

Deep Learning Features in RF Signal Classification

Rachneet Singh Sachdeva

Master Thesis

October 9, 2020

Examiners

Prof. Dr. Petri Mähönen
Prof. Dr.-Ing. Marina Petrova

Supervisors

Prof. Dr. Petri Mähönen
Dr. Ljiljana Simić

Institute for Networked Systems
RWTH Aachen University



The present work was submitted to the Institute for Networked Systems

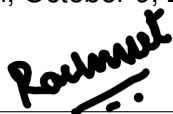
Deep Learning Features in RF Signal Classification

Master Thesis

presented by
Rachneet Singh Sachdeva

Prof. Dr. Petri Mähönen
Prof. Dr.-Ing. Marina Petrova

Aachen, October 9, 2020

A handwritten signature in black ink, appearing to read 'Rachneet', written over a horizontal line.

(Rachneet Singh Sachdeva)

ACKNOWLEDGEMENTS

I would like to thank Prof. Dr. Petri Mähönen and Dr. Ljiljana Simić for giving me the opportunity to work on this thesis. I sincerely appreciate their time, motivation and expert guidance during the course of this thesis. I am deeply grateful for their clear and concise advice and feedback on my work. I would also like to thank the entire staff of iNETS for providing a flexible and comfortable working environment, both from home and office during these testing times. Moreover, I would like to extend my regards to my family and friends for their constant support and encouragement.

All computations were performed using the resources provided by RWTH Aachen University.

CONTENTS

ACKNOWLEDGEMENTS	II
CONTENTS	III
ABSTRACT	VI
1 INTRODUCTION	1
2 BACKGROUND	3
2.1 RF SIGNALS FOR WIRELESS COMMUNICATIONS	3
2.1.1 BASEBAND SIGNAL GENERATION	3
2.1.2 SIGNAL TRANSMISSION	8
2.1.3 SIGNAL PROPAGATION	10
2.1.4 SIGNAL RECEPTION	10
2.2 MACHINE LEARNING FOR CLASSIFICATION	11
2.2.1 CLASSIFICATION ALGORITHMS	12
2.2.2 ALGORITHMS FOR EXPLORATORY ANALYSIS	23
2.2.3 TRANSFER LEARNING	25
2.2.4 MODEL INTERPRETATION	27
2.2.5 OPEN SET CLASSIFICATION	30
3 LITERATURE REVIEW AND STATE OF ART	33
4 METHODOLOGY	37
4.1 AVAILABLE RF SIGNAL DATASET	37
4.2 FEATURIZATION	39
4.2.1 NOMENCLATURE	39
4.2.2 FEATURE DEFINITIONS	42
4.3 DATA PRE-PROCESSING	48
4.4 EMPLOYED MACHINE LEARNING ALGORITHMS	48
4.5 EVALUATION PROCEDURE	50
4.5.1 EXPLORATORY DATA ANALYSIS	51
4.5.2 EVALUATION OF OVERALL CLASSIFICATION ACCURACY	52
4.5.3 EVALUATION FOR COMPARATIVE ANALYSIS	52
4.5.4 EVALUATION OF GENERALIZATION ACCURACY	52
4.5.5 EVALUATION USING TRANSFER LEARNING	53

4.5.6	SEQUENTIAL TESTING	53
4.5.7	EVALUATION OF FEATURE IMPORTANCE	54
4.5.8	EVALUATION OVER MULTIPLE DATASETS	54
4.6	EVALUATION SCENARIOS	55
4.6.1	BASLINE SCENARIO	55
4.6.2	SENSITIVITY ANALYSIS	55
4.6.3	INTERFERENCE SCENARIO	57
4.6.4	REAL-WORLD SCENARIO	57
5	RESULTS	58
5.1	BASLINE SCENARIO	58
5.1.1	EXPLORATORY DATA ANALYSIS	58
5.1.2	EVALUATION OF MODEL PERFORMANCE	59
5.2	SENSITIVITY ANALYSIS	61
5.2.1	EFFECT OF NOISE	62
5.2.2	EFFECT OF HARDWARE HETEROGENEITY	64
5.2.3	EFFECT OF CARRIER FREQUENCY OFFSET	67
5.2.4	EFFECT OF LENGTH OF SIGNAL SEGMENTS	69
5.2.5	EFFECT OF SIZE OF TRAINING DATA	71
5.2.6	EFFECT OF UNKNOWN SIGNALS	71
5.3	INTERFERENCE SCENARIO	72
5.4	REAL-WORLD SCENARIO	76
5.5	SEQUENTIAL TESTING	77
5.6	EXPLAINING MODEL PREDICTIONS	79
5.6.1	FEATURE-BASED CLASSIFICATION	79
5.6.2	MODEL INTERPRETATION	80
5.7	EVALUATION OVER MULTIPLE DATASETS	85
6	CONCLUSIONS AND FUTURE WORK	88
A	APPENDIX	90
A.1	DATASET FILE FORMAT	90
A.1.1	FILE STRUCTURE	90
A.1.2	ACCESSING THE DATA	90
A.2	AUXILIARY RESULTS	93
A.2.1	EVALUATION USING DEEP NEURAL NETWORK (DNN)	93
A.2.2	EVALUATION FOR FOUR CLASS MODULATION	93
A.2.3	XGBOOST PERFORMANCE USING LESSER SAMPLES FOR FEAT- TURIZATION	94
B	ABBREVIATIONS	98
	LIST OF TABLES	101

CONTENTS	V
LIST OF FIGURES	102
BIBLIOGRAPHY	105
DECLARATION	112

ABSTRACT

Emerging mobile network technologies such as 5G promise high data rates with increased reliability and low latency. However, the implementation of these technologies requires efficient spectrum management and utilization. Due to an exponential increase in the number of connected devices over the past decade, the spectrum has become a scarce and expensive resource. To increase spectral efficiency, cognitive radios have generated keen interest. To enhance the performance of these devices for signal detection, real-time modulation recognition tasks have to be performed.

