On the Use of Convolutional Neural Networks for Specific Emitter Identification

Lauren Joy Wong

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Masters of Science

in

Electrical Engineering

Alan J. Michaels, Co-chair

Jia-Bin Huang, Co-chair

William C. Headley

A. A. (Louis) Beex

April 12, 2018

Blacksburg, Virginia

Keywords: Specific Emitter Identification, Convolutional Neural Networks, IQ Imbalance, Estimation, Feature Learning, Clustering

On the Use of Convolutional Neural Networks for Specific Emitter Identification

Lauren Joy Wong

ABSTRACT

Specific Emitter Identification (SEI) is the association of a received signal to an emitter, and is made possible by the unique and unintentional characteristics an emitter imparts onto each transmission, known as its radio frequency (RF) fingerprint. SEI systems are of vital importance to the military for applications such as early warning systems, emitter tracking, and emitter location. More recently, cognitive radio systems have started making use of SEI systems to enforce Dynamic Spectrum Access (DSA) rules.

The use of pre-determined and expert defined signal features to characterize the RF fingerprint of emitters of interest limits current state-of-the-art SEI systems in numerous ways. Recent work in RF Machine Learning (RFML) and Convolutional Neural Networks (CNNs) has shown the capability to perform signal processing tasks such as modulation classification, without the need for pre-defined expert features. Given this success, the work presented in this thesis investigates the ability to use CNNs, in place of a traditional expert-defined feature extraction process, to improve upon traditional SEI systems, by developing and analyzing two distinct approaches for performing SEI using CNNs. Neither approach assumes *a priori* knowledge of the emitters of interest. Further, both approaches use only raw IQ data as input, and are designed to be easily tuned or modified for new operating environments. Results show CNNs can be used to both estimate expert-defined features and to learn emitter-specific features to effectively identify emitters.

On the Use of Convolutional Neural Networks for Specific Emitter Identification

Lauren Joy Wong

GENERAL AUDIENCE ABSTRACT

When a device sends a signal, it unintentionally modifies the signal due to small variations and imperfections in the device's hardware. These modifications, which are typically called the device's radio frequency (RF) fingerprint, are unique to each device, and, generally, are independent of the data contained within the signal.

The goal of a Specific Emitter Identification (SEI) system is to use these RF fingerprints to match received signals to the devices, or emitters, which sent the given signals. SEI systems are often used for military applications, and, more recently, have been used to help make more efficient use of the highly congested RF spectrum.

Traditional state-of-the-art SEI systems detect the RF fingerprint embedded in each received signal by extracting one or more features from the signal. These features have been defined by experts in the field, and are determined ahead of time, in order to best capture the RF fingerprints of the emitters the system will likely encounter. However, this use of pre-determined expert features in traditional SEI systems limits the system in a variety of ways.

The work presented in this thesis investigates the ability to use Machine Learning (ML) techniques in place of the typically used expert-defined feature extraction processes, in order to improve upon traditional SEI systems. More specifically, in this thesis, two distinct approaches for performing SEI using Convolutional Neural Networks (CNNs) are developed and evaluated. These approaches are designed to have no knowledge of the emitters they may encounter and to be easily modified, unlike traditional SEI systems.

Contents

List of Figures

List of Tables

List of Abbreviations

AMI	Adjusted Mutual Information
API	Application Programming Interface
AWGN	Additive White Gaussian Noise
CNN	Convolutional Neural Network
CPU	Central Processing Unit
dB	deci-Bel
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DSA	Dynamic Spectrum Access
FFT	Fast Fourier Transform
GPU	Graphical Processing Unit
ID	Identification
IQ	In-phase/Quadrature

kNN	k-Nearest Neighbors
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
NMSE	Normalized Mean Squared Error
OFDM	Orthogonal Frequency Division Multiplexing
pdf	Probability Density Function
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
QPSK	Quaternary Phase Shift Keying
ReLU	Rectified Linear Unit
RF	Radio Frequency
RFML	Radio Frequency Machine Learning
RMS	Root Mean Square
RNN	Recurrent Neural Network
SEI	Specific Emitter Identification
SNR	Signal-to-Noise Ratio

SP	Signal Processing
SVM	Support Vector Machine
t-SNE	t-distributed Stochastic Neighbor Embedding
USRP	Universal Software Radio Peripheral

Chapter 1

Introduction

1.1 Motivation

Specific Emitter Identification (SEI) is the act of matching a received signal to an emitter using a database of radio frequency (RF) features belonging to known transmitters. SEI algorithms were developed for and continue to be used in military settings for emitter tracking [1]. However, emitter identification has also become a powerful tool for use in cognitive radio applications, such as enforcing Dynamic Spectrum Access (DSA) rules [2–4].

Successful SEI systems must be able to reliably identify emitters, but must also be fast, robust to changing environments, and easily and quickly adaptable. The speed of such systems is especially critical in a military setting, where SEI may be used to provide early warning [5]. Furthermore, in an environment in which the channel may be changing, the emitters may be changing modulation schemes, bandwidth, and transmission frequency, and new emitters may be encountered, it is important that the designed system be able to

operate under such conditions. However, should the system need to be tuned or modified in order to accommodate a new environment, such changes should be simple and efficient to implement [1].

State-of-the-art SEI systems rely on the accurate measurement of expert-defined signal features, which are then clustered by emitter for identification [1]. In addition to accurate measurement, it is also important that each emitter's features are consistent between transmissions, but different amongst emitters. Ideally, the selected features are also robust to effects such as noise, channel effects, and transmission bandwidth. However, this is often not achieved [4]. Furthermore, the extraction of expert features often requires considerable pre-processing of the raw signal data.

The development of current SEI systems starts with considerable visual examination of signals collected from emitters of interest, in order to determine possible features. Then, signal processing software is developed to extract the selected features. Finally, these features are clustered to determine the success of the selected features in describing the emitters of interest, determining whether the system is validated or the development process is restarted [1]. This development process is not only time-consuming, but requires considerable expert intervention to determine the specific signal features of interest. Furthermore, current SEI systems are designed for specific operating environments and emitters-of-interest and must be re-designed for new or changing environments and in order to identify new emitters.

With all of this in mind, it is clear that current SEI systems are most limited by the use of pre-determined and expert-defined signal features, which not only slow the execution

time of deployed SEI systems, but make for a prohibitively slow system development and modification process. As such, this thesis reports on an investigation into the feasibility of using Machine Learning (ML) techniques to overcome these weaknesses.

The field of ML uses concepts from computer science and statistics to develop algorithms and methods to identify patterns in large amounts of data [6]. More specifically, ML algorithms optimize a parameterized model, using a set of training data, to perform a desired task. ML methods vary widely, with common methods including discriminant analysis, kernel-based algorithms, hidden Markov models, and artificial neural networks [7]. The fields in which ML algorithms have been applied also vary widely, and include economics, linguistics, biology, image processing, and engineering [7].

Research in ML for Signal Processing (SP) investigates novel techniques and advancements in the application of ML to the processing of signals. However, work in ML for SP has primarily been limited to "audio, speech, image, multispectral, industrial, biomedical, and genomic signals" [8]. The scope of work using ML for RF signal processing, or Radio Frequency Machine Learning (RFML), is far newer, with applications including modulation and waveform classification [9–11], link adaptation [12], and cooperative spectrum sensing [13].

This thesis examines the application of ML for SEI with the primary goal of improving upon traditional SEI approaches by:

- Reducing the pre-processing time typically needed to extract emitter-specific features
- Eliminating the use of pre-selected and expert-defined features

In doing so, a less time-consuming system development process and more adaptive system is achieved, and a greater understanding of ML abilities for the purpose of SEI is gathered.

1.2 Outline and Contributions

This thesis reports on an investigation into the feasibility of using ML techniques, in place of the current expert-defined feature extraction process, to improve upon traditional SEI systems. To this end, the work presented in the following chapters develops and evaluates two distinct approaches to perform SEI, both using Convolutional Neural Networks (CNNs).

Chapters 2 and 3 provide the background necessary to motivate and develop the approaches presented in the following chapters. In Chapter 2, some of the traditional approaches to SEI are described, as well as their limitations, further motivating the investigation of deep-learning based approaches such as those described in this thesis. Chapter 3 starts by introducing three of the most popular artificial neural network architectures used in the literature, leading to a discussion of the selection of CNNs for SEI. Next, some of the practical considerations that allowed for the successful use of CNNs in this work are discussed including the software toolboxes used, parameter selection, network training, performance evaluation, and the collection and generation of real and synthetic training data.

Chapter 4 develops the first approach to perform SEI in which CNNs are used to estimate an expert feature, IQ imbalance. Chapter 4 starts by describing transmitter IQ imbalance and developing a signal model to be used throughout the chapter. Some of the traditional

approaches to estimating IQ imbalance are then discussed. Next, the CNN architecture designed to estimate IQ imbalance is described. The approach is evaluated by investigating the bias and sample variance of the designed CNN estimators as a function of signal-to-noise ratio (SNR), network input size, and the true IQ imbalance parameters, showing the capability to estimate gain and phase imbalances in both M-QAM and M-PSK signals. Finally, the approach to perform SEI using the CNN IQ imbalance estimators is described and evaluated, showing the ability to identify emitters by their gain imbalance only, even as they change modulation schemes, with better accuracy than a comparable feature-based approach.

While the SEI approach developed in Chapter 4 continues to rely on expert features, Chapter 5 eliminates the use of expert features completely, and presents an approach to perform SEI using CNN-learned features. The semi-supervised approach developed uses a supervised CNN to extract emitter-specific features from the received signal in conjunction with an unsupervised clustering step. This approach is evaluated in its ability to identify emitters transmitting at a single bandwidth, as well as at multiple bandwidths, and in its ability to identify emitters unseen in CNN training, showing the ability to use CNN-learned features and the DBSCAN clustering algorithm to perform specific emitter identification, even in the presence of emitters the CNN feature extractor did not see in training.

Finally, Chapter 6 provides overall conclusions, highlighting the benefits of the developed approaches over traditional SEI approaches, and presents directions for future work.

1.3 Publications

Conference Papers

- L. J. Wong, W. C. Headley, and A. J. Michaels, "Estimation of Transmitter I/Q Imbalance Using Convolutional Neural Networks," IEEE Annual Computing and Communication Workshop and Conference, 2018.

- L. J. Wong, W. C. Headley, S. Andrews, R. M. Gerdes, and A. J. Michaels, "Clustering of Learned CNN Features from Raw I/Q for Emitter Identification." *(in review MILCOM)*

Journal Papers

- L. J. Wong, W. C. Headley, and A. J. Michaels, "Emitter Identification Using CNN I/Q Imbalance Estimators." *(in review Springer ML)*

Chapter 2

Specific Emitter Identification

2.1 Traditional SEI Techniques

A typical SEI system is shown in Figure 2.1. The system includes an RF system, followed by data collection, signal processing, feature extraction/estimation, clustering, identification, and verification steps [1]. At a high level, the RF system and data collection steps pre-process the analog data gathered from the RF receiver, bringing it to the digital domain, before the signal processing stage. The steps taken at the signal processing stage are largely dependent upon the features that will be extracted in the feature extraction and estimation stage, but often include filtering and demodulation. Once processed, all versions of the data (raw, demodulated, etc.) are passed to the feature extraction and estimation stage where pre-determined features are measured. The resulting features are then used for clustering and identification.

An emitter's RF fingerprint or electromagnetic signature is most commonly caused by

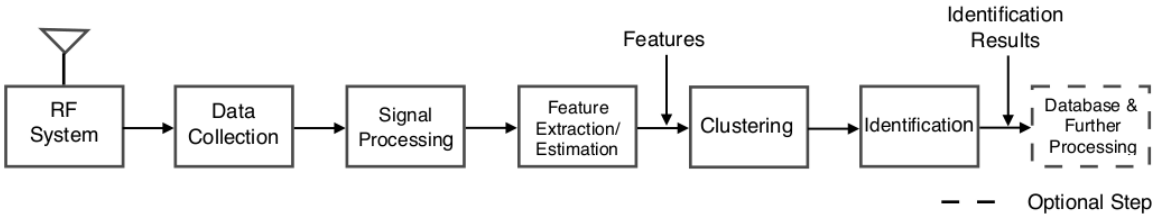


Figure 2.1: The components of a typical SEI system.

natural variations amongst RF components and architectures used by manufacturers and by non-idealities in the emitter's hardware [2, 14]. Further, RF fingerprints are generally independent of the signal transmitted or the data contained within said signal. The success of a traditional SEI systems is largely dependent upon selecting features which characterize some portion of the RF fingerprints of a group of emitters [1]. The features often used in traditional SEI systems are either taken from the transient portion or the steady-state portions of the received signal.

Techniques analyzing the transient signal often use expert features found in the time-domain, frequency-domain, or phase-space [15, 16]. Time-domain features are most commonly used to describe the transient portion of radar signals, but may also be used to fingerprint radio transmitters [15–17]. A popular frequency-domain feature used for SEI is the power spectral density (PSD), as it provides both the signal power and spectral shape of the transmitted signal, and often displays emitter-specific features [14]. Meanwhile, phase-space analysis often yields emitter specific features caused by the non-linear power amplifier of the transmitter [18].

Techniques analyzing the steady-state signal are more diverse and include use of wavelet-

based techniques [19], modulation-based techniques [20], preamble-based techniques [21], and cyclostationary-based techniques [22]. A popular wavelet-based technique, the dynamic wavelet fingerprint technique, applies a wavelet transform on the time-domain portion of the steady-state signal [19]. Features extracted in the modulation-domain often examine the error between the transmitted and ideal demodulated signal [20], while preamble-based techniques examine features of the extracted preamble, such as its periodicity [21]. Finally, cyclostationary-based techniques examine unique cyclic features within a signal [22].

With either transient or steady-state techniques, extracted features are used in conjunction with a clustering or classification algorithm to identify specific emitters and for verification [16]. Popular algorithms used in the literature include support vector machines (SVMs) and k-Nearest Neighbors (kNNs) [16, 19, 20, 23, 24], supported by dimensionality reduction techniques such as linear discriminant analysis (LDA) or principal component analysis (PCA), if needed [17].

While neural networks have been used for emitter identification applications in the literature, prior work could only be found in which they were used to perform the classification stage and not the feature extraction stage. More specifically, the neural network is used in place of the clustering stage in the typical SEI system, taking in pre-defined features as input and producing an identification result [25, 26].

2.2 Limitations of Traditional SEI Techniques

The first key limitation of traditional SEI approaches is the extraction and use of pre-determined and expert-defined features. The first step in designing a traditional SEI system relies on an "expert" to define quality features of interest. These pre-determined features are often only accurate over a valid range of parameters and require accurate and consistent measurement or estimation in order to ensure quality SEI performance.

For example, when using features extracted from the transient signal, SEI performance relies heavily on the accuracy and consistency of the transient detection and extraction process, as this directly affects the quality of the features [24]. Additionally, time-domain and frequency-domain features can vary according to the noise and channel conditions and can therefore be impacted by channel impairments such as multipath [16].

Though using features extracted from the steady-state portion of the received signal is generally more practical, expert features used to describe the steady-state signal often have their own limitations. For example, wavelet-based techniques are heavily impacted by the choice of wavelet function [19]. Preamble-based techniques fail in the case where the received signal does not have a pre-defined preamble [24]. Further, techniques analyzing the cyclostationary features of a signal are often inconsistent in the presence of frequency or phase uncertainties and time-consuming to compute [22].

Additionally, by using pre-defined expert features, traditional SEI approaches only consider specific aspects of the received signal. Therefore emitters that produce the same fea-

tures are indistinguishable to the SEI system. As such, the selection of these features is imperative and leads to a prohibitively long system development process, as discussed in Chapter 1. Further, feature extraction often requires pre-processing of the received signal, including synchronization, carrier tracking, demodulation, and SNR estimation, in addition to the computational cost of extracting the expert features.

Finally, traditional SEI systems are also limited by their choice of clustering or classification algorithm. Many clustering or classification algorithms used in the literature, such as SVMs, kNNs, and neural network classifiers, require the user to input the number of clusters or use an iterative process to determine the optimal number of clusters for the given dataset [24, 25, 27]. Unless in a cooperative environment, the number of clusters (emitters) will not be known in practice and is subject to change, severely limiting the ability to identify anomalous emitters and behaviors.

2.3 Summary

This chapter has described the traditional feature-based approaches to SEI, and further discussed the limitations of the traditional approaches. Namely, current approaches are hindered by the methods used to extract or estimate features, the use of expert features themselves, and the clustering and classification algorithms used.

As such, this thesis seeks to mitigate some of these limitations using Convolutional Neural Networks (CNNs). To eliminate the need for pre-processing steps often needed to extract

expert features, work in Chapter 4 examines the feasibility of using only the raw data as input to CNNs to estimate an expert feature, IQ imbalance. Further, Chapter 5 investigates eliminating expert features entirely, using CNNs as feature extractors.

While the choice of clustering algorithm used in Chapter 5 aims to alleviate some of the concerns associated with the popular algorithms outlined above, investigation into the appropriate choice or design of the clustering algorithm to be used with raw IQ data and/or features extracted from raw IQ data remains as future work.

Chapter 3

Convolutional Neural Networks

Though feed-forward neural networks existed in the literature as early as the 1940s and 1950s [28,29], limitations caused by the back-propagation training algorithm kept early feed-forward neural networks relatively shallow [30]. Consequently, the use of machine learning techniques in the literature has traditionally referred to shallow architectures such as Gaussian Mixture Models (GMMs), Hidden Markov Models (HMMs), SVMs, regression techniques, and shallow feed-forward neural networks [31].

However, in breakthrough work published by Hochreiter in 1991, the flaws of the traditional back-propagation training algorithm were exposed, identifying the problem of vanishing and exploding gradients [32]. This sparked the resurgence of neural network research, focused on methods of overcoming and working around vanishing and exploding gradients [30]. Additionally, recent advances in computing technology and software toolboxes have made the use of deep feed-forward neural networks both computationally practical and accessible,

further encouraging the popularity of neural networks in pattern recognition competitions, research, and the literature.

The CNN is one of the most popular deep feed-forward network architectures used today, with applications ranging from medical analysis to signal processing and image classification. This chapter first describes the three types of artificial neural network architectures considered for this work. Then the use of raw IQ data as input and the selection of CNNs for this work is motivated, and some of the infrastructure used and designed to facilitate their use is described.

3.1 Neural Network Architectures

The basic feed-forward neural network, known as the Multi-Layer Perceptron (MLP), is shown in Figure 3.1 [33]. MLPs are fully-connected such that every neuron in one layer is connected to every neuron in the next layer. Using an algorithm called stochastic gradient descent, MLPs are trained to appropriately weight each connection and bias each neuron so that the input produces a desired output [33].

Traditionally, MLPs have been used to solve a wide variety of problems, including modulation classification [34]. As universal approximators, MLPs can approximate any continuous function with only a single hidden layer, and are capable of solving many complex problems [35]. However, as these problems increase in complexity, the MLP needed to solve a given problem can become prohibitively large [36–38].

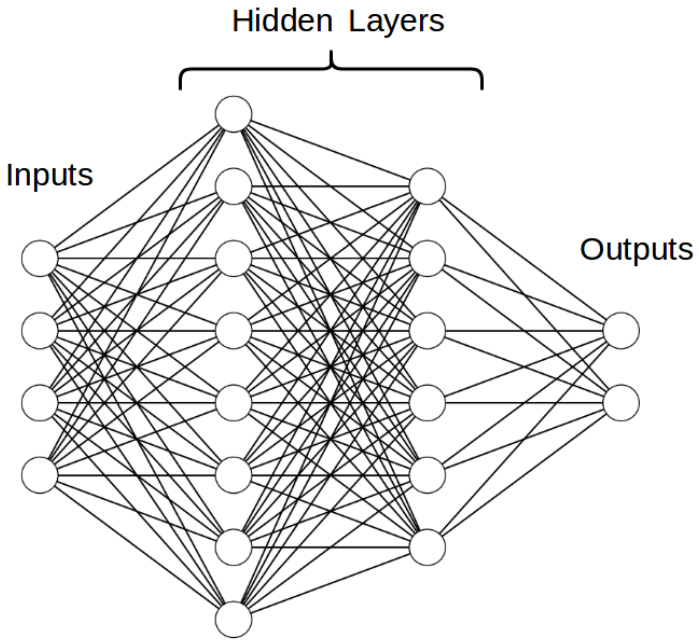


Figure 3.1: An example MLP.

Newer architectures such as CNNs and Recurrent Neural Networks (RNNs) have been shown to be more robust, scale better, and utilize computational resources better than MLPs [33, 38]. An example CNN is shown in Figure 3.2. CNNs perform mathematical convolutions over localized regions of the data as opposed to fully-connecting all nodes [39]. The result of convolving a *filter* over some input vector or matrix, shown in Figure 3.3, is known as a *feature map*, as the filter has extracted some piece of information from the input. Typically, the set of feature maps learned by the early layers of the CNN architecture are then passed to a set of dense fully-connected layers, much like an MLP, that performs the decision making on the features learned in the earlier layers.

