Real-World Considerations for Deep Learning in Spectrum Sensing

Steven C. Hauser

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Alan J. Michaels, Co-chair
A. A. (Louis) Beex, Co-chair
William C. Headley
Ryan K. Williams

May 7, 2018
Blacksburg, Virginia

Keywords: Machine Learning, Spectrum Sensing, Neural Networks, Automatic Modulation Classification, Communication Systems

Copyright 2018, Steven C. Hauser
Real-World Considerations for Deep Learning in Spectrum Sensing

Steven C. Hauser

(ABSTRACT)

Recently, automatic modulation classification techniques using deep neural networks on raw IQ samples have been investigated and show promise when compared to more traditional likelihood-based or feature-based techniques. While likelihood-based and feature-based techniques are effective, making classification decisions directly on the raw IQ samples removes the need for expertly crafted transformations and feature extractions. In practice, RF environments are typically very dense, and a receiver must first detect and isolate each signal of interest before classification can be performed. The errors introduced by this detection and isolation process will affect the accuracy of deep neural networks making automatic modulation classification decisions directly on raw IQ samples. The importance of defining upper limits on estimation errors in a detector is highlighted, and the negative effects of over-estimating or under-estimating these limits is explored. Additionally, to date, most of the published research has focused on synthetically generated data. While large amounts of synthetically generated data is generally much easier to obtain than real-world signal data, it requires expert knowledge and accurate models of the real world, which may not always be realistic. The experiments conducted in this work show how augmented real-world signal captures can be successfully used for training neural networks used in automatic modulation classification on raw IQ samples. It is shown that the quality and duration of real world signal captures is extremely important when creating training datasets, and that signal captures made from a single transmitter with one receiver can be broadly applicable to other radios through dataset augmentation.
Real-World Considerations for Deep Learning in Spectrum Sensing

Steven C. Hauser

(GENERAL AUDIENCE ABSTRACT)

With the increasing prevalence of wireless devices in every day life, communicating between them can become more difficult because the devices must contend with each other to send and receive information. Being able to communicate in a variety of environments can be challenging and, while devices can be pre-configured for certain situations, devices that are able to automatically adjust how they communicate are more reliable and robust. The research presented in this thesis will contribute to solving this challenge by considering machine-learning based, radio frequency signal processing algorithms that are able to automatically group different communication signals. Being able to automatically group different signals is helpful because it can provide information about the wireless environment, allowing a device to make intelligent decisions based on what it detects is happening around it. However, before these algorithms can be successfully used in wireless devices, their limitations must be better understood. To this end, the work in this thesis will show how sensitive these algorithms are to imperfections in wireless devices. This work will also show how information from new environments can be captured and manipulated to allow these algorithms to scale for unseen environments and communication signals.
Contents

List of Figures vii

List of Tables xvi

1 Introduction 1

2 Background 6

2.1 Deep Learning for Wireless Communications 7

2.1.1 Deep Learning Architectures 7

2.1.2 Deep Learning Applied to Communication Systems 15

3 Receiver Effects 19

3.1 Introduction 19

3.2 Convolutional Neural Network Architecture 21

3.3 Neural Network Training and Simulations 22

3.4 Simulations 24
3.4.1 Frequency Offset .......................................................... 25
3.4.2 Sample Rate Offset ....................................................... 31
3.4.3 Sample Rate Offset and Frequency Offset ......................... 37

4 Augmenting Over-the-Air Signal Captures ................................ 39

4.1 Introduction ................................................................. 39
4.2 Neural Network Architectures ............................................ 41
4.3 Datasets and Augmentation ............................................... 42
  4.3.1 System Setup .......................................................... 42
  4.3.2 Training Dataset Generation ....................................... 44
  4.3.3 Test Dataset Generation ............................................. 45
  4.3.4 Augmenting Existing Data ......................................... 46

4.4 Simulation Results ....................................................... 46
  4.4.1 Synthetic Datasets .................................................. 47
  4.4.2 Augmenting Dataset 1: 10,000 Examples at 15 dB SNR .......... 50
  4.4.3 Augmenting Dataset 2: 10,000 Examples at 7.5 dB SNR .......... 54
  4.4.4 Augmenting Dataset 3: 100,000 Examples at 15 dB SNR .......... 59
  4.4.5 Augmenting Dataset 4: 100,000 Examples at 7.5 dB SNR .......... 62
  4.4.6 Improving Performance with Dataset 2: 10,000 Examples at 7.5 dB SNR 64

4.5 Applying to New Hardware ............................................. 71
4.5.1 Overall Classification Accuracy ........................................ 71

5 Conclusions and Future Work .............................................. 78

Bibliography ............................................................................ 84
# List of Figures

2.1 Example of a simple multilayer perceptron neural network. Each dimension of the input vector is fully connected to every hidden node, and then the outputs of the hidden nodes are combined to get the desired output $y$ of the system. 9

2.2 Nonlinear activation functions used in this work. 10

2.3 Example of a simple convolutional neural network. The hidden nodes only connect to a subset of the input at once, and convolve across the input to calculate the output of the layer, which acts as the input to the next layer. 12

2.4 Example of a multi-layer convolutional neural network. As highlighted in orange, the input view of the deeper layers is larger than that of earlier layers. 13

2.5 Example of a simple recurrent neural network. The input is sequentially processed by the hidden layer, and the previous state of the hidden layer is also used as an input to the next state for the next point in the input. The output is calculated at the end of the sequence. As shown, the hidden layer consists of a single node, and that node is updated through time via the new time input and the previous state of the hidden node. 14
3.1 Imperfections introduced by the detection and isolation stage of a receiver. On the top is a wideband snapshot of a dense spectral environment, and on the bottom is a baseband signal after detection and isolation. Frequency error (left) and sample rate mismatch (right) are shown in red. 20

3.2 Diagram of a representative convolutional neural network architecture designed for AMC using only raw IQ samples. 22

3.3 Neural network testing accuracies using 256 raw IQ input samples averaged over all considered SNRs. As the offsets in the training set increase, the overall performance of the CNN decreases. The decrease is most noticeable for CNNs trained to generalize over frequency offsets. 24

3.4 Classification accuracy of Ideal CNN trained assuming no frequency offset. Accuracy quickly drops as the tested frequency offset deviates from 0. 26

3.5 Classification accuracy of Freq_2.5 CNN trained assuming frequency offsets from $\pm 2.5\%$ of the sample rate. Accuracy is relatively flat within the training range, but quickly drops as the tested frequency offset increases beyond 2.5%. 27

3.6 Classification accuracy of Freq_5 CNN trained assuming frequency offsets from $\pm 5\%$ of the sample rate. Accuracy is relatively flat within the training range, but quickly drops as the tested frequency offset increases beyond 5%. 28

3.7 Classification accuracy vs. SNR for Ideal, Freq_2.5, and Freq_5 CNNs tested with samples at no frequency offset. For high SNRs, the CNN is able to generalize over small frequency offsets without significant degradation in accuracy. However, accuracy is significantly degraded at lower SNRs. 29
3.8 Classification accuracy vs. frequency offset for Ideal, Freq_2.5, and Freq_5 CNNs tested at an SNR of 10 dB. Maximum classification accuracy decreases significantly as the CNN is trained for broader ranges of frequency offsets. 30

3.9 Classification accuracy vs. SNR for Freq_5 CNN trained with varying input size. Increasing the number of inputs to the CNN improves classification accuracy across all SNRs. 31

3.10 Classification accuracy of Ideal CNN trained assuming a sample rate of twice the bandwidth. Accuracy quickly drops as the tested sample rate deviates from twice the bandwidth. 32

3.11 Classification accuracy of Samp_2.5 CNN trained assuming sample rates between 1.5 and 2.5 times the bandwidth. Accuracy is relatively flat within the training range, but quickly drops as the tested sample rate decreases below 1.5 or increases above 2.5 times the bandwidth. 33

3.12 Classification accuracy of Samp_4 CNN trained assuming sample rates from the signal bandwidth to 4 times the bandwidth. Accuracy is relatively flat within the training range, but quickly drops as the tested sample rate increases beyond 4 times the bandwidth. 34

3.13 Classification accuracy vs. SNR for Ideal, Samp_2.5, and Samp_4 CNNs tested with samples at twice the bandwidth. All CNNs converge to the same accuracy at high SNRs, but noticeable degradation occurs at SNRs below approximately 11 dB. 35

3.14 Classification accuracy vs. sample rate for Ideal, Samp_2.5, and Samp_4 CNNs tested at an SNR of 10 dB. Maximum classification accuracy does not decrease significantly as the CNN is trained for broader ranges of input sample rates. 36
3.15 Classification accuracy vs. SNR for Samp.4 CNN trained with varying input size. Increasing the number of input samples improves performance, but the improvement is limited for higher SNRs. .......................................................... 37

3.16 Classification accuracy vs. SNR for Ideal, Freq.2.5, Samp.2.5, and Samp.Freq CNNs tested with signals at no frequency offset and sampled at twice the signal bandwidth. The maximum accuracy is measurably degraded when the neural network is trained to handle both frequency and sample rate offsets. ........ 38

4.1 GNURadio flowgraph used for collecting over-the-air signal captures. ....... 43

4.2 Classification accuracy of CNN architecture for various training scenarios using synthetic datasets. Synthetic training datasets that do not accurately model the real world result in poor neural network performance when deployed in real-world systems. ................................................................. 48

4.3 Classification accuracy of LSTM architecture for various training scenarios using synthetic datasets. Synthetic training datasets that do not accurately model the real world result in poor neural network performance when deployed in real-world systems. ................................................................. 49

4.4 Classification Accuracy of CNN architecture for various training scenarios using Dataset 1 and a single augmentation. None of the augmentations show improvements over the CNN trained with synthetic data, and only augmenting with noise is capable of doing better than random guessing at lower SNRs. 50
4.5 Classification Accuracy of LSTM architecture for various training scenarios using Dataset 1 and a single augmentation. None of the augmentations show improvements over the LSTM trained with synthetic data, and only augmenting with noise is capable of doing better than random guessing at lower SNRs. 51

4.6 Classification Accuracy of CNN architecture for various training scenarios using Dataset 1 and multiple augmentations. All three augmentations must be used to maximize the effectiveness of the dataset. 53

4.7 Classification Accuracy of LSTM architecture for various training scenarios using Dataset 1 and multiple augmentations. All three augmentations must be used to maximize the effectiveness of the dataset. 54

4.8 Classification Accuracy of CNN architecture for various training scenarios using Dataset 2 and a single augmentation. None of the augmentations show improvements over the CNN trained with synthetic data, once again demonstrating that multiple augmentation techniques are required. All but the resampling augmentation show almost no improvement throughout the entire SNR range. 55

4.9 Classification Accuracy of LSTM architecture for various training scenarios using Dataset 2 and a single augmentation. No scenario is better than the LSTM trained with synthetic data, once again demonstrating that multiple augmentation techniques are required. 56
4.10 Classification Accuracy of CNN architecture for various training scenarios using Dataset 2 and multiple augmentations. Augmenting the training data by applying noise, resampling, and a frequency shift is critical to improving classification accuracy, but accuracy is still relatively poor throughout the entire SNR range when compared to the network trained with synthetic data.

4.11 Classification Accuracy of LSTM architecture for various training scenarios using Dataset 2 and multiple augmentations. Augmenting the training data by applying noise, frequency shifts, and resampling is critical to maximizing classification accuracy across the entire SNR range, but still does not match the performance of the network trained with synthetic data.

4.12 Classification Accuracy of CNN architecture for various training scenarios using Dataset 3. Augmenting the training data is critical to improving classification accuracy, and the larger initial capture size helps to improve classification accuracy when compared to using Dataset 1. The CNN trained with synthetic data still has the best performance over the entire SNR range.