In this thesis, we explore deep learning algorithms for Automatic Modulation Classification (AMC) tasks using raw signal data in the form of I/Q samples. A comprehensive analysis of the performance of algorithms under the influence of noise, hardware heterogeneity, Carrier Frequency Offset (CFO), and modulated co-channel interference is performed. Further, a method to explain model predictions is also discussed.

The results indicate that deep learning algorithms perform well for Signal-to-Noise Ratio (SNR) as low as 5 dB. Below that, the accuracy begins to deteriorate. However, if we can gather a large amount of training data, the accuracy can be improved further. Moreover, our models can distinguish signal modulations under challenging practical scenarios with interference, CFOs, and random noise. The further exhaustive analysis shows that models trained on data recorded in a certain environment do not generalize well for all environments one may encounter in a practical scenario. Nevertheless, we show that the concept of Transfer Learning (TL) can be used to fine-tune models and hugely enhance the accuracy of the model in a real-world scenario.

Our work has three main consequences concerning AMC tasks. Firstly, we deduce that deep learning algorithms can produce comparable results to models trained using handcrafted features. Thus, these models can be deployed in practical wireless systems with greater ease and without much human intervention and expertise. Secondly, we observe that these models can be fine-tuned for varying environments without compromising model accuracy. Thirdly, we identify the features important for the model predictions.

INTRODUCTION

The rapid evolution of the wireless communication system has put the spectrum resources under immense pressure. Upcoming 5G technology provides an opportunity for massive Internet of Things (IoT). However, the number of connected devices has been increasing exponentially over the past decade. Projections by Cisco [1] report that Machine-to-Machine (M2M) connections will amount to half of the globally connected devices by 2023. Therefore, there is an immediate need for efficient spectrum management. To this end, spectrum aware radios have shown huge potential. AMC is an important task for these devices as it aids in monitoring spectrum interference, detecting signal distortion [2], and allows for opportunistic routing among nodes. Due to this, it has been adopted for use in military applications.

Approaches used for Modulation Recognition (MR) can be classified into three groups: Likelihood-based (LB) [3–8], Feature-based (FB) [9–16] and Artificial Neural Network (ANN) [17–21] based. LB approach is formulated as a hypothesis-testing problem wherein the likelihood ratio of each possible hypothesis is compared against a threshold that is derived from the probability density function of the signal under observation. LB approach has high computational complexity and works only for limited modulation types. The FB approach focuses on extracting spectral and statistical features from the received signal samples. These featurized signals are then classified via decision-theoretic approaches or input to Machine Learning (ML) algorithms to identify modulation types. FB approaches lead to decent performance but require human expertise to extract the optimal set of features for a particular environment. This approach can also be very time consuming and thus less suitable for practical use. The ANN-based approach makes use of a Multi Layer Perceptron (MLP) [22] where each layer provides automatic and adaptive learning to classify signals based on the input features. ANN-based approaches have performed well on simple MR tasks but the advent of Deep Learning (DL) architectures have paved way for complex MR in real-world scenarios.

Over the past few years, DL algorithms have gained popularity for AMC due to their ability to capture complex features without human intervention and produce at par or even better accuracy in comparison to FB approaches. Convolutional Neural Networks (CNNs) [23] have been successfully used for AMC tasks in a realistic wireless propagation environment [24].

In this thesis, our goal is to study DL algorithms for AMC tasks using raw Over-the-Air (OTA) signal data in the form of I/Q samples. To this end, we develop our custom CNN and evaluate its performance in a realistic wireless propagation environment. We take into account the effects of hardware heterogeneity, carrier frequency offsets, modulated interference, and random noise. Our results show that the perfor-

mance of CNN degrades in an interference environment. To mitigate this, we use Residual Networks (ResNets) [25] which improves the classification accuracy of our model by a substantial margin. Generally, these models do not generalize well for varying environments. However, our results show that these models can be fine-tuned to produce good accuracy using the concept of TL [26]. For example, a model trained with signals recorded in an interference-free environment can be fine-tuned for use in an interfering environment just by using a subset of the recorded interfering signals. This approach has two potential advantages: first, we save a huge amount of training time and second, this works well in cases where large training sets are not available.

The rest of the thesis is structured as follows. Chapter 2 gives information about RF signals for the wireless communication system. Further, we describe ML and DL algorithms leveraged for the task of AMC. Chapter 3 presents an extensive review of the prior work based on our study. In Chapter 4, we first detail our approach of creating, featurizing, and processing the signal datasets. Then, we describe the implementation details of the employed learning algorithms along with the simulation environment, simulation parameters, and scenarios. Chapter 5 presents and analyses the results of our simulations in the realistic wireless propagation environment. Finally, Chapter 6 presents the conclusions, future research prospects, and limitations of our work.

BACKGROUND

This chapter presents the relevant theoretical concepts to familiarize with the task of modulation classification. Section 2.1 provides information about the generation, transmission, and reception of RF signals in a wireless communication system. Section 2.2 describes in detail the machine learning and deep learning algorithms leveraged for the AMC task.

2.1 RF SIGNALS FOR WIRELESS COMMUNICATIONS

This section explains the fundamentals of an end to end wireless communication system. Owing to the robust and flexible nature of *digitally* modulated signals, and given their adoption by most modern radios, we will only be considering these for our work. The following subsections describe in detail the process of signal generation, transmission, and reception in a wireless propagation environment.

2.1.1 Baseband Signal Generation

Signals having near-zero frequencies are called Baseband (BB) signals. The information in these signals starts at low frequencies. For transmission, this information is modulated to a higher frequency range so that it can be put in the spectrum. These modulated signals are called Passband (PB) signals. Signals can be modulated using a Single Carrier (SC) or Orthogonal Frequency Division Multiplexing (OFDM) scheme. In SC modulation, the data is sent in a serial form whereas, in OFDM, the data can be clubbed together and sent in parallel. The following subsections explain the process of SC and OFDM modulation techniques for the generation of transmission signals.

