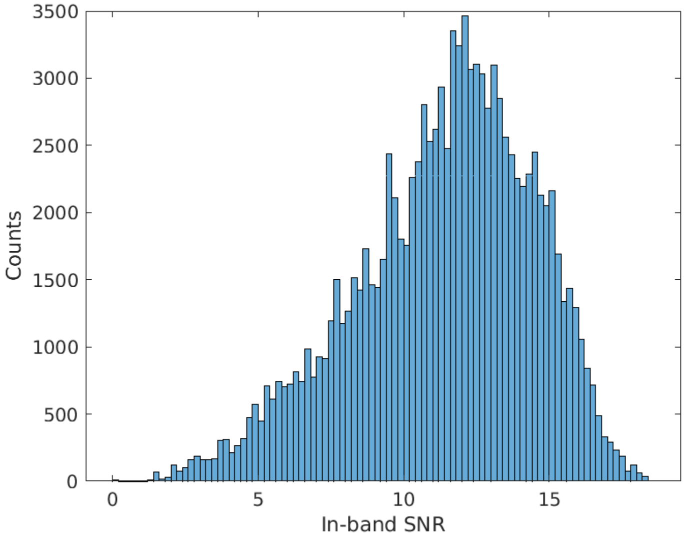


FIG. 24: *DataSet2* In-band SNR range for all signals.

largely overlaps with the *DataSet1* symbol rate characteristic. However, the CFOs Δf_c interval is disjoint between these two datasets. If a network is trained using one dataset and tested using the other, it is supposed that this will be a good test by which to gauge how well the trained network will be able to generalize. This experiment will be conducted and results displayed in a later chapter. Furthermore, the in-band SNR ranges vary slightly between the two datasets. The in-band SNR range for *DataSet2* is as shown in Fig. 24.

Using multiple separate datasets to test the proposed network will be useful in evaluating the overall performance of the proposed network. Also, by using different networks with varying characteristics, the problem of dataset shift [27] may be assessed.

5.2 DATA AUGMENTATION

It is expected that the trained digital modulation classifier will perform well at classifying high in-band SNR signals and perform less desirably as the SNR decreases. This is visible in similar digital modulation classification approaches [8, 5, 6, 7]. A close inspection of Fig. 23 will show that there are few samples for lower in-band SNR values. This small sample size for the lower in-band SNR values will not be very useful in evaluating the network performance for this SNR interval. To remedy this dilemma, data augmentation is used.

Let's go back to the example of facial recognition ANN that is trained only using upright images. If instead the dataset used to train the ANN is augmented by randomly flipping, rotating and inverting then the network will be exposed to such characteristics. Networks trained by augmenting the data can achieve better generalization. In this case, random noise will be added to higher SNR signals which will reduce their overall SNR. This noise will increase the difficulty in symbol excursion recognition. In no way will adding noise make the modulation scheme easier to classify.

For the entire dataset, the in-band SNR values are known for each signal, represented as P_{signal}. The signals will be augmented by adding a specific amount of noise to reduce the in-band SNR to a lower value. This value will be referred to as P_{add}. Because not all in-band SNR values are initially the same and because it is desirable to add a random amount of noise, the term P_{rand} is presented. P_{rand} is a pseudo random amount of noise that will be added to make the P_{add} fall in a desirable range. It is important that all classes be augmented by similar amounts of noise.

To begin, P_{add} must be determined which is simply the sum of the current in-band SNR value with the pseudo random amount of noise and the noise power. This is shown in equation (42). Note, some of these power values are in dB and some are in watts and P_n

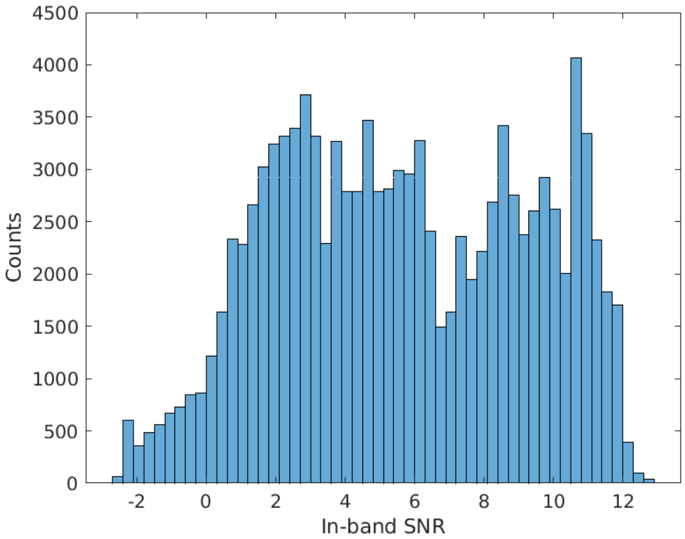


FIG. 25: *DataSet1* In-band SNR range for all signals after augmentation.

denotes the noise power which is known.

$$P_{add} = 10^{(P_{signal} - P_{rand})/10} - P_n \tag{42}$$

Next, two random lists of in-phase and quadrature data must be generated. The size of these arrays must match the dataset size which is $32,768$ samples for both in-phase and quadrature components. This can be represented as shown in equation (43) for $i = 1, \ldots, 32768$ and r and q are random values.

$$n(i) = r + jq. \tag{43}$$

Next, this $n(\cdot)$ must be scaled by the power factor. This new scaled value is represented in equation (44) where $n_s(\cdot)$ is the scaled value of the added noise; VAR is the statistical variance operator.

$$n_s(i) = n(i)\sqrt{\frac{P_{add}}{VAR(n(i))}} \tag{44}$$

This new $n_s(i)$ is then separated into its real and imaginary components (in-phase and quadrature components) and appropriately added back into the dataset. This results in a new histogram of the in-band SNR which is shown in Fig. 25. This histogram shows a better distribution of SNR values for the lower interval when compared to Fig. 23. These new SNR values were cross checked using a band of interest estimator [12]. This will be useful in determining the accuracy of the network for a wider range of SNR values. As can be seen in Fig. 24, *DataSet2* also does not have many values for the lower in-band SNR ranges. This dataset could also be augmented with noise to increase the samples for the lower in-band SNR range. However, the goal of introducing *DataSet2* is to see how well the network behaves for different characteristics and not necessarily for lower in-band SNR values.

5.3 SIMULATION SETUP

Supervised training is computationally intensive and is best performed on a high performed cluster (HPC). Training occurred on the WAHAB high performance cluster. Currently, WAHAB has available 10 NVidia Tesla V100 graphical processing unit (GPU) nodes available with each node having 128 GB of memory. For these nodes, the CPU is an Intel(R) Xeon(R) Gold 6130 @ 2.1GHz, with 32 slots available (that is, 32 threads can occur simultaneously). The proposed network has been implemented in MATLAB and is trained using the stochastic gradient descent with momentum (SGDM) algorithm [28]. The actual training of the network is computationally intensive; however, the HPC resources allow for the entire dataset to be loaded into RAM which greatly deceases training time. When working with large datasets, the slowest aspect of training is in loading a batch of training data into memory. By initially loading all data into RAM, a great deal of time is saved.

Now that the datasets have been properly introduced, the actual simulation environment can be established. For each dataset, the entire dataset is broken up into training, validation and testing subsets each with 70%, 5%, 25% respectively. All subsets have similar SNR ranges and contain the same class distribution. The training data is exclusively used for training the proposed network. During training, the actual training process will be paused and the validation data will be passed through the partially trained network. The validation data is not used to update weights; rather, it is used to evaluate the current performance of the network. Because the network was never trained with the validation data, it should be a good predictor of how the network is performing.

Because the network attempts to best fit the training data, it is possible for the network to memorize the training dataset thus producing artificially good results. To avoid this, separate testing data is used. Once all training has been concluded, the network may be tested with the testing data. This testing data has not been used in any way to update the weights of the network; thus, the classification accuracy of this network will be a good metric for evaluating the network's performance.

After training has been completed and the network has been evaluated with the testing data, the results will need to be displayed in a useful way. The three main ways that performance will be evaluated are overall accuracy, confusion matrix, and classification accuracy over in-band SNR range. The overall accuracy is a convenient way to quickly see how well the network performed overall. The confusion matrix is a very helpful graphic because it captures the overall classification accuracy per class as well as provides a metric for when incorrect classifications occur. This can be thought of as measuring the confusion of the network. Finally, a plot that shows the performance of the network for varying in-band SNR values will be helpful as lower in-band SNR values are not expected to perform as well. These three metrics will be useful in quantifying the proposed network's performance.

CHAPTER 6

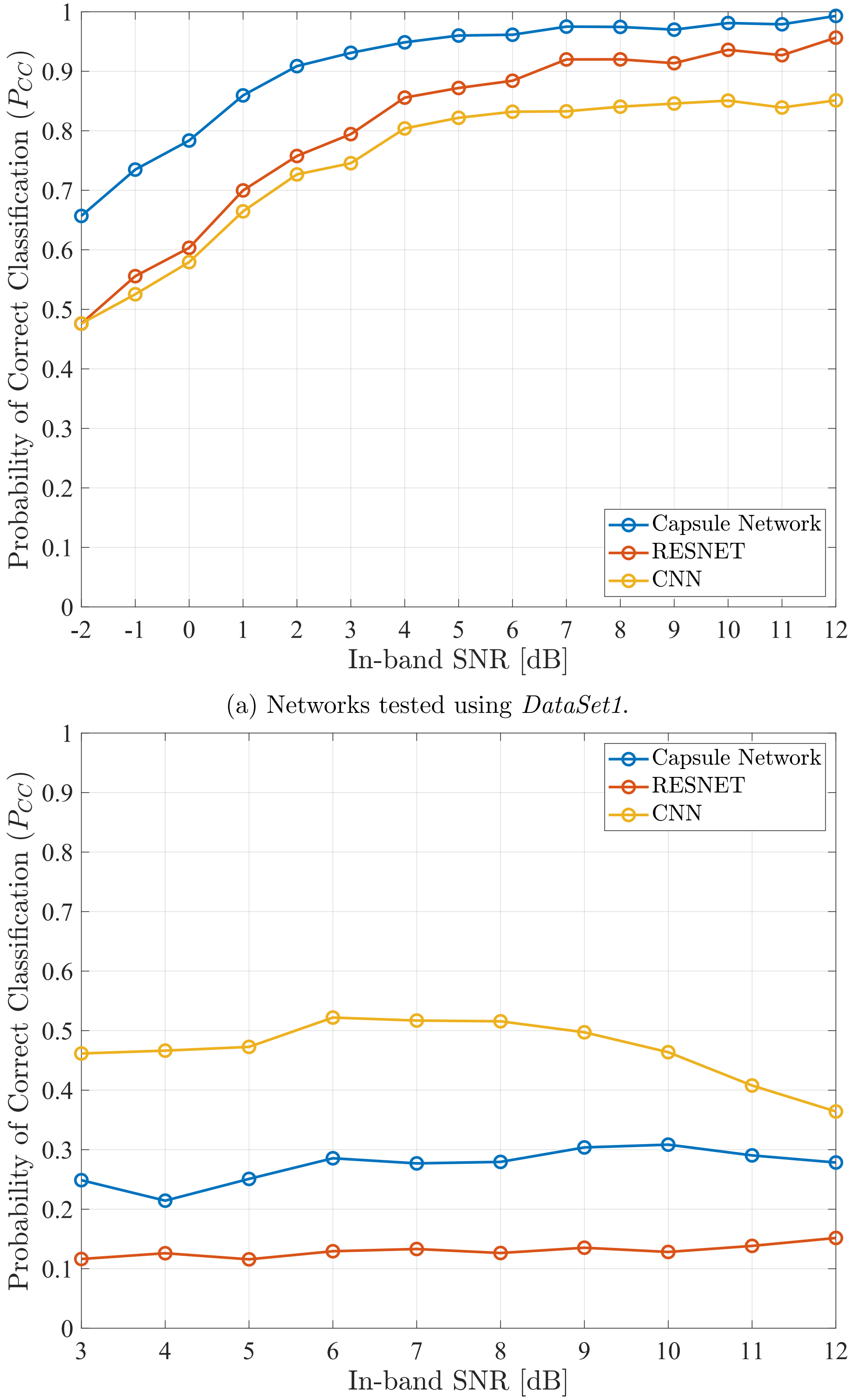
SIMULATION RESULTS AND ANALYSIS

All previous chapters have been presented in order to provide useful information which will now be culminated in this chapter. A background to the various digital modulation schemes has been provided. The signal space and symbol representations were shown in attempts to describe what kind of differences the classification network will be expected to learn. A background on machine learning as well as common layers were provided. This was in an attempt to provide a high level overview of the topic as it pertains to a classification ANN. Specific discussion of the various network topologies as well as detailed parameters for those networks were provided. The datasets and their similarities and dissimilarities were discussed. A description of the simulation environment and how performance will be measured was provided in detail. All of the above was given with the goal of providing the background information necessary to understand the goals of the simulations and the analysis of their results.