4.13 Classification Accuracy of LSTM architecture for various training scenarios using Dataset 3. Augmenting the training data is critical to improving classification accuracy, and the larger initial capture size does not provide any benefit over using the smaller captures from Dataset 1. The LSTM trained with synthetic data still has the best performance over the entire SNR range.
4.14 Classification Accuracy of CNN architecture for various training scenarios using Dataset 4 and multiple augmentations. Augmenting the training data is critical to improving classification accuracy at lower SNRs, but the classification accuracy plateaus at higher SNRs. With a larger initial capture, it is possible, through augmentation, to match the classification accuracy of the CNN trained with synthetic data at lower SNRs. ......................................................... 63

4.15 Classification Accuracy of LSTM architecture for various training scenarios using Dataset 4 and multiple augmentations. Augmenting the training data is critical to improving classification accuracy at lower SNRs, but the classification accuracy plateaus at higher SNRs. With a larger initial capture, it is possible, through augmentation, to match the classification accuracy of the LSTM trained with synthetic data at lower SNRs. ......................................................... 64

4.16 Training and validation loss for the CNN architecture augmenting Dataset 2 with noise, frequency shifts, and resampling. The CNN begins to overfit immediately. .......................................................... 65

4.17 Training and validation loss for the LSTM architecture augmenting Dataset 2 with noise, frequency shifts, and resampling. The validation loss of the LSTM network tracks the training loss for the first 5 epochs, but then quickly begins to overfit the training data. .......................................................... 66

4.18 Training and validation loss for the CNN architecture augmenting Dataset 2 with a non-repeating number of noise, frequency shift, and resampling augmentations. The CNN tracks the training loss for the duration of the training, although begins to overfit around epoch 70. .......................................................... 67
4.19 Training and validation loss for the LSTM architecture augmenting Dataset 2 with a non-repeating number of noise, frequency shift, and resampling augmentations. The validation loss of the LSTM network is consistently better than the training loss, but plateaus around epoch 300.

4.20 Classification accuracy comparison for CNN architecture across four different training scenarios. Training with non-repeated augmentations results in improved accuracy, but is still not as good as the CNN trained with a larger original data capture.

4.21 Classification accuracy comparison for LSTM architecture across four different training scenarios. Training with non-repeated augmentations is clearly the best technique when augmenting over-the-air signal captures.

4.22 Overall classification accuracy for different receiver and transmitter pairs with neural networks trained using synthetic data. The overall classification accuracy varies fairly significantly, as much as about 8% across the different receiver/transmitter pairs.

4.23 Overall classification accuracy for different receiver and transmitter pairs with neural networks trained using Dataset 3 augmented with noise, frequency shifts, and resampling. The receiver and transmitter pair used to create the initial training dataset is highlighted in red, and the overall accuracy varies fairly significantly, as much as about 8% across the different receiver/transmitter pairs.
4.24 Overall classification accuracy for different receiver and transmitter pairs with neural networks trained using Dataset 4 augmented with noise, frequency shifts, and resampling. The receiver and transmitter pair used to create the initial training dataset is highlighted in red, and the overall accuracy is remarkably consistent for all hardware combinations. 74

4.25 Classification accuracy of the CNN architecture with Radio B as the transmitter and Radio C as the receiver under various training scenarios. Training on Dataset 4 augmented with noise, frequency shifts, and resampling results in the best performance at lower SNRs. 75

4.26 Classification accuracy of the LSTM architecture with Radio B as the transmitter and Radio C as the receiver under various training scenarios. Training on Dataset 4 augmented with noise, frequency shifts, and resampling results in the best performance at lower SNRs. 76
List of Tables

3.1 Considered Neural Network Training Scenarios . . . . . . . . . . . . . . . . . . . . 23

4.1 Summary of Over-the-Air Capture Parameters . . . . . . . . . . . . . . . . . . . . 45
Abbreviations

AGC  Automatic Gain Control
AMC  Automatic Modulation Classification
AWGN Additive White Gaussian Noise
CNN Convolutional Neural Network
DSP Digital Signal Processing
FIR Finite Impulse Response
FSK Frequency Shift Keying
GPU Graphical Processing Unit
IIR Infinite Impulse Response
IQ In-phase/Quadrature
LSTM Long Short-Term Memory
MLP Multilayer Perceptron
PSK Phase Shift Keying
QAM Quadrature Amplitude Modulation
ReLU Rectified Linear Unit
RF Radio Frequency
RNN Recurrent Neural Network
SNR Signal-to-Noise Ratio
Chapter 1

Introduction

With the recent success of machine learning, and specifically deep neural networks, in fields such as image processing [1], machine translation [2], and speech transcription [3], there has been a surge of interest in how deep neural networks can be applied to communication systems, particularly using only raw in-phase and quadrature (IQ) samples. For example, in [4], a convolutional neural network (CNN) is used to estimate IQ imbalance using only IQ samples; in [5], a CNN is used for timing and center frequency offset estimation using only raw IQ samples; and in [6], a CNN is used for interference detection using only raw IQ samples. For automatic modulation classification (AMC) specifically, designs such as stacked sparse auto-encoders [7], CNNs [8], and recurrent neural networks (RNNs) [9], as well as more complicated implementations such as those explored in [10], have been used to autonomously extract features for AMC. In particular, [8] and [9] use a CNN and RNN respectively for AMC using only raw IQ samples with promising results.

Automatic modulation classification has been a topic of interest for many years in wireless communications, enabling blind characterization of a signals' modulation with little, to no, a priori knowledge. AMC is of particular interest in cognitive radio, spectrum sensing/dy-
dynamic spectrum access, and military applications. In cognitive radio, one use of AMC is that it allows for dynamically changing modulation formats without increasing the communication overhead. Similarly, being able to easily identify primary users and secondary users based strictly on modulation format can be highly desirable not only in cognitive radios, but in broader spectrum sensing/dynamic spectrum access scenarios, where it is important to find open channels without interfering with primary users. Finally, for military applications, AMC can enable easier characterization and information gathering of both adversary and friendly radios, as well as enabling intelligent jamming of specific types of signals.

As explored in works such as [11, 12, 13], AMC has traditionally been accomplished using likelihood functions or by extracting modulation specific expert features from the received IQ samples. Works such as [14, 15, 16, 17, 18, 19] have demonstrated that these features can be used as inputs to neural networks to achieve high classification accuracy. While both likelihood-based and feature-based methods are effective, they require expert design or knowledge of the signal, require large numbers of samples, and/or can add complexity to a receiver given that the raw IQ samples must be processed before classification. Techniques that can autonomously classify modulations directly from the raw IQ samples can not only remove the need for expert feature design, but also potentially free resources in the receiver for other tasks. Additionally, the likelihood-based and feature-based approaches to AMC offer no one-size-fits-all solution, requiring different techniques depending on environmental variables and the modulations at hand [20].

The use of deep neural networks for AMC directly on raw IQ samples offers several benefits over traditional likelihood-based and feature-based methods. First, these networks offer a potential universal solution, in that rather than having to hand pick or expertly design AMC features/algorithms for every type of modulation, deep neural networks can autonomously extract and classify on useful features by simply adding the modulations to a training dataset.
Not only is this helpful for existing modulations, but it also allows for easy scaling to new modulation formats, making deep neural network based AMC on raw IQ a potentially future-proof technology. Second, by processing the raw IQ samples directly, receiver complexity can potentially be reduced, freeing up processing resources for other tasks. Finally, traditional techniques can require large numbers of samples to be effective [21], whereas neural networks may be able to make more accurate classification decisions using fewer samples [5, 8].

Much of the published literature uses datasets either directly from, or derived from [22]. As outlined in [22], these datasets attempt to model the real world by incorporating effects such as sample rate offset, center frequency offset, channel fading, and additive white Gaussian noise. However, the sample rate and center frequency offsets considered are largely intended to model small inaccuracies such as sampling errors from imperfect clocks, and carrier frequency drift due to local oscillator drift. While considering small inaccuracies is important, in modern military, dynamic spectrum access, and cognitive radio applications, the exact center frequency and bandwidth of signals of interest will not generally be known, but will instead need to be estimated by a receiver. These estimates can create significantly larger offsets than simple errors due to oscillator drift and digital sampling.

As the research into deep neural networks on raw IQ for modern AMC applications progresses, and in particular as the research transitions to real-world deployments, it will become increasingly important to understand how deep neural networks will fit into the system design, and what real-world effects need to be taken into consideration, particularly in blind receiver applications. To that end, two primary contributions are added by the work presented in this thesis. First, the importance of blind receiver characteristics on deep neural networks trained for AMC on raw IQ is discussed [23]. Next, an approach for augmenting real, over-the-air signal captures is discussed. This approach will show how augmenting signal captures can improve deep neural network based AMC on raw IQ in blind receiver ap-
plications, and what trade-offs exist between signal captures and future performance. More specifically, this work is organized as follows:

In Chapter 2, a background on neural networks and AMC is presented. This chapter provides a brief overview of neural networks, with a focus on the types of networks used throughout the work presented in this thesis. Additionally, traditional AMC techniques and how neural networks may revolutionize the field is also explored.

Chapter 3 provides an in-depth study on the effect that imperfections in the detection stage of a receiver can have on neural networks designed for AMC on raw IQ. Specifically, this chapter introduces two primary types of common receiver imperfections: frequency offset and sample rate offset. Through simulation, it is shown that there is a close relationship between the frequency and sample rate estimates in a receiver and the overall classification accuracy of neural network based AMC on raw IQ. This study provides two major takeaways. First, taking into account receiver imperfections such as frequency offset and sample rate mismatch are essential if neural network based AMC on raw IQ is to be realized in real-world systems. Second, training a neural network to generalize over a broad range of frequency and sample rate offsets has a negative effect on classification accuracy, even if subsequent signals have 100% accurate frequency and sample rate estimates. Training for the specific error bounds of a receiver is critical to maximizing classification accuracy.

Chapter 4 then explores how real, over-the-air signal captures can be used to train neural networks designed for AMC on raw IQ. Using real, over-the-air signal captures for training offers two primary benefits over the synthetic datasets from Chapter 3. First, while receiver imperfections such as frequency offset and sample rate offset are relatively easy to account for in synthetic datasets, other imperfections such as emitter non-idealities and complex channels may not be as easy to model through simulation. Second, creating synthetic training datasets inherently requires expert knowledge of all modulations of interest. Being able
to capture and train with previously unknown modulations can allow deep neural networks to classify arbitrary modulations, without the need for expertly created synthetic datasets. Through real-world simulation, it is shown that, in the absence of synthetic training data, augmenting over-the-air signal captures with simple techniques such as adding noise, applying frequency shifts, and resampling is essential to maximizing classification accuracy of neural network based AMC on raw IQ. It is also shown that classification accuracy can be limited by the quality of the original data capture used for training. Although presented through the framework of AMC, the results of this chapter can readily apply to other fields such as deep neural network based emitter identification, where the presented augmentation techniques could enable creating generalized training datasets from emitter specific, real-world transmissions.

Finally, in Chapter 5, conclusions and future work are provided. Here major takeaways will be discussed, reaffirming the applicability and limitations of using both synthetic and augmented datasets for neural network based AMC on raw IQ. Additionally, this chapter highlights current hurdles that will need to be addressed such as neural network scalability, generalizability, and hardware sensitivity before neural network based AMC on raw IQ can be realized in communication systems.

The following publications were a direct result of the work presented in this thesis.


Chapter 2

Background

In recent years, deep neural networks have seen an explosion in use. While neural networks have been around for decades, modern advances in hardware like Graphical Processing Units (GPUs), as well as improved algorithms have enabled training of larger, deeper neural networks capable of learning more complicated tasks [24]. Additionally, open source software libraries such as Keras [25] and TensorFlow [26] have significantly lowered the barrier to entry for building, training, and testing neural networks. This has helped to open the door for deep learning in new domains by enabling rapid design, prototyping, and testing of advanced neural network architectures with just a few lines of code. In combination with ever-improving hardware, this makes new applications for neural networks readily accessible, and has provided a framework for state-of-the-art algorithms and widespread use in multiple fields [1, 2, 3].