2.1.1.1 Single Carrier Modulation

Fig. 2.1 shows a typical block diagram of a SC modulator for the generation of BB signals. The block diagram depicts the process of modulating raw data bits (x_i) to time domain in-phase $I(t)$ and quadrature $Q(t)$ BB signals for transmission over a wireless channel. The following paragraphs explain the role of each processing block in the signal generation process.

Digital Data Source. The information generated by transmitting sources is generally analog in nature. This can be in the form of audio, video, or control signals. These signals are input to the digital data sources which outputs a series of bits (x_i) with the least amount of redundancy.

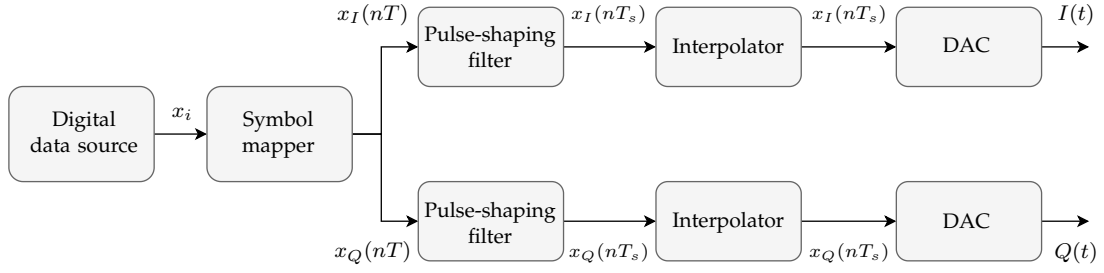


Figure 2.1: Block diagram of SC modulator.

Symbol Mapper. The process of mapping information bits (x_i) to a complex signal $x(nT) \in \mathbb{C}$, $n \in \mathbb{N}$ is called *modulation*. The information bits are grouped and mapped to a symbol in the complex plane based on the modulation order M , of the modulation scheme. The number of bits per symbol can be calculated using the formula $\log_2 M$ where M depicts the total number of symbols. These symbols are concatenated in a pulse train with an interval of one *symbol time* T to form the discrete-time signal

$$x(nT) = \sum_k x_k \delta(n - kT), \quad (2.1)$$

where $x(nT)$ is composed of the in-phase and quadrature component $x_I(nT)$ and $x_Q(nT)$, respectively.

Fig. 2.2 illustrates the mapping of bits to symbols for variants of Phase Shift Keying (PSK) and Quadrature Amplitude Modulation (QAM) modulation schemes in the normalized I/Q plane. Higher modulation schemes can convey more information per symbol which results in a higher data rate. But as M increases, the symbols are placed more closely which makes them less robust against noise.

Pulse-Shaping Filter. The purpose of using a pulse shaping filter is two-fold: first, it reduces the channel bandwidth, and second, it reduces Intersymbol Interference (ISI) caused due to multi-path effects of the channel. The modulated output from the symbol mapper has sharp transitions between the subsequent symbols. Due to this, the signal bandwidth exceeds the channel bandwidth leading to inter-channel interference. Pulse shaping filter mitigates this problem by smoothing transitions between subsequent symbols and limiting signal power to a specific bandwidth.

Another problem posed by channel multi-path effects causes the symbols in signal to spread beyond their time interval and interfere with previous and subsequent symbols. This phenomenon is called ISI. Application of pulse shaping filter to the symbols can reduce channel bandwidth, hence reducing the ISI. According to the *Nyquist criterion for zero ISI*, the channel impulse response must be zero for all sampling instants except the desired one

$$h(kT) = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases}. \quad (2.2)$$

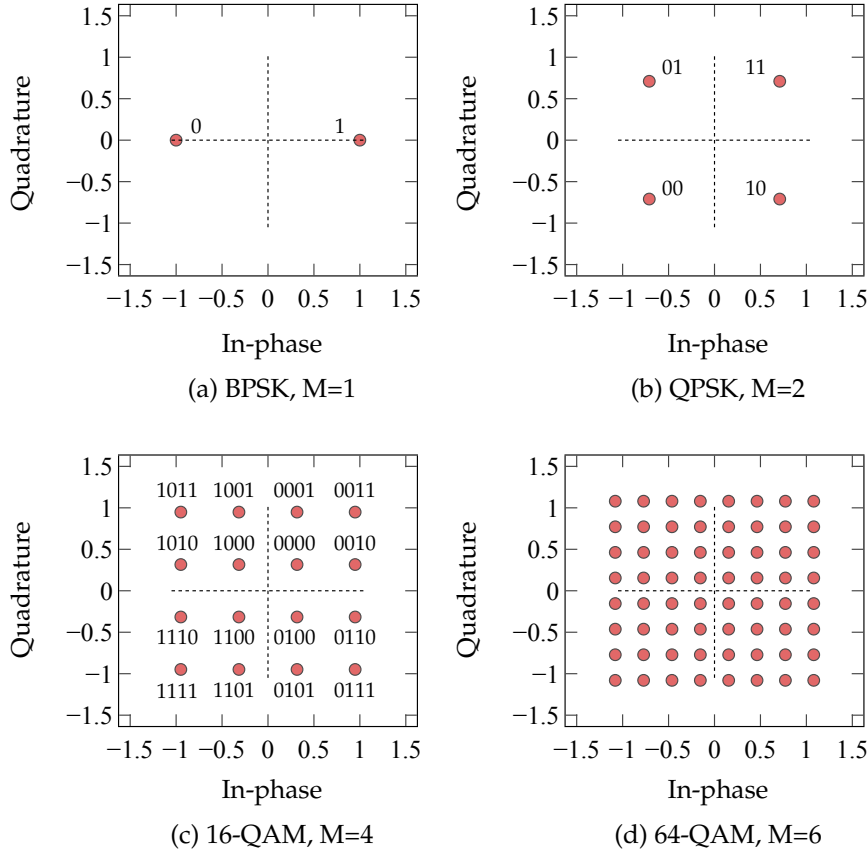


Figure 2.2: Constellation diagrams of SC modulation schemes of different modulation orders along with their bit mapping sequences where M denotes the number of bits per symbol¹.