This chapter will begin by discussing the results obtained by training and testing with the *DataSet1*. This will demonstrate the network's ability to learn how to classify digital modulation schemes. In the same section, training with *DataSet1* and testing with *DataSet2* will be simulated. The goal of this second simulation is to see how well the network is able to generalize to out of distribution characteristics. These simulations will be repeated the other way around by evaluating the performance when the networks are trained and tested using *DataSet2* and also for when they are trained with *DataSet2* and tested with *DataSet1*.

6.1 DATASET1 RESULTS

The networks with the parameters specified in Tables 1, 2 and 3 were each trained using *DataSet1*. All networks were trained using $78,400$ training signals. The capsule network was trained for 8 epochs, the RESNET was trained for 14 epochs and the CNN was trained for 12 epochs. Once trained, the networks were tested using $28,000$ testing signals which were not used during training in any way. The network performance can be assessed by viewing a probability of correct classification (PCC) vs in-band SNR plot. These results are displayed in Fig. 26a. These same trained networks were then tested with all $112,000$ testing signals from *DataSet2* with results shown in Fig. 26b.



(a) Networks tested using *DataSet1*.

(b) Networks tested using *DataSet2*.

FIG. 26: Probability of correct classification vs in-band SNR for networks trained using *DataSet1*.

The network performance for an individual class can be better assessed with a confusion matrix. The confusion matrix will show the percentage of correct classifications for each particular class as well as the percentage of incorrect classification with respect to the labeled class. This is very useful as it shows the accuracy of network predictions while capturing which classes the network is likely to incorrectly classify. The confusion matrices for each of the trained networks will be discussed in subsequent sections.

CNN Performance:

The CNN performance as shown in Fig. 26a is reasonable. Moderately high classification accuracy is seen for higher in-band SNR values. As the in-band SNR is decreased, the classification accuracy is reduced. The network performance never drops below $\frac{1}{8} = 12.5\%$ which is a good indicator that, even for undesirable in-band SNR values, the network is able to activate on key features in the input signal which allow the network to make predictions at an accuracy level that is greater than a random guess. Similarly, the performance for PCC vs in-band SNR for the network tested with *DataSet2* can be evaluated by viewing Fig. 26b. The CNN trend line shows decreased performance for the entire in-band SNR range which suggests that this trained CNN network is not as readily able to activate on the features in *DataSet2* with characteristics it was not trained with. The PCC vs in-band SNR is a valuable plot, but it does not provide individual class performance. To view the individual class performance, a confusion matrix is used.

The confusion matrix for the CNN that was trained and tested using *DataSet1* is provided in Fig 27a. This confusion matrix shows that the two classes that perform best are BPSK and MSK. It is interesting to note that the PSK type signals are most likely to be confused with each other. There are similarities between the different PSK type signals. This suggests that the CNN is learning key features and characteristics present in the PSK type classes and is making predictions based on these features. Similarly, the QAM type signals also are heavily confused with each other. The QAM constellations are very similar to each other which explains why there is so much confusion among the QAM type signals. Overall, this confusion matrix provides valuable insight into the CNN performance and suggests that the network is truly learning features present in the respective classes. The confusion matrix for the CNN that was trained with *DataSet1* and tested using *DataSet2* is shown in Fig. 27b. Here, the results are not very desirable with large confusions happening among the PSK type and QAM type signals. Furthermore, BPSK signals now have the lowest overall performance. It is apparent that the features learned by training with *DataSet1* did not generalize well to the features with different characteristics present in *DataSet2*.

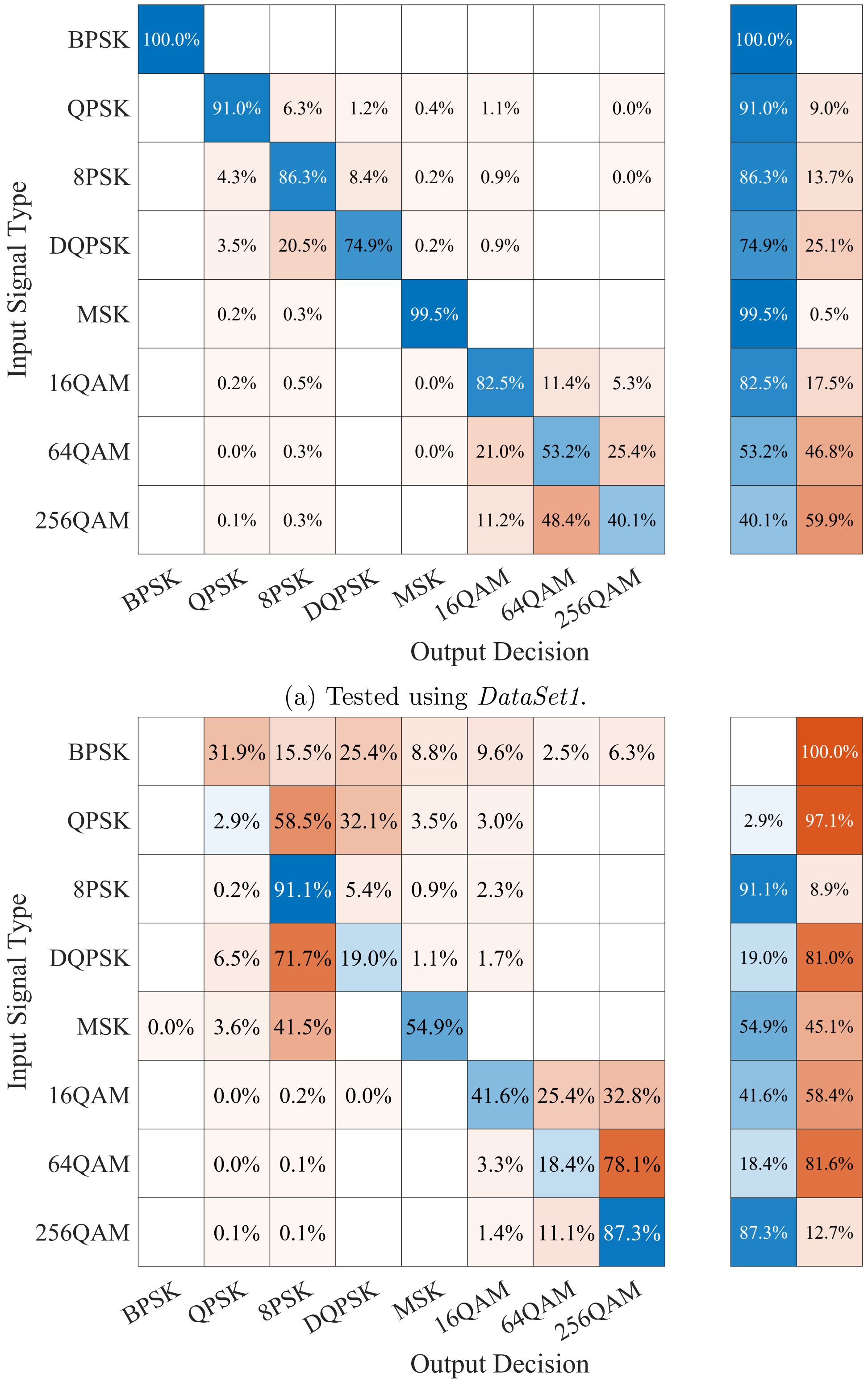
RESNET Performance:

The RESNET has a slightly more complicated architecture than the CNN. As seen in Fig. 26a, the RESNET either matched or outperformed the CNN across the entire in-band SNR range. The RESNET has excellent PCC at high in-band SNR values which is reduced as the in-band SNR is decreased. It is interesting to note that the CNN and RESNET are very similar for in-band SNR values at or below $2dB$. The results for RESNET, which is tested using *DataSet2*, are shown in Fig. 26b. These results are very poor and appear to hover around 12.5% which suggests that the network is unable to detect any features in *DataSet2* and is simply guessing. A confusion matrix can be used to assess each individual class performance.

The confusion matrix in Fig. 28a displays the RESNET class performance. It is very interesting to note that all classes in the RESNET perform around 84% with MSK having the best performance. The confusion of this network shows that confusion for any class is spread among all classes. Upon close inspection, there is still a slightly higher chance of PSK type signals being confused with PSK type signals and QAM type signals being confused with other QAM type signals. This signature was more noticeable in the CNN confusion matrix. Overall, these are good results which suggest that the RESNET is learning to recognize features in the input signals and activate upon them. The confusion matrix capturing the results of the trained RESNET being tested with *DataSet2* are shown in 28b. These results show poor performance and suggest that the network is simply guessing with a slight preference for the MSK class. The RESNET is very sensitive to the out of distribution characteristics present in *DataSet2* and thus did not generalize well.

Capsule Network Performance:

The capsule network has the best overall performance as illustrated in Fig. 28a. Here, the capsule network is higher than both the CNN and RESNET for the entire in-band SNR range. This network has excellent performance at high in-band SNR values. This performance is reduced when the in-band SNR is lowered which is to be expected. Performance is severely decreased when this trained network is tested using *DataSet2* as shown in Fig. 26b. The PCC is greater than 12.5% suggesting that the network is not guessing and is still able to activate on features in the input signals but with severely reduced accuracy. A confusion matrix can be used to assess individual class performance.



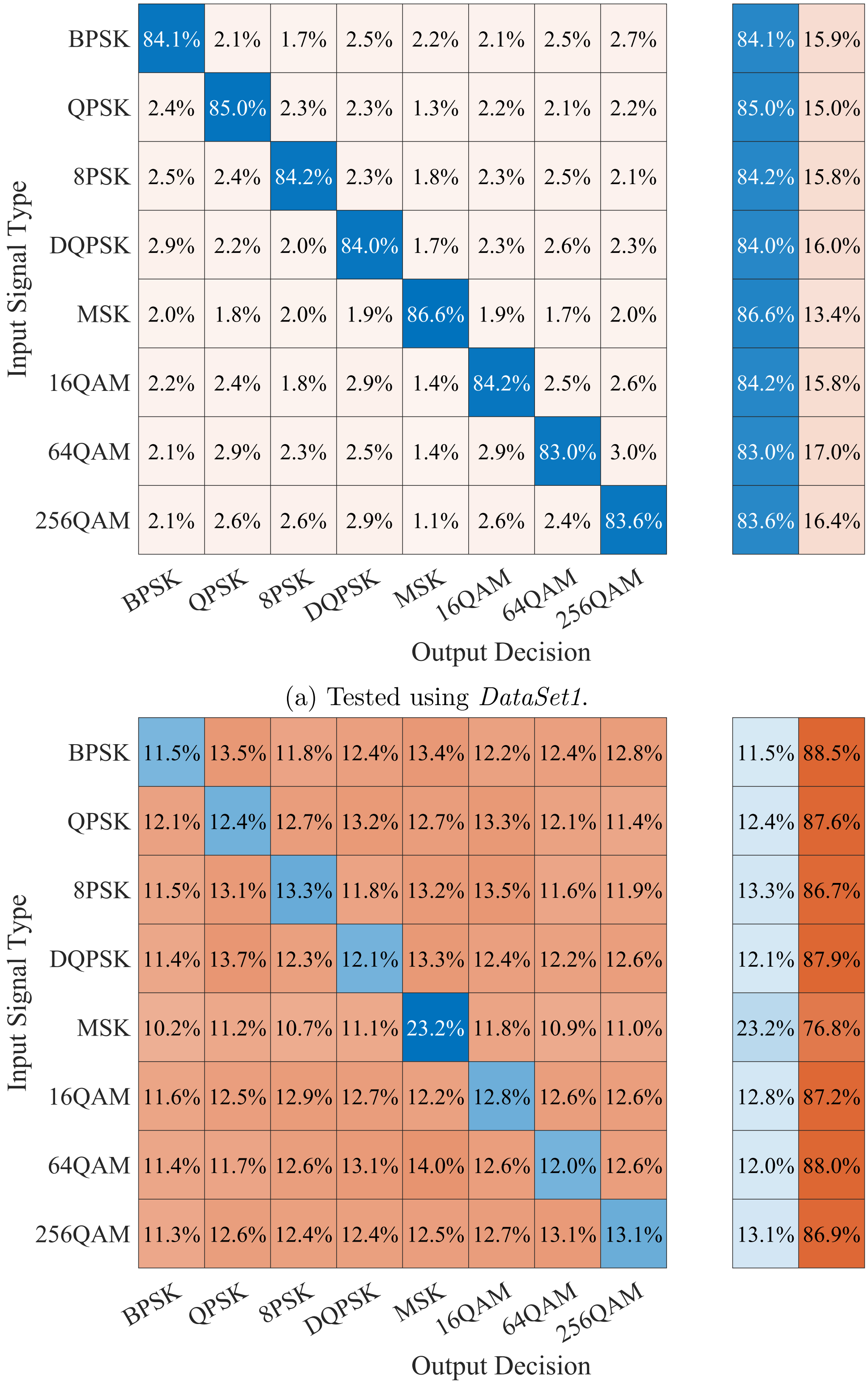
Input Signal Type	BPSK	QPSK	8PSK	DQPSK	MSK	16QAM	64QAM	256QAM		
BPSK	100.0%								100.0%	
QPSK		91.0%	6.3%	1.2%	0.4%	1.1%		0.0%	91.0%	9.0%
8PSK		4.3%	86.3%	8.4%	0.2%	0.9%		0.0%	86.3%	13.7%
DQPSK		3.5%	20.5%	74.9%	0.2%	0.9%			74.9%	25.1%
MSK		0.2%	0.3%		99.5%				99.5%	0.5%
16QAM		0.2%	0.5%		0.0%	82.5%	11.4%	5.3%	82.5%	17.5%
64QAM		0.0%	0.3%		0.0%	21.0%	53.2%	25.4%	53.2%	46.8%
256QAM		0.1%	0.3%			11.2%	48.4%	40.1%	40.1%	59.9%

(a) Tested using *DataSet1*.