In training, CNNs learn the set of filters that will convolve over the data, as well as the

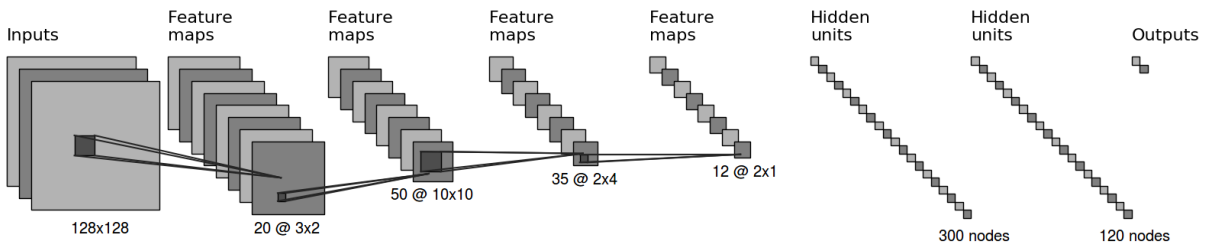


Figure 3.2: An example CNN.

weights and biases of the following dense layers. The benefits of learning a set of shared filters is three-fold:

1. It allows the network to work with inputs of varying size [38].

2. It increases memory and computational efficiency, as these filters contain far fewer weights and biases than a fully-connected MLP [38]. These increases in efficiency vary according to network size. However, as an example, consider performing edge detection on an $n \times n$ image:

   - Using a convolutional operation, edge detection can be performed using a 2×1 filter (2 matrix entries), and $(n-1) \times n \times 3$ floating point operations; $\mathcal{O}(n^2)$.
   - The equivalent matrix operation requires $(n \times n) \times ((n-1) \times n)$ matrix entries, and $2 \times (n \times n) \times ((n-1) \times n)$ floating point operations; $\mathcal{O}(n^4)$.

   Therefore, for simple edge detection, the convolutional operation is $\frac{n^3 \times (n-1)}{2}$ times more memory efficient and $\frac{2n^2}{3}$ times more computationally efficient.

3. It encourages the learning of relevant features within the data [33]. As such, pertinent

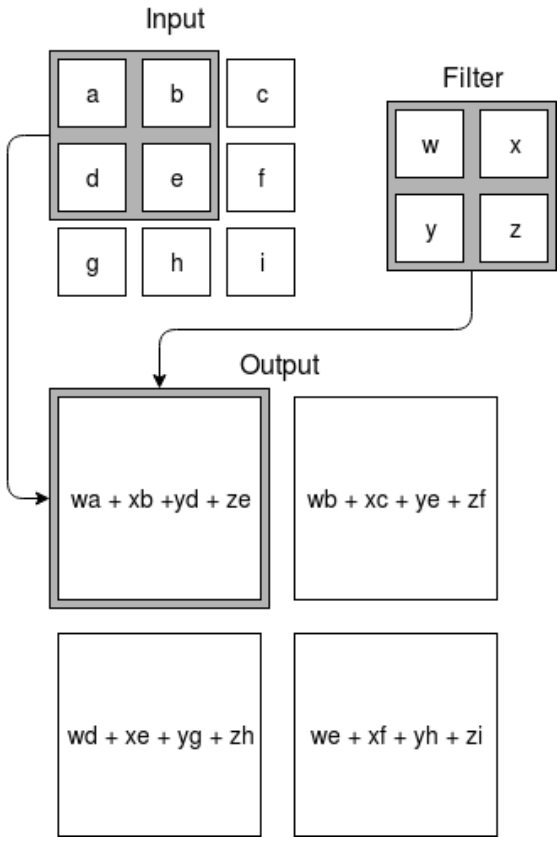


Figure 3.3: The CNN convolution operation.

features do not need to be determined *a priori*. This is the key discriminator of CNNs amongst neural network architectures.

RNNs differ from MLPs and CNNs in that they allow for cyclic connections between neurons, and therefore are not feed-forward networks [33,38]. This allows RNNs to effectively model sequences and temporal behavior by allowing the network to retain a notion of state or memory [40].

There are many different types of RNNs used in the literature, such as Recursive Neural Networks, Bi-directional RNNs, Long Short-Term Memory Networks (LSTMs), and Gated

Recurrent Units (GRUs) [38]. Training RNNs requires modification of the traditional Back-propagation training algorithm to properly address the looping in recurrent networks. In deep RNNs, this causes instabilities in the gradients that update the weights and biases during training, which can make training RNNs extremely difficult [41].

3.2 The Selection and Use of CNNs in this Work

While neural networks have been used for wireless communications applications, most of this prior work has used expert features as input [10, 11, 26, 42, 43]. As discussed in the previous chapter, by using pre-defined features, only a few of a received signal's attributes are considered, leaving behind information in the raw data that may be useful in identifying an emitter. In this work, the raw IQ data is used as input to the network, in order to allow the network access to all content within the received signal [44, 45], and to eliminate the need for pre-processing steps such as synchronization, carrier tracking, demodulation, or SNR estimation typically needed in prior works.

Though MLPs have been used for wireless communications applications with expert features as input, they lack the scalability needed to process raw IQ data efficiently. On the other hand, CNNs are able to learn to model features of raw data, and to do so efficiently. Though the use of raw IQ as input to CNNs is a relatively new concept, it has shown success in prior works [9, 46, 47], indicating CNNs can learn directly from raw signal data. Additionally, the recent success of CNNs in the field of wireless communications, using both expert features

and raw IQ data, further shows CNNs are capable of modeling communications data to perform tasks including emitter fingerprinting. Because CNNs have inherent feature learning abilities and due to their successes in the wireless communications domain, they were selected for this work.

RNNs would likely model raw IQ data well also, as the recurrent connections would allow RNNs to model the sequential nature of raw IQ data streams. However, the wide variety of architecture designs and training methods used in the literature combined with their volatility in training makes it difficult to create robust decision engines. As a result, RNNs were not considered in this work. Despite this, recent work has shown success using deep LSTM networks for anomaly detection with raw bio-signal data as input and using a hybrid CNN-LSTM model for time series classification with raw audio data as input, suggesting RNNs may be an appropriate direction for future work [48, 49].

3.2.1 Applications of CNNs in the Literature

CNNs are most commonly used in the image processing domain for tasks such as image classification, object detection, and filtering [50–52]. Additionally, because images are easily visualized, novel tools and techniques have been developed for visualizing the filters and filter activations of the trained network [53] [54], allowing for an increasing understanding of the features extracted by CNNs when trained on images.

However, CNNs have also shown great success when applied to time-series data such as in the natural language processing (NLP) and audio realms [55, 56]. More recently, CNNs

have been applied in the wireless communications domain, and have shown success performing modulation and waveform classification [9–11], emitter fingerprinting [26], interference identification [42], and device localization [43].

Much of the prior work using CNNs for wireless communications applications frame the problem as a classification problem, using the modulation type, emitter, interference type, or device location as the output class. As previously mentioned, features are often used as input to the network [10,11,26,42,43]. Further, because CNNs are so commonly used in the image processing domain, some prior works have used different image representations of the signal data as input to the network, such as the constellation diagram [10], the Choi-Williams distribution [11], or visualizations of angle of arrival [43]. However, in [9], raw IQ data was used as input to a CNN to successfully perform modulation classification, and in [42], the Fast Fourier Transform (FFT) of the raw IQ, a lossless transform, was used as input to a CNN to perform interference identification.

3.2.2 Network Design and Training

There are numerous open-source software libraries and toolboxes available that provide support for neural network design and training, including Caffe [57], Torch [58], TensorFlow [59], Theano [60], and Keras [61]. In addition to allowing for development in high-level languages such as Python, C++, and MATLAB, these libraries also support the optimal use of multi-core CPU systems and GPUs.

All networks used in this work were designed and trained in Python using Keras, a neural

networks applications programming interface (API) running on top of Tensorflow or Theano [59–61]. Keras provides an additional level of abstraction, with common layer types, cost functions, optimizers, activation functions, and regularizers provided as standalone modules that can easily be connected, modified, and extended. This allowed for the rapid design and testing of networks, due to the ease of architecture modification and the ability to parameterize much of the network.

The increased level of abstraction provided by Keras also allowed for the automation of the parameter tuning required when designing neural networks. Table 3.1 shows a portion of the large number of parameters that need to be tuned and optimized to find the best performing network for a problem. In order to efficiently determine the best network parameters for the architecture designed, a script was developed to select parameters using a method akin to the Monte-Carlo method.

The network architectures developed for this thesis, that is the number of layers and activation functions, were chosen by hand, influenced by networks used in the literature. Then, given a designed network architecture, the developed script randomly selected network parameters, built and trained networks with the given parameters, and evaluated the performance of the networks. The parameters producing the network with the best performance was then selected, as shown in Figure 3.4. Network performance was evaluated according to the purpose of the network being designed. For example, networks being designed to perform a classification task will use classification accuracy as the evaluation metric, while networks being designed for an estimation task may use the root mean squared error or the

Category	**Parameter**	**Description**
Convolutional Layers	# of filters	The number of filters per convolutional layer in the network, an integer value.
	filter size	The filter size, per layer. Can be 1-dimensional, 2-dimensional, or 3-dimensional.
Dense Layers	# of nodes	The number of nodes or neurons in each dense fully-connected layer.
All Layers	activation function	The activation function used in each layer (ex. ReLU, sigmoid, tanh, linear).
	pooling	Whether or not pooling is used after each layer, and if so, the size of the pool and pooling method (ex. Max Pooling of size = 2).
	dropout	Whether dropout is used after each layer, and if so, by how much (ex. dropout = 0.5).
Network Parameters	# of Convolutional layers	The number of convolutional layers in the network, an integer value.
	# of Dense layers	The number of dense layers in the network, an integer value.
	batch size	The number of training samples the network is given at a time, an integer value.
	loss function	The loss function, also known as the objective function, used to compile the network which modifies the back-propagation algorithm (ex. mean squared error, categorical crossentropy).
	optimizer	The optimizer used to compile the network which modifies the back-propagation algorithm (ex. Stochastic Gradient Descent, RMSprop, Adadelta).

Table 3.1: The tunable parameters of a CNN.

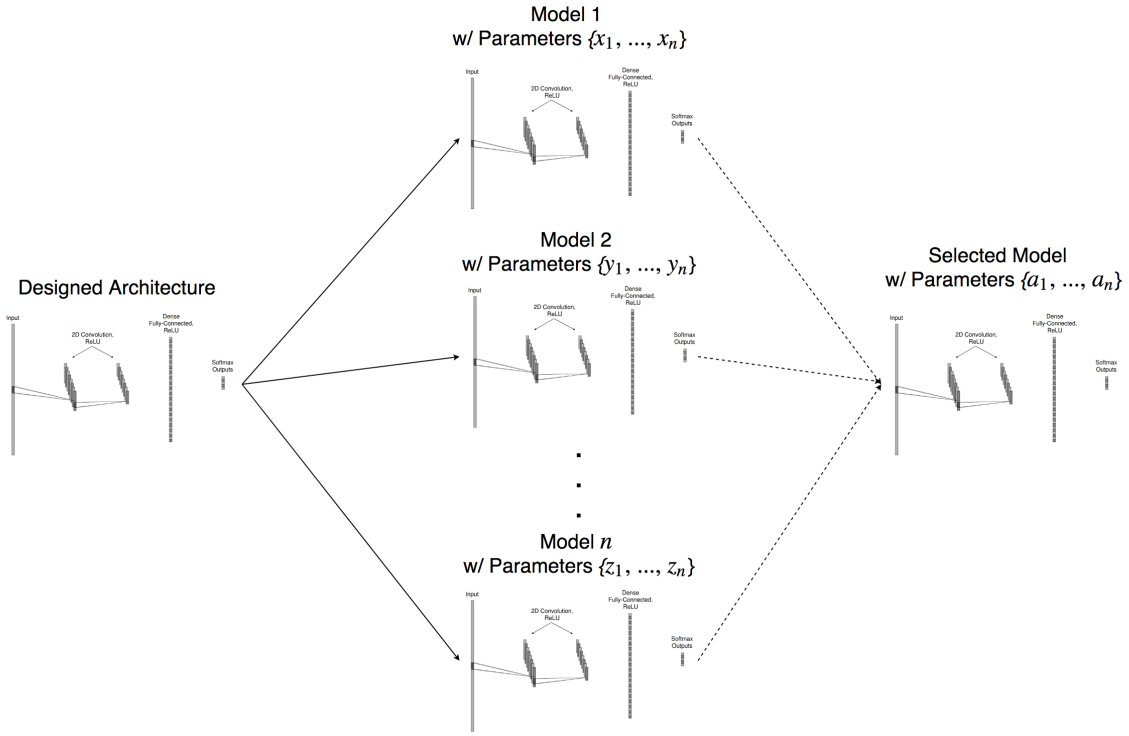


Figure 3.4: The CNN parameter selection process.

normalized mean squared error. Due to the variety of approaches investigated in this work, numerous metrics were used to evaluate the different networks developed. Each metric used will be described in detail in the following chapters.

While CNNs have the potential for unsupervised and semi-supervised learning applications, where inferences are drawn from unlabeled or partially labeled datasets, because the CNNs used in this work perform estimation and identification tasks, supervised learning approaches provided simpler and more appropriate solutions than unsupervised learning approaches. As such, the approaches developed in this work used supervised learning methods, meaning each set of training and testing samples is labeled with their true offset values [47, 62].

In order to prevent the networks from overfitting, or learning the training data too well, as this would keep the networks from performing well on unseen data, a validation split on the training data was used to monitor the accuracy of the networks as they trained. When the performance of the network on the validation split stopped improving, training was stopped and the network evaluated appropriately on a separate validation set, a method called early stopping [33].

3.2.3 Training Data

When training neural network models, the quantity and quality of training data is of the utmost importance. What constitutes "enough" training data varies widely between neural network types and applications [63]. However, in general, more training data improves network performance, at the cost of either having to gather more real-world data or to generate more simulated data [33, 38].

In the case that gathering or simulating additional data is not feasible, learning may be improved by either improving the learning algorithm itself, using methods such as regularization, tuning network parameters, altering the network architecture, or by improving the quality of the training data [38]. When training a neural network, the quality of training data refers to the similarity of the training data to the test data or to data the network may see once deployed, as the network cannot be expected to perform well on something it has never seen before in training. Though online learning algorithms may allow the network to continue learning once deployed, they can be difficult to assess and are subject to forgetting

previously learned information, a phenomenon known as *catastrophic forgetting* [7, 64].

Many prior works utilizing CNNs for wireless communications applications have used real data in training [26, 42, 43]. However, tools such as GNU Radio have allowed for the generation of simulated data containing real-world effects such as channel effects and hardware impairments [65]. As such, GNU Radio has become popular for generating simulated data, but MATLAB is also used [9, 11].

This work uses both simulated and real RF data in training. In addition to channel effects, RF fingerprints are captured in the real data. Real RF training data is essential to performing tasks such as emitter identification, as the features which make an emitter unique are not always entirely known and therefore cannot be easily simulated. However, the process by which real RF data is collected can be tedious and time-consuming.

In comparison, simulated data is far easier to generate. However, channel effects, transmitter impairments, and the effects of an imperfect signal detection stage needed to be considered during dataset generation, in order to simulate the effects inherent in the real data [46]. By training on data generated at a variety of SNRs, frequency offsets, and sampling rates, the network is encouraged to generalize over different noise levels as well as different center frequencies and bandwidths that may be caused by an imperfect signal detection stage. However, it should be understood that overall performance does decrease when the network needs to generalize over these parameters, as was also shown in [46]. All simulated data used in this work was generated using the open-source *gr-signal_exciter* module in GNU Radio [66].

3.3 Summary

This chapter has provided background on the types of artificial neural network architectures considered for this work, and discussed some of the practical considerations when training neural networks. Further, this chapter has addressed the key design choices that are foundational to this work: the use of raw IQ as input to the network and the selection of the CNN architecture.

More specifically, using raw IQ as the input to the network allows the network access to all the content within the signal and eliminates pre-processing steps typically needed to extract expert features. CNNs are designed to learn features most important to their task, have shown success in the field of wireless communications, and have been used with raw IQ data in the recent literature. As such, this work seeks to further understand the abilities of CNNs as feature learners, when applied to the raw data, for the purpose of SEI.

Chapter 4

Emitter Identification Using CNN IQ Imbalance Estimators

As discussed in Chapter 3, prior work using CNNs for wireless communications applications often uses expert features as input. While these expert features can be extracted using traditional methods, these methods often make a variety of assumptions and/or require various operating conditions that may not always be satisfied, as will be discussed in Section 4.2. In this chapter, an approach using CNNs to extract an expert feature, transmitter IQ imbalance, is developed and analyzed. Further, using the developed CNN IQ imbalance estimators, an approach is presented to identify emitters across numerous modulation schemes.

To this end, in Section 4.1, transmitter IQ imbalance is discussed and an appropriate signal model is developed, for use in the generation of the simulated data used for training and testing of the approach. Section 4.3.1 then describes the models designed for the estimation of IQ imbalance. Using the simulated data described in Section 4.3.2, the impact

of network input size, SNR, and imbalance value on the performance of the estimators is thoroughly evaluated in Section 4.3.3, for both QAM and PSK modulation schemes. Section 4.4 presents the SEI approach using the developed CNN IQ imbalance estimators and evaluates the performance of the developed approach, showing that the accuracy of developed approach exceeds that of a traditional feature-based approach using less data and making fewer assumptions. Finally, Section 4.5 summarizes the work presented in this chapter and describes future work that may improve the approach.

4.1 Transmitter IQ Imbalance

4.1.1 Causes and Implications

Transmitter-induced frequency-independent IQ imbalance is caused by non-idealities in the local oscillators and mixers of the transmitter which cause the in-phase and quadrature components of the modulator to be non-orthogonal. The result is the real and imaginary components of the complex signal interfering with each other. In addition to potentially degrading the performance of the transmitter, IQ imbalance can also be used as an identifying feature when performing SEI techniques.

IQ imbalance in the constellation diagram, shown with exaggerated imbalance values and after demodulation for clarity, is shown for 16QAM in Figure 4.1. The result of a phase imbalance on a signal, shown in the lower left constellation, is a rotation of the real component of the symbols in the IQ plane. The result of a gain imbalance on a signal, shown in the

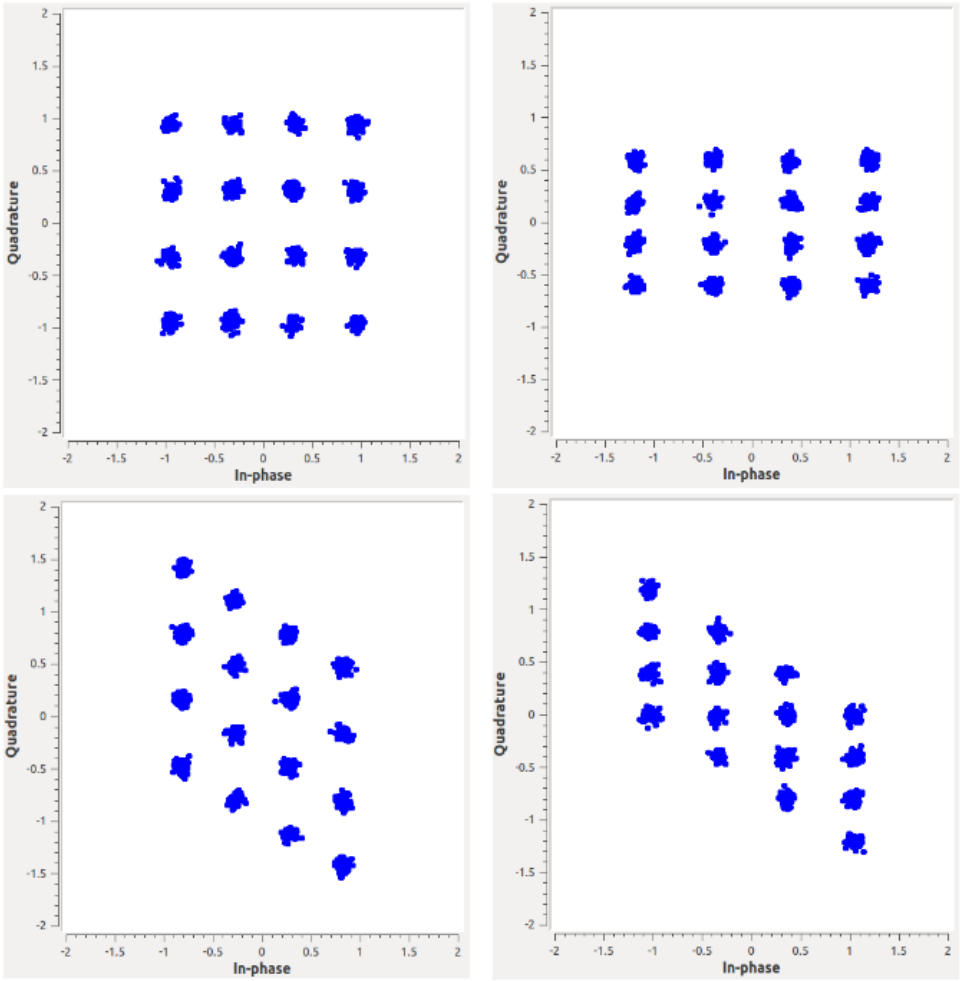


Figure 4.1: The result of transmitter IQ imbalance applied to the in-phase component of a 16QAM signal in the constellation diagram, SNR = 20dB. Top Left: no imbalances. Top Right: phase imbalance = 30°, gain imbalance = 0. Bottom Left: phase imbalance = 0, linear gain imbalance = 0.9. Bottom Right: phase imbalance = 30°, linear gain imbalance = 0.9.

upper right constellation, is a stretching or contracting of the real component of symbols along the in-phase axis. However, in many systems, it may be impractical to obtain the symbols, such as in a blind system where synchronization cannot be assumed. Given this, the proposed approach uses raw IQ as input, eliminating the need for demodulation, used in many traditional methods [67].