The most commonly used pulse-shaping filter in the communications system is a raised-cosine (RC) filter. This is because it satisfies the Nyquist criterion stated in Equation 2.2. The roll-off factor α of the RC filter decides the steepness of filter in the frequency domain and the level at which the filter impulse response decreases in the time domain, respectively. The value of α can be varied between $[0,1]$ and gives a compromise between the bandwidth consumption and complexity of the filter design to prevent ISI. The impulse response of an RC filter can be defined by the equation

$$h_{RC}(t) = \begin{cases} 1 & t = 0 \\ \frac{\alpha}{2} \sin\left(\frac{\pi}{2\alpha}\right) & t = \pm \frac{T}{2\alpha} \\ \left(\frac{\sin \frac{\pi t}{T}}{\frac{\pi t}{T}}\right) \times \left(\frac{\cos \frac{\pi \alpha t}{T}}{1 - (\frac{2\alpha t}{T})^2}\right) & \text{otherwise} \end{cases} \quad (2.3)$$

¹Note: Bit mapping sequence for 64-QAM is not shown due to space constraints

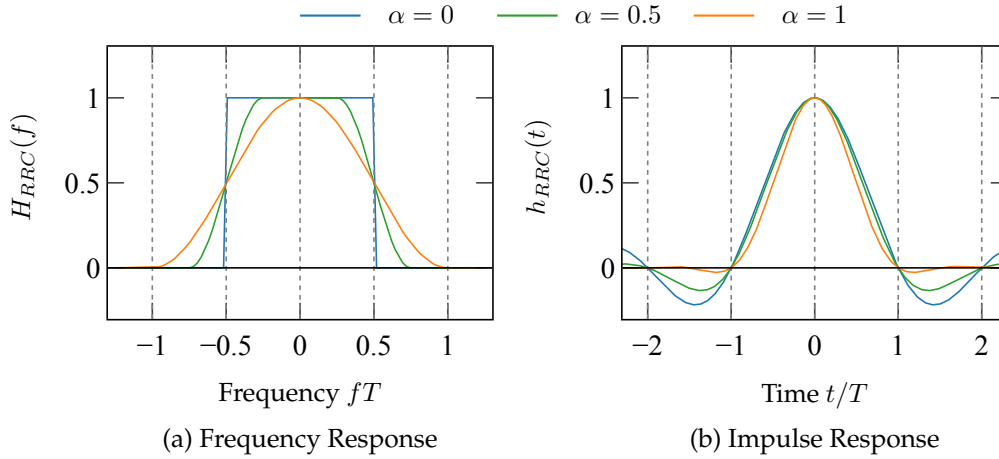


Figure 2.3: Filter responses of a RC filter with different roll-off factors α .

Fig. 2.3 shows frequency and time domain response of a RC filter. It is evident from the time domain response of the filter that the impulse response equals 0 at multiples of sampling intervals T and thus helps prevent ISI.

Interpolator. The digital BB signal has to be converted to analog form using a Digital-to-Analog Converter (DAC). The output of DAC consists of undesired images along with the desired analog signal. These images are filtered with the help of reconstruction filters. Due to sharp transitions between frequency bands, a higher-order analog filter is required. The implementation of such a filter is very complex and expensive. Thus, we need to interpolate the signal before it is input to DAC. Interpolation consists of oversampling the signal to increase its sampling rate so that the image frequencies at the DAC output are far from the desired signal frequency and hence the signal can be reconstructed using a lower order analog filter. The oversampled signal can be depicted by

$$x(nT_s) = \sum_l x_l \delta(n - lT_s) \quad ; \quad \frac{1}{T_s} \geq \frac{1}{T} \quad (2.4)$$

This resultant signal is input to the DAC and the corresponding BB signal is generated and ready to be transmitted over a wireless channel.

2.1.1.2 Orthogonal Frequency Division Multiplexing

OFDM is used to transmit high data rates with narrowband signals. It uses multiple orthogonal subcarriers, each having a lower data rate, to transmit data in parallel. Thus, the overall data rate of OFDM signal is similar to that of a wideband signal while being less prone to interference. Fig. 2.4 shows the process of OFDM signal generation for transmission in a wireless channel. The purpose of each block is explained in the following paragraphs.

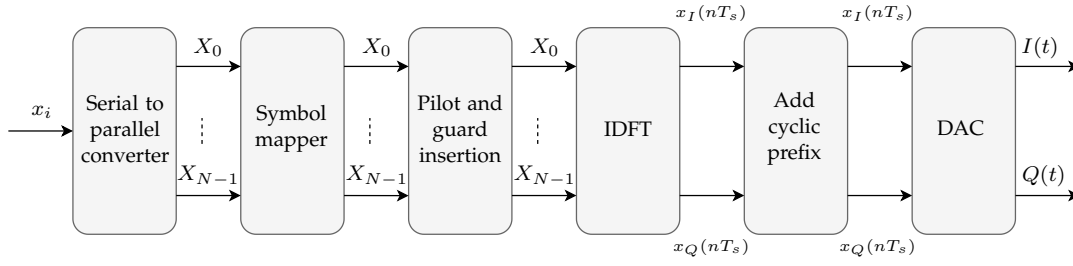


Figure 2.4: Block diagram of OFDM signal generator.

Data Source. We consider the generation of a continuous serial bitstream x_i by a memoryless stochastic data source. This serial data is converted to N different parallel substreams using a serial-to-parallel converter. Each substream represents a sub-carrier of the OFDM signal. Each subcarrier is processed independently and combined in the end to form an OFDM signal.

Symbol Mapper. The role of symbol mapper is to map the individual sub-carriers to complex symbols X_k in the I/Q plane. Each sub-carrier can be modulated via a different modulation scheme based on the channel conditions.