Input Signal Type	BPSK	QPSK	8PSK	DQPSK	MSK	16QAM	64QAM	256QAM		
BPSK		31.9%	15.5%	25.4%	8.8%	9.6%	2.5%	6.3%		100.0%
QPSK		2.9%	58.5%	32.1%	3.5%	3.0%			2.9%	97.1%
8PSK		0.2%	91.1%	5.4%	0.9%	2.3%			91.1%	8.9%
DQPSK		6.5%	71.7%	19.0%	1.1%	1.7%			19.0%	81.0%
MSK	0.0%	3.6%	41.5%		54.9%				54.9%	45.1%
16QAM		0.0%	0.2%	0.0%		41.6%	25.4%	32.8%	41.6%	58.4%
64QAM		0.0%	0.1%			3.3%	18.4%	78.1%	18.4%	81.6%
256QAM		0.1%	0.1%			1.4%	11.1%	87.3%	87.3%	12.7%

(b) Tested using *DataSet2*.

FIG. 27: Confusion Matrix of CNN trained using *DataSet1*.



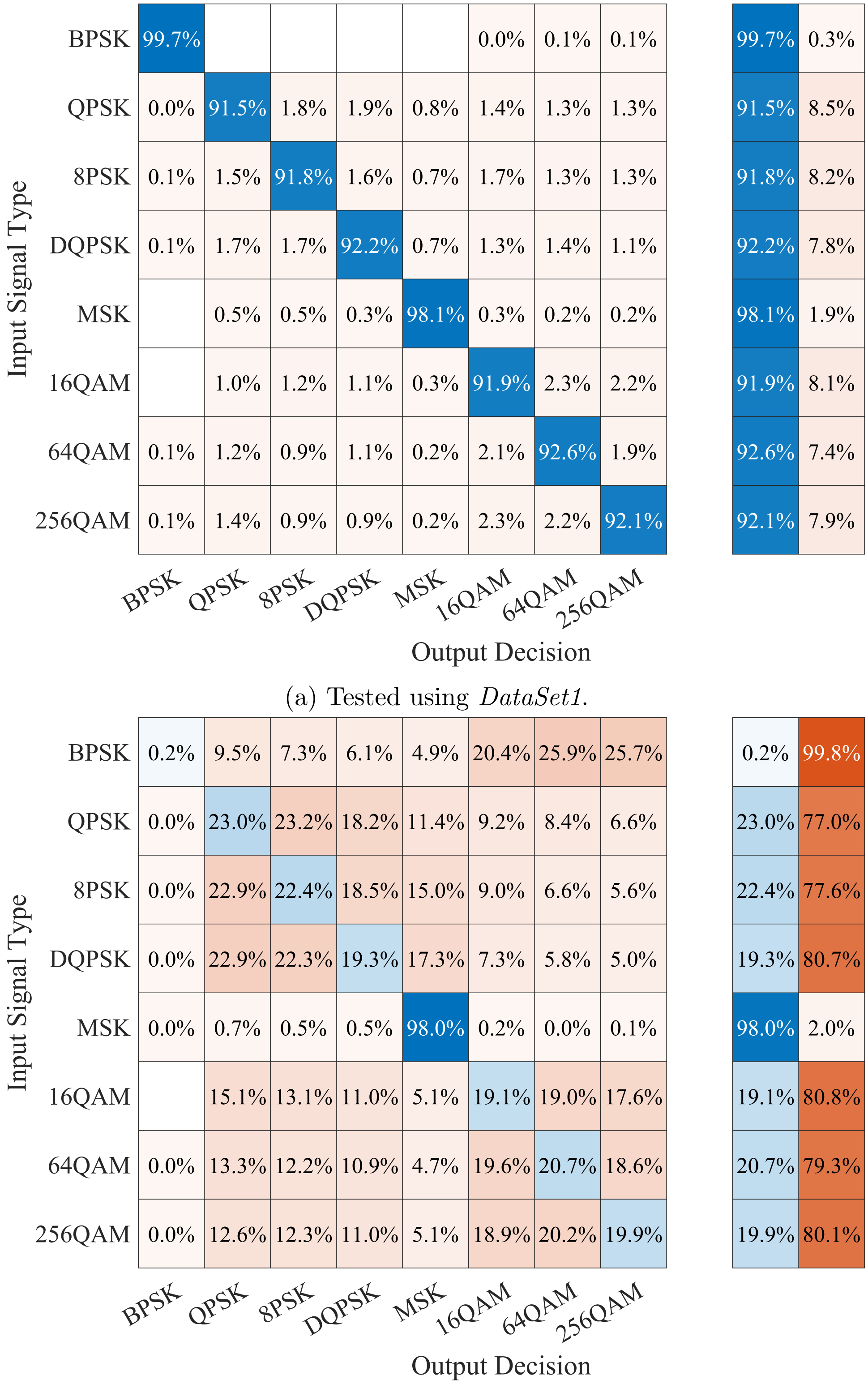
	BPSK	QPSK	8PSK	DQPSK	MSK	16QAM	64QAM	256QAM		
BPSK	84.1%	2.1%	1.7%	2.5%	2.2%	2.1%	2.5%	2.7%	84.1%	15.9%
QPSK	2.4%	85.0%	2.3%	2.3%	1.3%	2.2%	2.1%	2.2%	85.0%	15.0%
8PSK	2.5%	2.4%	84.2%	2.3%	1.8%	2.3%	2.5%	2.1%	84.2%	15.8%
DQPSK	2.9%	2.2%	2.0%	84.0%	1.7%	2.3%	2.6%	2.3%	84.0%	16.0%
MSK	2.0%	1.8%	2.0%	1.9%	86.6%	1.9%	1.7%	2.0%	86.6%	13.4%
16QAM	2.2%	2.4%	1.8%	2.9%	1.4%	84.2%	2.5%	2.6%	84.2%	15.8%
64QAM	2.1%	2.9%	2.3%	2.5%	1.4%	2.9%	83.0%	3.0%	83.0%	17.0%
256QAM	2.1%	2.6%	2.6%	2.9%	1.1%	2.6%	2.4%	83.6%	83.6%	16.4%

(a) Tested using *DataSet1*.

	BPSK	QPSK	8PSK	DQPSK	MSK	16QAM	64QAM	256QAM		
BPSK	11.5%	13.5%	11.8%	12.4%	13.4%	12.2%	12.4%	12.8%	11.5%	88.5%
QPSK	12.1%	12.4%	12.7%	13.2%	12.7%	13.3%	12.1%	11.4%	12.4%	87.6%
8PSK	11.5%	13.1%	13.3%	11.8%	13.2%	13.5%	11.6%	11.9%	13.3%	86.7%
DQPSK	11.4%	13.7%	12.3%	12.1%	13.3%	12.4%	12.2%	12.6%	12.1%	87.9%
MSK	10.2%	11.2%	10.7%	11.1%	23.2%	11.8%	10.9%	11.0%	23.2%	76.8%
16QAM	11.6%	12.5%	12.9%	12.7%	12.2%	12.8%	12.6%	12.6%	12.8%	87.2%
64QAM	11.4%	11.7%	12.6%	13.1%	14.0%	12.6%	12.0%	12.6%	12.0%	88.0%
256QAM	11.3%	12.6%	12.4%	12.4%	12.5%	12.7%	13.1%	13.1%	13.1%	86.9%

(b) Tested using *DataSet2*.

FIG. 28: Confusion Matrix of RESNET trained using *DataSet1*.



Input Signal Type	BPSK	QPSK	8PSK	DQPSK	MSK	16QAM	64QAM	256QAM		
BPSK	99.7%					0.0%	0.1%	0.1%	99.7%	0.3%
QPSK	0.0%	91.5%	1.8%	1.9%	0.8%	1.4%	1.3%	1.3%	91.5%	8.5%
8PSK	0.1%	1.5%	91.8%	1.6%	0.7%	1.7%	1.3%	1.3%	91.8%	8.2%
DQPSK	0.1%	1.7%	1.7%	92.2%	0.7%	1.3%	1.4%	1.1%	92.2%	7.8%
MSK		0.5%	0.5%	0.3%	98.1%	0.3%	0.2%	0.2%	98.1%	1.9%
16QAM		1.0%	1.2%	1.1%	0.3%	91.9%	2.3%	2.2%	91.9%	8.1%
64QAM	0.1%	1.2%	0.9%	1.1%	0.2%	2.1%	92.6%	1.9%	92.6%	7.4%
256QAM	0.1%	1.4%	0.9%	0.9%	0.2%	2.3%	2.2%	92.1%	92.1%	7.9%

Output Decision

(a) Tested using *DataSet1*.

Input Signal Type	BPSK	QPSK	8PSK	DQPSK	MSK	16QAM	64QAM	256QAM		
BPSK	0.2%	9.5%	7.3%	6.1%	4.9%	20.4%	25.9%	25.7%	0.2%	99.8%
QPSK	0.0%	23.0%	23.2%	18.2%	11.4%	9.2%	8.4%	6.6%	23.0%	77.0%
8PSK	0.0%	22.9%	22.4%	18.5%	15.0%	9.0%	6.6%	5.6%	22.4%	77.6%
DQPSK	0.0%	22.9%	22.3%	19.3%	17.3%	7.3%	5.8%	5.0%	19.3%	80.7%
MSK	0.0%	0.7%	0.5%	0.5%	98.0%	0.2%	0.0%	0.1%	98.0%	2.0%
16QAM		15.1%	13.1%	11.0%	5.1%	19.1%	19.0%	17.6%	19.1%	80.8%
64QAM	0.0%	13.3%	12.2%	10.9%	4.7%	19.6%	20.7%	18.6%	20.7%	79.3%
256QAM	0.0%	12.6%	12.3%	11.0%	5.1%	18.9%	20.2%	19.9%	19.9%	80.1%

(b) Tested using *DataSet2*.

FIG. 29: Confusion Matrix of the capsule network trained using *DataSet1*.

The confusion matrix shown in Fig. 29a shows overall good performance with all average performances being at or greater than 91%. BPSK and MSK have the best overall classification accuracy. As observed for the CNN and RESNET confusion matrices, there is a tendency for PSK type signals to be confused with other PSK type signals and likewise for QAM type signals to be confused with other QAM type signals. The greatest confusions are occurring among the QAM type signals which is reasonable considering their similarities. Overall, this is excellent performance which is noticeably superior to the CNN and RESNET. The confusion matrix in Fig. 29b shows the results for when the network is tested using *DataSet2*. These results show similar confusions occurring for the respective PSK type and QAM type signals with MSK performing the best overall. As seen for the CNN network, the capsule network is also able to activate on the features with different characteristics present in *DataSet2* but with reduced accuracy.