IQ imbalance in the time domain is shown for 16QAM in Figure 4.2. The result of a gain imbalance on a signal in the time domain is an increase or decrease in the amplitude of the real component of the signal. The result of a phase imbalance on a signal in the time domain is a shifting of the phasor of the real component of the signal. To the human eye, a phase imbalance is much harder to see than a gain imbalance, though both become hard to detect at low SNR. However, as will be shown in Section 4.3.3, using the learned features, CNNs are able to identify small differences between sets of samples to estimate these imbalances, given enough samples and reasonable SNR values.

4.1.2 Signal Model

This work assumes only frequency-independent IQ imbalance. Though most modern communications systems are affected by frequency-dependent IQ imbalance, frequency-independence is often assumed in the existing literature, for simplicity [68]. Frequency-independent IQ imbalance is also a valid approximation for imbalanced narrowband systems and imbalance due to the analog components of emitters [68, 69]. Without loss of generality, all imbalances are modeled on the in-phase component of the modulated signal before transmission through an

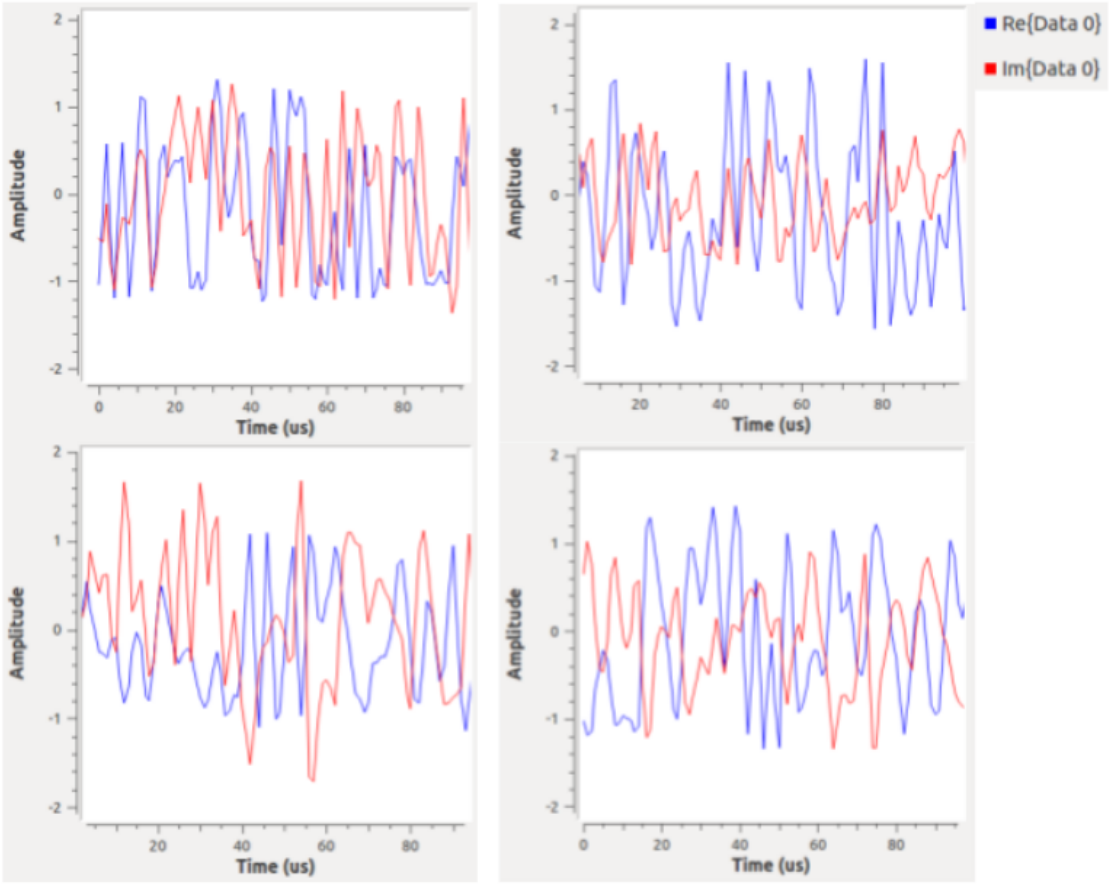


Figure 4.2: The result of transmitter IQ imbalance applied to the in-phase component of a 16QAM signal in the time domain, SNR = 20dB. Top Left: no imbalances. Top Right: phase imbalance = 30°, gain imbalance = 0. Bottom Left: phase imbalance = 0, linear gain imbalance = 0.9. Bottom Right: phase imbalance = 30°, linear gain imbalance = 0.9.

AWGN channel [70], as follows:

Consider the baseband signal

$$x(t) = x_i(t) + jx_q(t), \tag{4.1}$$

where $x_i(t)$ and $x_q(t)$ are real-valued time-varying baseband signals. An IQ modulator with imbalance, as shown in Figure 4.3, modulates this baseband signal to its bandpass equivalent through

$$x(t) = (1+\alpha)\cos(2\pi f_0 t + \theta)x_i(t) - j\sin(2\pi f_0 t)x_q(t), \tag{4.2}$$

where f_0 is the carrier frequency, the transmitter's gain imbalance is represented by α, and the transmitter's phase imbalance is represented by θ, such that for an ideal transmitter, with no IQ imbalance, $\alpha = 0$ and $\theta = 0$. Transmission through an AWGN channel gives the received signal

$$y(t) = \mathbb{R}\left\{ \sum_{k=-\infty}^{\infty} (1+\alpha)\cos(2\pi f_0 t + \theta)x_{k_i}(t) - j\sin(2\pi f_0 t)x_{k_q}(t) \right\} + n(t) \tag{4.3}$$

where $n(t)$ is a zero mean white Gaussian noise process [44, 71].

Though gain and phase imbalance values for real systems are not easily found, prior works in IQ imbalance estimation and compensation use test values ranging from 0.02 to 0.82 for absolute gain imbalance and from 2° to 11.42° for phase imbalance, with most works using test values on the orders of 0.05 and 5° for gain and phase imbalance respectively [67, 70–76].

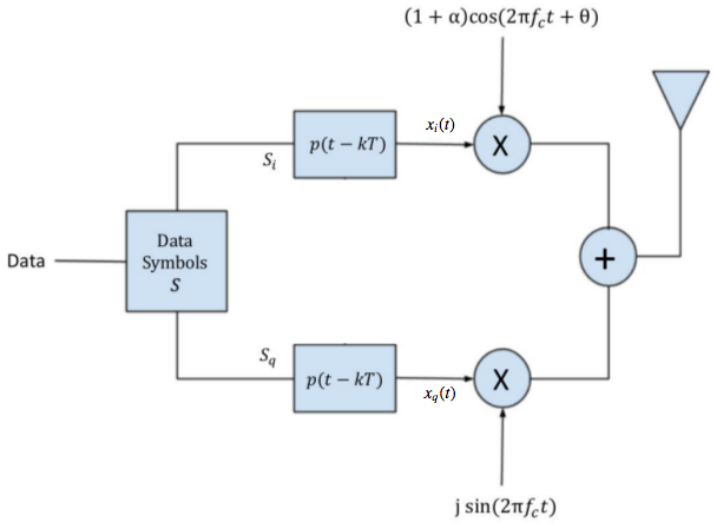


Figure 4.3: IQ modulation with IQ imbalance on the in-phase component.

4.2 Traditional IQ Imbalance Estimation Approaches

Many methods for estimating or compensating for IQ imbalance exist in the literature [67, 68, 72–78]. For example, in [67], a clustering method was developed to match the received symbols to their ideal positions in the I/Q plane, a non-linear regression technique was used to estimate the gain and phase imbalance values using a training sequence in [77], and the methods developed in [73] and [76] use second-order statistics to estimate the terms used to compensate for IQ imbalance at the transmitter and receiver.

However, many of these approaches rely on various assumptions. For example, the approaches developed in [72, 77] assume the use of a training sequence or test signal for calibration. Additionally, many of these approaches require the use of successive iterations [73], demodulation [67], adjacent power measurements [74], or the use of statistical measures such as cross-correlation or expectation [73, 76]. The approach developed in this chapter

makes no such assumptions and uses only raw IQ as input to the CNN, eliminating the need for typically assumed pre-processing steps such as synchronization, carrier tracking, feature extraction, or SNR estimation.

Additionally, an abundance of IQ imbalance estimation or compensation techniques have been developed specifically for systems utilizing an OFDM modulation scheme, and are not applicable to modulation schemes that do not utilize multiple carriers [68]. Likewise, some IQ imbalance compensation approaches require modification of the transmitter hardware, such as the addition of diodes for IQ imbalance measurements to be used in a feedback loop [75, 78]. The approach developed in this chapter is modulation agnostic and requires no hardware modifications.

Furthermore, the majority of current work in the area of estimating IQ imbalance focuses on IQ imbalance compensation [67,68,72–78]. While IQ imbalance compensation is certainly an important application of this work, this assumes cooperation between the transmitter and receiver. This work considers a non-cooperative scenario. In such a scenario, little to no assumptions can be made about what is being received, eliminating the ability to use many existing estimators. For example, training sequences or test signals can not be used because such sequences or signals will not be known *a priori*.

4.3 CNN IQ Imbalance Estimators

4.3.1 Model Design, Training, and Evaluation

The network architecture designed for the approach is shown in Figure 4.4, and is loosely based off of the network architecture used in [9]. To investigate the trade-offs between input size and performance, models were trained and evaluated using input sizes of 512, 1024, and 2048 raw IQ samples. Following the input layer, the network is composed of two two-dimensional convolutional layers and four dense fully-connected layers. Intuitively, the convolutional layers in this architecture are designed to identify and extract the relevant features, and the fully connected layers that follow are intended to perform the estimation [79].

All layers, excluding the output layer, utilize a Rectified Linear Unit (ReLU) activation function, shown in Figure 4.5. The ReLU function is a popular activation function in the literature, as it has been shown to be robust to saturation (when output is near zero or one) which usually causes learning to slow. However, because the function has a range of $[0, \infty)$ it cannot be used at the output layer, as it cannot produce negative estimates. Therefore, the final layer of the network uses the linear activation function, shown in Figure 4.6, to allow the network to estimate negative gain and phase imbalance values. The stochastic gradient descent algorithm modified with a RMSProp optimizer and a mean squared error loss function was used to train the networks [80].

The work in this chapter uses simulated data in training and testing which allowed for

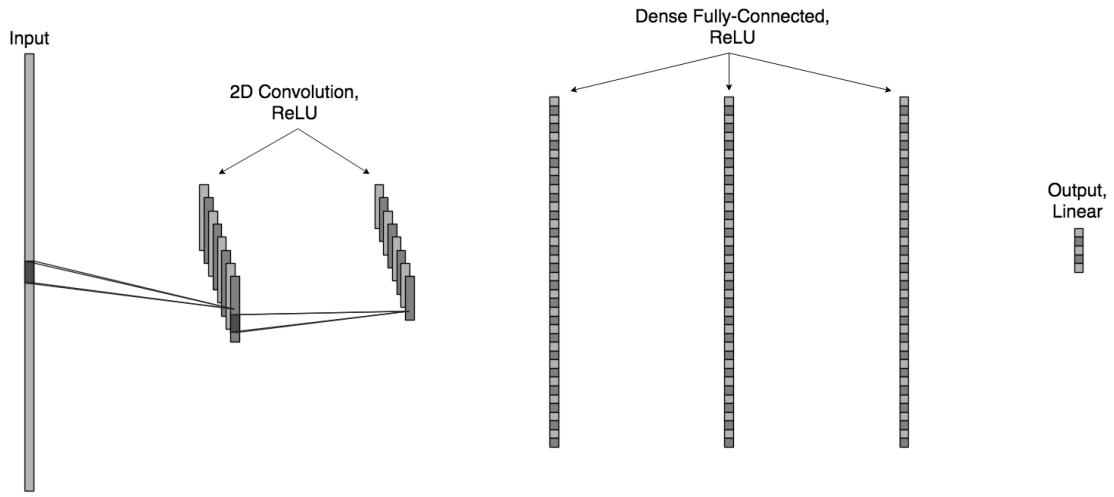


Figure 4.4: The CNN architecture designed for estimation of transmitter IQ imbalance.

control over the range of IQ imbalance values in the training set, and thus the range of IQ imbalance values the designed networks learn to estimate. Most prior works use test values on the orders of 0.05 and 5° for absolute gain and phase imbalance [67, 70–76]. However, test values in the literature ranged from 0.02 to 0.82 for absolute gain imbalance and from 2° to 11.42° for phase imbalance [67, 70–76]. Therefore, the training, validation, and test sets for this work were simulated with gain imbalances of [-0.9, 0.9], uniformly distributed, and phase imbalances between $[-10°, 10°]$, uniformly distributed, in order to train over a range of imbalance values incorporating anything the networks might see in a real system.

Due to the complexity of estimating IQ imbalance, this approach estimates gain imbalance and phase imbalance separately using two different neural networks. Though both networks share the same underlying architecture, shown in Figure 4.4, using two networks allows each network to be optimized for the specific problem of gain imbalance estimation or phase imbalance estimation. The resulting networks therefore have different sized convolutional and

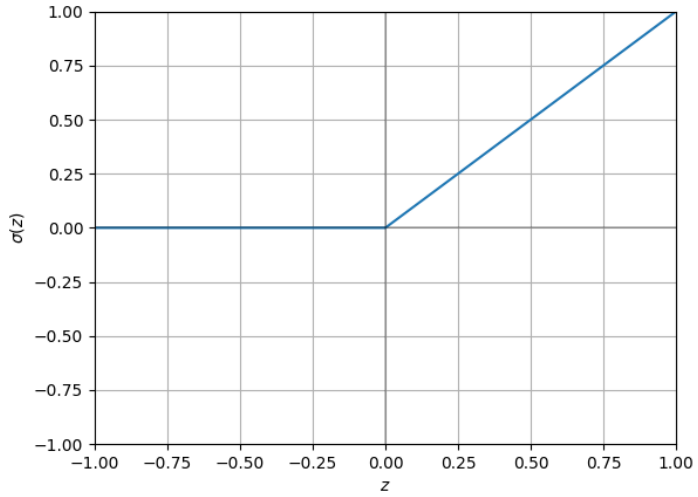


Figure 4.5: The ReLU activation function.

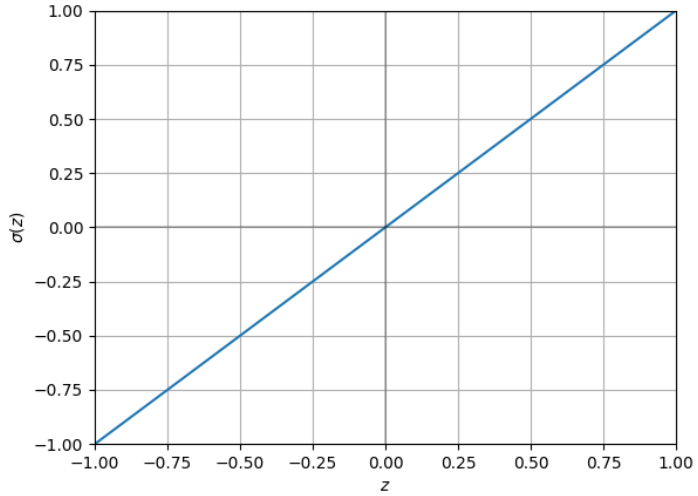


Figure 4.6: The linear activation function.

dense layers, as well as different weights and biases, as they have been trained separately using the scripts described in Section 3.2.2. However, it should be noted that these two networks are not dependent upon each other, and therefore can be trained and run in parallel.

Similarly, separate networks were trained to estimate IQ imbalance for the simulated QAM and PSK signals. However, results in Section 4.3.3 will show that the performance of these networks is comparable, indicating the designed network architecture is not modulation specific. Additionally, though the networks have been trained per modulation type, they are generalizing over modulation order (i.e. the networks trained to estimate IQ imbalance for QAM can estimate gain and phase imbalances for QAM signals of orders 8, 16, 32, and 64).

Each network used 2,020,000 sets of labeled samples in training: 2,000,000 sets of samples were used for training, 10,000 for validation, and 10,000 for testing. The normalized mean squared error (NMSE) was used as the performance metric to determine the best network design and to evaluate performance, and is defined as

$$NMSE = \frac{1}{N} \sum_{i} \frac{(P_i - M_i)^2}{\overline{P}\,\overline{M}}, \tag{4.4}$$

where P is the vector of estimated imbalance values, M is the vector of measured imbalance values, $\overline{P}$ is the mean of vector P, $\overline{M}$ is the mean of vector M, and N is the length of vectors P and M [81].

To further evaluate the performance of the estimators, evaluation sets were constructed with 180,000 sets of samples for the gain estimator and 200,000 sets of samples for the phase estimator. For each evaluation set, 1,000 sets of samples were generated at evenly spaced

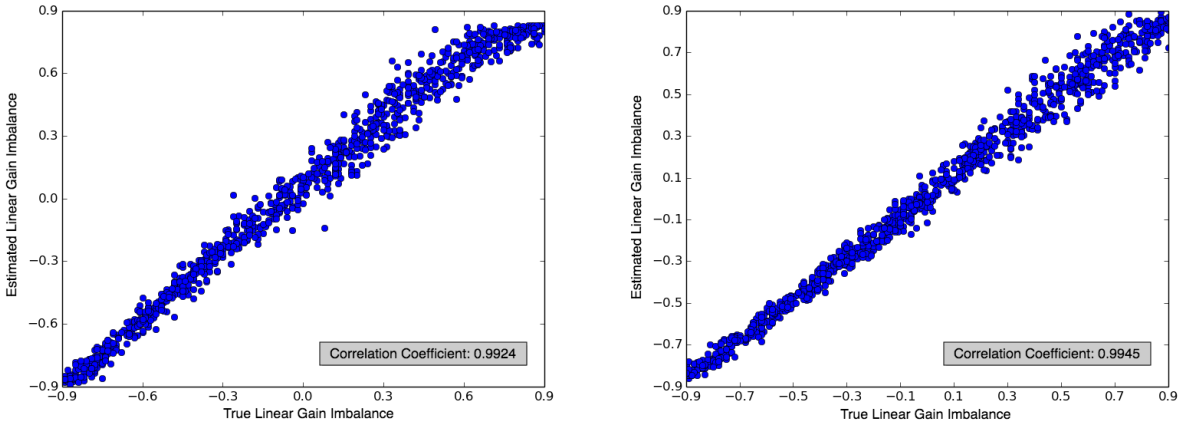
intervals of $\Delta\alpha = \pm 0.01$ for gain imbalance and evenly spaced intervals of $\Delta\theta = \pm 0.1^\circ$ for phase imbalance within the training range. These evaluation sets were used to determine the bias of the estimators and to generate the histograms shown in Section 4.4.1.1.

4.3.2 Dataset Generation

All data used in the following simulations was generated using the open-source *gr-signal_exciter* module in GNU Radio [66]. QAM signals of orders 8, 16, 32, and 64 and PSK signals of orders 2, 4, 8, and 16 were simulated with linear gain imbalances between [-0.9, 0.9], uniformly distributed, and phase imbalances between $[-10^\circ, 10^\circ]$, uniformly distributed. Additionally, frequency imbalances between [-0.1, 0.1] times the sample rate, uniformly distributed, were simulated and the simulated signal was sampled between [1.2, 4] times Nyquist, uniformly distributed, in order to simulate the effects of an imperfect signal detection stage. The sampled signal was passed through a root-raised cosine filter with a roll-off factor of 0.35 and normalized so that the average symbol power is 1dB. Finally, white Gaussian noise was added such that all signals had SNRs between [0dB, 25dB], uniformly distributed.

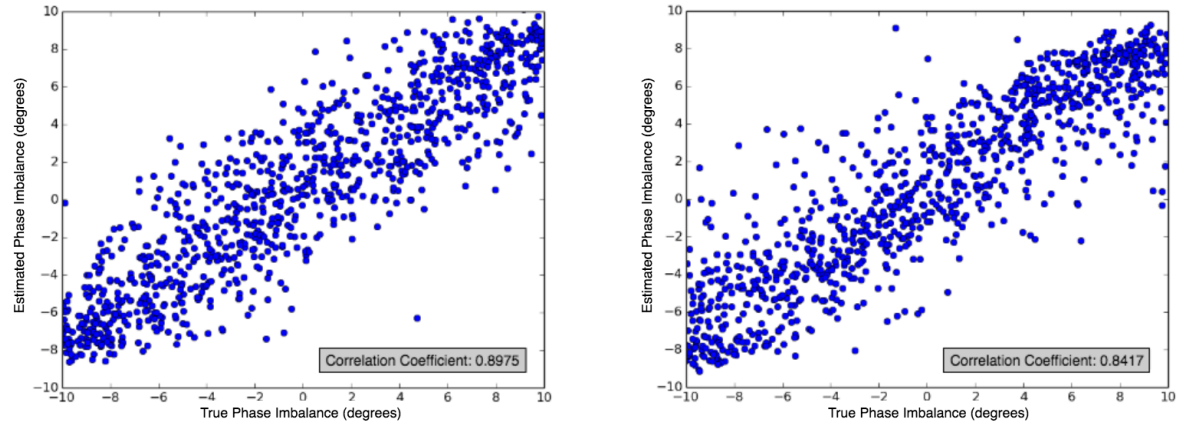
4.3.3 Simulation Results and Discussion

Initial results can be seen in Figures 4.7 and 4.8. The extremely strong linear correlations in Figure 4.7 shows each network's ability to estimate gain imbalance, for all imbalances in the training range, using 1024 input samples. The phase imbalance estimators similarly show linear correlations in Figure 4.8, though with correlation coefficient values 0.1-0.15



(a) QAM (b) PSK

Figure 4.7: The true linear gain imbalance value versus the linear gain imbalance value estimated by the 1024-input CNN gain imbalance estimators with input signals at 10dB SNR.