Pilot and Guard Insertion. Since the wireless communication channel is frequency selective and time-varying, dynamic channel estimation is important. This is done by inserting pilot symbols in the OFDM signal. This can be accomplished in two ways, *block-type* or *comb-type* pilot arrangement. In the block-type arrangement, pilot tones are inserted to all the sub-carriers of an OFDM symbol at specific periods whereas, in the comb-type arrangement, pilot tones are uniformly inserted to each OFDM symbol. A block-type pilot arrangement is used when we have a slow fading channel and a comb-type pilot arrangement is used for a fast fading channel. Fig. 2.5 shows the comb-type pilot arrangement technique which is frequently employed.

Additionally, guard bands or guard intervals are added between symbols of individual channels to prevent Inter-channel Interference (ICI).

Inverse Discrete Fourier Transform (IDFT). The complex symbols X_k are modulated by IDFT and converted to a serial stream for further processing. The equation of an N -point IDFT can be given as shown

$$\begin{aligned}
 x(nT_s) &= \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{j2\pi k \Delta f (nT_s)} \\
 &= \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{j2\pi kn/N},
 \end{aligned} \tag{2.5}$$

where choosing $\Delta f T_s = 1/N$ leads to orthogonality between any two sub-carriers of an OFDM symbol. Orthogonality between two sub-carriers at $k_1 \Delta f$ and $k_2 \Delta f$ can be defined by the following equation

$$\frac{1}{N} \sum_{n=0}^{N-1} X_{k_1} e^{j2\pi k_1 n/N} X_{k_2}^* e^{-j2\pi k_2 n/N} = \begin{cases} 0 & k_1 \neq k_2 \\ 1 & k_1 = k_2 \end{cases}. \quad (2.6)$$

Usually, the number of sub-carriers N is chosen to be in the power of 2 because the IDFT can be efficiently calculated using Fast Fourier Transform (FFT).

Cyclic Prefix. The wireless transmission channel is not ideal. Due to the multipath and dispersive effects of the channel, delayed versions of previous OFDM symbols can overlap with the incoming symbols causing ISI. Moreover, channel variations can also cause interference between different carriers of an OFDM symbol leading to Inter-carrier Interference (ICI). To mitigate these problems, the last few samples of each symbol are appended to the starting of the symbol. This provides with a guard between the symbols and alleviates the aforementioned issues.

This is the last step in the generation of an OFDM baseband signal. Fig. 2.5 shows the typical arrangement of an OFDM signal in the frequency-time domain. Successive steps for signal upconversion and transmission will be discussed in the following sections.

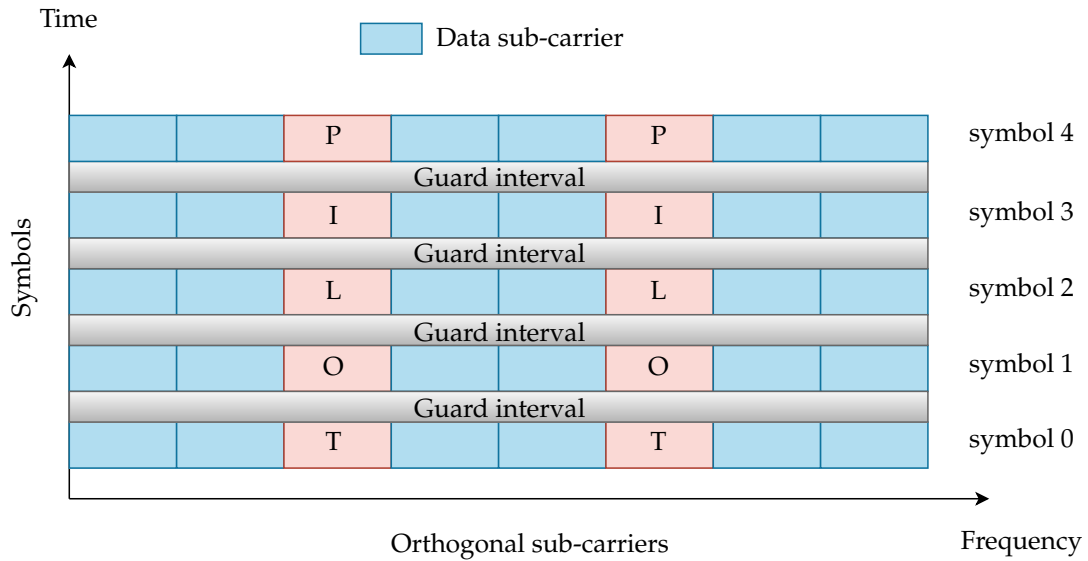


Figure 2.5: Frequency-Time representation of an OFDM signal.

2.1.2 Signal Transmission

The modulated BB signal has to be filtered, up-converted, and amplified before it can be transmitted to the wireless channel. These tasks are performed by the analog part

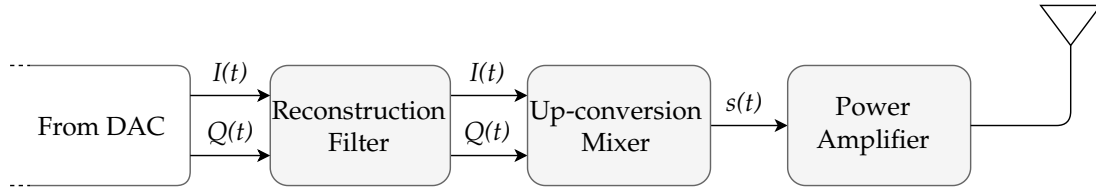


Figure 2.6: Typical block diagram of the RF transmit chain.

of the transmitter chain shown in Fig. 2.6. All the modulated BB signals go through the same processing irrespective of the modulation type, SC, or OFDM. The following paragraphs present brief descriptions of the various blocks of this processing chain.

The DAC outputs the desired frequency signal along with undesired image frequencies at multiples of the sampling frequency. These unwanted images are filtered out using a low-pass filter called the *reconstruction* or *anti-alias* filter. The image frequencies are aliases of the desired frequency which needs to be reconstructed, hence the name.