Taking a cue from rapid advances in areas such as image processing, in communication systems this has inspired moving away from more traditional, expertly designed solutions, to adaptable solutions that are ‘learned’ by deep neural networks. In the context of AMC, deep neural networks are causing a shift from simply using neural networks to classify modulation
2.1. Deep Learning for Wireless Communications

2.1.1 Deep Learning Architectures

In this work, two primary types of deep neural networks are used, namely feedforward neural networks and recurrent neural networks. Both types of networks are very similar in that they attempt to learn a function that maps a set of inputs to a set of outputs. More specifically, they attempt to learn an output $y$ given a set of inputs $x$, using a set of learnable parameters defined by $\theta$. The key distinction between feedforward and recurrent neural networks being that recurrent neural networks incorporate feedback connections as highlighted later in this section [28].

To get a basic idea of how, in general, deep neural networks work, the multilayer perceptron (MLP) is introduced. As described in [28], neural networks largely consist of the following...
Chapter 2. Background

mathematical constructs. Formally, the neural network attempts to learn a function \( f \) such that

\[
y = f(x|\theta).
\]  

(2.1)

Typically \( \theta \) consists of a series of weights and biases, and therefore Equation (2.1) can be re-written as

\[
y = f(x|w, b).
\]  

(2.2)

Drawing from [29], Figure 2.1 shows a simple example of a MLP network with a single hidden layer, where a hidden layer is any layer within the structure of the neural network that is not an input or an output layer. Each line represents a learnable weight in the network. The term deep learning really just means that the neural network has multiple hidden layers, each with its own set of weights and biases so that

\[
f(x|\theta) = f^{(3)}(f^{(2)}(f^{(1)}(x|\theta_1)|\theta_2)|\theta_3).
\]  

(2.3)

For simplicity only a single hidden layer is shown in Figure 2.1.

In order to map to more complex functions, such as non-linear functions, the weights and biases are typically passed through an activation function before the output of a particular layer is calculated. From [28], letting an activation function be defined by the function \( g(x) \), and the hidden layer output being \( h \), the whole layer can be written as

\[
h = g(W^Tx + b)
\]  

(2.4)
2.1. Deep Learning for Wireless Communications

where $W$ is a learnable weight matrix, $\mathbf{T}$ is the matrix transpose, and $b$ is an optional set of learned biases. In the simple case as shown in Equation (2.2), $h$ from Equation (2.4) is equal to $y$, and if there are multiple hidden layers Equation (2.3) can be re-written as

$$f(x; \theta) = g^{(3)} (W_3^T g^{(2)} (W_2^T g^{(1)} (W_1^T x + b_1) + b_2) + b_3). \quad (2.5)$$

In this work, three primary activation functions are used throughout all neural network architectures considered: the rectified linear unit (ReLU), the hyperbolic tangent, and the logistic sigmoid function. The ReLU was used because of its widespread success \[1, 30, 31, 32\], while the hyperbolic tangent and logistic sigmoid functions are widely used in long short-
Nonlinear Activation Functions

![Nonlinear Activation Functions](image)

Figure 2.2: Nonlinear activation functions used in this work.

term memory recurrent neural networks [33, 34, 35]. For reference, these three activation functions are shown in Figure 2.2. Also, because all simulations presented in this work are in the context of classification, the neural networks constructed are designed to choose the most likely class. This is accomplished by increasing the number of output nodes $y$ in Figure 2.1 to be equal to the number of desired classes, applying the softmax function to this output vector, and then choosing the most likely class from the distribution. Mathematically, the chosen class $H$ is determined by

$$ H = \arg \max_c \left( \frac{e^{y_c}}{\sum_{j=0}^{M-1} e^{y_j}} \right), \quad (2.6) $$

where $y_j$ is a single element from the output vector and $\sum_{j=0}^{M-1}$ is the sum over all possible classes $M$. 
Convolutional Neural Networks

While effective, the MLP architecture suffers from scaling issues due to the number of parameters in the densely connected layers [28], and as networks become larger and learning tasks more difficult, approximating complicated functions with only a MLP architecture can be challenging. One particular architecture that can drastically reduce the number of parameters within the neural network is a convolutional neural network [36]. Unlike MLP neural networks, each node is only connected to a subset of nodes from the previous layer, and the learned weights act as filters that are convolved across the previous layer. In a machine learning context, the filters created by the weights are known as kernels [28]. Mathematically, Equation (2.4) becomes

\[ h = g \left( (W^T * x) + b \right) , \]  

(2.7)

where \( * \) is the convolution operator, defined as

\[ y[n] = x[n] * w[n] = \sum_{m=-\infty}^{\infty} x[m]w[n - m] = \sum_{m=-\infty}^{\infty} x[n - m]w[m] \]  

(2.8)

for a given input \( x \) and a given weight vector \( w \). Note that in the context of machine learning, convolution is often implemented as a cross-correlation, defined as

\[ y[n] = x[n] * w[n] = \sum_{m=-\infty}^{\infty} x[n + m]w[m] \]  

(2.9)

and used interchangeably with convolution [28].

A simple diagram of a convolutional neural network is shown in Figure 2.3. Unlike the MLP, here the weights are only connected to a subset of the input for any given hidden node. Additionally, the learned weights of each node are the same; the key difference being
that the weights are being applied to different parts of the input vector. It is important to note that, as pictured in Figure 2.3, there is only a single kernel being learned. While not shown, this diagram could easily extend along a third dimension, where each dimension is a new, learned kernel. At the output all kernels are connected as they would be in a MLP architecture.

Convolutional neural networks offer three primary advantages over MLP neural networks: sparse interactions, parameter sharing, and equivariant representations [28]. As illustrated in Figure 2.3, each node in a convolutional neural network is only connected to a subset of the nodes in the previous layer, resulting in sparse interactions. Because each node is not connected to every other node, this structure requires fewer parameters, decreasing the
memory footprint and increasing the efficiency of the neural network estimator [28]. Similarly, convolutional neural networks take advantage of parameter sharing in that each kernel has fixed weights applied everywhere on the previous layer. This is helpful because rather than having to learn the same set of weights for any feature, anywhere that said feature could occur in the previous layer, only a single set of weights is learned, and every part of every kernel is efficiently used throughout the previous layer [28]. Finally, convolutional neural networks offer equivariant representations, effectively preserving changes in the previous layers. Note that due to the sparse interactions in CNN architectures, each learned kernel can only operate on \( n \) inputs at a time, where \( n \) is the size of the kernel. For example, in Figure 2.3 \( n \) would correspond to 4 inputs. This view of the input can be increased with depth as shown by the orange connections in Figure 2.4 [28].

**Recurrent Neural Networks**

Recurrent neural networks [37] are similar to one-dimensional convolutional neural networks, but, through the use of recurrent connections, RNNs are specifically designed for sequential
Figure 2.5: Example of a simple recurrent neural network. The input is sequentially processed by the hidden layer, and the previous state of the hidden layer is also used as an input to the next state for the next point in the input. The output is calculated at the end of the sequence. As shown, the hidden layer consists of a single node, and that node is updated through time via the new time input and the previous state of the hidden node.

data [28], and have a broad range of uses from stock market prediction [38] to natural language processing [39]. While the input view of a CNN can be expanded with depth, the recurrent connections in a RNN allow RNNs to model significantly longer sequences, well beyond the immediate timestep of any particular node [28]. There are many ways to build RNNs, but a typical construct, and the one used in this work, is shown in Figure 2.5 [28]. The weights learned by the hidden recurrent layer are the same from timestep to timestep, but in addition to the input layer, the hidden state is also passed from timestep to timestep. Note that while the output of the recurrent layer can be calculated at every timestep, only the output at the end of the sequence is considered in this particular implementation. Similar to the CNN architecture, Figure 2.5 could easily extend along a third dimension, allowing for separate sets of weights (hidden states) to be learned at each timestep.

In this work, the recurrent neural network that was used follows the structure of Figure 2.5,
but leverages long short-term memory (LSTM) cells [40] as the recurrent units. LSTM cells are used because LSTM networks are still considered to be one of the best RNN architectures for sequence learning [35] and have shown to be exceptionally effective at many sequence based tasks such as video to text translation [34], image captioning [41], sequence to sequence learning [2], and image generation [42]. At a high level, LSTM networks are similar to the generic RNN networks previously described, except that they allow for the network to more easily learn long term dependencies by incorporating gated internal recurrent loops, in addition to the recurrent connections described above [40, 43].

### 2.1.2 Deep Learning Applied to Communication Systems

The use of CNNs and RNNs are a natural fit for communication systems. Convolution is fundamental to the field of communications and digital signal processing (DSP), particularly in filtering to manipulate IQ data and extract useful information. The use of CNNs on the raw IQ samples extends naturally from these signal processing concepts, where the neural network is, by design, convolving over the inputs in an attempt to autonomously extract high level features [28]. Each kernel in the CNN can effectively be thought of as a non-linear, finite impulse response (FIR) filter [44]. While the RNN is not explicitly performing a convolution in the same sense as the CNN, it is designed to learn sequential information from the data, and the learned hidden units can effectively be thought of as non-linear, infinite impulse response (IIR) filters [45, 46]. Both FIR and IIR filters are used extensively in modern communication systems, so the CNN and RNN architectures have the potential to offer unique improvements over traditional systems, with the added benefit of not requiring expert filter design.

As previously discussed, given the recent success of deep learning in fields such as image
processing, where deep neural networks use only pixels as input values [1, 47, 48], there has been a surge of interest looking into applying deep learning directly to raw IQ in communication systems [4, 5, 6, 8, 9, 10]. One particular application that may benefit from deep learning is AMC. Traditional AMC techniques can broadly be grouped into two categories: likelihood-based and feature-based [11, 12, 13]. Likelihood-based methods offer theoretically optimum AMC performance, but suffer from computational complexity and the need for a priori knowledge such as channel gain, noise variance, and phase offset [21]. This can make the likelihood-based algorithms very difficult to use in practical scenarios [11]. Feature-based techniques offer an alternative to likelihood-based techniques, trading simplified complexity for sub-optimal performance [11].

The literature on likelihood-based AMC techniques extends back decades [49, 50, 51, 52, 53, 54, 55, 56]. In general, likelihood-based techniques center around making modulation classification decisions by looking at the probability of the received samples conditioned on the candidate modulations, and choosing the most likely distribution [11, 21]. While useful from a theoretical point of view, two of the biggest drawbacks to likelihood-based AMC are the computational complexity and the assumptions that need to be made. Additionally, poor estimation of assumed parameters such as the channel state can negatively affect the classification accuracy of the likelihood-based AMC technique [21].

On the other hand, feature-based techniques [14, 15, 16, 17, 18, 19, 57, 58, 59] can decrease the computational complexity of classification algorithms by first extracting expertly designed features such as statistical moments of phase [60, 61, 62], higher order cumulants and cyclic cumulants [63, 64, 65, 66, 67], and zero-crossing variance of the signal [68, 69]. Classification decisions can then be made from the relevant features. This classification decision can be performed with different machine learning algorithms such as support vector machines [70, 71, 72], decision trees [73], or neural networks [14, 15, 16, 17, 18, 19, 58, 59].
While effective, feature-based approaches require expert feature design, which can become particularly difficult as new modulation formats and channel assumptions are incorporated into communication systems. Therefore, techniques that can autonomously learn relevant features can not only simplify system design, but also allow for a universal approach to AMC that scales as new modulation formats are created.