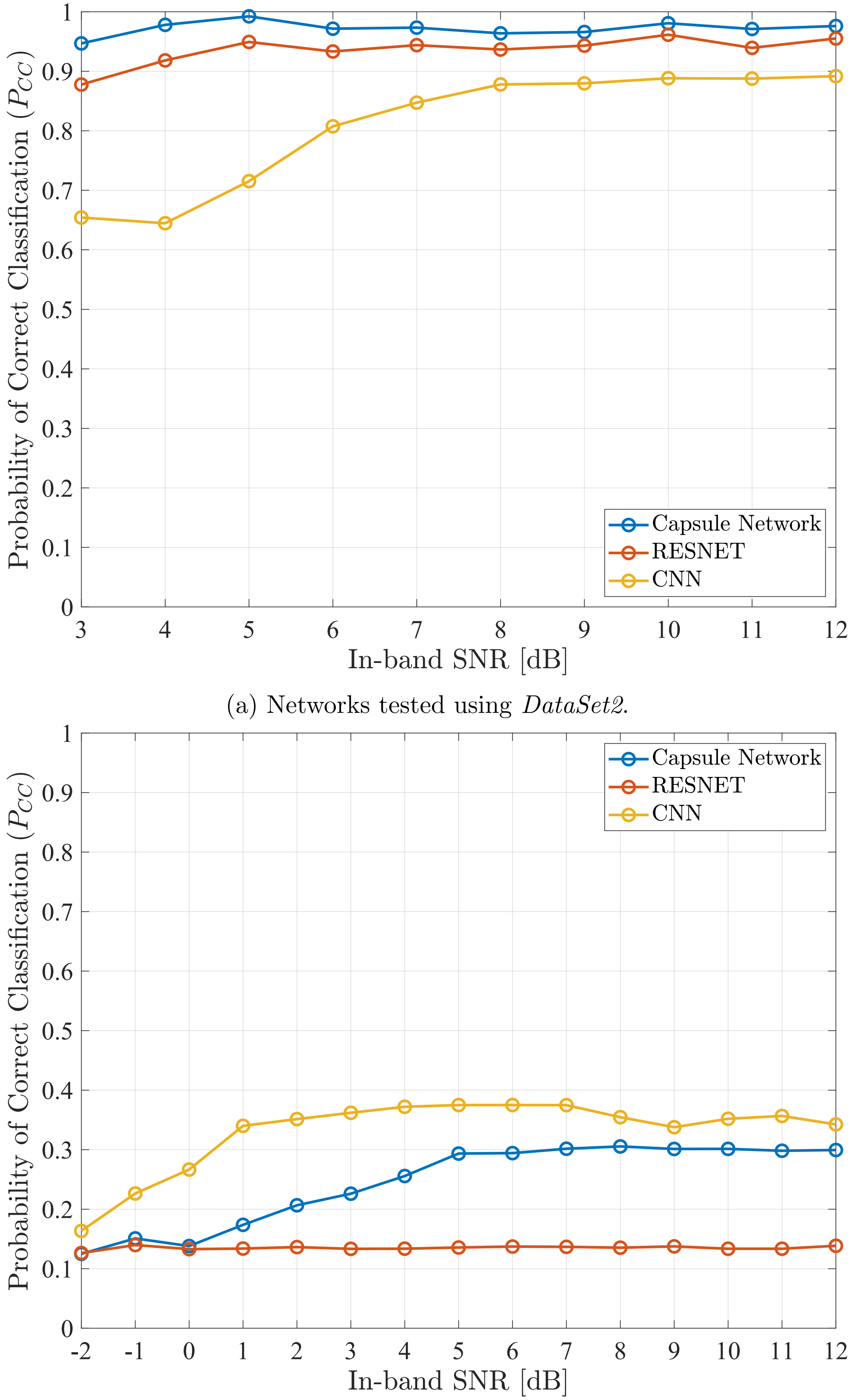
6.2 DATASET2 RESULTS

As introduced in Chapter 5, there is a second dataset that will be used to assess the performance of all networks. The exact same parameters specified in Tables 1, 2 and 3 were used for training with *DataSet2*. All networks were trained using $78,400$ training signals. The capsule network was trained for 5 epochs, the RESNET was trained for 14 epochs and the CNN was trained for 12 epochs. As before, once each respective network was trained it was then tested using $28,000$ testing signals. The PCC vs in-band SNR can be assessed using Fig. 30a. These same networks that were trained with *DataSet2* were then tested using $112,000$ signals from *DataSet1*. These PCC vs in-band SNR results are displayed in Fig. 30b

The main intention of the *DataSet2* is to assess how well the various networks generalize to out of distribution characteristics present in the testing data which were not present in the training data. However, viewing the performance of the various networks that are trained with *DataSet2* will determine if the network is able to learn these features and thus establishes baseline performance. This baseline performance is important as it will determine if the networks are able to learn these different feature characteristics. The performance of each network will be addressed in subsequent sections by looking at the confusion matrices.

CNN Performance:

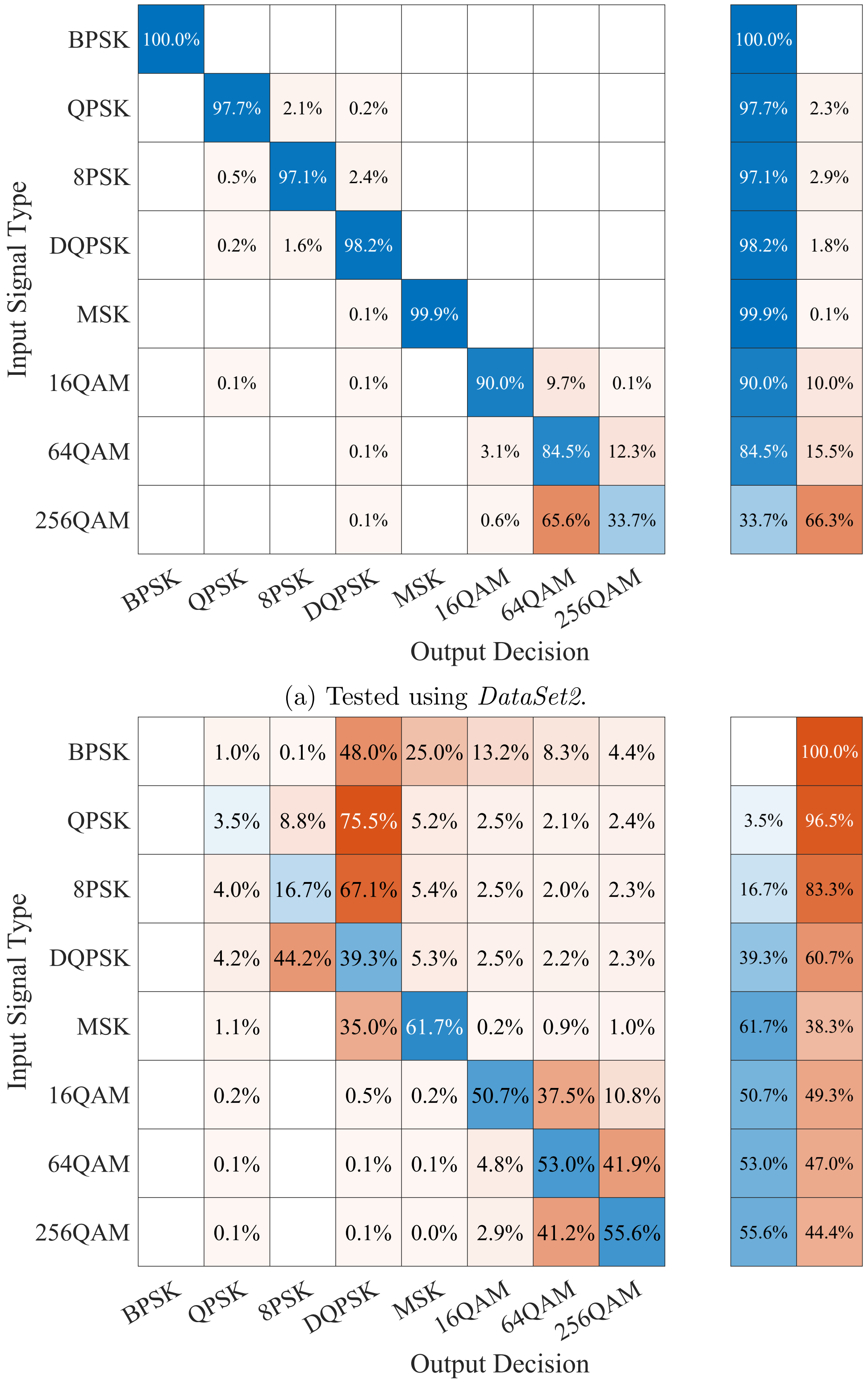
As seen in Fig. 30a, the CNN consistently has lower PCC across the entire in-band SNR range when compared to RESNET and capsule network. Nevertheless, these are still good results especially for the higher in-band SNR range. The confusion matrix in Fig. 31a suggests that similar class confusions among the PSK type and QAM type classes is present. For the CNN, the QAM type signals experience the worst confusion with the 64-QAM and 256-QAM being the worst offenders. What is unusual is that 16-QAM has 90% classification accuracy and 256-QAM has 33.7% classification accuracy. This is a substantial variance within the QAM type classes. At any rate, these results suggest that the CNN is able to activate on features and characteristics present in *DataSet2*. This same network trained with *DataSet2* was also tested with *DataSet1* with the confusion matrix results of this simulation being presented in Fig. 31b. This confusion matrix shows that the CNN is not able to activate on the features in *DataSet1* when only trained with *DataSet2*.



(a) Networks tested using *DataSet2*.

(b) Networks tested using *DataSet1*.

FIG. 30: Probability of correct classification vs in-band SNR for networks trained using *DataSet2*.



(a) Tested using *DataSet2*.

(b) Tested using *DataSet1*.

FIG. 31: Confusion Matrix of CNN trained using *DataSet2*.

RESNET Performance:

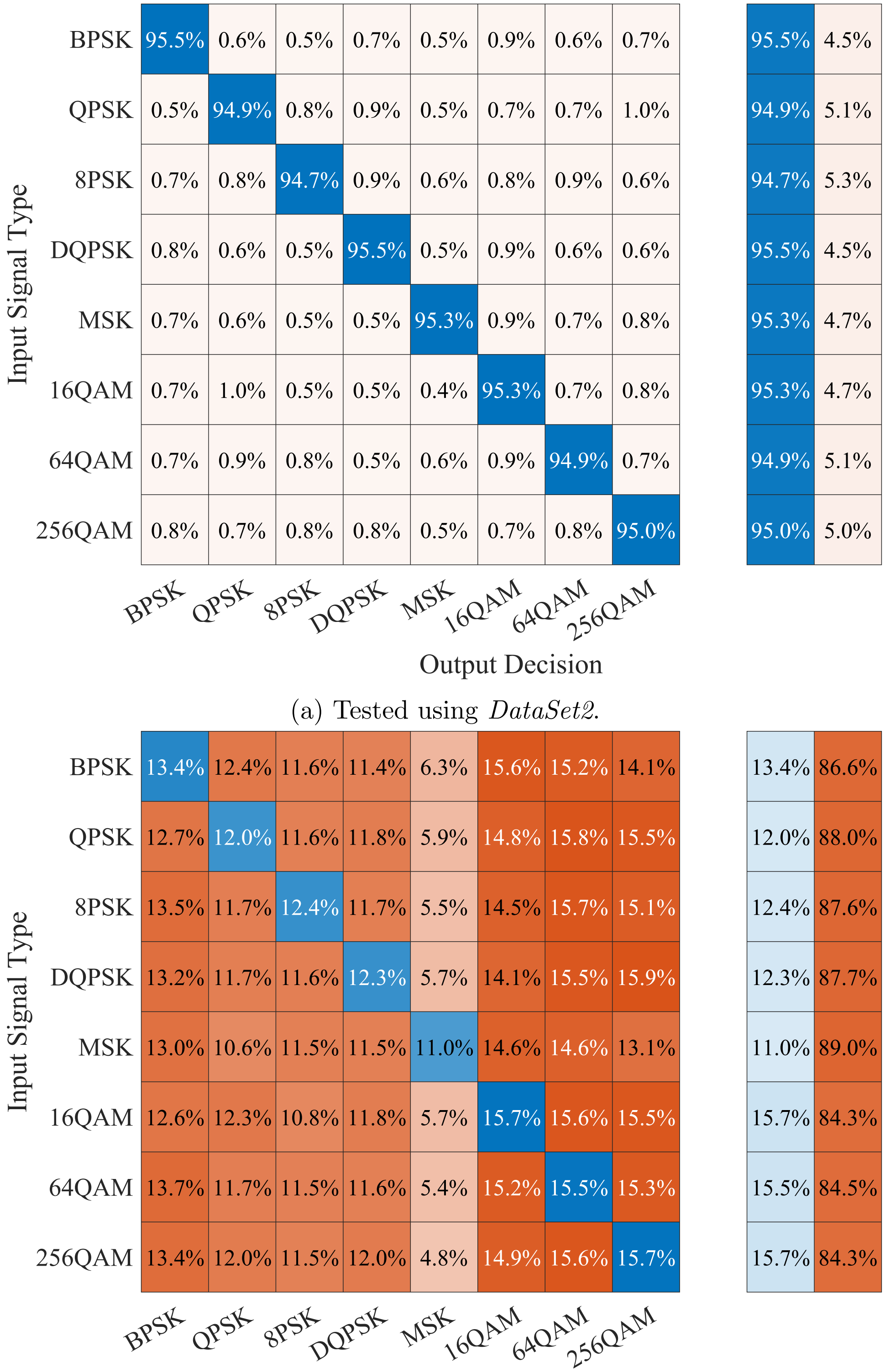
The performance of the RESNET is excellent across the entire in-band SNR range as shown in Fig. 30a. It can also be observed that the performance is not severely impacted for the lower end of in-band SNR values. However, when this trained network is then tested with signals from *DataSet1* it has lower PCC results as shown in Fig. 30b. In fact, these results hover around 12.5% suggesting that the RESNET is simply guessing and is not able to activate on features with varied characteristics in *DataSet1*. These results can be assessed further by viewing the confusion matrix corresponding to this plot.