(a) QAM (b) PSK

Figure 4.8: The true phase imbalance value versus the phase imbalance value estimated by the 1024-input CNN phase imbalance estimators with input signals at 10dB SNR.

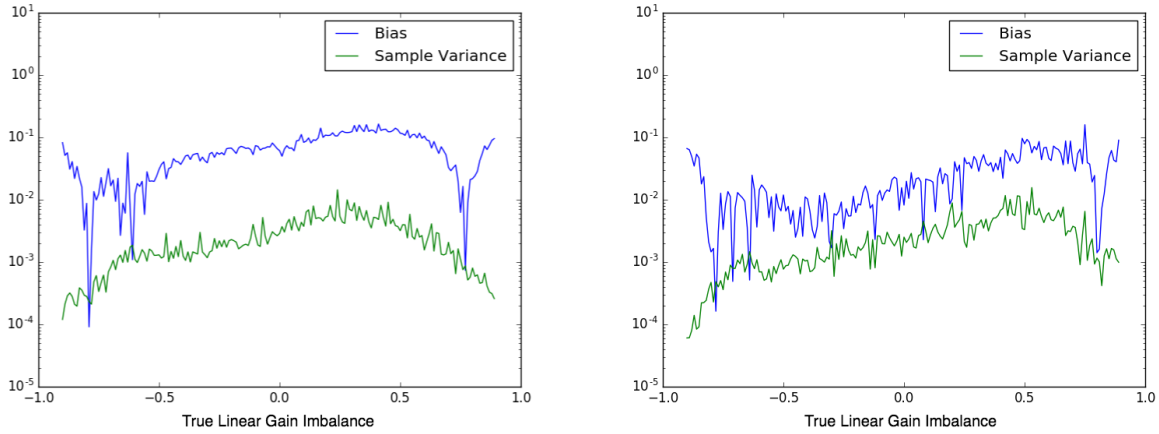
lower than the gain imbalance estimators. This indicates phase imbalance is more difficult to estimate than gain imbalance, using the designed network architecture with 1024 input samples, as a strong linear correlation indicates a clear relationship between the estimated and true imbalance value.

Bias of the Estimators

To examine the bias of the estimators, the cumulative average of the estimator outputs was taken for 1,000 sets of samples, each with the same imbalance value. The estimator can be called *unbiased* if the cumulative moving average converges to the true imbalance value, and is *biased* otherwise [82].

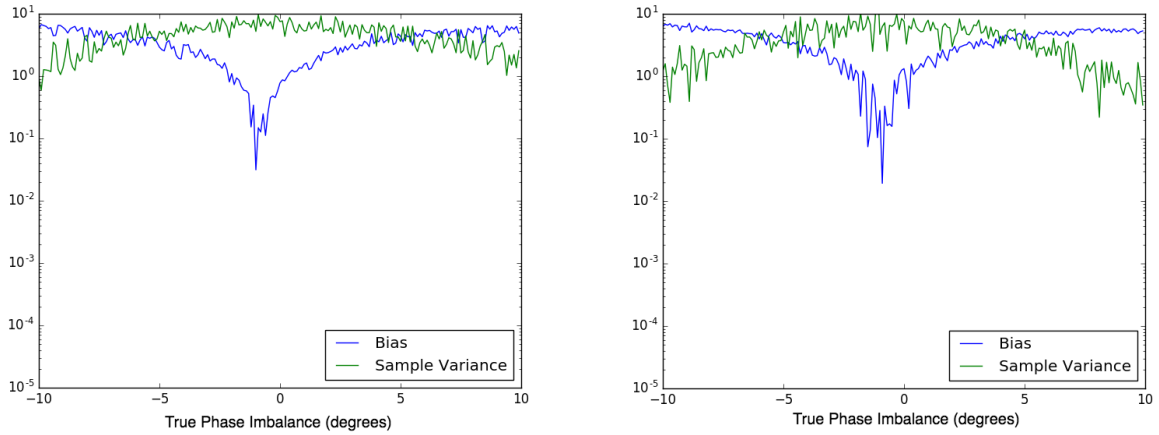
Figures 4.9 and 4.10 show the bias and sample variance of the gain imbalance estimators and phase imbalance estimators respectively, as a function of the true imbalance value. The gain imbalance estimators produce estimates with low bias for all values within the training range $(-0.9,\ 0.9)$, with slightly higher bias values at the positive imbalance values. Additionally, the sample variance is also very low across all values within the training range. However, both the bias and the sample variance of the gain imbalance estimators are negligible in comparison to the bias and variance of the phase imbalance estimators, further indicating phase imbalance is far more difficult to estimate at 10dB SNR using this network architecture.

The phase imbalance estimators produce estimates with lowest bias when the true imbalance value is near zero. The bias then increases as the true imbalance value gets farther from



(a) QAM (b) PSK

Figure 4.9: The bias and sample variance versus the true linear gain imbalance value for the 1024-input CNN gain imbalance estimator and signals simulated at 10dB SNR.



(a) QAM (b) PSK

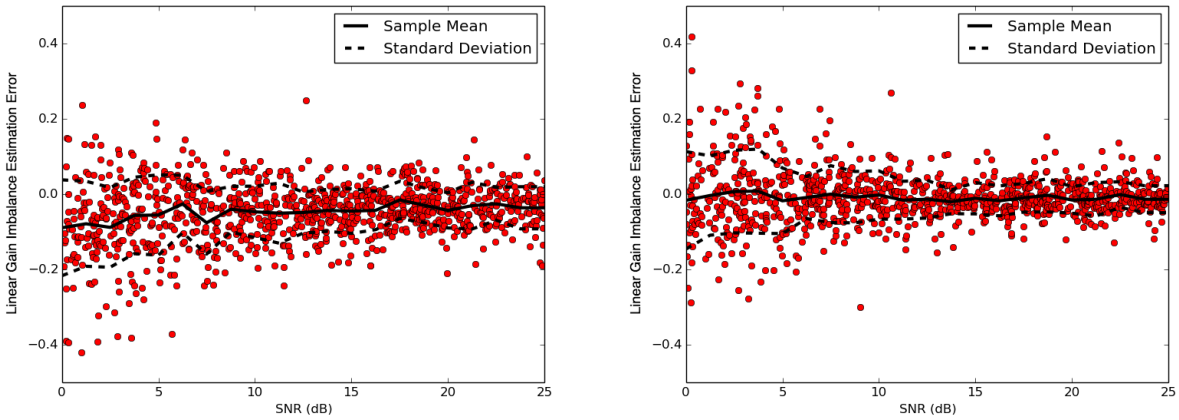
Figure 4.10: The bias and sample variance versus the true phase imbalance value for the 1024-input CNN phase imbalance estimator and input signals simulated at 10dB SNR.

zero in either direction. The sample variance shows an inverse trend, with maximum sample variance near zero and minimum sample variance at $-10°$ and $10°$. This indicates that small phase imbalance values are more difficult for the designed CNN to estimate. These trends further emphasize the inaccuracy of the phase imbalance estimators across all values.

Impact of SNR and Network Input Size on Performance

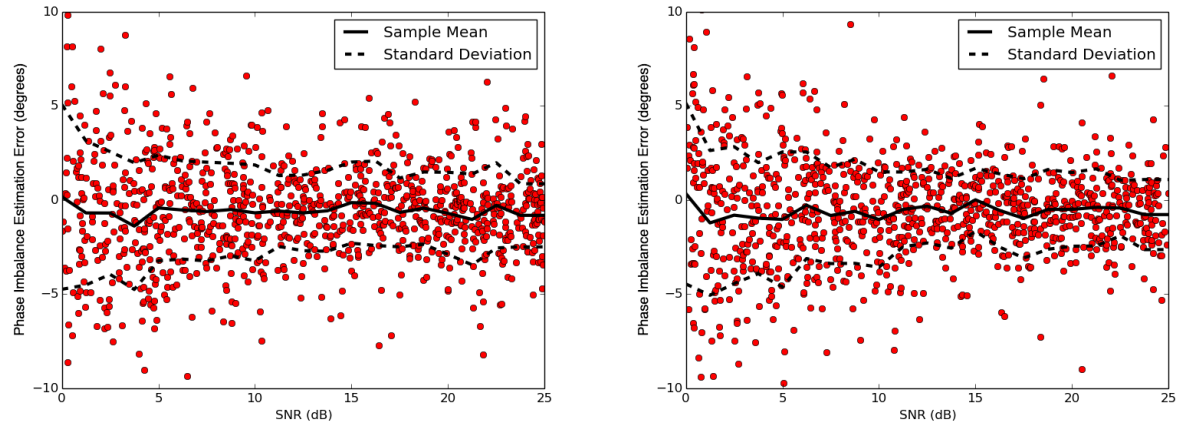
The effect of SNR on the performance of the estimators can be seen in Figures 4.11 and 4.12. For both imbalance estimators trained for both modulation types, it is shown that as the SNR increases, the imbalance estimation error (the difference between the true and estimated imbalances) decreases. However, the mean imbalance error stays almost constant near zero for all SNR values with the standard deviation of the imbalance error decreasing as SNR increases, with diminishing returns after 10dB.

The effect of the network input size and the SNR of the input signal on the performance of the gain and phase imbalance estimators was further investigated using the average bias and the sample variance of the output. These results are shown in Figures 4.13, 4.14, 4.15, and 4.16. Figures 4.13 and 4.14 show that as the SNR increases, the bias of the estimators decreases. Additionally, Figure 4.15 shows that for the gain imbalance estimators, as the SNR of the input signal increases, the sample variance also decreases, with dramatic improvement between $0-10$dB and diminishing returns after 15dB. Shown in Figure 4.16, the PSK phase imbalance estimators and the 1024- and 2048-input QAM phase imbalance estimators behave similarly. However, the sample variance of the 512-input QAM phase imbalance estimator



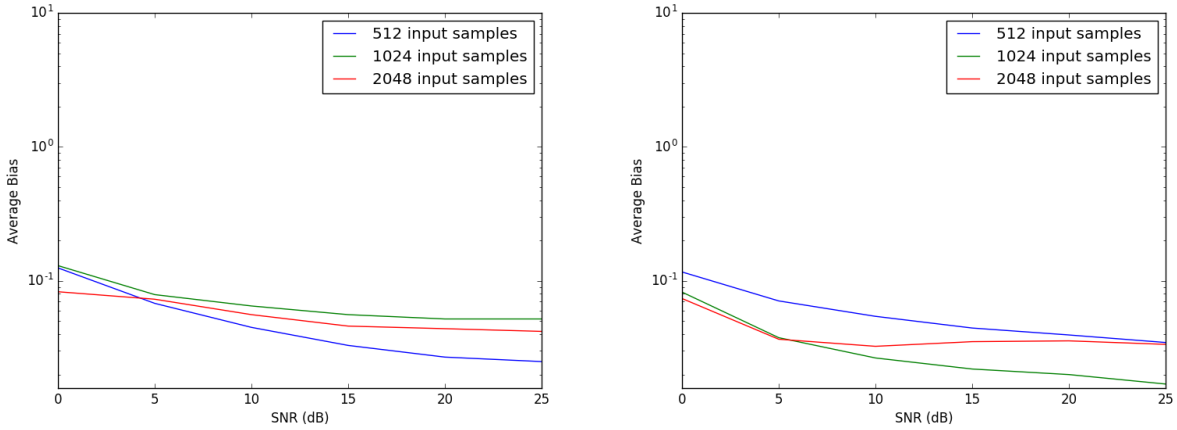
(a) QAM (b) PSK

Figure 4.11: The Linear Gain Imbalance Estimation Errors for signals simulated with SNRs between 0dB and 25dB. True linear gain imbalances vary uniformly between [-0.9, 0.9].



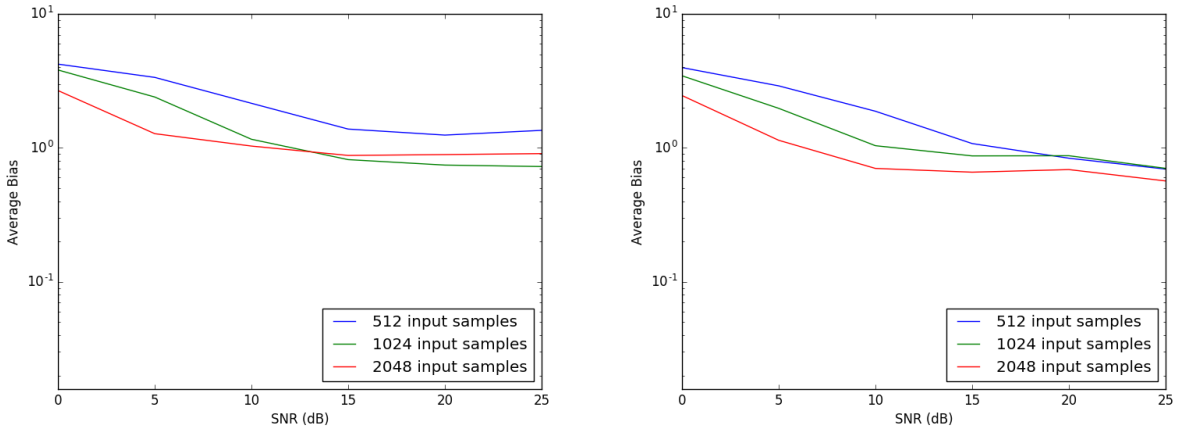
(a) QAM (b) PSK

Figure 4.12: The Phase Imbalance Estimation Errors for signals simulated with SNRs between 0dB and 25dB. True phase imbalances are uniformly distributed between [-10°, 10°].



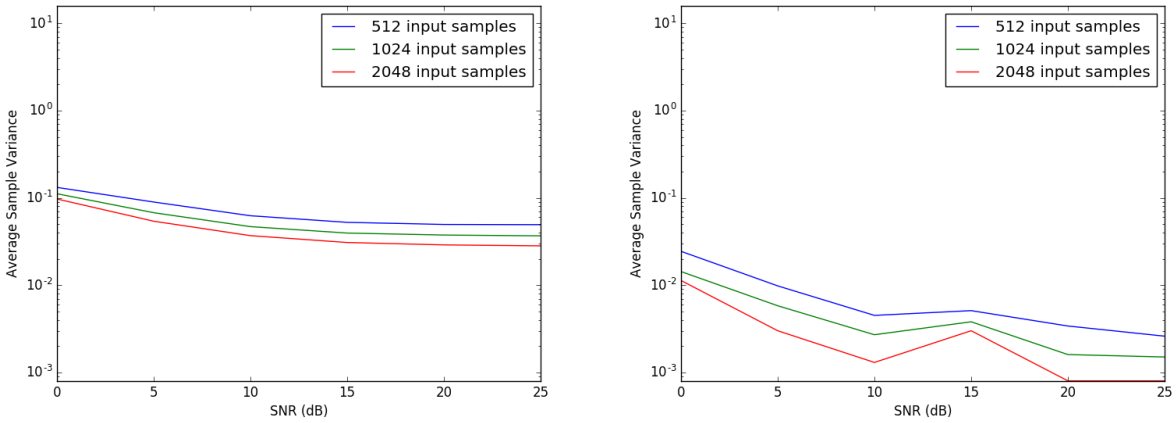
(a) QAM (b) PSK

Figure 4.13: The average bias versus SNR for CNN gain imbalance estimators with input sizes of 512 samples, 1024 samples, and 2048 samples.



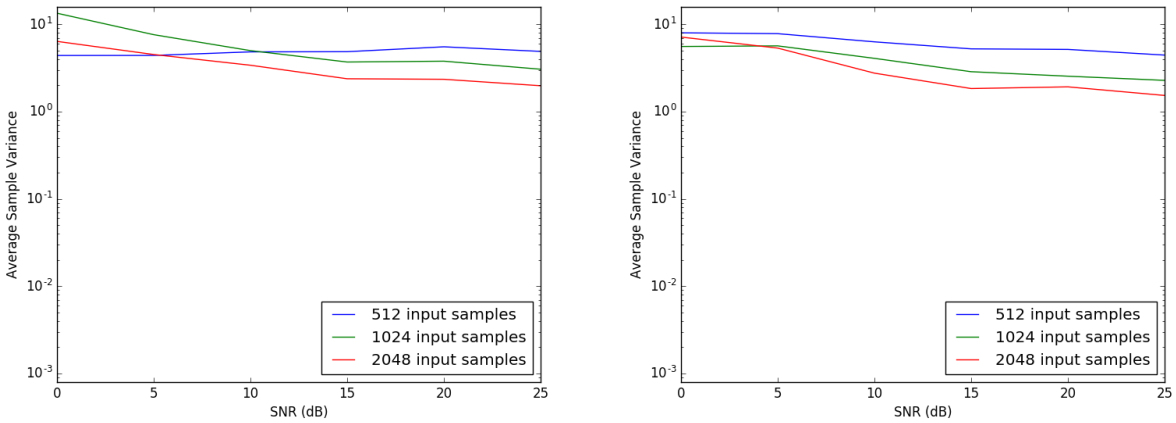
(a) QAM (b) PSK

Figure 4.14: The average bias versus SNR for CNN phase imbalance estimators with input sizes of 512 samples, 1024 samples, and 2048 samples.



(a) QAM (b) PSK

Figure 4.15: The sample variance of the histograms for the 512-input, 1024-input, and 2048-input CNN gain imbalance estimators as a function of SNR.



(a) QAM (b) PSK

Figure 4.16: The sample variance of the histograms for the 512-input, 1024-input, and 2048-input CNN phase imbalance estimators as a function of SNR.

histogram remains constant for all SNRs. This, in addition to the high average bias of the 512-input QAM phase imbalance estimator, suggests that 512-input samples does not give the network enough information to learn phase imbalance for the QAM modulation type. Therefore the network produces very similar outputs for all inputs at all SNRs.

Figures 4.15 and 4.16 also show, as the number of input samples increases, the sample variance decreases, excluding the 512-input QAM phase imbalance estimator. This behavior is expected, as with more input samples, the network observes the signal for longer, and therefore has more information about the signal to use in its estimation.

From the results shown above, it can be concluded that though increasing the number of input samples to the network may not increase the accuracy of the estimate, the network does become more sure of the estimate it produces. However, it should be noted that though using more inputs generally improves some aspects of performance, it also slows the network and increases training time, as it has to process more information. Additionally, increasing the input size to the network also requires more training data, as n sets of 1024 samples requires twice the memory as n sets of 512 samples.

4.4 Transmitter Gain Imbalance Estimation for SEI

This section presents an approach for performing SEI using a modulation classifier and the CNN gain imbalance estimators developed previously. The approach uses the CNN gain imbalance estimators, in order to limit the scope of the problem and because the gain

imbalance estimators far outperformed the phase imbalance estimators. Because an emitter's IQ imbalance parameters will not change as it changes modulation schemes, the proposed approach has the ability to track emitters, even as they change modulation scheme.

The performance of the developed approach is analyzed in terms of the probability of incorrect identification, considering the impact of SNR, gain imbalance value, and modulation scheme. Finally, the developed approach is compared to a traditional feature-based approach.

4.4.1 Approach

Three main steps are required to perform emitter identification using the proposed approach, shown in Figure 4.17: modulation classification, gain imbalance estimation, and decision making. The result is a decision tree-like structure in which the output of each step informs the next action, as described below.

The first step is modulation classification because the pre-trained CNN gain imbalance estimators are modulation-specific. It is important to note that while any modulation classifier may be used, a key advantage of the developed approach over traditional approaches is the use of only the raw IQ as input. In order to retain this advantage, the modulation classifier should only use raw IQ as input as well. Such modulation classifiers exist in the literature. For example, [46] uses a CNN architecture to perform modulation classification using only raw IQ as input.