Some of the early results with neural networks designed for AMC using only raw IQ samples were first published in [8]. In [8], the authors show improved performance over machine learning algorithms applied to expertly designed cyclic-moment features [74] across 11 different modulations. Since [8], other works have expanded the results to include different neural network architectures [9, 10], as well as different modulations and performance with over-the-air captures [27] using only raw IQ samples. While promising, most of the results in the work to date focus on training with synthetic datasets from [22], or derivatives thereof. While real-world effects such as sample rate offset, center frequency offset, channel fading, and additive white Gaussian noise are implicitly modeled in the synthetic datasets, none of the work attempts to understand exactly how important each effect may or may not be on classification tasks with raw IQ signal data. Additionally, the sample rate and center frequency offsets considered are largely intended to model small inaccuracies from hardware such as local oscillators, not larger inaccuracies as would be typical in blind receiver scenarios, where true bandwidths and center frequencies of received signals are unknown and must be estimated. The work in Chapter 3 will demonstrate not only how important it is to account for larger inaccuracies due to blind estimation, but also that there is a close relationship between expected classification accuracies and the quality of initial bandwidth and center frequency estimates.

A natural extension to the synthetic datasets from [22] is to use over-the-air signal captures for training and evaluation. [27] explores this extension with encouraging results. However,
the authors again assume perfectly tuning to signals of interest, with a signal-to-noise ratio (SNR) of about 10 dB, and only briefly consider training with over-the-air signal captures. Considering that one of the primary benefits to using deep learning based AMC on raw IQ is that the approach can autonomously learn features directly from the raw IQ, and therefore be extended for arbitrary modulations, being able to effectively train deep neural networks from real-world signal captures will be vital toward realizing this benefit. Chapter 4 will reinforce the results of Chapter 3, while demonstrating that if synthetic data correctly models the real-world it can be an effective tool, but that in the absence of synthetic data, real-world signal captures can be just as effective provided augmentations such as noise, frequency shifts, and resampling are applied.
Chapter 3

Receiver Effects

3.1 Introduction

As part of the investigation into real-world effects that must be considered when using deep neural networks for AMC with raw IQ data, the signal imperfections that can be introduced by a blind receiver are first considered. Real-world RF environments can be very dense, such as in dynamic spectrum access applications, often with numerous signals being observed in any particular band of interest. For most applications, signals must first be detected and isolated before any further processing. In real systems, the detection and isolation stage is not perfect. While receivers can directly tune to pre-defined channels with specific bandwidths and center frequencies, there will always be a small source of error in center frequency and sample rate due to hardware imperfections such as oscillator drift, non-ideal analog to digital converters, and filtering. If pre-tuning is not possible, the center frequency and sample rate errors can be exacerbated because the signal detection stage must estimate where each signal is within the wideband spectrum, and the isolation stage must filter and resample the signal to baseband. This entire process introduces two primary forms of error, namely a center
frequency offset and a sample rate offset. A diagram of the detection problem and how these imperfections manifest is shown in Figure 3.1.

While prior works like [8] include datasets with real world signal effects such as center frequency and sample rate offsets, the authors only consider small effects due to non-ideal hardware and do not attempt to quantify the impact, whether positive or negative, that these effects have on classification performance. Through a thorough study of the effect of sample rate offset and frequency offset in baseband signals, the simulations in this chapter will show that not only must these effects be accounted for in the training datasets, but also that defining error bounds in the detection and isolation stage of a receiver can be critical to maximizing performance of downstream neural network processing. While only one CNN is considered in this chapter, the CNN analyzed is of sufficient size to perform generally well across all simulation datasets, without optimizing for any specific dataset, and thus the results are representative of what can be expected for similar neural network architectures.

The rest of this chapter is outlined as follows. In Section 3.2, the exact CNN architecture
used is described. In Section 3.3, information about how the neural network is trained and how data is generated is presented. Finally, in Section 3.4, the simulations and results are presented. Conclusions and future work will be summarized in Chapter 5.

3.2 Convolutional Neural Network Architecture

Leveraging the convolutional neural network architecture proposed in [8] for AMC and the more finely tuned parameters explored in [10], here a representative CNN is developed to classify an input vector of raw IQ samples into one of eight different modulation formats: BPSK, QPSK, 8PSK, 2FSK, 4FSK, 8FSK, 16QAM, and 64QAM. While not an exhaustive search of all modulation formats, these eight representative classes span many of the most common ways to modulate signals, i.e. by amplitude, phase, or frequency, while including a mix of higher and lower order modulations. Note that while works such as [6] show potential improvements in certain tasks if the raw IQ samples are represented as magnitude/phase or frequency domain samples, only raw IQ is considered in this work, since a hypothesized advantage of using deep learning frameworks in AMC applications is the ability to eliminate expertly designed features and upfront data transformations.

The considered neural network architecture given 256 input samples is shown in Figure 3.2. The input vector consists of IQ samples divided into separate I and Q dimensions, and the first convolutional layer (Conv1) uses a filter size of 1x16, with 32 distinct filters. The second convolutional layer (Conv2) uses a filter size of 2x16 and again has 32 distinct filters. To avoid overfitting, dropout [75] is applied at the output of Conv2 before being fed to a fully connected layer with 256 nodes. Conv1, Conv2, and the first fully connected layer use a rectified linear unit for their activation function. The final layer is another fully connected layer with the softmax function applied to the output. Again, the architecture and results
are intended to be representative, and the results presented in this work are the takeaways, not the exact performance of the considered neural network.

3.3 Neural Network Training and Simulations

In order to determine how estimation errors in the detection and isolation stage of a receiver impact classification accuracy, multiple neural networks are trained with datasets that simulate varying degrees of estimation error. These neural networks are outlined in Table 3.1. Four primary scenarios are considered: an ideal detector with perfect frequency and sample rate estimates (Ideal), a detector with varying maximum center frequency offset (Freq_2.5 and Freq_5), a detector with varying sample rate offset (Samp_2.5 and Samp_4), and a detector with both center frequency offset and sample rate offset (Samp_Freq). The nominal sample rate is twice the bandwidth of a detected signal, where, without loss of generality, the bandwidth is defined as the null to null bandwidth of the signal.

To generate data for training and testing the neural networks, signals are created using GNU
Table 3.1: Considered Neural Network Training Scenarios

<table>
<thead>
<tr>
<th>Network Name</th>
<th>Max Frequency Offset</th>
<th>Sample Rate Range (Multiple of Bandwidth)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal</td>
<td>no offset</td>
<td>no mismatch</td>
</tr>
<tr>
<td>Freq.2.5</td>
<td>2.5% of sample rate</td>
<td>no mismatch</td>
</tr>
<tr>
<td>Freq.5</td>
<td>5% of sample rate</td>
<td>no mismatch</td>
</tr>
<tr>
<td>Samp.2.5</td>
<td>no offset</td>
<td>1.5-2.5</td>
</tr>
<tr>
<td>Samp.4</td>
<td>no offset</td>
<td>1-4</td>
</tr>
<tr>
<td>Samp_Freq</td>
<td>2.5% of sample rate</td>
<td>1.5-2.5</td>
</tr>
</tbody>
</table>

Radio and offsets are drawn from a uniform distribution bound by the ranges defined in Table 3.1. For each scenario, all signals assume an additive white Gaussian noise (AWGN) channel and a signal-to-noise ratio (SNR) that varies uniformly between 0 and 20 dB. The neural networks are built and trained using Keras [25] and TensorFlow [26], and optimized using the ADADELTA [76] algorithm, with early stopping to avoid overfitting. Approximately 240,000 signal captures are used for training, and approximately 100,000 signal captures are used for testing.

Overall testing accuracies of these neural networks are shown in Figure 3.3. Here the overall classification accuracy for each neural network is calculated with signals whose frequency and sample rate offsets vary over the entire training range of the neural network, as specified in Table 3.1. Observing Figure 3.3, it is clear that the overall classification accuracy decreases as the training data includes larger offsets. Note that while increasing the size of the neural network or increasing the number of training samples might improve the classification accuracy of the training sets with larger offsets, the size of the neural network and training set is kept fixed throughout all of the simulations. This is done so that direct comparisons can be made across the various detector assumptions. Specific breakdown points and trade-offs are explored more fully in Section 3.4.
Figure 3.3: Neural network testing accuracies using 256 raw IQ input samples averaged over all considered SNRs. As the offsets in the training set increase, the overall performance of the CNN decreases. The decrease is most noticeable for CNNs trained to generalize over frequency offsets.

3.4 Simulations

For all results presented in this section, classification accuracy is determined by averaging 4,000 signal captures (500 per modulation) for each data point. The signals were created with GNU Radio as outlined in Section 3.3. If not otherwise stated, trained neural networks use 256 input IQ samples. Simulations demonstrating how accuracy is impacted by a frequency offset are presented first, followed by simulations demonstrating how accuracy is impacted by a sample rate offset. Finally, simulations demonstrating the combined effect of frequency offset and sample rate offset are presented.
3.4. Simulations

3.4.1 Frequency Offset

To evaluate the effect of center frequency offset on modulation classification accuracy, the frequency offset is swept from -10\% to 10\% of the sample rate. Classification accuracy is then evaluated for the following three neural networks: Ideal, Freq.2.5, and Freq.5. Plots of each neural network’s classification accuracy over the entire sample space are shown in Figures 3.4–3.6. As can be seen, classification accuracy of each CNN decreases quickly for frequency offsets outside of the training range, and this degradation in accuracy is consistent across all SNRs and all three neural networks. As the frequency offset in the training data is increased, peak classification accuracy also noticeably decreases, even at a frequency offset of 0. This demonstrates that there is a penalty for arbitrarily training a CNN on a wide range of frequency offsets.
Figure 3.4: Classification accuracy of Ideal CNN trained assuming no frequency offset. Accuracy quickly drops as the tested frequency offset deviates from 0.
3.4. Simulations

Figure 3.5: Classification accuracy of Freq_2.5 CNN trained assuming frequency offsets from ±2.5% of the sample rate. Accuracy is relatively flat within the training range, but quickly drops as the tested frequency offset increases beyond 2.5%.
Figure 3.6: Classification accuracy of Freq-5 CNN trained assuming frequency offsets from ±5% of the sample rate. Accuracy is relatively flat within the training range, but quickly drops as the tested frequency offset increases beyond 5%.
3.4. Simulations

A cross-sectional view of classification accuracy as a function of SNR, with frequency offset fixed at 0, is shown in Figure 3.7. Similarly, a cross-sectional view of classification accuracy as a function of the frequency offset, with SNR fixed at 10 dB, is shown in Figure 3.8. These figures help to further illustrate the trends observed in Figures 3.4–3.6. For example, in Figure 3.7 the Freq.2.5 CNN approaches the accuracy of the Ideal CNN as SNR increases, while the accuracy of the Freq.5 CNN remains about 15% worse.
Figure 3.8: Classification accuracy vs. frequency offset for Ideal, Freq.2.5, and Freq.5 CNNs tested at an SNR of 10 dB. Maximum classification accuracy decreases significantly as the CNN is trained for broader ranges of frequency offsets.

Finally, to evaluate if classification accuracy can be improved by changing the number of input samples, the neural network with the poorest accuracy, Freq.5, is trained and tested varying the number of input samples from 128 to 512. An SNR sweep with samples at no frequency offset for three different input sizes is shown in Figure 3.9. Each increase in the number of input samples improves classification accuracy across all SNRs, illustrating that classification accuracy degradations due to errors in frequency offset can be mitigated by increasing the number of input samples. Detailed analysis of this trend is left as future work.
3.4. Simulations

Figure 3.9: Classification accuracy vs. SNR for Freq.5 CNN trained with varying input size. Increasing the number of inputs to the CNN improves classification accuracy across all SNRs.