The confusion matrix for the RESNET trained and tested using *DataSet2* is illustrated in 32a. This confusion matrix shows little confusion for the various classes with all classes scoring in the mid $90's\%$. Because of the high classification accuracy values, there does not seem to be the same PSK type and QAM type confusion signatures that were apparent in previous confusion matrices. Overall, this is excellent performance and suggests that the RESNET has learned to distinguish the various digital modulation schemes with great accuracy. However, the confusion matrix for the RESNET that is tested with signals from *DataSet1* reveals that the network is simply guessing with a slight preference for the QAM type classes. This confusion matrix is found in Fig. 32b. Ultimately, this is undesirable performance which suggests that the RESNET is very sensitive to characteristic changes present in the features.

Capsule Network Performance:

Finally, the results of the capsule network when trained and tested with *DataSet2* will be assessed. As is clear in Fig. 30a, the capsule network has excellent PCC over the entire in-band SNR range. Even for the lower in-band SNR values, the performance is still very desirable. When this trained network is then tested using signals from *DataSet1* it has decreased PCC performance over the entire in-band SNR range as shown in Fig. 30b.

By looking at a confusion matrix, the class performance may be assessed. This confusion matrix for the capsule network trained and tested using *DataSet2* is provided in Fig. 33a. This shows excellent performance with all classes scoring in the mid to high $90's\%$. The respective PSK type and QAM type confusion signatures are present in this network. This suggests that the capsule network is learning to recognize PSK type and QAM type signatures with more consistency than the RESNET was able to. Overall, these are excellent results and suggest that the capsule network is able to activate on the features present in *DataSet2*. The confusion matrix in Fig. 33b shows the results for when this trained network is tested with



(a) Tested using *DataSet2*.

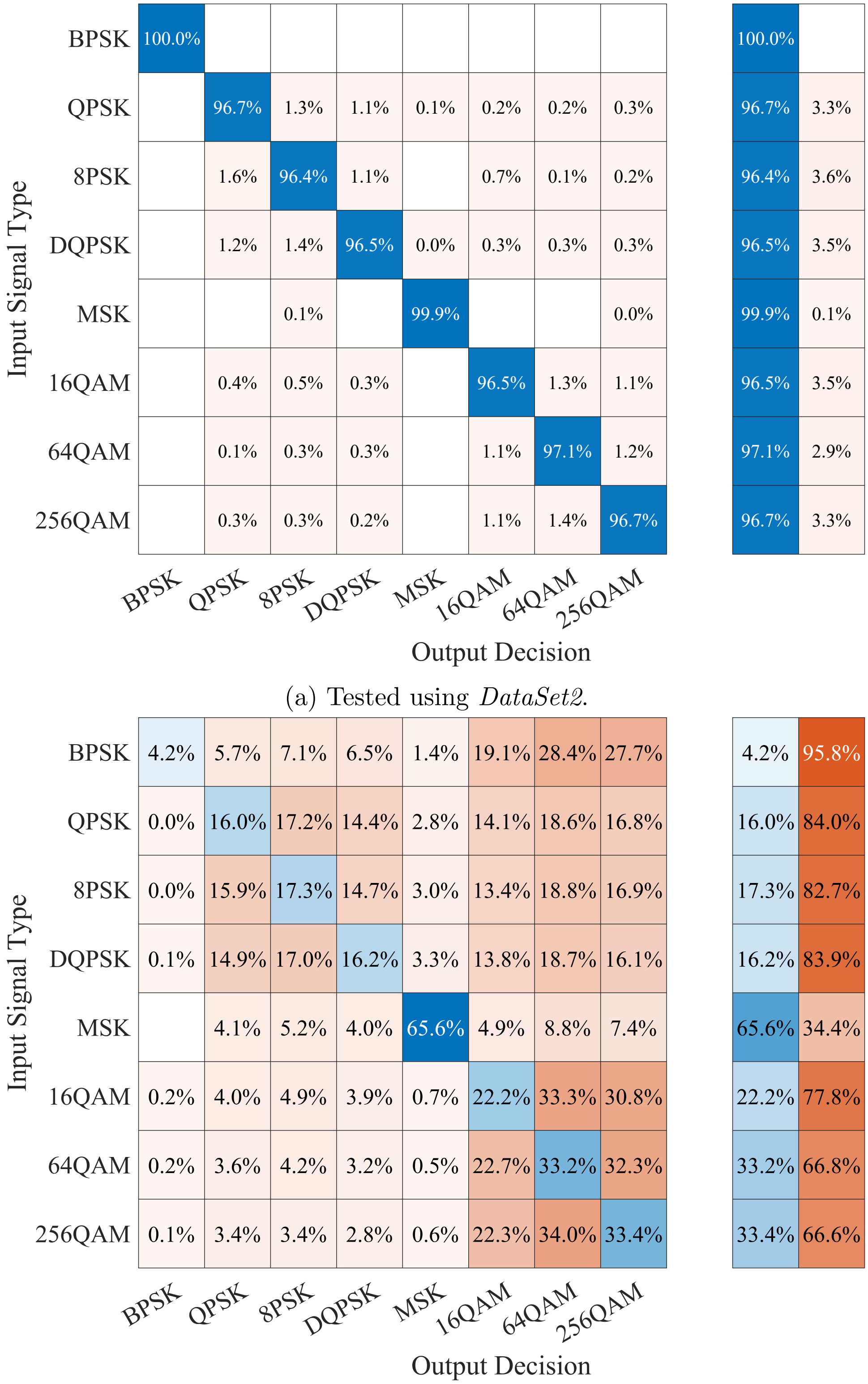
(b) Tested using *DataSet1*.

FIG. 32: Confusion Matrix of RESNET trained using *DataSet2*.

signals from *DataSet1*. In this case the MSK has excellent class performance, but all other classes have severely decreased performance. The PSK type and QAM type are confused with each other respectively.

The goal of these networks is to successfully classify digital modulation schemes in *DataSet1* and *DataSet2*. It was understood that the similarities between classes may cause inter-class confusion, and this was visible in the confusion matrix plots. Specifically, PSK type classes and QAM type classes were more readily incorrectly classified as a class of the same type. It was also expected that the networks would learn to recognize excursions and characteristics that belong to that respective class. It seems reasonable that some of these excursions or characteristics may be similar between the PSK types and QAM types. This suggests that the networks are truly learning PSK type and QAM type features and are activating on these features.

Furthermore, it was expected that the networks would be better able to classify higher in-band SNR signals because the signal power was much greater than the noise power. However, as the in-band SNR was decreased the performance was impacted. This also suggests that the characteristics that are learned are harder to distinguish when saturated with noise. This is more evidence suggesting that the networks are truly learning class specific features and characteristics. Thus far, the experiments involving training and testing with the same dataset are very desirable and seem to suggest that the networks are able to correctly classify signals when trained with data that represents all underlying feature characteristics in the testing data.



(a) Tested using *DataSet2*.

(b) Tested using *DataSet1*.

FIG. 33: Confusion Matrix of the capsule network trained using *DataSet2*.

6.3 GENERALIZATION DISCUSSION

The above results show that excellent classification accuracy can be achieved with the respective datasets. As noted in the chapter introducing the datasets, these datasets have certain characteristics not present in the other, namely their CFO values, Δf_c, are on disjoint intervals. When a network is not exposed to specific feature characteristics during training, it is not clear how the network will perform when exposed to new characteristics during testing. This is why the simulations trained with one dataset were then independently tested with both datasets. The goal was to determine if the respective networks would still be capable of activating on features with different CFO characteristics.

It is apparent from the simulation results that variations in the CFO that were not present during training have adverse impacts on performance. As pointed out in the background chapter on digital modulation schemes, different values for the CFO may cause inter carrier interference or attenuate the overall received signal. However, when the proposed network was trained and tested with *DataSet1*, it performed well and also when it was trained and tested with *DataSet2* it performed well. Thus, all networks are capable of learning to activate on any range of CFO characteristics when these characteristics are present in the training data.

Ultimately, these simulations show that whatever characteristics were learned by the respective networks are very sensitive to changes in the CFO. This is clearly demonstrated in the above simulations as the primary difference between *DataSet1* and *DataSet2* are the CFO intervals. One argument might be that, because the networks were not exposed to the respective CFO characteristics during training, they cannot be expected to recognize the different characteristics. To assess this idea, the datasets can be combined and then trained and tested using a combined dataset.

6.4 COMBINED DATASET RESULTS

By combining the datasets, a wider range of CFO characteristics will be present during training. This will theoretically enable the network to activate on features with a broader range of CFO characteristics. To do this $80,000$ signals from *DataSet1* and $80,000$ signals from *DataSet2* were combined together and used for training. Careful attention was given to have even distributions of class labels from each dataset. Of the $160,000$ signals, $112,000$ were used for training. The capsule network was trained for 10 epochs, the RESNET was trained for 14 epochs and the CNN was trained for 12 epochs. Once training was concluded,

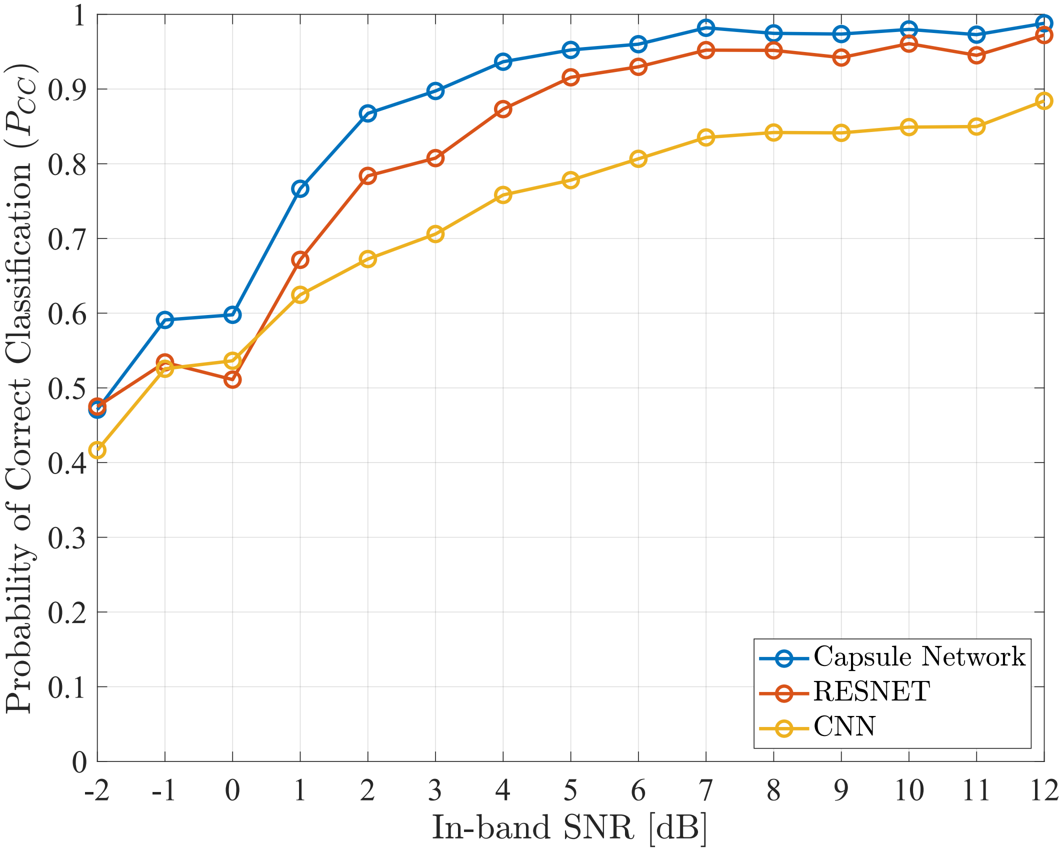


FIG. 34: Probability of correct classification vs in-band SNR for networks trained and tested using a combined dataset.