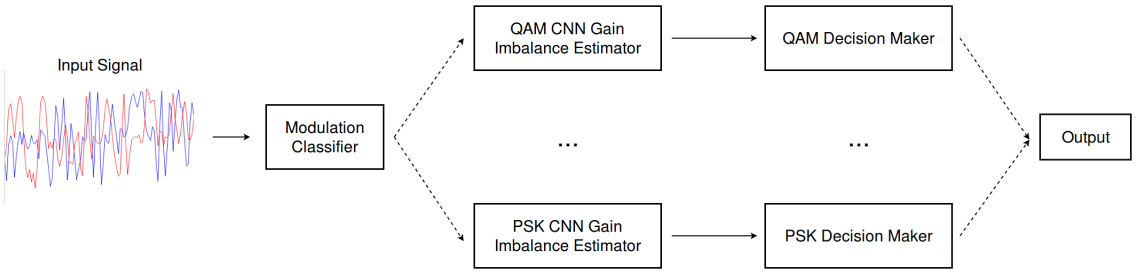
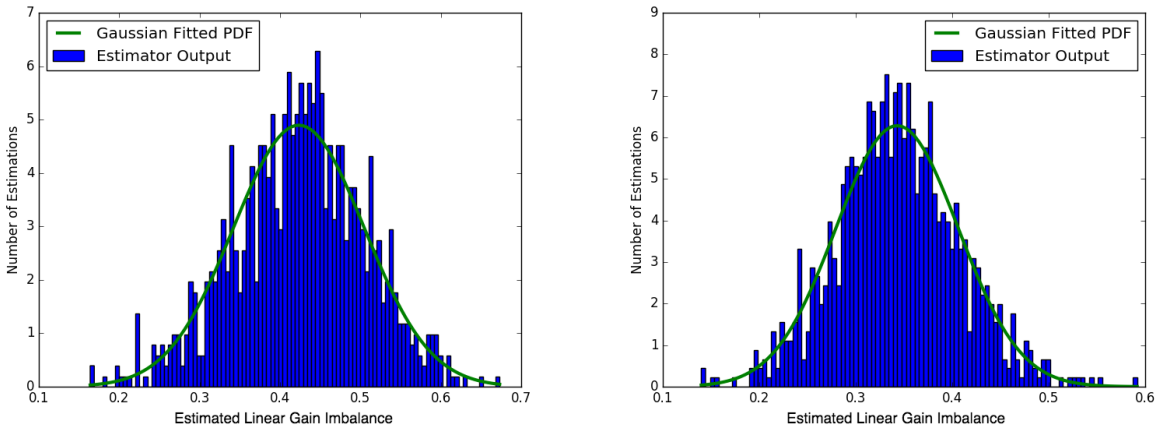


Figure 4.17: The designed emitter identification approach using CNN IQ imbalance estimators.

The output of the modulation classifier determines which modulation-specific CNN gain imbalance estimator the input signal is fed to, and the gain imbalance of the emitter can be appropriately estimated. The point estimate produced by the CNN gain imbalance estimator in the previous step is then used to determine the identity of the transmitter using modulation-specific decision makers built using Gaussian probability density functions (*pdf*s) and Bayes optimal decision boundaries, to be discussed below.

The next sections will describe the components of the modulation specific decision makers. Using the evaluation sets described in Section 4.3.1, histograms of the estimator outputs can be produced. In Section 4.4.1.1, the fit of a Gaussian *pdf* to these histograms will be discussed. The derivation of the Bayesian decision boundaries between these *pdf*s for differing known imbalance values is shown in Section 4.4.1.2. Finally, the use of the Gaussian *pdf*s and Bayesian decision boundaries for determining emitter identity is described in Section 4.4.1.3 and for determining the probability of mis-identification in Section 4.4.1.4.



(a) QAM (b) PSK

Figure 4.18: The fitted Gaussian curve for the 1024-input CNN gain imbalance estimator output histogram with input signals at 10dB SNR and true linear gain imbalance = 0.30.

4.4.1.1 Gaussian Curve Fit to CNN Output Histograms

Using the evaluation sets described in Section 4.3.1, histograms can be produced for the CNN estimator outputs at evenly spaced intervals of 0.01 within the training interval, [-0.9, 0.9]. Because a Gaussian trend was observed, the *pdf* was fitted to the gain imbalance estimator output histograms, as shown in Figure 4.18, using SciPy's statistics package [83].

The goodness of fit was tested using the Chi-Squared Goodness of Fit (GoF) test, as follows [84]: Letting the null hypothesis (H_0) be that the data is consistent with a Gaussian distribution, and the alternate hypothesis (H_1) be that the data is not consistent with a Gaussian distribution, the χ^2 test produces a p-value representing the probability of incorrectly rejecting the null hypothesis. The null hypothesis is rejected if the p-value is less than the chosen significance level. In the literature, 0.05 is a commonly chosen significance level

	QAM	PSK
0dB	0.519	0.717
5dB	0.644	0.565
10dB	0.707	0.710
15dB	0.659	0.600
20dB	0.591	0.558
25dB	0.618	0.525

Table 4.1: The p-values produced by the χ^2 GoF test, averaged over all gain imbalance values.

and is used here [84, 85].

As shown in Table 4.1, the χ^2 test produced average p-values greater than 0.05 for both the QAM and PSK gain imbalance estimators for SNRs varying from 0 to 25dB, over all imbalance values, using 1024 input samples, so the null hypothesis, and thus the Gaussian fit for the CNN outputs, was accepted.

4.4.1.2 Bayesian Decision Boundaries

Given two imbalance values, i and j, the Bayesian decision boundary between the fitted *pdf*s, $p(x|i)$ and $p(x|j)$, is calculated as follows. The following calculations assume that each imbalance value is equally likely to occur in any given emitter, but are not specific to the Gaussian *pdf* and can therefore be used for any curve fit.

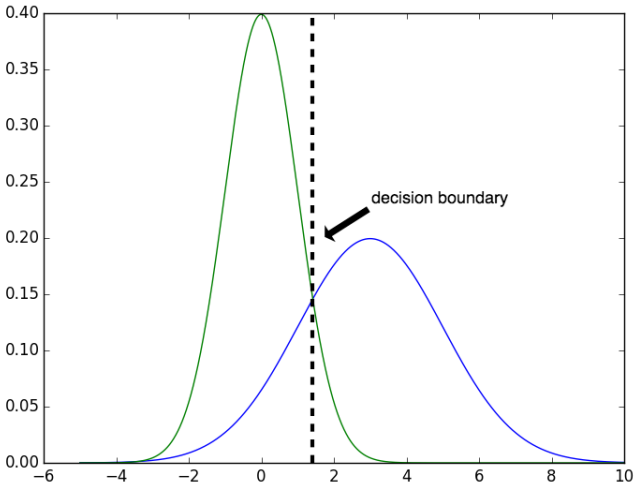


Figure 4.19: The Bayesian decision boundary given two equally likely Gaussian *pdf*s.

Letting x be the received signal data,

$$\text{Decide } i \text{ if } P(i|x) > P(j|x); \text{ otherwise decide } j.$$

Using Bayes Rule, this decision rule can be expressed in terms of the fitted *pdf*s $(p(x|i), p(x|j))$ and the probability of the emitter having a given imbalance value $(P(i), P(j))$:

$$\text{Decide } i \text{ if } p(x|i)P(i) > p(x|j)P(j); \text{ otherwise decide } j.$$

Finally, assuming each imbalance value is equally likely to occur (i.e. $P(i) = P(j)$), the final decision rule is

$$\text{Decide } i \text{ if } p(x|i) > p(x|j); \text{ otherwise decide } j,$$

making the decision boundary the intersection point(s) of the two fitted *pdf*s $p(x|i)$ and $p(x|j)$ for imbalance values i and j, i.e. where $p(x|i) = p(x|j)$, shown in Figure 4.19 [86].

4.4.1.3 Decision Making

Given a decision boundary calculated between the two *pdf*s, $p(x|i)$ and $p(x|j)$, for imbalance values i and j and a received signal, a decision can be made about emitter identity. After modulation classification, the received signal can be fed to the appropriate gain imbalance estimator, producing a point estimate of the gain imbalance of the emitter which sent the signal.

Without loss of generality, let the mean of $p(x|i)$ be less than the mean of $p(x|j)$. If the point estimate falls on the left side of the decision boundary, it is decided the transmitted signal came from an emitter with imbalance value i. Otherwise, it is decided the transmitted signal came from an emitter with imbalance value j.

For illustrative purposes, consider the example decision scenario shown in Figure 4.20. Because the point estimate falls on the right side of the decision boundary, it is decided the transmitted signal came from Emitter 2. However, this assumes the gain imbalance values of each emitter in the system is known, the first of two ways the proposed approach may be used.

In this case, the *pdf*s of the known imbalance values can be selected and the decision boundaries between these *pdf*s calculated. Decisions on point estimates are then made as described above. While this method has use cases for Dynamic Spectrum Access and cooperative scenarios [87], the ability to perform SEI in non-cooperative and blind scenarios is a primary motivator of this work. In the case that the emitters in the system are unknown,

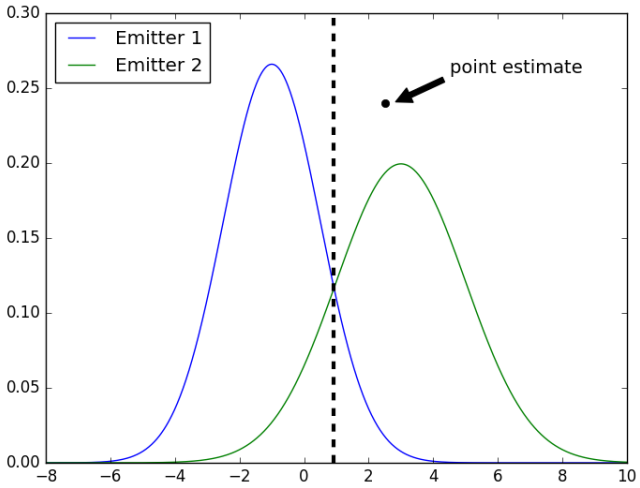


Figure 4.20: An example decision scenario identifying an emitter by its estimated gain imbalance using the calculated Bayesian decision boundary.

the approach may still be used. However, because the *pdf*s of the known imbalance values cannot be selected, it is only possible to "bin" the emitters into intervals of possible imbalance values. To do this, *pdf*s are selected at evenly spaced values, and the decision boundaries calculated, indicating the endpoints of each bin. Then, as in the first case, decisions on point estimates are made as described above. For simplicity, the results shown in Section 4.4.3 will consider only this second case.

4.4.1.4 The Probability of Mis-Identifying Emitters

Given two fitted *pdf*s, $p(x|i)$ and $p(x|j)$, and the decision boundary, d, between the *pdf*s, it is also possible to determine the probability of misidentifying an emitter:

Consider the scenario in Figure 4.21, where a point estimate, x_i, is produced from a set

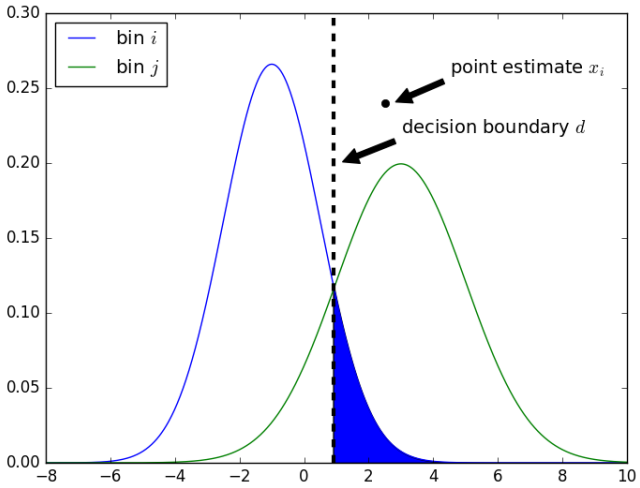


Figure 4.21: The region representing the probability of mis-identifying the point estimate.

of samples received from an emitter belonging to imbalance bin i. Again, without loss of generality, let the mean of $p(x|i)$ be less than the mean of $p(x|j)$. A correct classification occurs when the estimate from the CNN, x_i, is less than the decision boundary d. Therefore, an incorrect classification occurs when $x_i > d$. Because the area under a *pdf* is 1, the probability of this occurring is

$$\int_d^\infty p(x|i)dx,$$

and is represented by the shaded blue region in Figure 4.21.

4.4.2 Model Design, Training, and Evaluation

The model developed previously contained two two-dimensional convolutional layers followed by four dense fully-connected layers. The final layer used a linear activation function, while all other layers used the ReLU activation function. This approach uses the same model,

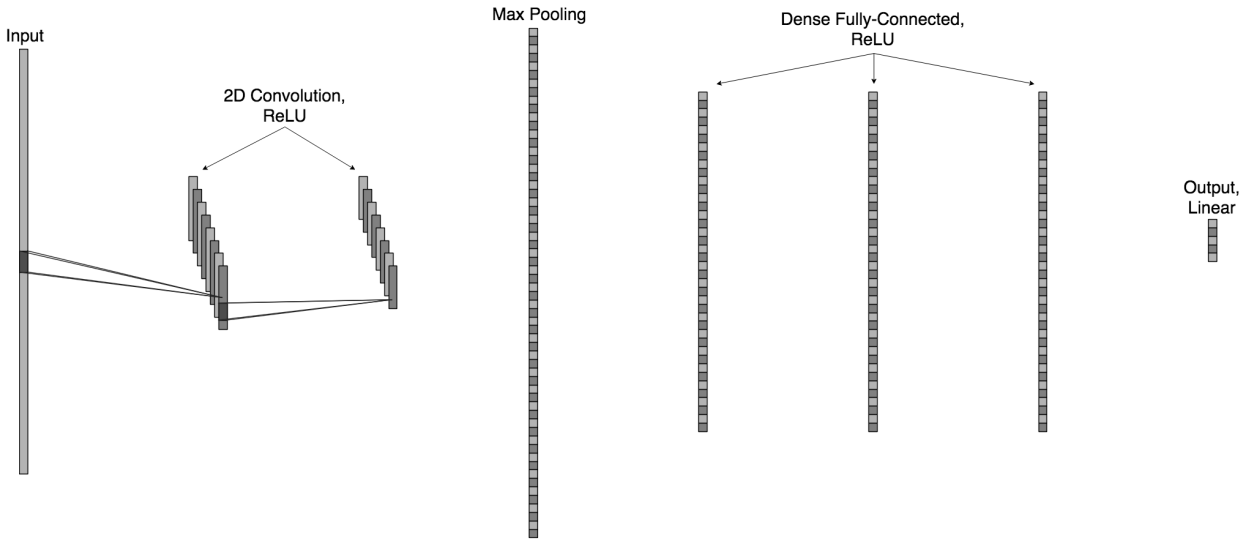


Figure 4.22: The CNN architecture designed for estimation of transmitter gain imbalance to perform SEI.

modified with one max-pooling layer, with size = 2, inserted between the convolutional layers and the dense layers, as shown in Figure 4.22.

When determining which networks performed best, the NMSE was no longer a helpful evaluation metric, as a network with a low average NMSE could produce histograms with larger variance than networks with a higher average NMSE. Therefore, to evaluate the performance of trained networks, the evaluation sets previously described in Section 4.3.1 were used. For a given trained network, *pdf*s were fitted for each of the gain imbalance values, and the minimum gain imbalance separation needed to obtain a probability of mis-identification of less than 5% was calculated. This value was used to determine which networks were performing better than others.

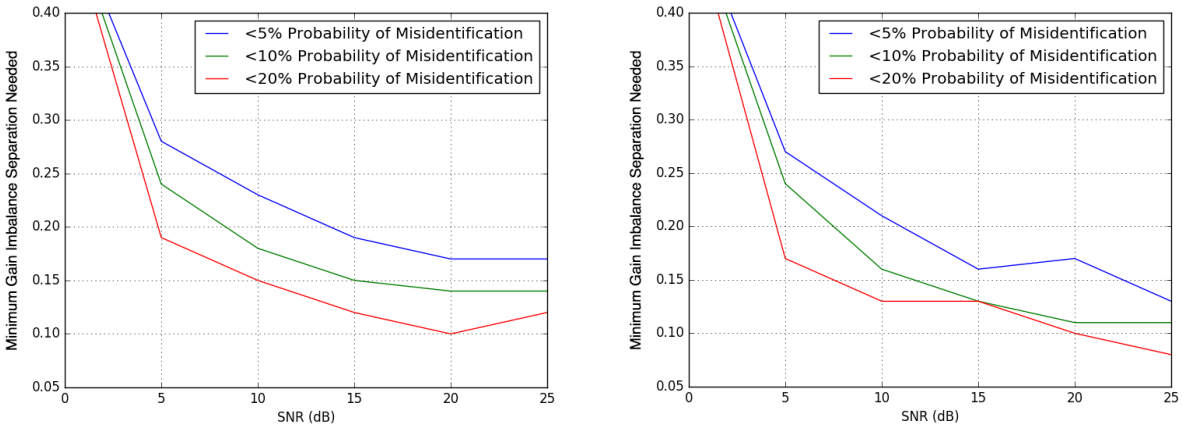
4.4.3 Simulation Results and Discussion

Impact of SNR on SEI Ability

Using the evaluation sets constructed for QAM and PSK, the minimum gain imbalance separations needed to obtain average probabilities of mis-identification of less than 20%, 10%, and 5% across all imbalance values were calculated, as described in Section 4.4.1.4. The impact of SNR on the ability to identify emitters at these levels of accuracy is shown in Figure 4.23. At 0dB, the estimators cannot be used to perform emitter identification to even a 80% probability of correct identification. However, as the SNR increases, the minimum gain imbalance separation needed to obtain $< 5\%$, $< 10\%$, and $< 20\%$ probabilities of mis-identification decreases with diminishing returns at around 20dB. Therefore, the lower the probability of mis-identification needed in a system, the higher the gain imbalance separation needed.

Impact of imbalance Value on SEI Ability

In Section 4.3.3 and Figure 4.9, it was shown that the sample variance of the gain imbalance estimators is slightly lower when the true imbalance value is at the limits of the training range (near -0.9 and 0.9). As a result, the variance of the fitted pdfs is lower when the true imbalance value is near the limits of the training range, in comparison to when the true imbalance value is near zero. Therefore, the probability of mis-identification is lower when the true imbalance value is near the limits of the training range.



(a) QAM

(b) PSK

Figure 4.23: The SNR versus minimum gain imbalance separation needed to obtain $< 5\%$, $< 10\%$, and $< 20\%$ probability of mis-identification using the CNN gain imbalance estimator.

Impact of Modulation Scheme on SEI Ability

The ability to perform gain imbalance estimation on QAM and PSK signals using the designed CNN architecture was shown in Section 4.3.3. Though the use of CNN estimators for gain imbalance estimation on other signal types was not investigated, the comparable results of the CNN gain imbalance estimators trained for QAM and PSK showed that the designed network architecture described in Section 4.3.1 was not modulation specific. Investigation into the performance of the estimators on further modulation schemes is left for future work.

Figure 4.24 shows the importance of having separate decision boundaries for each modulation class. Though the true imbalance value of the input signal to the estimators is the same, the output histograms produced by the modulation specific CNN gain imbalance estimators are not. This yields different decision boundaries for each modulation class.

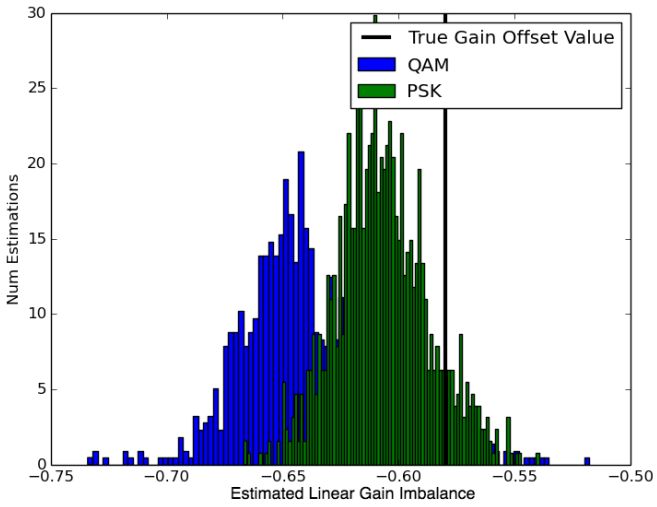


Figure 4.24: The histogram outputs for the PSK and QAM estimators both with true gain imbalance values = -0.58.

As discussed in Section 4.3.3 and shown in Figure 4.15, the CNN gain imbalance estimator trained for PSK showed a lower average sample variance than the CNN gain imbalance estimator trained for QAM. This resulted in fitted pdfs with lower variance for PSK than QAM. Therefore, in general, lower minimum gain imbalances are needed to identify emitters with the same accuracy as the QAM estimators, as shown in Figure 4.23, and more emitters can be identified uniquely.

Practical Considerations

As expected, for both QAM and PSK, the lower the probability of mis-identification needed in a system, the higher the gain imbalance separation needed to achieve the needed level of accuracy. Therefore, in systems with a higher tolerance for mis-identification, more emitters

can be uniquely identified, and in systems that require a low probability of mis-identification, fewer emitters can be uniquely identified.

However, even in systems with a 20% tolerance for mis-identification receiving signals exceeding 20dB SNR, emitters need to have a linear gain imbalance separation of at least 0.15. While few publications indicate measured gain imbalance values for real systems, most prior works in IQ imbalance estimation and compensation use test values on the order of 0.05 [67,70–76], indicating the gain imbalance values necessary to obtain even 80% accuracy are not practical in real systems. Narrowing the range of gain imbalance values included in the training set would likely help combat this problem.

The training set was simulated with gain imbalances between $[-0.9, 0.9]$, uniformly distributed, in order to incorporate any possible gain imbalance value the CNN estimator might encounter in a real system. However, training over such a large range has likely hindered the estimator's accuracy, as the network has had to learn to generalize over such a large range [46]. Given that 0.9 is likely much larger than anything one might find in a real system, the training range could be narrowed to yield better results in estimator accuracy and therefore SEI ability.