The reconstructed signal is then up-converted to get the final RF signal by mixing the $I(t)$ and $Q(t)$ BB signals with the sinusoidal components of the Local Oscillator (LO). The LO frequency is already set to the desired RF carrier frequency (f_c) in the simplified case (homodyne transmitter) and no intermediate frequency (IF) is required. The LO carrier signal can be denoted by

$$c(t) = e^{2\pi f_c t} = \cos(2\pi f_c t) + j \sin(2\pi f_c t) \quad (2.7)$$

and the resulting bandpass RF signal can be given by

$$s(t) = I(t) \cos(2\pi f_c t) + Q(t) \sin(2\pi f_c t). \quad (2.8)$$

Due to the simplicity and cost-effectiveness of the direct-conversion transmitter, it is widely used but is also error-prone. The architecture suffers from LO phase noise, DC offsets and gain imbalance due to the non-linearities of the analog components. These can be overcome by up-converting the analog signal to an IF frequency which can then be converted to an RF frequency using a series of mixers [27].

The last stage in the transmit chain comprises the power amplifier which serves the purpose of merely increasing the signal strength of a low-power RF signal to the desired power level to drive the antenna and compensate for the path loss along the wireless channel. The amplified signal is then optionally filtered to remove the harmonics and then fed into the antenna using an impedance-matching network for getting maximum power transfer. Finally, the antenna is responsible for transmitting the signal over the air.

2.1.3 Signal Propagation

The real-world wireless channel is not ideal and distorts the transmitted signals. Due to the presence of diverse structures between the transmitter and receiver, there is no line-of-sight (LOS) path between them. This leads to signals being reflected and taking varying paths and times before reaching the receiver where they sum up to form a signal having distortions in amplitude and phase. This phenomenon is called *multipath fading*. Moreover, when the transmitter is not stationary, the multipath signals undergo varying *doppler shifts* that further deteriorate the signal quality. Considering stationary sources and a time-varying multipath channel, the channel can be characterized by an impulse response $h(t, T)$ and the received signal $r(t)$ can be computed by convoluting the transmitted signal $s(t)$ with the channel impulse response and is given as

$$r(t) = s(t) \otimes h(t, T), \quad (2.9)$$

where $h(t, T)$ is defined as

$$\begin{aligned} h(t, T) &= \sum_{i=0}^{N-1} a_i(t, T) e^{(j2\pi f_c T_i(t) + \phi_i(t, T))} \delta(T - T_i(t)) \\ &= \sum_{i=0}^{N-1} a_i(t, T) e^{j\theta_i(t, T)} \delta(T - T_i(t)), \end{aligned} \quad (2.10)$$

where $a_i(t, T)$, $T_i(t)$, and $\theta_i(t, T)$ denote the amplitude, excess delay, and phase shift due to free space propagation of the i^{th} multipath component at time t . Note that the Equation 2.10 does not take into account the *path loss* component. The received signal power P_r usually decays as a power law of the distance d between the transmitter and the receiver and can be approximated as

$$P_r \propto d^{-n}, \quad (2.11)$$

where n is the path loss component and varies between the value of 2-4 for cellular systems. For real-world signals, this value can be even higher owing to the diffraction, absorption, and multipath losses in the wireless channel. For a further in-depth review of propagation characteristics, we refer the reader to [28].

2.1.4 Signal Reception

The RF receiver chain follows similar steps as the transmit chain in the reverse order. Fig. 2.7 shows a typical block diagram of a RF receiver. The received signal is attenuated due to imperfections of the channel as described previously. Thus, the signal needs to be amplified for further processing at the receiver. For this task, usually, a Low Noise Amplifier (LNA) with a low noise figure is used so that high SNR can be maintained. Sometimes a bandpass filter (not shown in Fig. 2.7) is also used between the antenna and LNA to select the signal in the desired frequency range.

The next step is to down-convert the amplified RF signal to an intermediate frequency (heterodyne receiver) or directly to baseband (direct-conversion receiver). In the heterodyne receiver, the RF signal is down-converted to a low IF for easier pro-

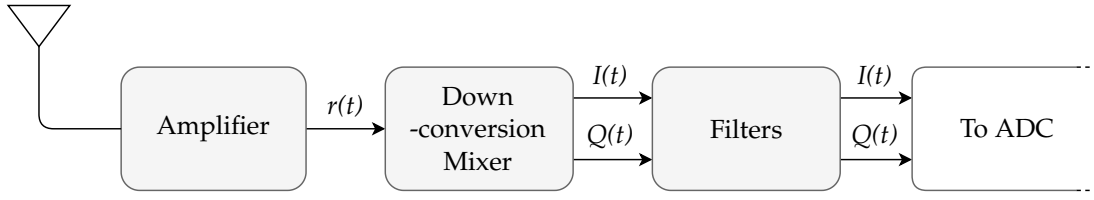


Figure 2.7: Typical block diagram of the RF receive chain.

cessing, but this leads to the so-called effect of *image*. Any spectra lying symmetrically around the LO frequency is converted to the same IF leading to interference between the desired signal and the unwanted signal called the image. This problem can be mitigated by the use of a direct-conversion receiver in which the LO frequency is set to the desired center frequency, hence avoiding the issue of image. But the direct-conversion receiver is prone to the problem of *LO leakage*. Due to its high frequency, the LO signal leaks to other parts of the circuit. If it appears at the input of the LNA, it is mixed with the RF signal and amplified. When this impaired signal is multiplied with the LO signal at the mixer, we see an effect called *LO self-mixing* and a *dc term* is seen at the baseband output due to the multiplication of similar sinusoids. These *DC offsets* are difficult to filter out as they are placed close to the desired spectrum. On the other hand, this problem is not seen in the heterodyne receivers as the dc offsets are far from the IF. So, there is a trade-off between the simplicity and the precision of both the receivers. Nevertheless, a heterodyne receiver with an image-reject filter is preferred. There are more sophisticated receiver architectures that we do not discuss here and refer the interested reader to [27].

The output signal from the mixer is then filtered to remove the unwanted frequency components using a low-pass filter and input to the analog-to-digital converter for conversion to the digital domain. The following processing steps involve mitigating the channel propagation effects (*cf.* Section 2.1.3) and demodulating the signal to retrieve the embedded information bits. Since these steps depend on the particular modulation used, we do not go into much detail and refer the reader to [29].