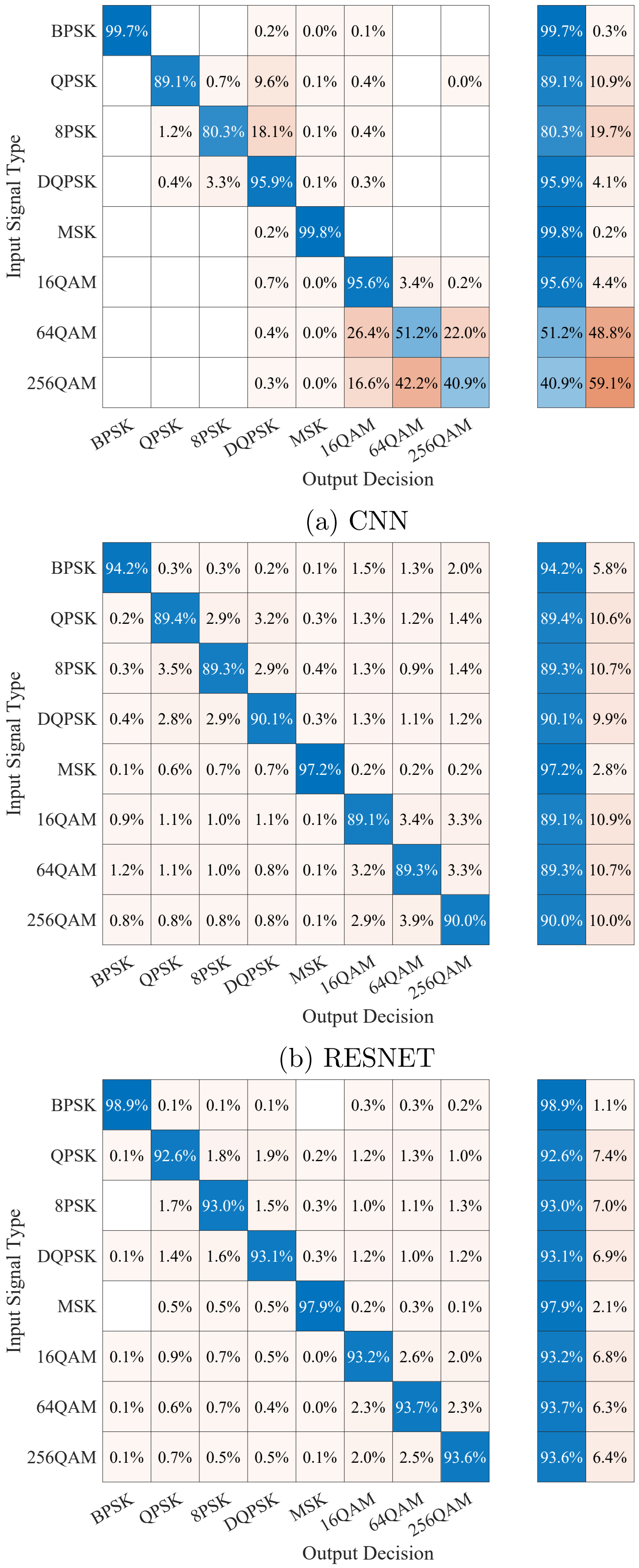
$40,000$ signals were used for testing each network. The PCC vs in-band SNR for the various networks is shown in Fig. 34. This shows good performance for high in-band SNR values with performance being reduced as the in-band SNR is decreased.

The results of using a combined dataset are desirable and show that the networks are able to achieve high accuracy for a wide range of CFO values only if those CFO characteristics are present during training. At a high level, the network learns by observing a difference between what was predicted and the labeled data. When an incorrect classification occurs, error is propagated to the weights. This error tunes individual weights to have a higher activation on features present in the input signal. If the network is only exposed to inputs with specific feature characteristics, then only these feature characteristics will be correlated with the respective class. At any rate, the networks are proficient at learning to classify digital modulation schemes. The confusion matrices for these various networks can be found in Fig. 35. These matrices show very similar results as described for the other simulations

TABLE 4: Ranked Network Performance

Rank	Network	Overall Accuracy	Training Dataset	Testing Dataset
1	Capsule	97.5%	*DataSet2*	*DataSet2*
2	RESNET	95.1%	*DataSet2*	*DataSet2*
3	Capsule	94.5%	Mix	Mix
4	Capsule	93.7%	*DataSet1*	*DataSet1*
5	RESNET	91.1%	Mix	Mix
6	CNN	87.7%	*DataSet2*	*DataSet2*
7	RESNET	84.3%	*DataSet1*	*DataSet1*
8	CNN	81.6%	Mix	Mix
9	CNN	78.4%	*DataSet1*	*DataSet1*
10	CNN	39.4%	*DataSet1*	*DataSet2*
11	CNN	34.9%	*DataSet2*	*DataSet1*
12	Capsule	27.9%	*DataSet2*	*DataSet1*
13	Capsule	26.2%	*DataSet1*	*DataSet2*
14	RESNET	14.0%	*DataSet1*	*DataSet2*
15	RESNET	13.4%	*DataSet2*	*DataSet1*

with confusions occurring most often among PSK type signals and QAM type signals respectively. The capsule network has the best overall performance but is closely followed by the RESNET. Table 4 ranks the performances of the above simulations with regards to their overall accuracy. This is provided to help summarize the performance of all simulations. This comparison of performance for various artificial neural network topologies for digital modulation classification have been submitted for presentation to IEEE Communication Conference COMM2022.



	BPSK	QPSK	8PSK	DQPSK	MSK	16QAM	64QAM	256QAM		
BPSK	99.7%			0.2%	0.0%	0.1%			99.7%	0.3%
QPSK		89.1%	0.7%	9.6%	0.1%	0.4%		0.0%	89.1%	10.9%
8PSK		1.2%	80.3%	18.1%	0.1%	0.4%			80.3%	19.7%
DQPSK		0.4%	3.3%	95.9%	0.1%	0.3%			95.9%	4.1%
MSK				0.2%	99.8%				99.8%	0.2%
16QAM				0.7%	0.0%	95.6%	3.4%	0.2%	95.6%	4.4%
64QAM				0.4%	0.0%	26.4%	51.2%	22.0%	51.2%	48.8%
256QAM				0.3%	0.0%	16.6%	42.2%	40.9%	40.9%	59.1%

(a) CNN

	BPSK	QPSK	8PSK	DQPSK	MSK	16QAM	64QAM	256QAM		
BPSK	94.2%	0.3%	0.3%	0.2%	0.1%	1.5%	1.3%	2.0%	94.2%	5.8%
QPSK	0.2%	89.4%	2.9%	3.2%	0.3%	1.3%	1.2%	1.4%	89.4%	10.6%
8PSK	0.3%	3.5%	89.3%	2.9%	0.4%	1.3%	0.9%	1.4%	89.3%	10.7%
DQPSK	0.4%	2.8%	2.9%	90.1%	0.3%	1.3%	1.1%	1.2%	90.1%	9.9%
MSK	0.1%	0.6%	0.7%	0.7%	97.2%	0.2%	0.2%	0.2%	97.2%	2.8%
16QAM	0.9%	1.1%	1.0%	1.1%	0.1%	89.1%	3.4%	3.3%	89.1%	10.9%
64QAM	1.2%	1.1%	1.0%	0.8%	0.1%	3.2%	89.3%	3.3%	89.3%	10.7%
256QAM	0.8%	0.8%	0.8%	0.8%	0.1%	2.9%	3.9%	90.0%	90.0%	10.0%

(b) RESNET

	BPSK	QPSK	8PSK	DQPSK	MSK	16QAM	64QAM	256QAM		
BPSK	98.9%	0.1%	0.1%	0.1%		0.3%	0.3%	0.2%	98.9%	1.1%
QPSK	0.1%	92.6%	1.8%	1.9%	0.2%	1.2%	1.3%	1.0%	92.6%	7.4%
8PSK		1.7%	93.0%	1.5%	0.3%	1.0%	1.1%	1.3%	93.0%	7.0%
DQPSK	0.1%	1.4%	1.6%	93.1%	0.3%	1.2%	1.0%	1.2%	93.1%	6.9%
MSK		0.5%	0.5%	0.5%	97.9%	0.2%	0.3%	0.1%	97.9%	2.1%
16QAM	0.1%	0.9%	0.7%	0.5%	0.0%	93.2%	2.6%	2.0%	93.2%	6.8%
64QAM	0.1%	0.6%	0.7%	0.4%	0.0%	2.3%	93.7%	2.3%	93.7%	6.3%
256QAM	0.1%	0.7%	0.5%	0.5%	0.1%	2.0%	2.5%	93.6%	93.6%	6.4%

(c) Capsule Network

FIG. 35: Confusion Matrix for each network that is trained and tested using the combined dataset.

CHAPTER 7

FPGA IMPLEMENTATION

Field Programmable Gate Arrays (FPGAs) excel at the implementation of real time, complex architectures into hardware. The internal architecture of an FPGA is defined by a hardware descriptive language (HDL) such as VHDL or Verilog. Though it has the appearance of software, this HDL code is often referred to as firmware because it is using verbose language to describe the implementation of synthesized hardware circuitry. FPGAs are highly customizable and lend well to applications with redundancy in the system architecture. These factors make FPGAs an ideal candidate for implementing feed-forward ANN for real world applications.

There are a number of inputs that must be considered when attempting to implement an architecture onto an FPGA such as the inputs/outputs needed, overall throughput of data, memory required for processing and the type of mathematical operations that are needed. There is often a balancing act that goes on between these areas which will be addressed in the following sections.

7.1 INPUT DATA AND REPRESENTATION

Most hardware designs begin with a black box diagram that specifies the inputs entering the box and the desired output leaving the box. In order for a feed-forward ANN to perform digital modulation classification, the raw in-phase and quadrature components would need to be extracted and digitized so that they may be processed by an FPGA. This could be achieved by a signal analyzer such as the one already introduced in Fig. 7. This signal analyzer would require additional analog to digital converters (ADC) on the in-phase and quadrature signal components respectively. The FPGA can only work with digitized signals; thus, the ADCs are necessary. The ADCs will need to be carefully chosen to have sufficient sample rates and bit precision. The ADCs will likely communicate using a serial peripheral interface (SPI). The SPI lines are what will effectively communicate the digitized in-phase and quadrature data to the FPGA. The goal would be to quickly perform live classifications; thus, there needs to be a trigger output indicating which digital modulation scheme is present. This could be as simple as having eight dedicated output wires (one for each digital modulation scheme).

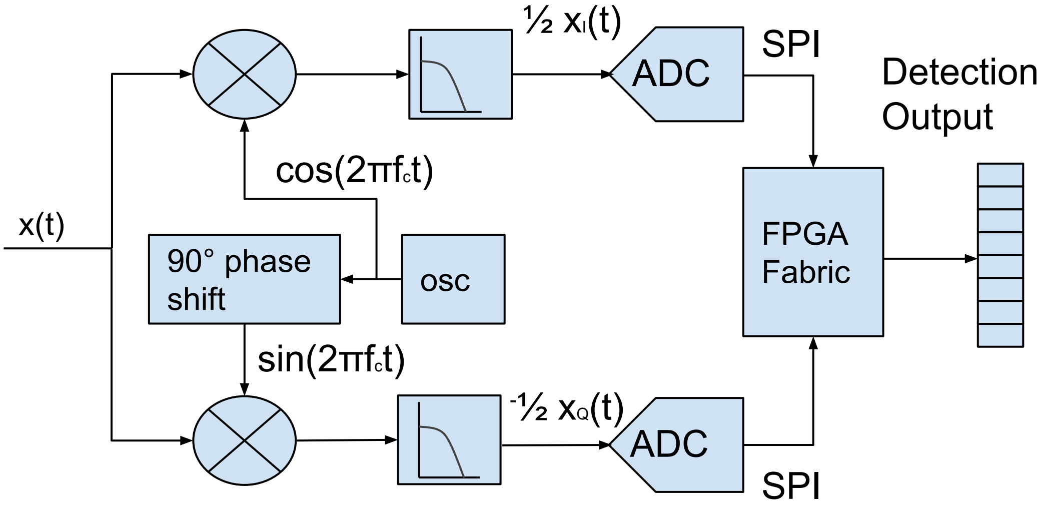


FIG. 36: High level black box of inputs and outputs to FPGA.