Comparison to a Traditional Feature-Based Approach

In [88], Zhuo et al. develop an SEI approach using IQ imbalance estimates and SVMs. The IQ imbalance estimation algorithm proposed uses statistical methods to determine gain and phase imbalance using the received symbols, and has assumed perfect synchronization said

	Emitter 1	Emitter 2	Emitter 3	Emitter 4	Emitter 5
α	0.1	0.13	0.15	0.17	0.19
θ	3°	3.3°	3.6°	3.9°	4.2°

Table 4.2: The simulated IQ imbalance parameters used to produce the results in [88].

symbols. The approach also requires SNR estimation. They then plot gain versus phase imbalance in two dimensions and use SVMs to assign the received signal to an emitter.

In order to provide an accurate comparison of the approach proposed in [88], five QPSK emitters were simulated with gain and phase imbalance values given in Table 4.2, for SNRs between [5dB, 35dB] at intervals of 5dB, and perfect synchronization has been assumed, as in [88]. Additionally, the developed CNN gain imbalance estimator was retrained to accommodate these assumptions. More specifically, the CNN gain imbalance estimator was retrained using training, validation, and test sets with SNRs between [0dB, 35dB], gain imbalance values (α) between $[0.0, 0.3]$, phase imbalance values (θ) between $[0°, 5°]$, and no frequency and sample rate imbalances, in order to match the assumptions in [88].

The results in Figure 4.25 show the accuracy of the approach developed in this chapter to that proposed in [88], given one capture of 1024 raw IQ samples and given ten captures of 1024 raw IQ samples. When the approach developed in this chapter uses only one capture of 1024 raw IQ samples, the developed approach shows lower accuracies than the approach presented in [88] by as much as 15%. However, it is important to note the developed approach uses only estimates of gain imbalance, while the approach presented in [88] uses estimates of

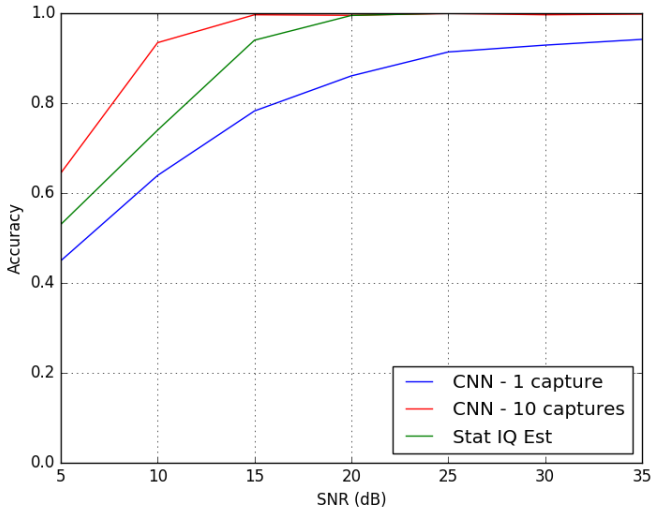


Figure 4.25: The accuracy of the developed SEI approach using CNN gain imbalance estimators compared to the accuracy of the approach proposed in [88], given one and ten captures of 1024 raw IQ samples.

both gain and phase imbalance, providing more information about the emitter-of-interest. Phase imbalance estimates could also be incorporated into the approach developed in this chapter, and would likely increase performance. Furthermore, the approach presented in [88] is dependent upon an estimate of SNR and assumes perfect synchronization, whereas the approach developed in this chapter needs no external measurements or estimates and can compensate for an imperfect receiver.

Additionally, the accuracy of the approach proposed in [88] is calculated given 1330 symbols, whereas the accuracy of the approach developed in this chapter is given for one capture of 1024 raw IQ samples. Given multiple captures of raw IQ samples, the outputs can be aggregated, and the accuracy of the approach developed in this chapter improves, as

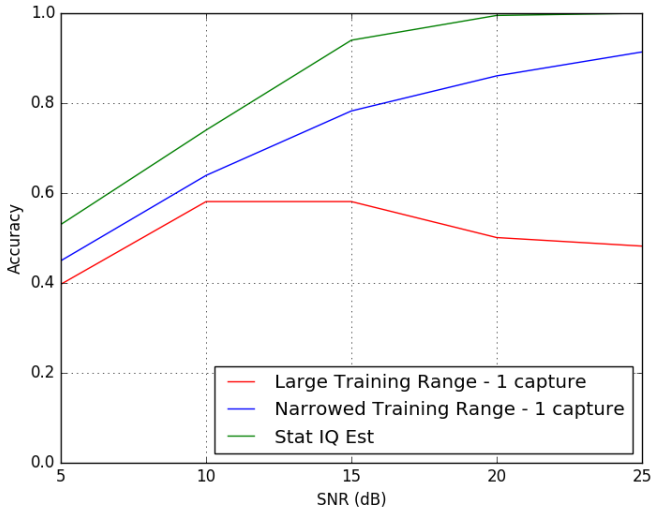


Figure 4.26: The accuracy of the developed SEI approach using the large training range described in Section 4.3.2 compared to the accuracy using the narrowed training range used to match the assumptions made in [88].

shown in Figure 4.25. Therefore, given as few as ten captures of 1024 raw IQ samples, the accuracy of the approach developed in this chapter exceeds the performance of the approach proposed in [88], using less data and making far fewer assumptions.

It should also be noted that limiting the range of IQ imbalance parameters and assuming perfect synchronization has improved the performance of the developed approach, as shown in Figure 4.26. This confirms that training over smaller parameter ranges, more closely aligned with those one might find in a real system, improves the performance of the approach, and is consistent with the results and discussion in [46].

4.5 Summary and Future Work

In this chapter, the capability of CNNs to estimate gain and phase imbalances between the in-phase and quadrature components of a signal was shown, assuming transmission through an AWGN channel. Performance analysis of the developed CNN IQ imbalance estimators, using QAM and PSK as test signals, showed the model to be modulation agnostic, and showed the model's ability to estimate both gain and phase imbalances, with performance increasing as SNR and network input size increases. However, phase imbalance proved to be far more difficult to estimate than gain imbalance with the designed network architecture, showing much higher bias and sample variance values. Therefore, an SEI approach using parallel modulation-specific gain imbalance estimators was designed and evaluated.

For both QAM and PSK modulation schemes, the proposed SEI approach showed increases in performance as SNR increases, in the form of smaller gain imbalance separations needed to achieve lower probabilities of mis-identification. Because the gain imbalance estimators trained for PSK slightly outperformed those trained for QAM, the proposed SEI approach performed better when the incoming signal was of a PSK modulation scheme. Though the approach was shown to need impractical gain imbalance separation values, even in high SNR scenarios, when the range of IQ imbalance parameters included in the training set is large, performance improved significantly when this range was narrowed. Further, the accuracy of the approach was shown to exceed that of a traditional feature-based approach, given as few as ten captures of 1024 raw IQ samples.

Therefore, the SEI approach developed in this chapter has improved upon traditional SEI approaches in the following ways:

- Pre-processing steps typically needed in traditional SEI systems, such as synchronization, carrier tracking, feature extraction, or SNR estimation, are no longer needed because raw IQ is used as input to the network.
- Gain imbalances can be estimated in low SNR scenarios where traditional feature extraction techniques, and therefore traditional SEI systems, may fail.
- The developed CNNs can easily be trained or retrained, to extend the model, using synthetic training data. Extending a traditional SEI system in such a way would first require data collection from emitters of interest and modification of the needed signal processing and feature extraction algorithms, before the system could be extended.
- The developed approach has outperformed a traditional feature-based approach, which also utilizes IQ imbalance to identify emitters, while using less data and making fewer assumptions.

To improve the proposed SEI approach, the range of gain imbalance values included in the training set can be narrowed, so that the network has to generalize less. This was shown when comparing the developed approach to the approach in [88]. The addition of more hardware impairments to the model, in order to further discriminate emitters, would also likely increase performance and is left for future work. Additionally, this work could be

extended to further modulation schemes and to receiver IQ imbalance. Finally, the ability to estimate IQ imbalance and perform SEI in the presence of different channel models may be explored.

Chapter 5

Clustering Learned CNN Features

In Chapter 4, CNNs were utilized to estimate transmitter IQ imbalance, and further, the estimated gain offset values were used to identify emitters across modulation schemes. In this chapter, CNNs are used to learn the features relevant to emitter identity themselves, utilizing their inherent feature learning abilities, as opposed to requiring the network to learn a specific feature.

More specifically, in this chapter a semi-supervised emitter identification approach is developed which utilizes a supervised CNN feature extraction step and an unsupervised clustering step, described further in Section 5.1. As in Chapter 4, the developed approach uses only raw IQ data as input. However, the dataset used for this work is comprised entirely of real data captures as opposed to synthetic data, like in Chapter 4, and is described in Section 5.1.1.

Because identifying emitters across bandwidth changes is a known challenge in the SEI

field [4], each emitter is first assumed to be transmitting at a single bandwidth. Under this assumption, it is shown that emitter identification can be performed using the developed approach, even in the presence of emitters the CNN feature extractor didn't see in training. This assumption is then relaxed, and the performance of the proposed approach is further evaluated.

Acknowledgements

Thank you to Seth Andrews and Dr. Ryan Gerdes for providing the dataset used in this chapter.

5.1 Approach

The developed approach, shown in Figure 5.1, utilizes CNNs to extract emitter-specific features by training the network to perform emitter identification on raw IQ data. Once trained, the CNN feature extractor simply serves as the pre-processor to a clustering step. To allow for the identification of emitters not seen in CNN training, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is then used to cluster the extracted features by emitter [89]. Finally, t-distributed Stochastic Neighbor Embedding (t-SNE) is used to reduce the dimensionality of the features, for visualization only [90]. An example output of the developed approach is shown in Figure 5.2.

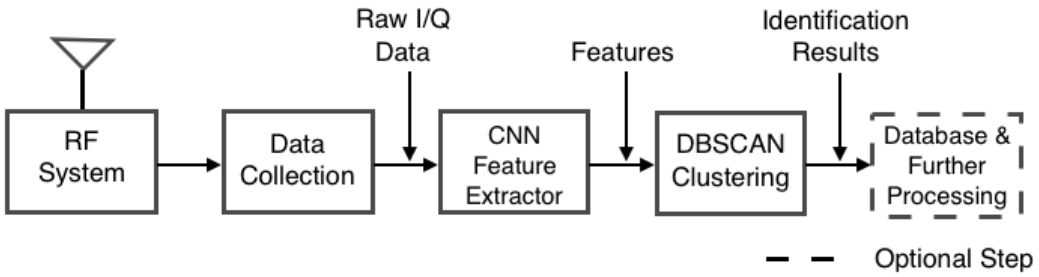


Figure 5.1: The developed CNN-learned feature clustering SEI approach.

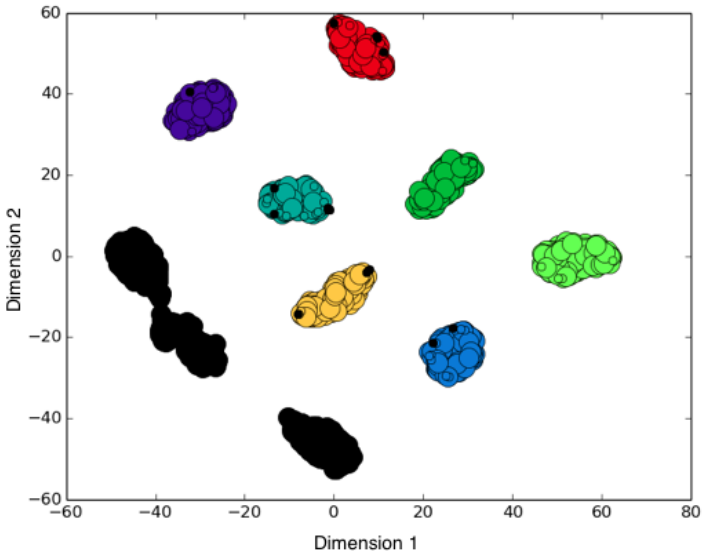


Figure 5.2: Clustering of features learned by the CNN trained on 10 emitters transmitting at a single bandwidth with 10 emitters in the system. 8 clusters (emitters) found by DBSCAN with an AMI of 0.85. Visualization of features embedded in 2 dimensions using t-SNE. Each color represents an emitter found by DBSCAN.

5.1.1 Dataset

The dataset used for this work contains wired transmissions from 53 Ettus USRP B210's and N210's. The dataset contains captures from each USRP transmitting at 11 different bandwidths, nine evenly distributed from [0.25MHz, 1.25MHz] and at 1.67MHz and 2.5MHz, using a QPSK modulation scheme. Each transmission contains the same sequence of bits, as the dataset was composed for emitter fingerprinting work, and traditional techniques often make this assumption [4].

5.1.2 Feature Extraction

In [47], it was shown that the learned features extracted from a CNN could be used in conjunction with a clustering algorithm to successfully perform modulation classification using a method called *supervised bootstrapping*. The method first required training a CNN in a supervised manner to perform modulation classification. After the network has been trained, the features relevant to the modulation scheme of an incoming signal were then obtained by feeding the raw IQ samples of the signal through the trained network. More specifically, the output of the first dense fully-connected layer, not the Softmax output, is then used for clustering.

This work used the *supervised bootstrapping* method to find features relevant to emitter identity. Therefore, instead of training a CNN to perform modulation classification like in [47], a CNN was trained to perform emitter identification. Once trained, incoming signals

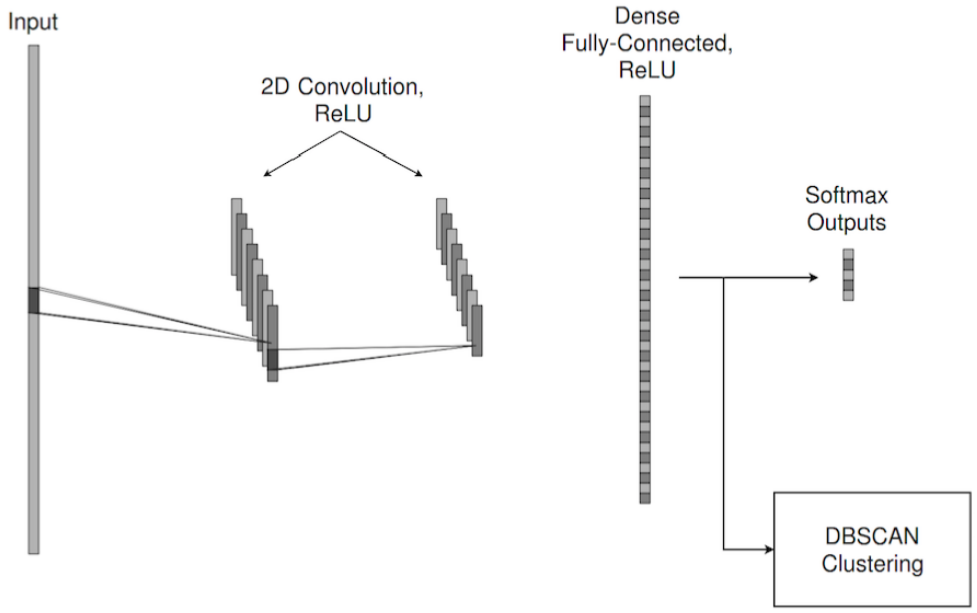


Figure 5.3: Feature extraction from a pre-trained CNN.

were fed to the trained network to obtain emitter-specific features. As described above and shown in Figure 5.3, the features used for clustering were not the Softmax outputs used during supervised training, but the output of the first dense fully-connected layer. These features are then fed to the DBSCAN clustering algorithm.

5.1.3 Clustering

After relevant features are extracted from the received signals, the features must be matched to features extracted from other received signals in order to identify emitters. Using an unsupervised clustering approach to group features, both the number of emitters in the system and which emitter each signal is being transmitted from can be identified while remaining blind to the environment.

The developed approach makes no assumptions about the number of emitters in a system or any knowledge about said emitters, once the CNN feature extractor has been deployed. As such, it is critical that the chosen clustering algorithm be able to determine the optimal number of clusters to fit the data. For this reason, the DBSCAN algorithm was chosen over those popularly used in the literature, such as SVMs and K-means [27, 91], and because it has shown success clustering high-dimensional and noisy data [89, 92, 93].

5.1.4 Visualization

Because the DBSCAN algorithm is able to cluster in any dimension, dimensionality reduction techniques are not integral to the approach. However, dimensionality reduction allowed for the visualization of the clustered features in a two-dimensional space, helping to tune the algorithm and validate the approach. It is important to understand that the CNN learned features have been clustered in the original high-dimensional space, not in the two-dimensional space. So, while clusters may not be co-located in the two-dimensional space, they may be in the high-dimensional space.

t-SNE was chosen for dimensionality reduction because of its ability to effectively map the high-dimensional CNN features onto a two-dimensional map [90]. Additionally, when compared to other popular dimensionality reduction techniques such as Sammon mapping, Isomap, and Locally Linear Embedding (LLE), t-SNE was shown to be more effective at preserving the structure of the high-dimensional space, when applied to real datasets [90].

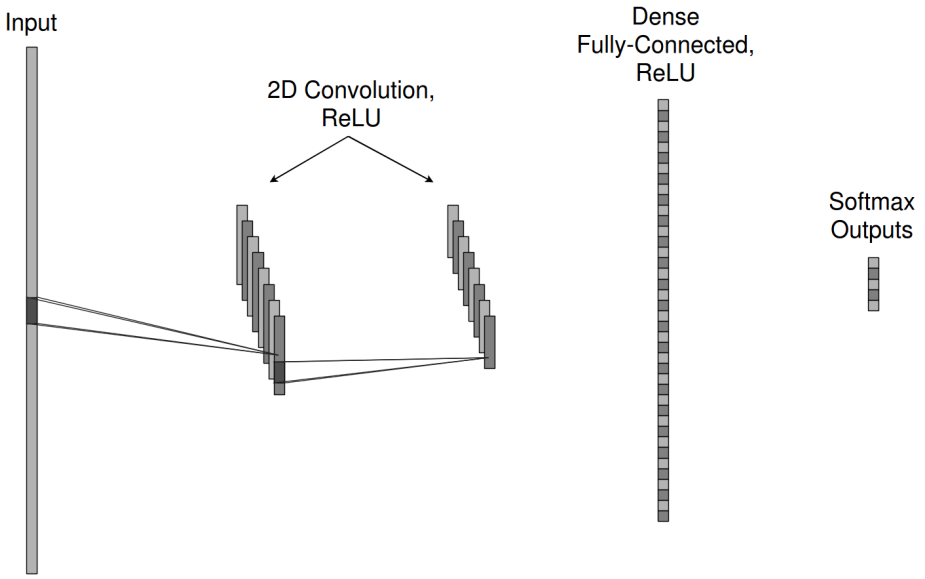


Figure 5.4: The CNN model designed to perform emitter identification and used for feature extraction in the clustering approach.

5.2 Model Design, Training, and Evaluation

The CNN feature extraction architecture designed for the approach is shown in Figure 5.4, and is modeled after the network used in [9]. However, the number of filters, filter sizes, and layer sizes in the developed network were altered to improve accuracy using the developed script described in Section 3.2.2.

The network designed uses 1024 raw IQ samples as input. Following the input layer are two 2D convolutional layers, a dense fully-connected layer, and an output layer. With the exception of the output layer, all layers use a ReLU activation function. The output layer uses a Softmax activation. Additionally, dropout is applied at a rate of 0.5 between the last convolutional layer and the dense layer to help control over-fitting [33].

To evaluate the robustness of the approach, 12 networks were trained: The first 6 networks were trained to perform emitter identification using training data such that all signals have the same bandwidth. These networks were trained to identify 2 emitters, 5 emitters, 10 emitters, 15 emitters, 20 emitters, and 25 emitters. Six corresponding networks were trained on all eleven bandwidths in the dataset. The sizes of the filters and layers in each of these networks differs, as each network's parameters have been selected by the developed training scripts separately.

The parameters for each network are shown in Tables 5.1 and 5.2. While there is no noticeable trend between network size and the number of emitters the network saw in training, the number of filters in the convolutional layers and the sizes of those filters generally increases as the network learns to generalize over bandwidth. More specifically, the convolutional layers generally increase in size from Table 5.1 to Table 5.2.

Because the dataset used provided a limited amount of data, each network was trained using approximately 390 sets of samples per emitter where 75% was randomly selected for training and the remaining 25% was used for testing. To further minimize the effects of overfitting, an early-stopping scheme was used to stop training when the output of the loss function stopped improving. The classification accuracy achieved on the test set was used to evaluate each network, and will be referred to as the *testing accuracy* from this point on, for clarity.

Number of Emitters	2D Conv 1: Number of Filters	2D Conv 1: Filter Size	2D Conv 2: Number of Filters	2D Conv 2: Filter Size	Dense Fully-Connected
2 emitters	65	4	19	12	231
5 emitters	82	4	57	4	352
10 emitters	91	5	9	15	298
15 emitters	11	4	45	18	76
20 emitters	28	13	28	14	321
25 emitters	82	11	81	11	395

Table 5.1: The network parameters for each CNN feature extractor, assuming all transmissions are at a single bandwidth.

Number of Emitters	2D Conv 1: Number of Filters	2D Conv 1: Filter Size	2D Conv 2: Number of Filters	2D Conv 2: Filter Size	Dense Fully-Connected
2 emitters	75	15	79	15	336
5 emitters	68	11	68	7	45
10 emitters	78	9	33	16	227
15 emitters	49	17	48	16	231
20 emitters	78	11	91	16	87
25 emitters	52	14	53	13	106

Table 5.2: The network parameters for each CNN feature extractor trained across all 11 bandwidths in the training dataset.