2.2 MACHINE LEARNING FOR CLASSIFICATION

In the current landscape, data has become the most important resource and helps organizations to make decisions that increase productivity and efficiency. This data is available in vast abundance which makes it difficult to be processed by humans. Instead, machines are used for this task and help make insightful decisions. Thus, one could say that these machines use intelligence the same as humans to accomplish a task. This field of research is called *Artificial Intelligence (AI)* wherein machine intelligence is used to make data-driven decisions. We will be concentrating on a sub-field of AI known as *Machine Learning (ML)*. ML involves the creation of algorithms that help machines learn to derive knowledge from a large amount of data and make predictive decisions without human intervention. Some real-world examples where ML is used are recommender systems in e-commerce, voice assistants, and web-search

optimization. Further, ML can be sub-divided into three categories: *Supervised Learning*, *Unsupervised Learning* and *Reinforcement Learning*.

In supervised learning, the training data consists of both input variable $x^{(i)} \in X$ and an output variable $y^{(i)} \in Y$ and the machine learns a mapping function $f : X \mapsto Y$ such that when unknown data x is input to the machine, it can predict the target variable y as illustrated in Fig. 2.8. For a classification task, X can be defined as a feature matrix $X \in \mathbb{R}^{(N,m)}$ where N is the total number of samples in the dataset and m denotes the number of features per sample. Similarly, the target variable Y can be defined as a column vector $Y \in \{0, 1, \dots, (n-1)\}^{N \times 1}$ where n is the number of classes.

On the other hand, in unsupervised learning, the training data only consists of the input variable $x^{(i)} \in X$ and no corresponding output variable. The task of the ML algorithm is to find the underlying structure in the data and group classes that share similar attributes. Lastly, in reinforcement learning, the goal is to develop a system that continuously improves its performance by interacting with the environment. Specifically, the environment provides feedback to the system which is called a *reward signal* and the system tries to maximize this reward through a series of steps. We do not delve into the intricacies of reinforcement learning here and refer the interested reader to [30] for further details.

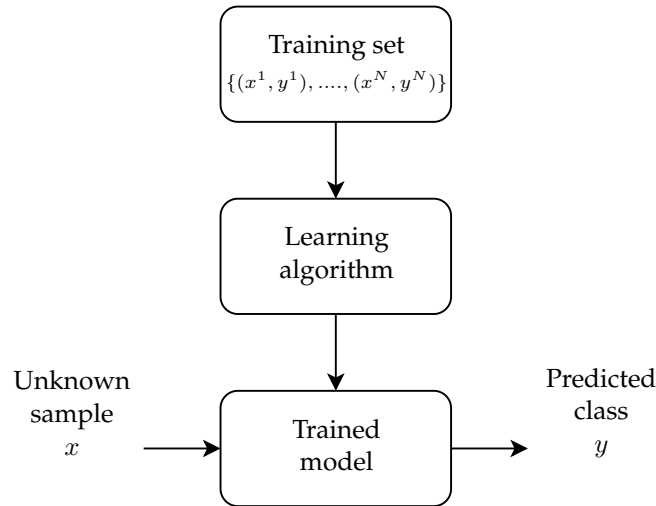


Figure 2.8: A simplified schematic of the supervised learning technique.

2.2.1 Classification Algorithms

This section details various machine learning algorithms that we use for the AMC task. First, we describe a decision-tree-based gradient boosting ML algorithm called XGBoost that uses a featurized set for the classification task. We then go on to explain DL algorithms, a sub-field of ML, which can extract complex features from raw data. Specifically, we present CNN and ResNet as they have shown great potential in the field of image and signal processing.

2.2.1.1 XGBoost

XGBoost [31] is a supervised machine learning algorithm that sequentially uses an ensemble of trees wherein each subsequent tree improves on the error made by the previous tree using an optimized gradient descent algorithm [32]. The tree ensemble model consists of a set of Classification and Regression Trees (CARTs) [33]. For a classification task, the leaves of the tree give the output scores for different classes.