3.4.2 Sample Rate Offset

To evaluate the effect of sample rate offset on modulation classification accuracy, the sample rate is swept from the signal bandwidth to approximately eight times the bandwidth. Classification accuracy is then evaluated for the following three neural networks: Ideal, Samp_2.5, and Samp_4. Plots of each neural network’s classification accuracy over the entire sample space are shown in Figures 3.10–3.12. Note that for all three neural networks, the nominal, ideal sample rate is twice the bandwidth. As can be seen, the classification accuracy of each CNN decreases quickly for sample rates outside of the training range, and this decrease is consistent across all SNRs and all neural networks. However, unlike for frequency offset, a noticeable difference cannot easily be observed between peak classification accuracies as the sample rate range in the training data is increased.
Figure 3.10: Classification accuracy of Ideal CNN trained assuming a sample rate of twice the bandwidth. Accuracy quickly drops as the tested sample rate deviates from twice the bandwidth.
Figure 3.11: Classification accuracy of Samp.2.5 CNN trained assuming sample rates between 1.5 and 2.5 times the bandwidth. Accuracy is relatively flat within the training range, but quickly drops as the tested sample rate decreases below 1.5 or increases above 2.5 times the bandwidth.
Figure 3.12: Classification accuracy of Samp.4 CNN trained assuming sample rates from the signal bandwidth to 4 times the bandwidth. Accuracy is relatively flat within the training range, but quickly drops as the tested sample rate increases beyond 4 times the bandwidth.
A cross-sectional view of classification accuracy as a function of SNR, with sample rate fixed at 2 times the bandwidth, is shown in Figure 3.13. Similarly, a cross-sectional view of classification accuracy as a function of the sample rate, with SNR fixed at 10 dB, is shown in Figure 3.14. These plots help to further illustrate the trends observed in Figures 3.10–3.12. All three neural networks converge to the same classification accuracy for SNRs above approximately 11 dB, and the decrease in classification accuracy for lower SNRs is not as pronounced as it is for frequency offset. This suggests that when designing a detector and accounting for estimation errors in training, an accurate sample rate estimation is less important than an accurate center frequency estimation.
Figure 3.14: Classification accuracy vs. sample rate for Ideal, Samp_2.5, and Samp_4 CNNs tested at an SNR of 10 dB. Maximum classification accuracy does not decrease significantly as the CNN is trained for broader ranges of input sample rates.

Finally, as was done for frequency offset, the number of input samples is varied to evaluate if classification accuracy can be improved by changing the number of inputs to the neural network. The neural network with the poorest accuracy, Samp_4, is trained and tested varying the number of input samples from 128 to 512. An SNR sweep with samples at twice the signal bandwidth for three different input sizes is shown in Figure 3.15. While the frequency offset simulation showed notable improvement across all SNRs as the number of input samples increased, the CNN trained to generalize over sample rate offset only achieves a measurable improvement at lower SNRs.
3.4. Simulations

3.4.3 Sample Rate Offset and Frequency Offset

To analyze how a neural network that must account for both frequency offset and sample rate offset performs under ideal conditions, the Samp_Freq neural network is compared to the Ideal, Freq_2.5, and Samp_2.5 neural networks over the SNR range of 0 to 20 dB. Once again, the data used has a frequency offset of zero, and is sampled at the ideal rate of twice the bandwidth. The results are shown in Figure 3.16. Apart from a dip in accuracy for the Freq_2.5 neural network around 6 dB, the Samp_Freq neural network is consistently less accurate than the other networks across the entire SNR range. The drop in accuracy shows that arbitrarily training a neural network to generalize over frequency and sample rate offsets can negatively impact overall performance, even with an ideal detector. Additionally, while the Ideal, Freq_2.5, and Samp_2.5 neural networks all converge to roughly the same peak classification accuracy.
classification accuracy (around 90%), the Samp_Freq neural network converges to a lower peak classification accuracy just above 80%. Over the given offset ranges, generalizing for a single offset only has a marginal effect at high SNRs. However, the lower peak classification accuracy of the Samp_Freq neural network, which is the most realistic of the given scenarios, is something that will need to be accounted for when designing real world systems.
Chapter 4

Augmenting Over-the-Air Signal Captures

4.1 Introduction

One of the important advantages of deep learning AMC on raw IQ is the ability of deep neural networks to autonomously learn relevant features directly from the raw IQ samples in order to make classification decisions. This advantage allows for scaling to arbitrary modulations, without the need for expert feature design. However, in order to create the synthetic datasets from Chapter 3, expert knowledge of modulation types and channel models had to be known, and examples had to be carefully created in order to accurately model the real world. In this chapter, the feasibility of training deep neural networks for AMC using only over-the-air signal captures is explored, and the importance of expanding over-the-air signal capture datasets through augmentations that directly manipulate the raw IQ samples is shown. By using over-the-air signal captures, prior knowledge of channels and modulations can be eliminated, removing the need for expertly designed synthetic datasets and allowing
deep neural networks used for AMC to more easily scale to arbitrary scenarios through directly training on signals captured in those scenarios. However, as shown in Chapter 3, it will still be crucial to account for inaccuracies in the detection and isolation stage of the receiver, which may be very difficult to do with a single signal capture.

One of the primary downsides to using real-world data is that labeled training datasets can be difficult and/or expensive to obtain [28]. This is a familiar problem in other domains such as image processing, where labeled training datasets are augmented to expand the size of the datasets and improve neural network generalization performance [48, 77, 78]. However, augmenting datasets has not been well studied in the radio frequency (RF) domain. This chapter will show how real-world, over-the-air signal captures can be augmented to not only address the receiver effects considered in Chapter 3, but also effectively expand the size of the training datasets to improve generalization performance of the neural network. Specifically, this chapter demonstrates how real, over-the-air signal captures can be augmented by applying noise, frequency shifts, and resampling to improve classification accuracy between the eight modulation classes from Chapter 3 using deep neural networks trained on raw IQ samples. These techniques are extensible to permit generalized labeled training dataset collection without acquiring or storing massive amounts of streaming signal captures.

The rest of this chapter is outlined as follows. In Section 4.2, a description of the two neural networks used in the experiments from this chapter are outlined. In Section 4.3, a high level description of the over-the-air capture environment is described, and the considered datasets and augmentation techniques are introduced. In Section 4.4, the primary augmentation simulation results are presented. Finally, in Section 4.5, applicability to other hardware and transmitter/receiver pairs is considered. Conclusions and future work will be summarized in Chapter 5.
4.2 Neural Network Architectures

For all simulations presented in this chapter, two different neural network architectures were used for analysis. One is the convolutional neural network architecture used in Chapter 3 and shown in Figure 3.2, and the other is a simple recurrent neural network. The reader is referred to Section 3.2 for details on the CNN architecture. The RNN consisted of a single layer of 128 LSTM cells [40]. Each LSTM cell was configured with hyperbolic tangent activation functions on the forward connections, and hard sigmoid functions on the recurrent connections. Following the layer of 128 LSTM cells was a single fully connected layer which, like the CNN, used a softmax function on the output for choosing between one of eight possible modulation classes. Unlike the CNN architecture which uses two-dimensional convolutions over the I and Q values, the inputs into the LSTM architecture are formatted so that each sample, or timestep, contains two dimensions, namely the I and Q values for each particular sample. Also, similar to the CNN architecture, this particular RNN architecture was chosen because it is relatively simple and quick to train, and generally performed better than the CNN architecture on the streaming IQ input. All training and evaluation of both neural networks was performed using the Keras neural network API [25].

It is worth noting that in terms of number of trainable parameters, and therefore the capacity/complexity of the model, the LSTM architecture is significantly smaller than the CNN architecture. While the CNN architecture has about 2,000,000 trainable parameters, the LSTM architecture only has about 70,000. Despite the smaller number of parameters, in general the LSTM architecture shows better performance than the CNN architecture, and this improvement will be highlighted in the coming sections. The improved performance of the LSTM architecture is likely due to the architecture being explicitly designed for sequential data and learning only temporally relevant features, whereas the comparative size and design of the CNN architecture may make it prone to learning insignificant and/or un-
helpful features. A more thorough investigation of the relationship between network size, architecture, and performance is left as future work.

4.3 Datasets and Augmentation

4.3.1 System Setup

Data captures were performed using over-the-air, line-of-sight channels in a lab environment, with radios about 1-2 meters apart. For the particular environment, a center frequency of 2.4 GHz was found to be relatively quiet and was used as the nominal center frequency for all signal captures. Both the transmitter and receiver were Ettus Research USRP B205minis, capable of tuning between 70 MHz and 6 GHz while supporting an instantaneous bandwidth of up to 56 MHz [79].

The GNU Radio flowgraph developed to automate signal captures is shown in Figure 4.1. This flowgraph is made up of three primary components: the transmit chain, the receive chain, and the SNR estimation chain. For the transmit chain, the random signal block from the open source gr-signal_exciter [80] is connected to a USRP sink block. The sink block is configured to transmit using a USRP B205mini at an arbitrary center frequency of 2.4 GHz. Similarly, the receive chain combines a USRP source block with automatic gain control (AGC), followed by a skip head block, a head block, and a file sink block. Similar to the way the USRP sink block was set up, the USRP source block is configured for a USRP B205mini tuned to 2.4 GHz. The AGC is used to help stabilize the input signal to have relatively constant amplitude, while the skip head block removes any transients that may be present during startup of the flowgraph. Finally a head block and file sink block are connected to save a specific number of raw IQ samples to disk.
4.3. Datasets and Augmentation

While not strictly necessary for the data capture, the flowgraph also incorporates an estimate of the SNR by connecting a polyphase clock sync block, a Costas loop block, and an MPSK SNR estimation block to the output of the AGC block. The MPSK SNR estimation block estimates the SNR of PSK signals by using the 2nd and 4th order moments, but only works well on the symbols (not the raw IQ) of the PSK signal. Therefore, the polyphase clock sync is used to time-align and match filter the signal, while the Costas loop accounts for any errors in carrier frequency. Note that while modulations beyond PSK were captured, SNR estimates were only being performed on a QPSK signal, and the SNR was assumed to be approximately the same for the other modulation formats.
4.3.2 Training Dataset Generation

A summary of the data capture parameters is shown in Table 4.1. To evaluate the effect of the noise, frequency shift, and resample augmentation techniques on neural networks used for AMC, as well as how important a good data capture is, four primary datasets were generated using the flowgraph shown in Figure 4.1. All datasets consisted of an equal number of examples for each class in the set BPSK, QPSK, 8PSK, 2FSK, 4FSK, 8FSK, 16QAM, and 64QAM. Two of the datasets contained 10,000 examples of each modulation, while the other two contained 100,000 examples each for total dataset sizes of 80,000 and 800,000, respectively. All datasets were created using a continuous data capture per class, and individual examples were formed by breaking up the continuous data capture into chunks of 256 raw IQ samples, which directly became the inputs to the neural networks. In addition to the sizes of the datasets, the SNR of the captures also varied. For two of the datasets the nominal SNR at the receiver was 15 dB, while for the other two the nominal SNR was 7.5 dB.