5.3 Evaluating the Approach

The effectiveness of the approach was evaluated based on the testing accuracy of the CNN feature extractor, the accuracy of the output, the number of emitters the approach can identify, and the ability to identify emitters outside of the training set of the CNN feature extractor.

The testing accuracy of the CNN being used for feature extraction is calculated using the Softmax output of the trained network and the testing portion of the labeled training data as described in Section 5.2. If the CNN being used for feature extraction cannot effectively differentiate between emitters, the features being extracted from the network will not be representative of emitter identity.

The output of the DBSCAN algorithm is a set of predicted labels such that each element within a cluster has the same label. The number of emitters the approach is able to identify can be measured as the number of clusters the DBSCAN algorithm identifies, given the nature of the approach. However, calculating the accuracy of the predicted labels is not straight-forward. Because the DBSCAN algorithm is unsupervised, the predicted labels may not match the ground truth, and further, the number of clusters found may not match the number of classes making comparison to the ground truth an unreliable measure of accuracy. Instead, the adjusted mutual information score (AMI) was used to measure the similarity between the true and predicted labels while ignoring permutations [94]. The AMI has a range of [0, 1], where random label assignments produce a value of 0 and perfect label

assignments produce a value of 1. The AMI is also normalized against chance, to account for the mutual information of two random sets not being constant [94].

The AMI of two sets U, V is calculated as follows [94]:

$$AMI = \frac{MI - E[MI]}{\max\{H(U), H(V)\} - E[MI]}, \tag{5.1}$$

where MI is the mutual information between U and V calculated as

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log\left(\frac{N|U_i \cap V_j|}{|U_i||V_j|}\right), \tag{5.2}$$

$E[MI]$ is the expected value of the mutual information between U and V calculated as

$$E[MI(U,V)] = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \sum_{n_{ij}=(a_i+b_j-N)^+}^{\min a_i, b_j} \frac{n_{ij}}{N} \log\left(\frac{N \cdot n_{ij}}{a_i \cdot b_j}\right) \times \frac{a_i! b_j! (N-a_i)!(N-b_j)!}{N! n_{ij}! (a_i - n_{ij})!(b_j - n_{ij})!(N - a_i - b_j + n_{ij})!}, \tag{5.3}$$

$H(U)$ is the entropy of U calculated as

$$H(U) = -\sum_{i=1}^{|U|} \frac{|U_i|}{N} \log \frac{|U_i|}{N}, \tag{5.4}$$

and $H(V)$ is the entropy of V calculated as

$$H(V) = -\sum_{j=1}^{|V|} \frac{|V_j|}{N} \log \frac{|V_j|}{N}. \tag{5.5}$$

Because *a priori* knowledge of the emitters in the system after the CNN feature extractor has been deployed is not being assumed, it is important that the approach be able to identify emitters the CNN never saw in training. This was measured using the number of emitters the approach is able to identify and the AMI, as a function of the number of emitters in the system.

In the following sections, when the number of emitters in the system is stated as less than or equal to the number of emitters the CNN feature extractor saw in training, the CNN trained on those emitters. However, when the number of emitters in the system exceeds the number of emitters the CNN feature extractor saw in training, the algorithm is being asked to identify

$$\#\ of\ emitters\ in\ the\ system - \#\ emitters\ in\ the\ CNN\ training\ set$$

unseen emitters.

The ability to identify emitters outside of the CNN feature extractor's training set indicates the features extracted by the CNN are general enough to describe emitters outside of the training set. The approach can be said to have successfully found emitters outside of its training set, if the number of clusters found exceeds the number of emitters the CNN feature extractor saw in training, the AMI remains high, and there are emitters present that the CNN feature extractor has never seen.

5.4 Results and Discussion

5.4.1 Transmissions at a Single Bandwidth

As previously mentioned, tracking emitters as they change bandwidth is a known challenge in the SEI field [4]. Therefore, to limit the scope of the problem, it was first assumed that all transmissions were at the same bandwidth, in addition to containing the same bits and using

Number of Emitters	Testing Accuracy
2 emitters	1.0
5 emitters	0.998
10 emitters	0.995
15 emitters	0.939
20 emitters	0.870
25 emitters	0.787

(a) 0.25 MHz

Number of Emitters	Testing Accuracy
2 emitters	1.0
5 emitters	1.0
10 emitters	0.991
15 emitters	0.960
20 emitters	0.893
25 emitters	0.804

(b) 1.67 MHz

Table 5.3: The testing accuracies of each CNN feature extractor, assuming all transmissions are at a single bandwidth.

the same modulation scheme, as described in Section 5.1.1. The following results reflect this assumption.

The testing accuracy of the CNN feature extractors, assuming all received signals are at a single bandwidth are shown in Table 5.3 for signals transmitted at both 0.25MHz and 1.67MHz, for completeness. For signals transmitted at 0.25MHz, the networks classifying up to 15 emitters all had Softmax testing accuracies of over 90%, while the network classifying 20 emitters has a testing accuracy of 87% and the network classifying 25 emitters has a testing accuracy of 78.7%. This drop in performance is likely due to the small amount of training data available. However, given the probability of a random guess being correct is 0.5, 0.2, 0.1, 0.067, 0.05, and 0.04 respectively, this is sufficient evidence that the network has learned features relevant to the emitter's identity.

For signals transmitted at 1.67MHz, the trained networks showed very similar Softmax testing accuracies, with very slight performance improvements, most likely due to a higher total signal energy caused by transmission at a higher bandwidth and oversampling effects. However, as the performance improvements are marginal, it can be concluded network performance is not heavily dependent upon transmission bandwidth, and it can be concluded that the networks are learning features related to emitter identity, other than transmission bandwidth. As such, all further results are shown using transmissions at 0.25MHz.

Figures 5.5 and 5.6 show the results of the developed emitter identification scheme using each of the 6 CNN feature extractors. Figure 5.5 shows the percentage of emitters found by the DBSCAN clustering algorithm as a function of the true number of emitters in the system, and Figure 5.6 shows the AMI of the true and predicted labels as a function of the number of emitters in the system. Together, Figures 5.5 and 5.6 show that as the number of emitters in the system increases, both the percentage of emitters found by the DBSCAN algorithm and the AMI of the true and predicted labels decreases, regardless of how many emitters the CNN feature extractor saw in training. These results show that the approach can identify a limited number of emitters outside of the training set. However, each CNN feature extractor can only describe a finite number of emitters. More specifically, while the ability to identify emitters unseen in training indicates that the CNN extracted features describe emitter identity in general to some degree, the learned features are largely specific to the emitters the CNN was trained to identify.

Figures 5.5 and 5.6 further show both the percentage of emitters found by the DBSCAN

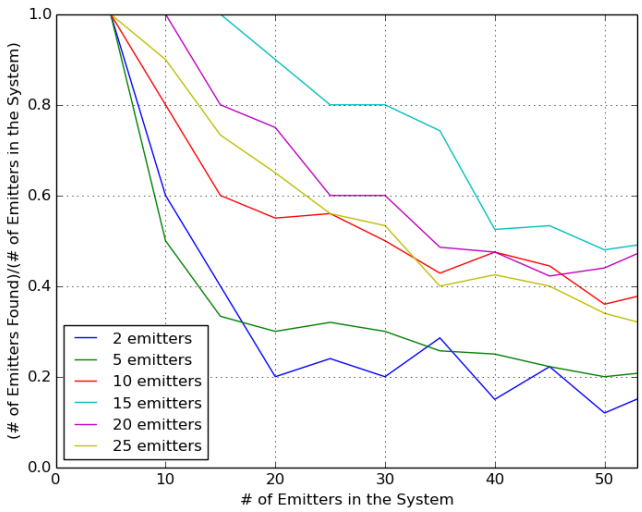


Figure 5.5: The percentage of emitters found by the DBSCAN algorithm, as a function of emitters in the system, for each CNN feature extractor trained assuming transmissions at a single bandwidth.

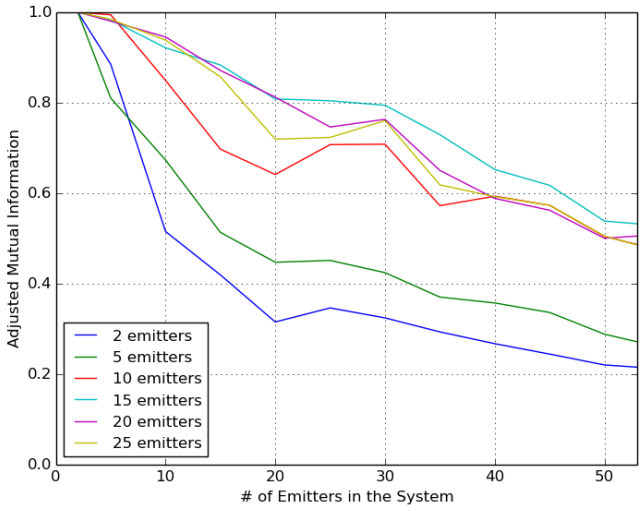


Figure 5.6: The AMI of the true labels and labels predicted using the CNN extracted features, as a function of the number of emitters in the system, for each CNN feature extractor trained assuming transmissions at a single bandwidth.

algorithm and the AMI increase as the number of emitters in the CNN feature extractor training set increases, until this value exceeds 15 emitters. After this point, both the percentage of emitters found and the AMI begin to decrease. From this trend, it can be concluded that the designed CNN feature extractor can only effectively differentiate between up to 15 emitters. However, it is likely that the limited amount of training data has hindered the ability to effectively identify more emitters, as more training data would reduce the effects of overfitting [33, 38].

As previously described in Section 5.3, Figures 5.5 and 5.6 also examine the ability of the approach to identify emitters outside of the training set of the CNN feature extractor. These results are consolidated into Figure 5.7, where the approach has successfully found emitters outside of the training range if the ratio of emitters found by DBSCAN to the number of emitters in the training set exceeds one (i.e. if the solid blue line is above the dashed blue line) while the AMI remains sufficiently high. What is considered "high" is largely application and system specific. For illustrative purposes, an AMI of greater 0.8 will be considered sufficient.

For clarity, examine the results from the CNN feature extractor trained with 15 emitters: When there are between 0 and 15 emitters in the system, DBSCAN finds all emitters with an AMI of 0.89 or higher, shown in Figure 5.7. When between 15 and 30 emitters exist in the system, DBSCAN is still able to find at least 80% of the emitters with an AMI of 0.80 or higher. This means the approach finds 24 emitters in a 30 emitter system when only trained to identify 15. This shows the developed approach has the ability to identify emitters

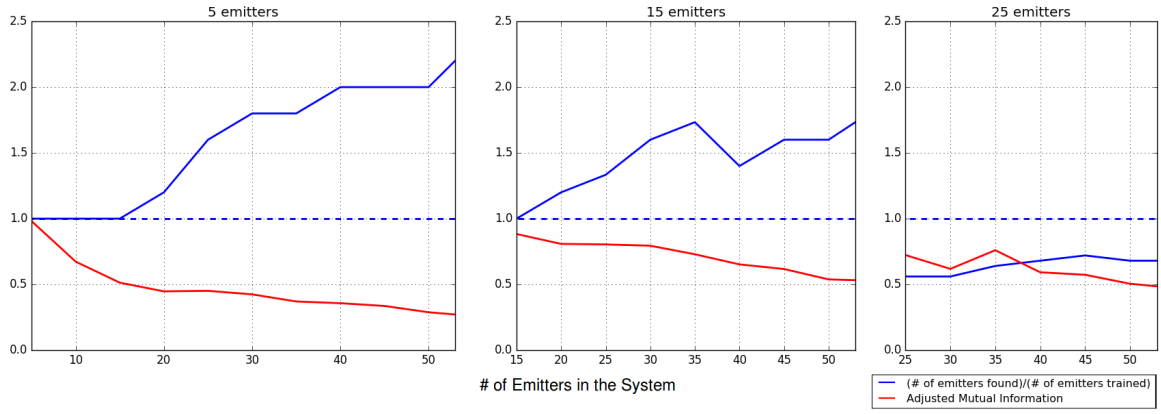


Figure 5.7: The ratio of emitters found to emitters trained and the AMI, as a function of the number of emitters in the system, for each CNN feature extractor trained assuming transmissions at a single bandwidth.

outside of the training set of the CNN feature extractor, given the assumption that all signals have common bandwidth. However, for each CNN feature extractor, the AMI decreases as the number of emitters in the system increases. This decrease in the AMI indicates the clusters found by the DBSCAN algorithm are no longer matching the true emitter identities. Therefore, as the number of emitters in the system continues to increase, the accuracy of the approach decreases.

More generally, the results in Figure 5.7 show the CNN feature extractors that have been trained on 5 and 15 emitters all show some ability to identify emitters outside their training ranges. However, while the ratio of emitters found to emitters trained generally increases as the number of emitters in the system increases, the AMI generally decreases. As described above, this limits the ability to identify emitters outside of the training set, as the drop in AMI indicates a drop in the accuracy of the assigned labels. Further, this trend indicates

the algorithm is likely over-clustering.

5.4.2 Transmissions Across Multiple Bandwidths

Given the success of the developed approach under the assumption that all received signals are at a single bandwidth, the feasibility of performing emitter identification across bandwidths using the developed approach was investigated. As such, the remaining results no longer assume all transmissions are at the same bandwidth. However, all transmissions still contain the same bits and use the same modulation scheme, as described in Section 5.1.1.

The testing accuracy for the CNN emitter identifier and feature extractors across all 11 bandwidths in the dataset are shown in Table 5.4. In comparison to those shown in Tables 5.3a and 5.3b, these testing accuracies are much higher, especially in the case of the CNN feature extractors trained on 20 and 25 emitters. This behavior is slightly unexpected given that the network has to generalize over bandwidth in this case, usually leading to a decrease in performance [46]. However, despite having more variation in the training data, training over all 11 bandwidths in the dataset also increased the amount of available training data by a factor of 11.

Further, Table 5.4 shows all CNN feature extractors trained across the 11 bandwidths provided in the dataset have achieved testing accuracies of well over 90%. As such, there is sufficient evidence that these networks have learned features relevant to emitter identity, just as there was when it was assumed all signals had the same bandwidth.

Number of Emitters	Testing Accuracy
2 emitters	1.0
5 emitters	0.9998
10 emitters	0.997
15 emitters	0.963
20 emitters	0.934
25 emitters	0.933

Table 5.4: The testing accuracies of each CNN feature extractor trained across all 11 bandwidths in the dataset.

However, when clustering is attempted using the DBSCAN algorithm, very interesting behavior emerges, shown in Figure 5.8. While the features are clustering by emitter, shown by the ability to detect two clusters using DBSCAN, they are also clustering according to another attribute. This additional attribute was shown to be bandwidth, as shown in Figure 5.9. The clustering of bandwidth in addition to emitter leads to the presence of 11 times as many clusters as there are emitters in the system. This makes clustering more than 2 emitters infeasible with the current approach, as shown in Figure 5.10.

In an effort to overcome this effect, each signal was resampled to the same effective bandwidth, to more closely resemble the operating conditions in Section 5.4.1, where it was assumed that all signals are at the same bandwidth. The testing accuracies of the CNN feature extractors, with all received signals resampled to the same effective bandwidth, are shown in Table 5.5. As expected, these accuracies are much higher than those shown in

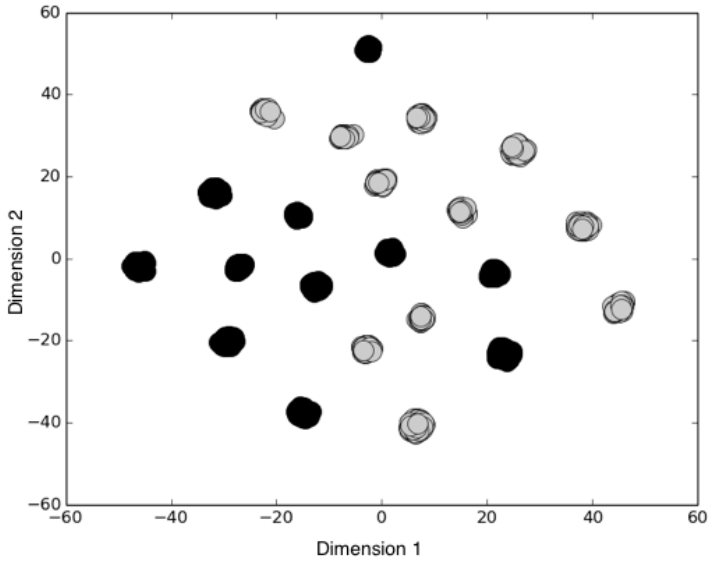


Figure 5.8: Clustering of the features learned by the CNN trained on 2 emitters across 11 bandwidths with 2 emitters in the system. 2 clusters (emitters) found by DBSCAN with an AMI of 1.0. Visualization of features embedded in 2 dimensions using t-SNE. Each color represents an emitter found by DBSCAN.

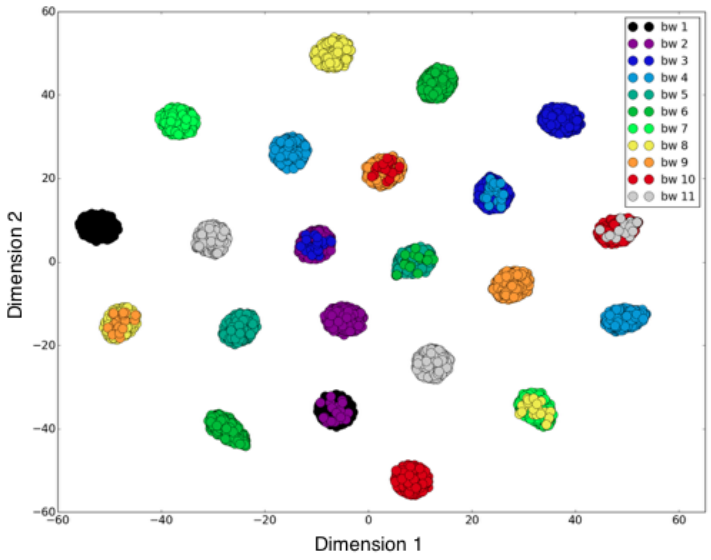


Figure 5.9: Clustering of the features learned by the CNN trained on 2 emitters across 11 bandwidths with 2 emitters in the system and each bandwidth labeled a different color.

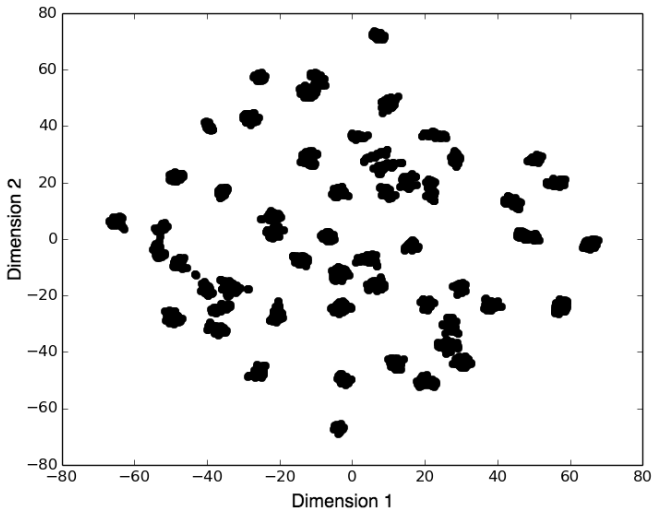


Figure 5.10: Attempted clustering of the features learned by the CNN trained on 5 emitters across 11 bandwidths with 5 emitters in the system. No clusters (emitters) found by DBSCAN. Visualization of features embedded in 2 dimensions using t-SNE.

Tables 5.3a and 5.3b, when all signals received were at the same bandwidth, as resampling provided 11 times the training data. More interestingly, the accuracies shown in Table 5.5 are slightly lower than those shown in Table 5.4, when the CNN was trained across all bandwidths. This indicates that the CNNs may be learning different emitter-specific features from signals at each bandwidth, aiding in it's decision making.

When clustering was attempted using the DBSCAN, the features continue to cluster by both emitter and bandwidth, despite resampling to the same effective bandwidth, as shown in Figures 5.11 and 5.12. This further indicates that CNNs may be learning different emitter-specific features from signals at different bandwidths, even if those signals came from the same emitter. Future work includes investigating this effect further, as it implies that the

Number of Emitters	Testing Accuracy
2	1.0
5	0.9986
10	0.9726
15	0.9546
20	0.9273
25	0.9030

Table 5.5: The testing accuracies of each CNN feature extractor when all received signals are resampled to the same effective bandwidth.

emitters may be impacting transmissions differently at each bandwidth.

5.5 Summary and Future Work

In this chapter, a semi-supervised approach was developed for performing specific emitter identification using CNNs to extract emitter-specific features from raw IQ data and the DBSCAN algorithm.