These detection wires would go high (turn on) if the corresponding digital modulation scheme is detected and stay low (stay off) otherwise. This black box diagram of the notional design can be found in Fig. 36. This diagram is very similar to a Software Defined Radio (SDR) RF front end. Now that the digital representation of the in-phase and quadrature data has been discussed, the data representation of the weights and bias values can be examined.

Weight values are often fractional values which are easily stored on an FGPA if fixed point notation is used. Fixed point notation can represent any signed value within a certain amount of precision based on the number of bits used. Normally, binary values represent a decimal value as a power of two. Thus, the nth binary bit represents 2^n in decimal for $n = 0, \dots, N$ with N being the bit length. Fixed point notation assumes that the decimal point is placed in the string of binary bits. Bits that are on the right hand side of the decimal point are treated as 2^{-m}. Here m represents the mth bit from the decimal point. Thus, a four bit binary value 1001, with the decimal point assumed to be as shown here, 1.001, would represent the value $1 \times 2^0 + 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-2} = 1 + \frac{1}{8} = 1.125$. The values to the right of the decimal are referred to as fractional bits. With fixed point precision a 16 bit binary value (with 1 bit being the sign bit and 15 fractional bits) precision to within $\frac{1}{2^{15}}$ can be achieved.

By representing weights as fixed point bit arrays, the hardware within the FPGA can be greatly simplified with a modest cost in precision. The weights can be stored either on the

FPGA as on chip memory or on a memory device external to the FPGA. For implementation of the capsule network on a mid grade FPGA such as a Cyclone 10, up to $11,740$ M20k on chip memory blocks are available which can be used to store the adaptive parameters internal to the FPGA (depending on parity bits used and ADC bit precision). Other FPGA such as Kintex-7 have available 34,380 k bit block RAM available. The proposed capsule network has $2,076,872$ adaptive parameters and will fit on Cyclone 10 or on the Kintex-7. Many embedded SDR platforms, such as USRP embedded options include external memory in the gigabyte range which will also fit this network. It may be preferable to utilize off chip memory to simplify the internal architecture of the FPGA.

Furthermore, the digitized in-phase and quadrature data may also be represented in fixed point notation. The available bit precision will be dependent on the chosen ADC, but 16 to 18 bits are commonly available. The proposed network requires inputs of $32,768$ samples of both in-phase and quadrature data. This means that a buffer size of $2 \times 32,768$ would be needed and thus an additional 66 M20k memory blocks may be needed.

The above discussion describes the type of memory constraints that are required for implementing the proposed network and what type of data representation could be used. Now that the data types are described, the conceptual architecture can be examined.

7.2 CONCEPTUAL ARCHITECTURE

To begin with, the received input will need to be conveniently stored in an accessible manner. The ADCs would be constantly sampling the receiving signal with the digitized value being stored in its respective buffer. Various options exist such as a first in first out (FIFO) buffer or a circular buffer. Circular buffers are easily implemented in firmware and can account for deterministic latency if needed. The ADC would write its value to the memory from a zero indexed first register to the maximum address value. A simple counter can be used to keep track of which address is written. When a new sample is ready, it will be written to the buffer and the counter will be incremented. When the max address is reached, the counter will reset. This process will overwrite old values with new values. The benefit of a circular buffer is that it is easy to implement and one does not need to worry about overfilling it, like with a FIFO buffer. A trigger event will be required to arrest the sampling process. This will freeze the buffer or cause it to pause temporarily so that its contents may be read by following modules.

Once the circular buffer containing the input data to the network is ready, this data will need to be normalized and passed to the feature extraction layer. Normalization of the input data requires that the maximum value in the buffer be divided by every element in the buffer. Finding the maximum value in the buffer may easily be obtained by reading each element in the buffer to find the greatest value. Alternatively, while the ADC values are being sampled, a watch process could be employed which watches for the maximum value. If the latest ADC sample is greater than the previous max value, then this becomes the new assumed maximum. Whenever the counter that controls the address is reset, this watchdog can be reset. This would allow the maximum value to be known before the trigger event. Division on an FPGA is possible but undesirable as it can take multiple clock cycles to process. Instead of dividing every memory element by the maximum sampled value, the inverse of the maximum value can be found once, and then this value may be used for multiplication of each element in the input circular buffer. The new normalized value may be rewritten to the circular buffer or it may be written to another circular buffer. These simple steps will result in normalization of the input data.

The normalized input data is now ready to be transferred to the feature extraction section which is a convolution layer. For the proposed network, all CNN kernel values are $1 \times X$ in length and the mathematical operation of the CNN is similar to a Finite Impulse Response (FIR) filter whose equation is given by equation (45) where $y(n)$ is the output of the filter which uses the impulse response coefficients b_i multiplied by the N previous values.

$$y(n) = \sum_{i=0}^{N} b_i x(n-i) \tag{45}$$

FIR filters are widely implemented on FPGAs for digital processing application and thus the mathematical operations of the CNN layers could be modeled after well established FIR designs. These would involve loading the weights from memory into a multiplier and then performing the multiplication with the weight values and the previous layer inputs. The results of these multiplications can be stored in flip-flops and do not need to be loaded into the on-chip memory. Modifications to FIR modules to include strides must be considered. As suggested in [29] and shown in Fig. 37, the convolutional operation of a $1 \times N$ filter can be achieved with cascaded multiply and accumulate modules. For CNN layers with multiple filters, this operation would have to be repeated multiple times for each layer, and those resulting values should also be stored in flip flops or in on chip memory. This process would be as shown in equation (46) with K kernels and s stride. As specified in [29] a controller would be necessary to ensure that the correct portions of the feature map and the correct corresponding weights are loaded into the convolutional module. It is worth noting that strides are implemented by an increment in address memory of s. Furthermore, padding is

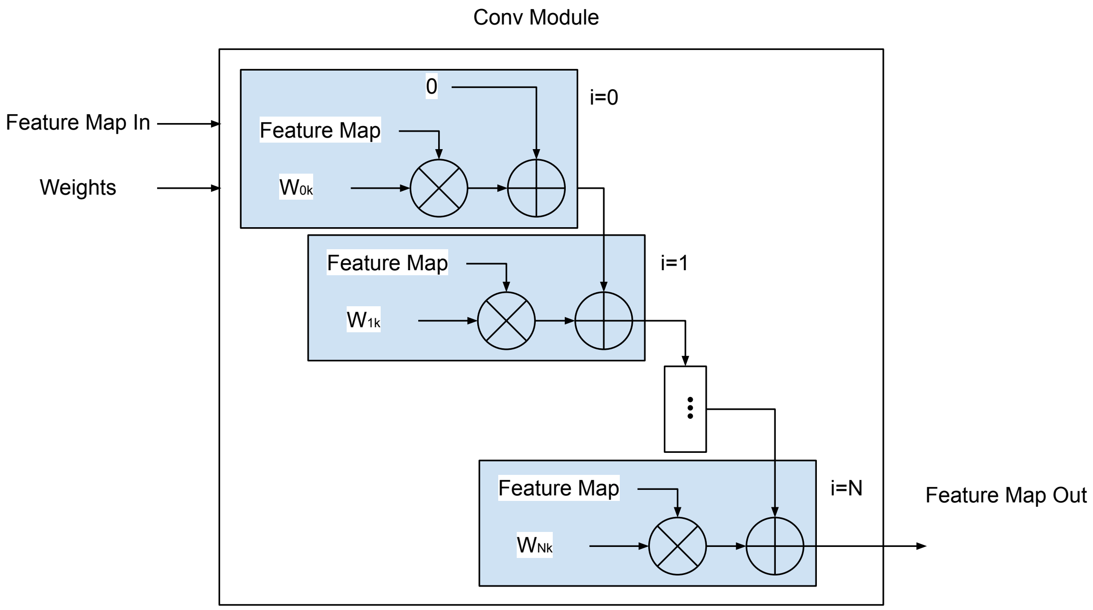


FIG. 37: Notional convolutional module.

implemented by adding zero values to the edges of the feature map. The controller is the key to successful implementation of the convolutional module. Furthermore, [29] also asserts that the cascaded MAC elements may require additional clock cycles for the output to be ready. To meet FPGA timing, delay registers or multi-cycle timing may be used.

$$a_{ji}(n) = \sum_{i=0}^{N} w_{ji} x((1+s)n - i), \text{for j=1,...,K} \tag{46}$$

The last step of the CNN layer is to be passed through the non-linear activation function. The hyperbolic tangent function, tanh, has many smooth components to it. This can be approximated to a specified precision by means of a CORDIC or look-up-table. As suggested in [29], the activation function may be included inside the convolutional block. This option may increase the throughput necessary for the convolutional module. Keeping the activation functions as a selectable, separate module allows the modules to be multipurpose and enables the controller to easily adapt to various activation functions. This chosen option will result in firmware that is easily converted to various network topologies. It is also worth noting that the batch normalization layers are readily used to normalize input data during training. For deployment these layers can be ignored.

The feature extraction layer would need to run the CNN over the entire input data sequence length. With an increase in complexity, multiple modules could process the initial input data sequence starting and stopping at specified points. This increase in complexity would require more multipliers, but it would result in faster data throughput. Depending on the input data rate, this might not be necessary.

Similarly, the primary caps layer is a series of CNN layers and activation functions as described above. The weights for each specific CNN would need to be loaded from memory and multiplied with the value stored in flip flops from the previous layer. Depending on the desired data throughput rate, all branches could be processed in parallel or they could be processed sequentially and their values stored in flip flops. By processing in parallel, more resources will be needed. By processing them sequentially, fewer resources would be needed, but the overall throughput rate would be less. It is also worth noting that, as shown in Fig. 22, fewer operations are needed for following layers; thus, the operations can be performed faster and with fewer resources.

The fully connected layers will need to implement equation (35). Here, a multiple and accumulate (MAC) circuit could be synthesized to perform the multiplications and summation to find the activation. The output of this network would also need to be passed through a ReLU activation function using either a CORDIC or look-up-table to approximate the ReLU. The result of the branch will be one value stored in a flip flop array. All of these values can then be combined depth wise and passed through the softmax as shown in equation (36).

To model the softmax in hardware, a CORDIC or look-up-table can also be used. Other approximations of the exponential terms may be pursued by means of an exponential series expansion. Multiple options exist depending on the desired complexity and throughput of data. This section was presented at a high level in order to describe the general process by which the proposed network could be implemented onto an FPGA. Depending on the required throughput and the available resources, various firmware architectures could be implemented. Perhaps one of the simplest would be as shown in Fig 38 where a sequential design is pursued.