At the transmitter, signals were transmitted at a center frequency of 2.4 GHz and a sample rate of 1 MHz, with an approximate bandwidth of 500 kHz. Similarly, at the receiver raw IQ streams were captured at a sample rate of 1 MHz and 2.4 GHz. This was done in an attempt to minimize any of the potential receiver effects explored in Chapter 3, such as sample rate mismatch and center frequency offset. While not strictly necessary to perfectly tune to the signal, the goal was to get as pristine examples as possible in order to ease the analysis of the feasibility of data augmentation when using over-the-air signal captures for neural network based AMC. Note that at these sample rates, each class in the smaller dataset consisted of a single 2.56 second continuous capture, while the larger dataset was formed using a 25.6 second continuous capture for each class.
### 4.3. Datasets and Augmentation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Dataset 1</th>
<th>Dataset 2</th>
<th>Dataset 3</th>
<th>Dataset 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples per Class</td>
<td>10,000</td>
<td>10,000</td>
<td>100,000</td>
<td>100,000</td>
</tr>
<tr>
<td>SNR at Receiver</td>
<td>15 dB</td>
<td>7.5 dB</td>
<td>15 dB</td>
<td>7.5 dB</td>
</tr>
<tr>
<td>Transmitter $F_c$</td>
<td>2.4 GHz</td>
<td>2.4 GHz</td>
<td>2.4 GHz</td>
<td>2.4 GHz</td>
</tr>
<tr>
<td>Receiver $F_c$</td>
<td>2.4 GHz</td>
<td>2.4 GHz</td>
<td>2.4 GHz</td>
<td>2.4 GHz</td>
</tr>
<tr>
<td>Sample Rate</td>
<td>1 MHz</td>
<td>1 MHz</td>
<td>1 MHz</td>
<td>1 MHz</td>
</tr>
<tr>
<td>Samples per Symbol</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Capture Duration</td>
<td>2.56 s</td>
<td>2.56 s</td>
<td>25.6 s</td>
<td>25.6 s</td>
</tr>
</tbody>
</table>

### 4.3.3 Test Dataset Generation

Unlike the training datasets, where the center frequency and sample rates were held fixed in an attempt to simulate a single data capture, the over-the-air test dataset was designed to model errors in a detection and isolation stage of a blind receiver as outlined in Chapter 3, where future signal acquisitions will include signals with varying power, as well as signals with less accurate center frequency and bandwidth estimates. To this end, an evaluation dataset was made with signals varying from about 0 dB to 15 dB SNR, sample rates that varied between approximately 2 and 2.5 samples per symbol, and frequency offsets that ranged from $\pm2.5\%$ of the sample rate. These ranges were leveraged from Chapter 3, allowing for a reasonable error range in the detection and isolation stage of a receiver without unnecessarily complicating the dataset. As an example, such a $\pm2.5\%$ frequency uncertainty translates to $\pm25$ kHz center frequency error for a 1 MHz signal, which is an easily achievable bound for most practical hardware platforms. Additionally, due to the augmenting constraints in this work, i.e. not having enough input samples if the signal captures were downsampled, data below 2 samples per symbol was not used in creating the test dataset. For each SNR point shown, the resulting classification accuracy was calculated using 50,000 underlying signals per signal class.
4.3.4 Augmenting Existing Data

As mentioned, dataset augmentation has been heavily employed in other machine learning applications such as image processing, but it has yet to be explored extensively in the RF domain. To expand the training dataset and evaluate generalization ability of the neural network, three types of data augmentation were explored: adding noise, introducing a frequency shift, and resampling. For all fixed size training datasets using augmented data, the original data was always kept as part of the training dataset, and any augmented examples were simply appended to the dataset, resulting in exponential growth of the training dataset for each augmentation applied. When adding noise, white Gaussian noise with zero mean and random variance was added so that the SNR of the augmented signals, rather than being fixed around 15 or 7.5 dB, varied from approximately 0 to 15 dB, or 0 to 7.5 dB depending on the dataset. Note that after the signal had been captured, it is only possible to decrease the SNR by adding noise; the noise cannot realistically be subtracted from the capture. For frequency shift augmentations, examples in the training dataset were randomly frequency shifted between $\pm 2.5\%$ of the sample rate. Similarly, resampling augmentations were applied so that rather than examples consisting of approximately 2 samples per symbol, examples were allowed to have up to 2.5 samples per symbol. These augmentation ranges were used because Chapter 3 clearly showed that classification accuracy is maximized when the training dataset is tuned to the expected estimation error ranges in a receiver, and these were the assumed estimation error ranges as described in Section 4.3.3.

4.4 Simulation Results

For all of the plots in this chapter, the CNN and LSTM architectures were evaluated with the same data, captured according to Section 4.3.3. Where appropriate, to get an idea of
baseline expectations for the neural networks, the performance of networks trained with synthetic data is shown as well, where both the CNN and LSTM networks are trained on 256 synthetically generated raw IQ samples, with 100,000 examples per class. The SNR range of the synthetically generated examples was 0 to 20 dB, the sample rate varied from approximately 1.5 to 2.5 samples per symbol, and the frequency shift ranged uniformly over ±2.5% of the sample rate, as described in Chapter 3. Note that for a fair baseline, the dataset from Chapter 3 was modified slightly so that modulation specific parameters such as pulse shape, rolloff factor, pulse length, and modulation index were fixed, in an attempt to be consistent with the over-the-air captures.

4.4.1 Synthetic Datasets

To demonstrate both the upside and downside of synthetic training datasets, both the CNN and the LSTM architectures are trained using synthetic data that represents the real world in different ways. In the over-the-air lab environment described in Section 4.3, the primary real-world effects that will need to be accurately modeled in the synthetic training dataset are AWGN, frequency offsets, and sample rate mismatches. Figure 4.2 and Figure 4.3 show the overall classification accuracy of the CNN and LSTM architectures, respectively, when only a portion of the real world is accurately modeled in the synthetic training dataset. If the model of the real world is incomplete, overall classification accuracy is poor when deployed in real-world scenarios. However, if the synthetic training dataset does accurately model the real world, as is the case with the ‘Noise, freq shift, resamp’ curves, neural networks can perform very well when used in real-world scenarios. Given that the lab environment is well defined, generating a synthetic training dataset that appropriately models the environment is relatively simple and can clearly act as an “oracle” for baseline comparison with the augmentation scenarios presented in the rest of this chapter.
Figure 4.2: Classification accuracy of CNN architecture for various training scenarios using synthetic datasets. Synthetic training datasets that do not accurately model the real world result in poor neural network performance when deployed in real-world systems.
4.4. Simulation Results

Figure 4.3: Classification accuracy of LSTM architecture for various training scenarios using synthetic datasets. Synthetic training datasets that do not accurately model the real world result in poor neural network performance when deployed in real-world systems.
Figure 4.4: Classification Accuracy of CNN architecture for various training scenarios using Dataset 1 and a single augmentation. None of the augmentations show improvements over the CNN trained with synthetic data, and only augmenting with noise is capable of doing better than random guessing at lower SNRs.

4.4.2 Augmenting Dataset 1: 10,000 Examples at 15 dB SNR

As a first test of augmenting datasets for neural network based AMC, both the CNN and LSTM architectures were trained by adding a single augmentation to the data from Dataset 1 in Table 4.1. Specifically, as described in Section 4.3.4, the augmentations were adding noise, applying a frequency shift, or resampling the data. Figure 4.4 shows the classification accuracy versus SNR of the CNN architecture when trained on synthetic data, over-the-air data that has not been augmented, and over-the-air data that has a single augmentation. Similarly, Figure 4.5 shows classification accuracies for the same scenarios, but for the LSTM architecture instead of the CNN architecture.
Figure 4.5: Classification Accuracy of LSTM architecture for various training scenarios using Dataset 1 and a single augmentation. None of the augmentations show improvements over the LSTM trained with synthetic data, and only augmenting with noise is capable of doing better than random guessing at lower SNRs.

In both cases, no single augmentation technique is adequate for significantly improving performance vs. the non-augmented dataset. As shown in Chapter 3, accounting for estimation errors in a receiver is important for maximizing classification accuracy, and of all the scenarios considered in Figure 4.4 and Figure 4.5, only the networks trained with synthetic data adequately account for these imperfections. Augmenting with noise is clearly necessary for improving performance at lower SNRs, however this augmentation comes at the expense of decreasing classification accuracy at higher SNRs because the neural networks must now generalize over a broader range of SNRs. Given that the evaluation dataset includes different SNRs, sample rate offsets, and frequency offsets, the results from Chapter 3 show that it is critical that all three of these must be accounted for in the training dataset. Figure 4.4 and
Figure 4.5 support this finding, and indeed it is not surprising that no single augmentation technique provides significant improvement. While the original, non-augmented capture will have some sample rate and frequency offset due to hardware inaccuracies, clearly augmenting the capture further will be vital to future performance on unseen data. Note that for this particular set of hardware, the largest gain in classification accuracy occurs when the resampling augmentation is applied to the original data capture. This is believed to be because the local oscillators in the hardware were poorly disciplined, resulting in the original data capture containing frequency offsets that covered a wider range when compared to sample rate errors. Also note that when compared to the neural networks trained with synthetic datasets in Figure 4.2 and Figure 4.3, the augmented datasets consistently improve over their synthetic counterparts, further emphasizing the usefulness of augmenting real-world signal captures, especially when it is difficult to accurately model the real world.

Figure 4.6 and Figure 4.7 show the overall performance of the CNN and LSTM architectures when multiple augmentations are applied to the training datasets. With both architectures, training with augmented data where only a frequency shift and a resample were applied is able to match the performance of the network trained with synthetic data at high SNRs. This is because the training set at higher SNRs is most representative of the test dataset, but performance quickly degrades once the SNR expands below the range that was used in training. As was shown in Figure 4.4 and Figure 4.5, augmenting with noise is once again crucial to maximizing accuracy of both networks at lower SNRs, emphasizing the fact that including lower SNR examples during training is vital to future performance on unseen data. Augmenting with all three techniques is clearly the best scenario, and the one that also most closely matches the unseen test dataset. Note that while the versions trained with synthetic datasets are still the best overall performing networks, augmenting with noise, frequency shifts, and resampling allows the performance of the networks to begin to approach their
Figure 4.6: Classification Accuracy of CNN architecture for various training scenarios using Dataset 1 and multiple augmentations. All three augmentations must be used to maximize the effectiveness of the dataset.

synthetic equivalents, and almost match performance in the case of the LSTM architecture. The discrepancy is believed to be due to a combination of the size of the training datasets, the size of the neural networks, and the underlying fidelity of the initial signal capture. Where the networks trained with synthetic data had 100,000 independent examples of each class, the networks trained with all three augmentations have 80,000 correlated examples per class. While the smaller LSTM architecture appears to be able to generalize well with this limited sample size, the CNN likely needs more training data to more closely match the performance when trained with synthetic data. Additionally, the 15 dB capture was near the maximum achievable SNR, and as a result there are likely hardware non-linearities/anomalies that are slightly corrupting the signal capture, affecting classification accuracy at lower SNRs.
Figure 4.7: Classification Accuracy of LSTM architecture for various training scenarios using Dataset 1 and multiple augmentations. All three augmentations must be used to maximize the effectiveness of the dataset.

4.4.3 Augmenting Dataset 2: 10,000 Examples at 7.5 dB SNR

While noise, frequency shift, and resampling augmentations on real world training datasets captured at high SNRs can approach performance of networks trained with synthetic datasets, achieving similar results with lower SNR signal captures is less straightforward. If the original capture is at a lower SNR, such as in Dataset 2 where the nominal SNR at the receiver is 7.5 dB, the same combination of augmentations from Section 4.4.2 appear to be inadequate to match or exceed the performance of the neural networks trained with synthetic datasets across the entire 0-15 dB SNR range. Note that while not explicitly explored, lower quality signal captures that initially contain frequency offsets or sample rate mismatches are likely not to have a significant impact when compared to SNR. This is because both frequency
Figure 4.8: Classification Accuracy of CNN architecture for various training scenarios using Dataset 2 and a single augmentation. None of the augmentations show improvements over the CNN trained with synthetic data, once again demonstrating that multiple augmentation techniques are required. All but the resampling augmentation show almost no improvement throughout the entire SNR range.

Offsets and sample rate mismatches can be mitigated after the fact with multiplication by complex exponentials or resampling. However, noise cannot be readily removed from a signal, and therefore the captured SNR corresponds to the best case SNR example in the training datasets.

Figure 4.8 shows the performance of the CNN architecture for single augmentations, and Figure 4.9 shows the equivalent for the LSTM architecture. As with the higher SNR data capture, no single augmentation technique is able to come close to the neural networks trained with synthetic data. The important take-away from these plots is that classification accuracy is roughly capped at the maximally trained SNR. Even when future test points have higher
Figure 4.9: Classification Accuracy of LSTM architecture for various training scenarios using Dataset 2 and a single augmentation. No scenario is better than the LSTM trained with synthetic data, once again demonstrating that multiple augmentation techniques are required.