When it was assumed that all incoming signals have the same bandwidth, it was shown that emitter identification can be performed using the developed approach, even in the presence of emitters unseen in CNN training. Additionally, performance analysis showed performance increases as the number of emitters the CNN feature extractors are trained on increases. Though, only to a point, after which performance degrades, indicating the

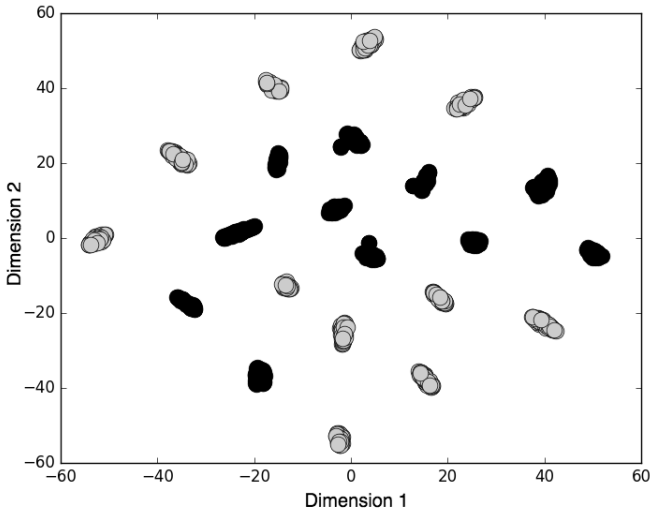


Figure 5.11: Clustering of the features learned by the CNN trained on 2 emitters with signals resampled to the same effective bandwidth and 2 emitters in the system. Visualization of features embedded in 2 dimensions using t-SNE. Each color represents an emitter.

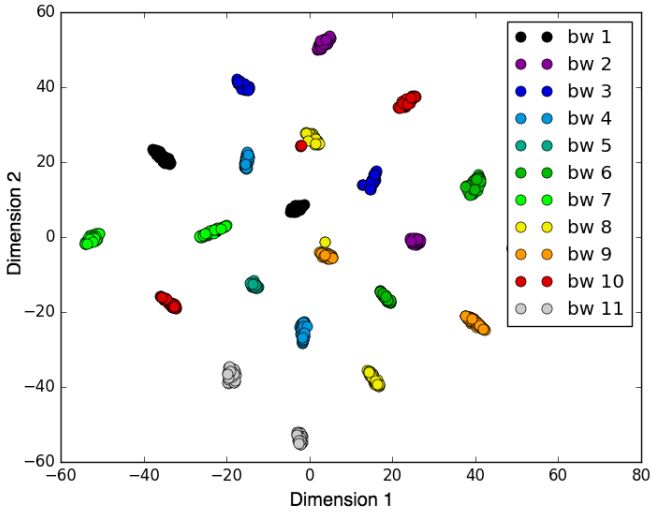


Figure 5.12: Clustering of the features learned by the CNN trained on 2 emitters with signals resampled to the same effective bandwidth, 2 emitters in the system, and each bandwidth labeled a different color.

designed CNN feature extractor is only capable of effectively describing a limited number of emitters, likely due to the small dataset size.

When it was no longer assumed all incoming signals have the same bandwidth, it was shown that the CNN extracted features clustered according to bandwidth as well as emitter identity, and therefore no more than two emitters could be identified. Additionally, the CNN extracted features clustered according to bandwidth as well as emitter identity when all received signals were resampled to the same effective bandwidth, indicating that the CNNs are likely learning different emitter-specific features from signals at different bandwidths, and that the emitters are impacting transmissions differently at each bandwidth.

As previously mentioned, future work includes examining the CNN-learned features at different bandwidths, in an effort to understand how an emitter's fingerprint changes as it changes transmission bandwidth. Future work also includes determining how to best modify the approach, to allow for the tracking of emitters as they change transmission bandwidth. Possible approaches include altering the feature extraction step by modifying the CNN architecture, using bi-clustering algorithms, or training a parallel architecture to identify or estimate the bandwidth of the incoming signal to be used during the clustering step of the approach. Additionally, alternative clustering algorithms should also be investigated because the output of the DBSCAN algorithm is extremely sensitive to the parameter which defines the maximum distance between two points in a cluster.

It is likely that the developed approach would also see performance increases given a larger training dataset. Therefore, additional data will be collected to allow for larger training

datasets, as well as to include more modulation schemes and wireless captures for further testing of the developed approach. With more training data, the scalability of the approach could also be examined, that is, whether increases in the number of inputs to the network or modifications to the network architecture would allow the CNN feature extractor to differentiate between more emitters.

Finally, the developed SEI approach has improved upon traditional SEI systems in the following ways:

- As in Chapter 4, the need for typically needed pre-processing steps has been eliminated by the use of raw IQ as input.
- The need for expert-defined features has been eliminated by allowing the CNN to learn emitter-specific features.
- The developed system is quickly and easily modified, requiring only appropriate training data.
- The developed approach is able to identify new or anomalous emitters, when traditional SEI approaches may have failed, because the DBSCAN clustering algorithm is able to identify the optimal number of clusters within the data.

Chapter 6

Conclusions

This work has reported on the development and evaluation of two methods by which to perform SEI using CNNs. In doing so, the work in this thesis has both examined the use of machine learning techniques to improve current SEI systems and the ability of CNNs to learn from raw IQ data for estimation and emitter identification tasks. Further, this work has studied the abilities of CNNs as feature learners, by examining the feasibility of using the features learned by a pre-trained CNN to differentiate between unseen emitters.

The first developed SEI approach utilizes the CNN IQ imbalance estimators described in Section 4.3. Performance analysis of the developed CNN IQ imbalance estimators showed that phase offset is far more difficult to estimate than phase offset. Additionally, it was shown that providing the network with more input samples (i.e. observing the signal for a longer period of time) does not necessarily increase the accuracy of the estimate the network produces, for the given architecture. However, the network does become more confident in

its estimate.

The first developed SEI approach, described in Section 4.4.1, uses parallel and modulation-specific gain offset estimators to identify emitters by their estimated gain offset. Due to the design, the performance of the first approach is directly affected by the sample variance of the CNN gain offset estimators. As a result, it was shown that the approach required gain offset values much higher than typically seen in a real RF system, when a large range of IQ imbalance values was used in training. However, the approach was shown to improve by narrowing the training range, outperforming a traditional feature-based approach which also uses IQ imbalance to identify emitters, while using less data and making fewer assumptions.

The second developed SEI approach is presented in Chapter 5 and improves upon the first proposed approach by allowing the CNN to learn the emitter-specific features, rather than constraining the network to learning a single expert-defined feature. The learned features are then clustered using the DBSCAN clustering algorithm. Performance analysis showed the ability to identify emitters both inside and outside of the training set, when all transmissions were assumed to be at the same bandwidth.

In light of the discussion in Chapters 1 and 2, the primary limitation of traditional SEI systems is the use of expert-defined features. Not only does the extraction of expert-defined features often require pre-processing of the raw data, slowing execution time of the system and stripping the raw data of potentially useful information, but the performance of the SEI system is heavily impacted by the accurate and consistent measurement of the features. Further, because only a few features are considered, the feature selection process is crucial

and leads to a restrictively long development process. The two approaches presented in this thesis have alleviated the effects described above in the following ways:

- Both approaches developed in this work use raw IQ data as input, eliminating the need for pre-processing steps typically needed when extracting expert-defined features and allowing the CNNs access to all of the information contained within the raw data.
- Both systems designed show the potential to be tuned or modified for a new operating environment by retraining the CNN(s), given the availability of appropriate training data, mitigating the lengthy system development process.
- The second approach uses CNNs to learn emitter-specific features, thereby eliminating the use of expert-features completely.

Finally, several areas have been highlighted as areas with potential for future work. First, both developed SEI approaches may be improved and/or extended, as discussed in Sections 4.5 and 5.5. Second, while this thesis explored the use of CNNs exclusively, RNNs have shown recent success when applied to raw data streams, and should be considered for future work, as they may aid in modeling temporal signal characteristics of emitters. Lastly, the appropriate choice or design of clustering algorithms for use with RF data will be investigated further.

Bibliography

[1] K. I. Talbot, P. R. Duley, and M. H. Hyatt, "Specific emitter identification and verification," *Tech. Rev.*, p. 113, 2003.

[2] K. Kim, C. M. Spooner, I. Akbar, and J. H. Reed, "Specific emitter identification for cognitive radio with application to IEEE 802.11," in *2008 IEEE Global Telecommunications Conf.*, Nov 2008, pp. 1–5.

[3] J. M. Park, J. H. Reed, A. A. Beex, T. C. Clancy, V. Kumar, and B. Bahrak, "Security and enforcement in spectrum sharing," *Proceedings of the IEEE*, vol. 102, no. 3, pp. 270–281, March 2014.

[4] S. Andrews, R. M. Gerdes, and M. Li, "Towards physical layer identification of cognitive radio devices," in *2017 IEEE Conf. on Comm. and Netw. Sec.*, Oct 2017, pp. 1–9.

[5] J. Matuszewski, "Specific emitter identification," in *2008 International Radar Symposium*, May 2008, pp. 1–4.

[6] E. Alpaydin, *Introduction to machine learning.* MIT press, 2014.

[7] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. The MIT Press, 2012.

[8] "IEEE machine learning for signal processing technical committee," Sep 2017. [Online]. Available: https://signalprocessingsociety.org/get-involved/machine-learning-signal-processing/mlsp-tc-home

[9] T. J. O'Shea, J. Corgan, and T. C. Clancy, *Convolutional Radio Modulation Recognition Networks*. Cham: Springer International Publishing, 2016, pp. 213–226.

[10] S. Peng, H. Jiang, H. Wang, H. Alwageed, and Y. D. Yao, "Modulation classification using convolutional neural network based deep learning model," in *2017 26th Wireless and Optical Communication Conference*, April 2017, pp. 1–5.

[11] M. Zhang, M. Diao, and L. Guo, "Convolutional neural networks for automatic cognitive radio waveform recognition," *IEEE Access*, vol. 5, pp. 11 074–11 082, 2017.

[12] Z. Puljiz, M. Park, and R. H. Jr., "A machine learning approach to link adaptation for SC-FDE system," in *2011 IEEE Global Telecommunications Conference*, Dec 2011, pp. 1–5.

[13] A. M. Mikaeil, B. Guo, and Z. Wang, "Machine learning to data fusion approach for cooperative spectrum sensing," in *2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Oct 2014, pp. 429–434.

[14] K. A. Remley, C. A. Grosvenor, R. T. Johnk, D. R. Novotny, P. D. Hale, M. D. McKinley, A. Karygiannis, and E. Antonakakis, "Electromagnetic signatures of WLAN cards and network security," in *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology, 2005.*, Dec 2005, pp. 484–488.

[15] K. J. Ellis and N. Serinken, "Characteristics of radio transmitter fingerprints," *Radio Science*, vol. 36, no. 4, pp. 585–597, 2001.

[16] C. Song, Y. Zhan, and L. Guo, "Specific emitter identification based on intrinsic time-scale decomposition," in *2010 6th Int. Conf. on Wireless Comm. Netw. and Mobile Comp.*, Sept 2010, pp. 1–4.

[17] A. Kawalec and R. Owczarek, "Specific emitter identification using intrapulse data," in *First European Radar Conference, 2004. EURAD.*, Oct 2004, pp. 249–252.

[18] T. L. Carroll, "A nonlinear dynamics method for signal identification," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 17, no. 2, p. 023109, 2007.

[19] C. Bertoncini, K. Rudd, B. Nousain, and M. Hinders, "Wavelet fingerprinting of radio-frequency identification (RFID) tags," *IEEE Trans. on Ind. Elec.*, vol. 59, no. 12, pp. 4843–4850, Dec 2012.

[20] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking.* New York, NY, USA: ACM, 2008, pp. 116–127.

[21] H. L. Yuan and A. Q. Hu, "Preamble-based detection of Wi-Fi transmitter RF fingerprints," *Electronics Letters*, vol. 46, no. 16, pp. 1165–1167, August 2010.

[22] K. W. Kim, "Exploiting cyclostationarity for radio environmental awareness in cognitive radios," Ph.D. dissertation, Virginia Polytechnic and State University, 2008.

[23] S. Xu, B. Huang, L. Xu, and Z. Xu, "Radio transmitter classification using a new method of stray features analysis combined with PCA," *2007 IEEE Military Communications Conference*, pp. 1–5, 2007.

[24] G. Huang, Y. Yuan, X. Wang, and Z. Huang, "Specific emitter identification based on nonlinear dynamical characteristics," *Canadian Journal of Elec. and Comp. Eng.*, vol. 39, no. 1, pp. 34–41, winter 2016.

[25] J. Matuszewski and K. Sikorska-ukasiewicz, "Neural network application for emitter identification," in *2017 18th Int. Radar Symp.*, June 2017, pp. 1–8.

[26] K. Merchant, S. Revay, G. Stantchev, and B. Nousain, "Deep learning for RF device fingerprinting in cognitive communication networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. PP, no. 99, pp. 1–1, 2018.

[27] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, Jun 1998.

[28] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec 1943.

[29] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, pp. 65–386, 1958.

[30] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015.

[31] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Transactions on Signal and Information Processing*, vol. 3, 2014.

[32] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation*, vol. 9, pp. 1735–1780, 1997.

[33] M. A. Nielson, *Neural Networks and Deep Learning*. Determination Press, 2015, http://neuralnetworksanddeeplearning.com/index.html.

[34] R. Hippenstiel, H. El-Kishky, C. Frick, and S. Dataprasad, "Modulation identification using neural networks and wavelet domain based approaches," in *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, vol. 2, Nov 2004, pp. 2116–2120 Vol.2.

[35] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, Mar. 1991.

[36] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 930–945, May 1993.

[37] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. Cambridge, MA, USA: MIT Press, 2014, pp. 2924–2932.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[39] Y. LeCun and Y. Bengio, "The handbook of brain theory and neural networks," M. A. Arbib, Ed. Cambridge, MA, USA: MIT Press, 1998, ch. Convolutional Networks for Images, Speech, and Time Series, pp. 255–258.

[40] J. Brownlee, "Crash course in recurrent neural networks for deep learning," Jul 2017. [Online]. Available: https://machinelearningmastery.com/crash-course-recurrent-neural-networks-deep-learning/

[41] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. JMLR.org, 2013, pp. III–1310–III–1318.

[42] M. Schmidt, D. Block, and U. Meier, "Wireless interference identification with convolutional neural networks," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, July 2017, pp. 180–185.

[43] X. Wang, X. Wang, and S. Mao, "Cifi: Deep convolutional neural networks for indoor localization with 5 GHz Wi-Fi," in *2017 IEEE International Conference on Communications*, May 2017, pp. 1–6.

[44] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed. McGraw-Hill, 2008.

[45] J. Kirkhorn, "Introduction to IQ-demodulation of RF-data," 1999.

[46] S. C. Hauser, W. C. Headley, and A. J. Michaels, "Signal detection effects on deep neural networks utilizing raw IQ for modulation classification," in *2017 IEEE Military Communications. Conference Proceedings*, vol. 1, 2017.

[47] T. J. O'Shea, N. West, M. Vondal, and T. C. Clancy, "Semi-supervised radio signal identification," in *2017 19th International Conference on Advanced Communication Technology*, Feb 2017, pp. 33–38.

[48] S. Chauhan and L. Vig, "Anomaly detection in ECG time signals via deep long short-term memory networks," in *2015 IEEE International Conference on Data Science and Advanced Analytics*, Oct 2015, pp. 1–7.

[49] Y. Guo, Z. Wu, and Y. Ji, "A hybrid deep representation learning model for time series classification and prediction," in *2017 3rd International Conference on Big Data Computing and Communications*, Aug 2017, pp. 226–231.

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition*, June 2015, pp. 1–9.

[51] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 2014, pp. 580–587.

[52] Y. Li, J. B. Huang, N. Ahuja, and M. H. Yang, "Deep joint image filtering," in *European Conference on Computer Vision*. Springer, 2016, pp. 154–169.

[53] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 818–833.

[54] A. Karpathy, "Visualizing what convnets learn." [Online]. Available: http://cs231n.github.io/understanding-cnn/

[55] C. N. dos Santos and M. A. de C. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *COLING*, 2014.

[56] S. Chakrabarty and E. A. P. Habets, "Broadband DOA estimation using convolutional neural networkstrained with noise signals," in *2017 IEEE Workshop on Applications of SignalProcessing to Audio and Acoustics*, Oct 2017, pp. 136–140.

[57] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[58] R. Collobert, S. Bengio, and J. Marithoz, "Torch: A modular machine learning software library," 2002.

[59] "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[60] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688

[61] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[62] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Unsupervised representation learning of structured radio communication signals," in *2016 First International Workshop on Sensing, Processing and Learning for Intelligent Machines*, July 2016, pp. 1–5.

[63] J. Brownlee, "How much training data is required for machine learning?" Jul 2017. [Online]. Available: https://machinelearningmastery.com/much-training-data-required-machine-learning/

[64] M. McCloskey and N. J. Cohen, "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem," ser. Psychology of Learning and Motivation, G. H. Bower, Ed. Academic Press, 1989, vol. 24, pp. 109–165.

[65] T. O'Shea and N. West, "Radio machine learning dataset generation with gnu radio," *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, 2016.

[66] B. Clark, "Efficient waveform spectrum aggregation for algorithm verification and validation," Sep 2016. [Online]. Available: https://gnuradio.org/grcon-2016/talks/

[67] L. Angrisani, M. D'Arco, and M. Vadursi, "Clustering-based method for detecting and evaluating I/Q impairments in radio-frequency digital transmitters," *IEEE Transactions on Instrumentation and Measurement*, vol. 56, no. 6, pp. 2139–2146, Dec 2007.

[68] Y. Li, *Frequency Independent IQ Imbalance Estimation and Compensation*. New York, NY: Springer New York, 2014, pp. 29–47.

[69] D. L. N. S. Inti, "Time-varying frequency selective IQ imbalance estimation and compensation," Master's thesis, Virginia Polytechnic and State University, 2017.

[70] C. J. Hsu and W. H. Sheen, "Joint calibration of transmitter and receiver impairments in direct-conversion radio architecture," *IEEE Transactions on Wireless Communications*, vol. 11, no. 2, pp. 832–841, February 2012.

[71] C.-L. Liu, "Impacts of I/Q imbalance on QPSK-OFDM-QAM detection," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 984–989, Aug 1998.

[72] F. E. Churchill, G. W. Ogar, and B. J. Thompson, "The correction of I and Q errors in a coherent processor," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-17, no. 1, pp. 131–137, Jan 1981.

[73] S. A. Bassam, S. Boumaiza, and F. M. Ghannouchi, "Block-wise estimation of and compensation for I/Q imbalance in direct-conversion transmitters," *IEEE Transactions on Signal Processing*, vol. 57, no. 12, pp. 4970–4973, Dec 2009.

[74] D. S. Hilborn, S. P. Stapleton, and J. K. Cavers, "An adaptive direct conversion transmitter," *IEEE Transactions on Vehicular Technology*, vol. 43, no. 2, pp. 223–233, May 1994.

[75] S. Burglechner, G. Hueber, and A. Springer, "On the estimation and compensation of IQ impairments in direct conversion transmitters," in *2008 European Conference on Wireless Technology*, Oct 2008, pp. 69–72.

[76] L. Anttila, M. Valkama, and M. Renfors, "Blind moment estimation techniques for I/Q imbalance compensation in quadrature receivers," in *2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, Sept 2006, pp. 1–5.

[77] R. A. Green, R. Anderson-Sprecher, and J. W. Pierre, "Quadrature receiver mismatch calibration," *IEEE Transactions on Signal Processing*, vol. 47, no. 11, pp. 3130–3133, Nov 1999.

[78] W. Li, Y. Zhang, J. Wang, L. k. Huang, J. Xiong, and C. Maple, "Diode-based IQ imbalance estimation in direct conversion transmitters," *Electronics Letters*, vol. 50, no. 5, pp. 409–411, Feb 2014.

[79] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *2009 IEEE 12th International Conference on Computer Vision*, Sept 2009, pp. 2146–2153.

[80] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," 2012.

[81] L. Patryla and D. Galeriua, "Statistical performances measuresmodels comparison," 2011.

[82] G. W. Brown, "On small-sample estimation," *Ann. Math. Statist.*, vol. 18, no. 4, pp. 582–585, 12 1947.

[83] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed ¡today¿]. [Online]. Available: http://www.scipy.org/

[84] K. Pearson, *On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling.* New York, NY: Springer New York, 1992, pp. 11–28.

[85] P. E. Greenwood and N. M. Stepanovic., *A guide to chi-squared testing.* Wiley, 1996.

[86] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[87] M. Song, C. Xin, Y. Zhao, and X. Cheng, "Dynamic spectrum access: from cognitive radio to network radio," *IEEE Wireless Communications*, vol. 19, no. 1, pp. 23–29, February 2012.

[88] F. Zhuo, Y. Huang, and J. Chen, "Radio Frequency Fingerprint Extraction of Radio Emitter Based on I/Q Imbalance," *Procedia Computer Science*, vol. 107, pp. 472–477, 2017.

[89] M. Ester, H. peter Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." AAAI Press, 1996, pp. 226–231.

[90] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[91] D. Arthur and S. Vassilvitskii, "K-means++: the advantages of careful seeding," in *In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[92] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: A new data clustering algorithm and its applications," *Data Mining and Knowledge Discovery*, vol. 1, no. 2, pp. 141–182, Jun 1997.

[93] B. J Frey and D. Dueck, "Clustering by passing messages between data points," vol. 315, pp. 972–6, March 2007.

[94] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, Dec. 2010.