Fig 38 is a high level diagram which shows the various blocks needed and the flow of data between them. This model assumes an SDR front end which samples and digitizes the in-phase and quadrature signal components which are then passed to a circular buffer. The controller can be a simple state machine running continuously on the FPGA. When the controller desires to detect a digital modulation scheme, it will freeze the circular buffer and

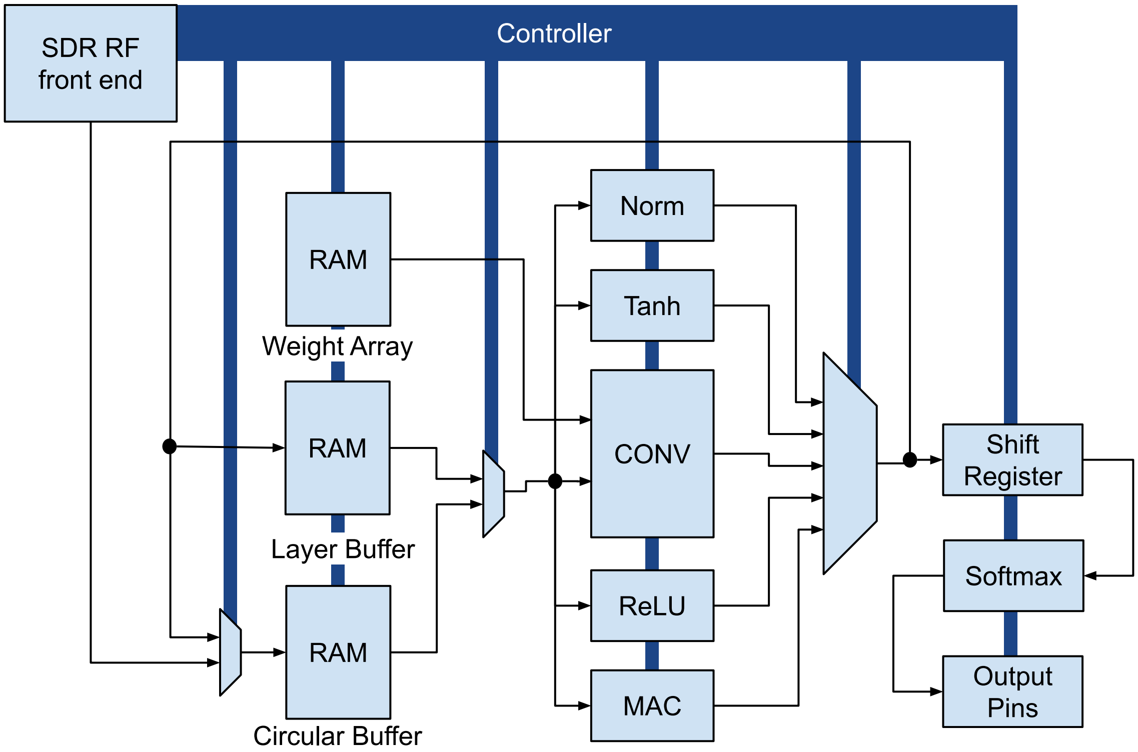


FIG. 38: High level block diagram of the proposed network being implemented onto an embedded SDR.

then manipulate the multiplexers so that data flows from the circular buffer to the normalization module which can then be re-written into the circular buffer. Then, the controller will load the appropriate weights from the weight array and select the corresponding normalized input values. These will then flow into the CONV module which acts as a convolutional layer with the results being stored in the layer buffer. This CONV module could actually be 64 parallel FIR modules to greatly reduce the throughput time. After this, the filtered data would need to be passed to the non-linear activation function which is the hyperbolic tangent module. Once the circular buffer has had all of its data processed, it can continue to sample live values from the ADCs for the next classification. The normalization module could be multi-purposed and also act as the pooling layer.

This process would then repeat for the various branches of the capsule network with the controller being in charge of loading the correct weights at the correct time. At the end of the convolutional layers, a similar process would occur for the fully connected layers. In this case, the data would be passed from the layer buffer and into the MAC module and then eventually into the ReLU module. At this point, the outputs of the ReLU may be shifted into a shift register. The shift register will contain a list of fixed point values, each being the output of one branch. These values would then be passed into the softmax layer which would perform the classification. The controller could have a threshold detector that reports a fault if the results are close to random. Ultimately, this is a very simple design which could be implemented onto an FPGA. Furthermore, this design could be used to implement any CNN architecture. If very short through-puts are desired, this design could be pipe-lined to improve performance at the cost of complexity.

7.3 OBSTACLES AND CONSIDERATIONS

The largest obstacles to implementing a complex architecture on an FPGA are access to memory and the number of multipliers needed. Either FPGA suggested earlier has sufficient memory to implement the proposed network. The Cyclone 10 has 384 multipliers available and the Kintex 7 has 1,920 multipliers. Either selection will support a sequential implementation. Depending on the desired throughput, it is very likely that either FPGA could be used to implement this design. However, prototyping and experimentation would be required.

In Table 5 is an estimate of the required resources for implementing the proposed network. This implementation assumes that the operations occur sequentially with fixed resources. Additionally, this estimate assumes that the branches of the network are processed one at a time. The output branch of each network is the output of the point neuron which is a single value. This value will be retained and the next branch will be processed and so on until all eight point neurons are computed. This vector will then be sent to the softmax which will perform the classification.

TABLE 5: Estimated FPGA Resources

Layer	M20k	Multipliers
Input Circular Buffer Bank	66	
Layer Buffer Bank	258	
Weight Bank	2077	
Tanh look up table	1	
Normalization Module		8
FIR Module bank		23
MAC		1
ReLU look up table	1	
Softmax look up table	1	
Total	2,404	32

CHAPTER 8

CONCLUSIONS AND FUTURE DIRECTIONS

This thesis studied application of modern machine learning techniques to the complex task of digital modulation classification. A modified version of a cutting edge deep learning architecture was used to achieve high classification accuracy on a very rigorous dataset. This application of the capsule network to perform digital modulation classification was the primary contribution of this thesis. Detailed background information was provided that described the needed information to understand the various digital modulation classifications and to deduce the difficulties associated with their accurate classification. Further background information regarding machine learning and the subset known as deep learning was provided in order to facilitate discussion of ANNs and how they may be leveraged for classification problems. The datasets and all pertinent details were provided, and simulations were conducted with all key configurations being listed. The classification accuracies listed in Table 4 demonstrate the success of the proposed network is achievable when a comprehensive dataset is employed. Ultimately, the proposed network has been successfully applied to the problem of digital modulation classification with high accuracy.

In researching this topic many additional lines of inquiry were established that were beyond the scope of this thesis. Some were mentioned in various sections throughout this document. Now, a short description of future possible work in these areas will be discussed. This section is specifically geared towards the topic of digital modulation classification, but the concepts are applicable to any classification problem.

- **Alternative Training Methods:**

  As noted in earlier chapters, the way in which an ANN learns is by backpropagation. Specifically, the error is the ultimate driver for updating weights. With the proposed network, there are eight branches, one per class, with each branch focused on learning characteristics of its class. When a BPSK signal is provided, then all of the other seven branches will have an error gradient that is adapting the other seven branches not to recognize BPSK. The concern is that individual branches are exposed to more examples of what their class is not rather than what their class is. One solution to this would be to incorporate special drop out layers that normally have that branch turned off during training. Then when either that branch's class is present in the input data or randomly with a $\frac{1}{7}$ chance, the branch will be turned on. This of course would require experimentation, but it might result in greater overall accuracy.

- **Modification of Topology:**

  As noted in earlier sections, matched filters are used in signal analyzers for symbol detection. It might be possible to implement some process inspired by matched filters in the routing stage of a capsule network. In [18], Gabor filters were used in the routing stage to help extract image characteristics. Similarly, there may be a type of filtering, likely specific for each digital modulation scheme, that could be applied. Experimentation would be required to explore this topic.

- **Iterative Augmentation of Training Data:**

  In addition to the in-band SNR ranges and varying CFO values, there are countless other characteristics that the training datasets might be augmented by. It would be very interesting to design augmentation functions that iteratively apply augmentations to each batch of training data for a wide range of characteristics. The goal here would be to expose the network to all possible characteristics in hopes of reaching a generalized network.

BIBLIOGRAPHY

[1] G. B. Giannakis, "Cyclostationary Signal Analysis," in *The Digital Signal Processing Handbook. DSP Fundamentals.* New York, NY: Taylor & Francis, CRC Press, 2009, pp. 365–416, v. K. Madisetti editor.

[2] F. Hameed, O. A. Dobre, and D. C. Popescu, "On the Likelihood-Based Approach to Modulation Classification," *IEEE Transactions on Wireless Communications*, vol. 8, no. 12, pp. 5884–5892, December 2009.

[3] J. L. Xu, W. Su, and M. Zhou, "Likelihood-Ratio Approaches to Automatic Modulation Classification," *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, vol. 41, no. 4, pp. 3072–3108, July 2011.

[4] J. A. Sills, "Maximum-likelihood modulation classification for psk/qam," in *MILCOM 1999. IEEE Military Communications. Conference Proceedings (Cat. No. 99CH36341)*, vol. 1. IEEE, 1999, pp. 217–220.

[5] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[6] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-Air Deep Learning Based Radio Signal Classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018.

[7] M. Zhou, Z. Yin, Z. Wu, C. Y., N. Zhao, and Z. Yang, "A Robust Modulation Classification Method Using Convolutional Neural Networks," *EURASIP Journal on Advances in Signal Processing*, vol. 2019, March 2019.

[8] J. A. Snoap, D. C. Popescu, and C. M. Spooner, "On Deep Learning Classification of Digitally Modulated Signals Using Raw I/Q Data," in *Proceedings* 19th *Annual IEEE Consumer Communications and Networking Conference – CCNC 2022*, Las Vegas, NV, January 2022.

[9] J. A. Latshaw, D. C. Popescu, J. A. Snoap, and C. M. Spooner, "Using Capsule Networks to Classify Digitally Modulated Signals with Raw I/Q Data," in *Proceedings 14th*

IEEE International Communications Conference – COMM 2022, Bucharest, Romania, June 2022.

[10] S. Haykin, *Digital Communication Systems.* Wiley, 2014.

[11] J. Proakis and M. Salehi, *Fundamentals of Communication Systems Second Edition.* Pearson, 2014.

[12] C. M. Spooner, "Multi-resolution white-space detection for cognitive radio," in *MILCOM 2007-IEEE Military Communications Conference.* IEEE, 2007, pp. 1–9.

[13] C. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2014.

[14] C. Aggarwal, *Neural Networks and Deep Learning.* Springer, 2018.

[15] R. D. Luce, "Luce's choice axiom," *Scholarpedia*, vol. 3, no. 12, p. 8077, 2008.

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

[17] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic Routing Between Capsules," in *Proceedings* 31st *International Conference on Neural Information Processing Systems – NIPS'17*, Long Beach, CA, December 2017, pp. 3859–3869.

[18] M. K. Patrick, B. A. Weyori, and A. A. Mighty, "Max-pooled fast learning gabor capsule network," in *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD).* IEEE, 2020, pp. 1–8.

[19] L. Li, J. Huang, Q. Cheng, H. Meng, and Z. Han, "Automatic modulation recognition: A few-shot learning method based on the capsule network," *IEEE Wireless Communications Letters*, vol. 10, no. 3, pp. 474–477, 2020.

[20] J. Sun, G. Wang, Z. Lin, S. G. Razul, and X. Lai, "Automatic Modulation Classification of Cochannel Signals using Deep Learning," in *Proceedings 23rd IEEE International Conference on Digital Signal Processing (DSP)*, 2018, pp. 1–5.

[21] D. Zhang, W. Ding, C. Liu, H. Wang, and B. Zhang, "Modulated Autocorrelation Convolution Networks for Automatic Modulation Classification Based on Small Sample Set," *IEEE Access*, vol. 8, pp. 27 097–27 105, 2020.

[22] M. Kulin, T. Kazaz, I. Moerman, and E. De Poorter, "End-to-End Learning From Spectrum Data: A Deep Learning Approach for Wireless Signal Identification in Spectrum Monitoring Applications," *IEEE Access*, vol. 6, pp. 18 484–18 501, 2018.

[23] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep Learning Models for Wireless Signal Classification With Distributed Low-Cost Spectrum Sensors," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, 2018.

[24] K. Bu, Y. He, X. Jing, and J. Han, "Adversarial Transfer Learning for Deep Learning Based Automatic Modulation Classification," *IEEE Signal Processing Letters*, vol. 27, pp. 880–884, 2020.

[25] Y. Sang and L. Li, "Application of novel architectures for modulation recognition," in *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. IEEE, 2018, pp. 159–162.

[26] C. M. Spooner, "Data set for the machine-learning challenge," cyclostationary.blog, Feb. 15, 2019, accessed Dec. 7, 2020. [Online]. Available: https://cyclostationary.blog/2019/02/15/data-set-for-the-machine-learning-challenge/.

[27] J. Djolonga, J. Yung, M. Tschannen, and et al., "On Robustness and Transferability of Convolutional Neural Networks," accessed: Feb. 18, 2021. [Online]. Available: https://arxiv.org/pdf/2007.08558.pdf.

[28] N. Qian, "On the Momentum Term in Gradient Descent Learning Algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.

[29] J. Silva, P. Pereira, R. Machado, R. Névoa, P. Melo-Pinto, and D. Fernandes, "Customizable fpga-based hardware accelerator for standard convolution processes empowered with quantization applied to lidar data," *Sensors*, vol. 22, no. 6, p. 2184, 2022.

VITA

James A. Latshaw
Department of Engineering - Electrical and Computer
Old Dominion University
Norfolk, VA 23529

Education:

MS in Electrical and Computer Engineering, Old Dominion University, Norfolk, VA. May 2022.

BS in Electrical Engineering, Liberty University, Lynchburg, VA. May, 2015.

Research Interests: Digital systems, communication systems and machine learning.

Work Experience:

Electronics Engineer at Jefferson Lab, as a member of the low level RF team. Primary tasks involve designing, fabricating and producing complex, interconnected embedded systems used to control and monitor US Department of Energy particle accelerators.

Electrical Engineer at Huntington Ingalls Industries, as a member of the sonar and combat systems group. Primarily tasked with support and construction of US Department of Defense projects and special research projects.

Hardware Engineer at Lockheed Martin, primarily tasked with testing and design validation of embedded systems.

Typeset using LaTeX.