SNRs, the networks are not able to classify them with any higher amount of reliability. Said another way, the features that the networks are using to make classification decisions are not features that become easier to recognize as the SNR improves. Also, while the classification accuracy trends of the LSTM architecture are very similar to those shown in Figure 4.5, just scaled and shifted due to the degraded SNR of the initial data capture, training the CNN architecture with non-augmented and frequency shift augmentation datasets results in poor generalization ability of the neural network, effectively never being able to improve beyond random guessing. This trend is likely only observed with the CNN architecture due to the significantly larger number of parameters when compared to the LSTM architecture.

Combining augmentation techniques when training the CNN and LSTM architectures with
4.4. Simulation Results

Figure 4.10: Classification Accuracy of CNN architecture for various training scenarios using Dataset 2 and multiple augmentations. Augmenting the training data by applying noise, resampling, and a frequency shift is critical to improving classification accuracy, but accuracy is still relatively poor throughout the entire SNR range when compared to the network trained with synthetic data.

Dataset 2 improves classification accuracy of both networks at lower SNRs, but still does not match the performance of the “oracles” trained with synthetic data, particularly at higher SNRs. Figure 4.10 shows the classification accuracy of the CNN architecture and Figure 4.11 shows the classification accuracy of the LSTM architecture for these combined training augmentation scenarios. Similar to the results from Section 4.4.2, adding noise, frequency shift, and resample augmentations is crucial for maximizing classification accuracy on future over-the-air scenarios. Comparing the results of Figures 4.8–4.11 to the results in Figures 4.4–4.7, it is clear that while the augmentations are helpful to an extent, the SNR of the initial data capture is crucial if the data is going to be used as training datasets for neural
network based AMC on raw IQ samples, especially for the one-shot augmentation scenarios considered here. Furthermore, an important takeaway from all of the results in Figures 4.8–4.11 is that regardless of architecture, the classification accuracy effectively peaks and remains relatively constant for all SNRs above the maximally trained SNR, in this case 7.5 dB. Expanding these training datasets even further to help bolster classification accuracy is explored more thoroughly in Section 4.4.6.
4.4. Simulation Results

4.4.4 Augmenting Dataset 3: 100,000 Examples at 15 dB SNR

As a third simulation, Dataset 3 is used for training, where the original capture size is increased by an order of magnitude so that rather than 10,000 examples per class as was the case in Datasets 1 and 2, now there is enough data to make 100,000 examples per class. An order of magnitude was chosen so that, in the non-augmented case, the over-the-air captured dataset has the same number of unique examples as the synthetically generated dataset. Note that while the number of unique examples between the synthetic dataset and the over-the-air dataset are now the same, the over-the-air dataset will have examples that are still slightly correlated due to effects such as aliasing and pulse shaping.

Figure 4.12 shows the classification accuracy of the CNN architecture, and Figure 4.13 shows the classification accuracy of the LSTM architecture when training data is augmented with noise, frequency shifts, and resampling, as well as how this combination of augmentations compare to training without augmentation and training with synthetic data. As before, in order to generalize for lower SNRs and improve classification accuracy at higher SNRs, augmentation is crucial during training. However, while the augmented CNN architecture improves over its counterpart in Figure 4.6, the LSTM architecture shows almost no improvement over its counterpart in Figure 4.7. This trend reinforces the conclusions from Section 4.4.2, and helps to illustrate that, for higher SNR signal captures, satisfactory performance can be achieved with augmentations of significantly smaller initial data captures, particularly if the neural network architecture is optimized for the task at hand.
Figure 4.12: Classification Accuracy of CNN architecture for various training scenarios using Dataset 3. Augmenting the training data is critical to improving classification accuracy, and the larger initial capture size helps to improve classification accuracy when compared to using Dataset 1. The CNN trained with synthetic data still has the best performance over the entire SNR range.
Figure 4.13: Classification Accuracy of LSTM architecture for various training scenarios using Dataset 3. Augmenting the training data is critical to improving classification accuracy, and the larger initial capture size does not provide any benefit over using the smaller captures from Dataset 1. The LSTM trained with synthetic data still has the best performance over the entire SNR range.
Chapter 4. Augmenting Over-the-Air Signal Captures

4.4.5 Augmenting Dataset 4: 100,000 Examples at 7.5 dB SNR

Depending on neural network architecture, the results from Section 4.4.2 and Section 4.4.4 show that if initial data captures contain signals at higher SNRs there is not much benefit to simply capturing more data. However, having a high SNR signal capture may not always be practical in real-world scenarios. Therefore, investigating how large, lower SNR captures such as Dataset 4, can be augmented for neural network based AMC on raw IQ is essential when determining the viability of the presented techniques, particularly if improvements might be had over the results shown in Figure 4.10 and Figure 4.11.

Figure 4.14 and Figure 4.15 show classification accuracy when Dataset 4, containing 100,000 signal examples per class captured at 7.5 dB, is augmented with noise, frequency shifts, and resampling, and is then used to train the CNN and LSTM architectures, respectively. The plots also show how this training scenario compares to training with non-augmented and synthetic datasets. Similar to the trends shown in Section 4.4.3, both the CNN and LSTM architectures approximately reach peak classification accuracy around the maximum training SNR of 7.5 dB, and remain relatively constant in performance for all higher SNRs, regardless of augmentation. Note that augmenting is once again essential for accurately classifying lower SNR signals, and that while the peak classification accuracy at high SNRs is lower than the equivalent from Section 4.4.4, training with the lower SNR dataset does improve classification accuracy for lower SNRs. This is likely due to the increased concentration of samples at lower SNRs because the range only goes from 0-7.5 dB instead of 0-15 dB, as well as the lower SNR signal captures occurring well within the linear operating range of the USRPs. By using a larger dataset captured at a lower SNR, all of the sub-optimal classification accuracies seen in Section 4.4.3 can be overcome, and classification accuracies of the neural networks when trained with augmented data are now capable of matching the performance when trained with synthetic data at lower SNRs. These results show that when
the initial signal capture is degraded, i.e. at a lower SNR, it is extremely important that more training examples be used in order to maximize future performance and match the accuracy of a “oracle” trained with synthetic data. However, regardless of the number of training examples, matching the accuracy of a “oracle” that has been trained with synthetic data can only be achieved for the SNR range seen during training, and fails to generalize well to higher SNRs.
Figure 4.15: Classification Accuracy of LSTM architecture for various training scenarios using Dataset 4 and multiple augmentations. Augmenting the training data is critical to improving classification accuracy at lower SNRs, but the classification accuracy plateaus at higher SNRs. With a larger initial capture, it is possible, through augmentation, to match the classification accuracy of the LSTM trained with synthetic data at lower SNRs.

4.4.6 Improving Performance with Dataset 2: 10,000 Examples at 7.5 dB SNR

While the results for training on larger captures at any SNR and smaller captures at high SNR are encouraging, obtaining large captures can be difficult in real-world scenarios, and a receiver may not be able to easily control the SNR of captured signals. Therefore, this section explores more robustly how small data captures at lower SNRs might be able to achieve classification accuracies on par with the other scenarios, by continually augmenting the data so that the same example is unlikely to be repeated during training.
4.4. Simulation Results

Figure 4.16: Training and validation loss for the CNN architecture augmenting Dataset 2 with noise, frequency shifts, and resampling. The CNN begins to overfit immediately.

Both the LSTM and CNN architectures used to generate most of the plots thus far exhibit significant amounts of overfitting with the smaller datasets (Datasets 1 and 2), particularly with the lower SNR dataset. Figure 4.16 shows the validation loss in a typical training run of the CNN for Dataset 2 using noise, frequency shifts, and resampling augmentations, and Figure 4.17 shows the same thing for the LSTM architecture. While the validation loss on the LSTM network follows the training loss for the first 5 epochs, it then begins to significantly overfit the training data. Overfitting on the CNN is even worse, as the validation loss immediately begins to increase. To address overfitting, and hopefully improve the performance of the neural networks, rather than adjust the neural network architectures or apply regularization techniques, one can simply continue to augment the training data, effectively creating a non-repeating number of training examples [81]. Doing this for the
Figure 4.17: Training and validation loss for the LSTM architecture augmenting Dataset 2 with noise, frequency shifts, and resampling. The validation loss of the LSTM network tracks the training loss for the first 5 epochs, but then quickly begins to overfit the training data.

combined noise, frequency shift, and resampling augmentations yields training runs such as Figure 4.18 for the CNN, and Figure 4.19 for the LSTM. By augmenting so that examples seen during training are not repeated, overfitting is drastically reduced for both networks.
Figure 4.18: Training and validation loss for the CNN architecture augmenting Dataset 2 with a non-repeating number of noise, frequency shift, and resampling augmentations. The CNN tracks the training loss for the duration of the training, although begins to overfit around epoch 70.
Figure 4.19: Training and validation loss for the LSTM architecture augmenting Dataset 2 with a non-repeating number of noise, frequency shift, and resampling augmentations. The validation loss of the LSTM network is consistently better than the training loss, but plateaus around epoch 300.
Figure 4.20: Classification accuracy comparison for CNN architecture across four different training scenarios. Training with non-repeated augmentations results in improved accuracy, but is still not as good as the CNN trained with a larger original data capture.

By reducing overfitting to the training data, the overall performance of the network is easily improved as shown in Figure 4.20 and Figure 4.21 for the CNN and LSTM architectures, respectively. In the case of the LSTM architecture, non-repeated augmentations with the smaller dataset show that even with this smaller, lower quality data capture, it is possible to match the performance of the network trained with synthetic data at lower SNRs. In the case of the CNN architecture, non-repeated augmentations show some improvement, but are still unable to match the performance of the CNN trained with synthetic data or the CNN trained on augmenting Dataset 4. The discrepancy is believed to be due to the underlying architectures of the neural networks, in that the CNN seems to, in general, be more inclined to learn unhelpful features in the training dataset and may not be the optimal network design.
Figure 4.21: Classification accuracy comparison for LSTM architecture across four different training scenarios. Training with non-repeated augmentations is clearly the best technique when augmenting over-the-air signal captures.

for this particular classification problem. Whether this is simply related to the relative size of the CNN, or the architecture itself, as well as a more thorough investigation into the possible dependence between architecture choice and training on over-the-air captures, is left as future work. Note that even with non-repeated data, both the CNN and LSTM architectures still effectively plateau for all SNRs above the maximum SNR seen during training, showing that the initial SNR of the captured data is potentially a physical limit for how well the neural network classifier will perform on future, higher SNR examples.
4.5 Applying to New Hardware

As a test of the feasibility and general applicability of augmenting over-the-air captures for training neural networks used for AMC on raw IQ data, some of the best performing networks are tested with different combinations of hardware to see what, if any, hardware specific impairments the networks might not be able to generalize over with the augmentation techniques presented in this thesis. All channels were once again over-the-air, line-of-sight in a lab environment, with USRPs 1-2 meters apart. The dataset capture was performed in exactly the same way as described in Section 4.3, with the addition of different combinations of receiver/transmitter pairs. The available hardware consisted of two USRP B205minis and one USRP B200. For purposes of discussion and simulation, the radios are labeled as follows.

1. **Radio A**: USRP B205 mini 1
2. **Radio B**: USRP B205 mini 2
3. **Radio C**: USRP B200

All training from Section 4.4, and therefore the only data used in training the CNN and LSTM networks, was performed with captures using Radio A as the transmitter, and Radio B as the receiver. Also in this section, any reference to “overall classification accuracy” or “overall performance” is the average accuracy of the neural network across the entire 0-15 dB SNR range.

### 4.5.1 Overall Classification Accuracy

The performance of the neural networks trained with synthetic datasets across the different receiver and transmitter pairs is shown in Figure 4.22. Additionally, the performance of the
Chapter 4. Augmenting Over-the-Air Signal Captures

Figure 4.22: Overall classification accuracy for different receiver and transmitter pairs with neural networks trained using synthetic data. The overall classification accuracy varies fairly significantly, as much as about 8% across the different receiver/transmitter pairs.

neural networks trained with noise, frequency shift, and resampling augmentations on the larger datasets (Datasets 3 and 4) across the different receiver/transmitter pairs is shown in Figure 4.23 and Figure 4.24 for Dataset 3 and Dataset 4, respectively. Note that in these figures, the receiver and transmitter pair that was used to create the original training dataset is highlighted in red.

While the overall performances shown in Figure 4.24 are remarkably consistent across different hardware scenarios, the same cannot be said of the overall performances shown in Figure 4.22 and Figure 4.23. The most noticeable degradation occurs when Radio B is configured as the transmitter, and Radio C is configured as the receiver. Here, when training
Figure 4.23: Overall classification accuracy for different receiver and transmitter pairs with neural networks trained using Dataset 3 augmented with noise, frequency shifts, and resampling. The receiver and transmitter pair used to create the initial training dataset is highlighted in red, and the overall accuracy varies fairly significantly, as much as about 8% across the different receiver/transmitter pairs.

with augmentations on Dataset 3 and synthetic data, the overall accuracy drops by about 8% for both networks. The discrepancies for the networks trained with synthetic data are likely due to missing assumptions in the synthetic dataset that do not completely model all possible hardware imperfections across the different receiver/transmitter setups. However, the consistency in performance when augmentations of Dataset 4 are used vs. the inconsistencies when augmentations of Dataset 3 are used is slightly unexpected and can be better analyzed by looking at overall classification accuracy vs. SNR.
Figure 4.24: Overall classification accuracy for different receiver and transmitter pairs with neural networks trained using Dataset 4 augmented with noise, frequency shifts, and resampling. The receiver and transmitter pair used to create the initial training dataset is highlighted in red, and the overall accuracy is remarkably consistent for all hardware combinations.

In all of the presented test scenarios, the worst overall classification accuracy across hardware configurations occurs when Radio B is the transmitter and Radio C is the receiver. Therefore, using this configuration as a representative ‘worst-case’ scenario, the overall classification accuracy over the entire SNR range of 0-15 dB is analyzed. Figure 4.25 shows the classification accuracy vs. SNR of the CNN architecture when trained with synthetic data, as well as Datasets 3 and 4 after they have been augmented by adding noise, frequency shifts, and resampling. Similarly Figure 4.26 shows the same plots, but for the LSTM architecture. Note that in prior simulations, training on augmented over-the-air captures only
Figure 4.25: Classification accuracy of the CNN architecture with Radio B as the transmitter and Radio C as the receiver under various training scenarios. Training on Dataset 4 augmented with noise, frequency shifts, and resampling results in the best performance at lower SNRs.

ever matched the performance of the models trained with synthetic data. However, while the classification accuracy of both networks trained with noise, frequency shift, and resample augmentations on Dataset 4 still roughly plateaus after 7.5 dB, the performance is now significantly better than both the synthetic and augmented Dataset 3 training scenarios at lower SNRs. This improvement reaffirms the importance of augmenting over-the-air signal captures and shows the benefit that they can have over synthetic datasets by being able to more accurately model real-world environments. As mentioned, the drop in accuracy with the network trained on synthetic data is likely due to an incomplete model of the environment, but the networks trained with augmented Dataset 3 also drop well below the accuracy of the networks trained with augmented Dataset 4 at lower SNRs. This is most likely due to
Chapter 4. Augmenting Over-the-Air Signal Captures

Figure 4.26: Classification accuracy of the LSTM architecture with Radio B as the transmitter and Radio C as the receiver under various training scenarios. Training on Dataset 4 augmented with noise, frequency shifts, and resampling results in the best performance at lower SNRs.

hardware specific non-linearities/anomalies being introduced into the training set at higher SNRs, decreasing the quality of the initial signal capture and forcing the networks to account for these hardware specific imperfections, therefore hurting the generalization ability of the networks across different hardware configurations.

In summary, Figures 4.22–4.26 help to illustrate that any hardware specific imperfections in the training data can slightly degrade performance for other receiver/transmitter pairs, but that these imperfections can be mitigated by ensuring that initial data captures are well within the linear operating regions of the hardware. Similarly, the consistency in overall classification accuracy shown in Figure 4.24 is likely due to the plateau at higher SNRs, and any hardware specific non-linearities that are being introduced at higher SNRs no longer
being incorporated into the training dataset. These results help to reinforce the importance of an initial, good quality, signal capture, as well as the effectiveness and limitations of the proposed augmentation techniques.
Chapter 5

Conclusions and Future Work

Using deep neural networks directly on raw IQ samples is a new and powerful tool for modulation classification applications. However, before these neural networks are realizable, it is important to understand how they will perform in practical scenarios. In most realistic scenarios, a receiver has no \textit{a priori} knowledge about the spectrum it is observing and must use a generic, blind, detection and isolation algorithm before classification can be performed. To address this fact, Chapter 3 investigated the relationship between convolutional neural network classification accuracy and detector quality. It was shown that as a convolutional neural network is trained to generalize over detection estimation parameters in the form of frequency offset and sample rate offset, overall classification accuracy is degraded, even for test signals with no offsets. For the offsets explored in Chapter 3, the degradation in classification accuracy is more pronounced for frequency offset than for sample rate offset. Classification accuracy is further degraded when training for both frequency offset and sample rate offset.

The presented results highlight the importance of accurate center frequency and sample rate estimates in a detector when using similar neural network architectures for AMC on raw
IQ samples. Furthermore, clearly defining expected estimation error ranges, and limiting neural network training to within these ranges, can maximize classification accuracy while minimizing complexity. In other words, it was shown that designing a neural network for an arbitrarily large range not only increases complexity, but negatively impacts the overall classification accuracy, even for signals with no center frequency or sample rate estimation errors.

In Chapter 4 the practicality of using real world, over-the-air signal captures when training neural networks for AMC on raw IQ was investigated and shows promising results, but only if over-the-air captures are augmented during training. While synthetically created datasets are typically much easier to create than over-the-air captures, and allow for an almost endless amount of training data, it can be very difficult to account for all types of future modulations and RF environments. Being able to effectively train neural networks with over-the-air captures will become exceedingly important as AMC with neural networks on raw IQ continues to develop.

Through simulation, it was shown that there is a reasonable trade-off between the fidelity of a signal capture and the classification accuracy of the neural networks, and that simple augmentations such as applying noise, applying a frequency shift, and resampling the training examples are essential to maximizing classification accuracy. In particular, all three augmentations are critical to include for the best performance on future signal captures. Additionally, it was shown that the SNR and duration of the original data capture is extremely important. For a given classification accuracy, and for the CNN considered in this work, the lower the SNR of the signal capture the longer the capture must be. The LSTM architecture was less sensitive to the length of the initial data capture at lower SNRs, but for both architectures the peak classification accuracy plateaus at the SNR of the original signal capture, and while augmenting can increase the classification accuracy at which the
networks plateau, none of the presented augmentation techniques allow the classification accuracy to continue to improve for SNRs above the training range.

The CNN and LSTM architectures considered showed similar trends for all simulations, although the LSTM architecture consistently out-performed the CNN architecture. Additionally, training with augmented over-the-air signal captures for a specific transmitter and receiver pair was able to generalize to other combinations of hardware, particularly when the signal captures were well within the linear operating ranges of the radios.

In summary, the key takeaways from this work are the following.

- **Chapter 3: Receiver Effects**
  - Errors in a detection and isolation stage of a blind receiver must be accounted for in training datasets, and the larger these errors the worse the performance of the neural network classifier.
  - Training for and generalizing across a range of frequency and sample rate offsets permanently and negatively affects the peak classification accuracy of neural network classifiers, even if future signals are perfectly tuned and have no frequency or sample rate offsets.
  - CNNs designed for AMC on raw IQ appear to be more robust to sample rate offsets than frequency offsets.
  - Classification accuracy of neural network classifiers can be improved as the number of input samples increases, helping to mitigate degradations from blind receiver estimation errors.

- **Chapter 4: Augmenting Over-the-Air Signal Captures**
  - When creating synthetic training datasets, inaccurate models of the real world
result in neural networks that are unable to accurately classify signals when used in real systems.

- Accurate models of the real world can create good training datasets that allow neural networks designed for AMC on raw IQ to perform very well when used in real systems.

- In the absence of accurate synthetic models, over-the-air signal captures can be used to create training datasets that allow neural network classifiers to achieve similar performance to networks trained with accurate synthetic datasets, provided the captures are appropriately augmented.

- Simple augmentations such as adding noise, applying a frequency shift, and resampling can not only increase the size of the over-the-air training dataset, but are required for general applicability to future blind receiver scenarios.

- With the presented augmentation techniques, the underlying fidelity of the initial signal capture used for training is crucial to maximizing classification accuracy, and the better the initial signal capture, the shorter the initial capture can be.

  * The SNR of the initial signal capture appears to be a limiting factor in how well neural networks are able to classify unseen, higher SNR signals.

  * If undesired, hardware specific effects are present in the initial signal capture, future performance of the neural network classifiers can be negatively affected. However, this effect is not detrimental and appears to only result in slightly worse classification accuracies when compared to networks trained with synthetic datasets.

- LSTM neural networks appear to be better suited to AMC on raw IQ than CNNs.

- Augmenting training datasets captured with a particular transmitter/receiver pair still allows for generalization to other hardware combinations, reinforcing that
through augmentation, any emitter specific features that are potentially being learned are likely not detrimental to the broad applicability of the neural network classifier.

To extend the results of Chapter 3, in future work it will be important to investigate different neural network architectures beyond just the CNN architecture described by Figure 3.2 in order to more broadly define and quantify blind receiver effects on the performance of neural network classifiers. Additionally, design choices such as increasing/decreasing the size of the neural network, increasing the number of training samples, or increasing the number of raw IQ input samples to the neural network may produce different trends and are all parameters that will need to be more thoroughly explored and quantified. Finally, determining the effect of estimation errors for an expanded dataset that includes other types of modulation will be crucial to further generalizing the conclusions of this chapter, beyond just the 8 modulations considered.

Future work to extend Chapter 4 will need to consider moving beyond just a line-of-sight lab environment to more complicated scenarios such as multipath environments. Similar to the future work from Chapter 3, future work for Chapter 4 will also need to evaluate how neural networks trained with augmented over-the-air captures extend to other types of modulation such as higher order QAM or orthogonal frequency-division multiplexing. Additionally, all over-the-air datasets in this work were captured at roughly the same center frequency, and it will be important to evaluate if there are any frequency dependent receiver/transmitter effects that can degrade the neural network classifier, and may require more novel forms of augmentation. While multiple transmitter and receiver pairs were considered in this work, all of the radios were similar models, and it will be important to more thoroughly evaluate hardware dependencies that may or may not be learned by raw IQ neural networks trained with over-the-air captures, particularly if these techniques are to scale to other applications.
such as emitter identification. Using other radio specific augmentation techniques such as IQ imbalance, combining augmented over-the-air captures with synthetically generated data, or conditioning classification decisions based on SNR may show even better results than the ones presented in this work. Furthermore, one of the dominant trends in Chapter 4 was the plateau seen for the networks trained by augmenting lower SNR signal captures. Continuing the analysis of this plateau and looking at trends beyond 15 dB will be vital towards determining if this plateau is in fact a limiting effect for any SNR, or if it is only a trend observed at lower SNRs. Finally, exploring how increasing the number of raw IQ input samples into the neural network can improve or degrade performance could be useful, particularly as more complicated/higher order modulations are explored.
Bibliography