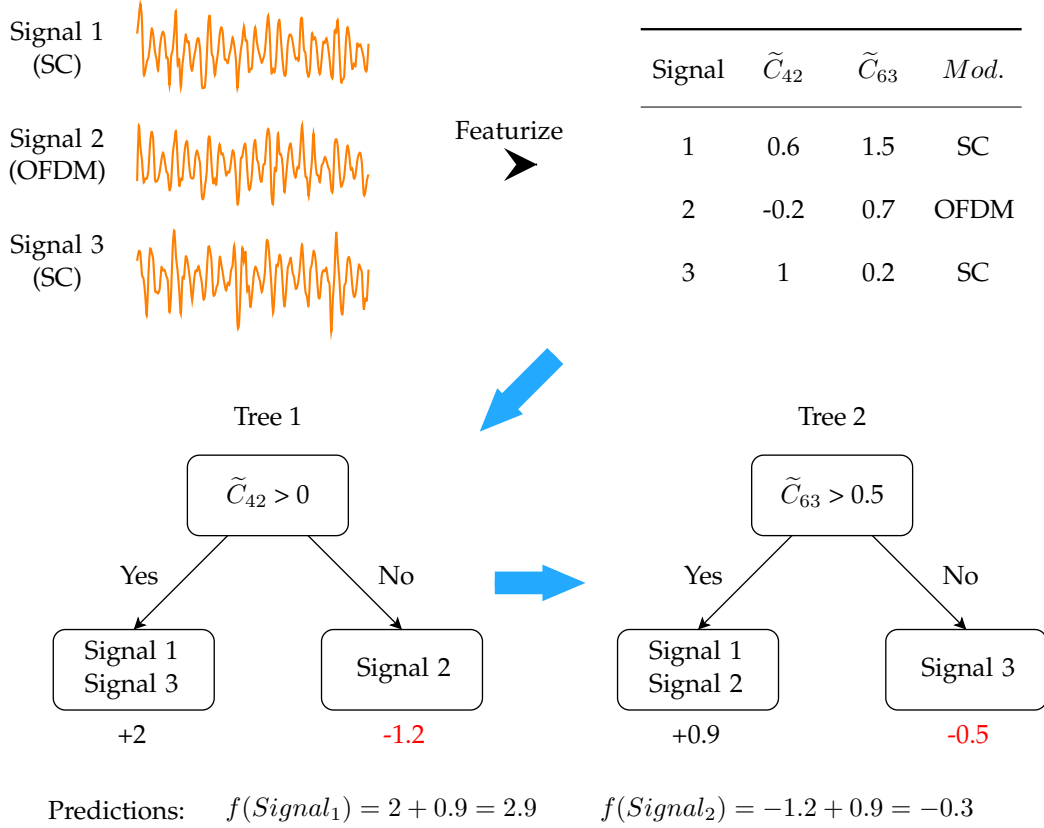


Figure 2.9: Schematic showing an ensemble of two trees to classify signals based on their modulation scheme.

Fig. 2.9 shows two classification trees that complement each other to predict if a signal is SC modulated or not. The predictions of both the trees are summed up to get the final output score. Mathematically, the prediction of the model \hat{y}_i for an input x_i can be given by the following equation

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), \quad f_k \in \mathcal{F}, \quad (2.12)$$

where K is the number of trees and \mathcal{F} is the set of all CARTs and the function $f(x)$ can be defined as

$$f(x) = w_{q(x)}, \quad (2.13)$$

where q denotes the tree structure and w gives the leaf weights for the corresponding tree. To learn the parameters of the model, the following objective function $\mathcal{L}(\phi)$ has to be minimized

$$\mathcal{L}(\phi) = \sum_i^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k), \quad (2.14)$$

where l is the loss function which calculates the difference between the actual y_i and predicted \hat{y}_i value. The second function Ω is the regularization term that penalizes the model weights to prevent overfitting.

Model training

Now that we have described our model, we can look into the training procedure. For supervised learning, the model is trained by optimizing the objective function. In case of XGBoost, we will optimize the function given in Equation 2.14. From this equation, we can identify that the learnable parameters are functions f_k that describes the tree structure. This optimization is not realizable in euclidean space as learning multiple trees at once is not feasible. Thus, XGBoost uses an additive strategy wherein one tree is learned at a time and the prediction from the previous tree is added to it. Thus, at iteration t , the prediction for sample i given as $y_i^{(t)}$ can be described as

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) \quad (2.15)$$

and the objective function can be re-formulated as

$$\mathcal{L}^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t). \quad (2.16)$$

This objective cannot yet be optimized for all loss functions and hence a *second-order taylor approximation* of the loss function is taken which gives the following objective

$$\mathcal{L}^{(t)} = \sum_{i=1}^n [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + h_i f_t^2(x_i)] + \Omega(f_t), \quad (2.17)$$

where the gradient g_i and hessian h_i are given as

$$\begin{aligned} g_i &= \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}), \text{ and} \\ h_i &= \partial_{\hat{y}_i^{(t-1)}}^2 l(y_i, \hat{y}_i^{(t-1)}). \end{aligned} \quad (2.18)$$

Now the objective function depends only on g_i and h_i and can thus be optimized for all loss functions.

Regularization

Tree models are prone to overfitting which results in the loss of generalization ability of the model and leads to degraded performance over the unknown set. Thus, XG-

Boost introduces a regularization term Ω in the objective function as seen in Equation 2.14 and it can be defined as

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2, \quad (2.19)$$

where γ is the parameter that controls the leaf node split based on loss reduction, λ is the regularization parameter, T is the number of leaves in the tree and w gives the corresponding leaf weights. To make the model more conservative, values of γ and λ can be increased but care must be taken as to not underfit the model.

Scoring the tree structure

At every iteration of the algorithm, the goal is to select a tree that will maximize the loss reduction. For this purpose, we need to find a way to score the tree structure. After re-formulating and removing the constants, the objective function for the t^{th} tree can be given by

$$\begin{aligned} \tilde{\mathcal{L}}^{(t)} &= \sum_{i=1}^n [g_i f_t(x_i) + h_i f_t^2(x_i)] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2 \\ &= \sum_{j=1}^T \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T, \end{aligned} \quad (2.20)$$

where $I_j = \{i | q(x_i) = j\}$ is the set of indices of data points that map to the j^{th} leaf. Thus, for a given tree structure $q(x)$, the optimal weight w_j^* at the j^{th} leaf is given as

$$w_j^* = - \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \quad (2.21)$$

and the corresponding optimal objective value can be calculated as

$$\tilde{\mathcal{L}}^{(t)}(q) = - \frac{1}{2} \sum_{j=1}^T \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T. \quad (2.22)$$

This equation gives the score for a particular tree structure and helps in determining the tree quality.

Learning the tree structure

Now that we have all the resources to calculate the tree scores, we can enumerate across all the tree structures q and find the best one. Unfortunately, this is not feasible in terms of computation time and resources. Instead, the algorithm takes a *greedy approach* and iteratively adds branches to the tree based on a gain value given by

$$Gain = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma \quad (2.23)$$

in a simplified form where $G_j = \sum_{i \in I_j} g_i$ and $H_j = \sum_{i \in I_j} h_i$ and the first, second and third parts of the equation give the scores on the new left leaf, new right leaf and their root leaf respectively. The γ parameter is of prime importance here because if the gain is less than γ , the branch will not be added and this is what we call the *pruning* technique used by the tree models.

We have now seen the most significant components of the XGBoost algorithm and can appreciate how this algorithm can scale to big datasets with optimal usage of computational resources and provide better accuracy than other tree models accessible. For further insights into the algorithm, we refer the reader to [31].

2.2.1.2 Convolutional Neural Network

CNN [34] is a deep learning-based supervised learning algorithm that can extract complex and high-dimensional features from raw data and use them for the classification task. These CNN features are invariant to shifting, scaling, and distortion which makes CNN a popular choice for image, video, and speech classification.

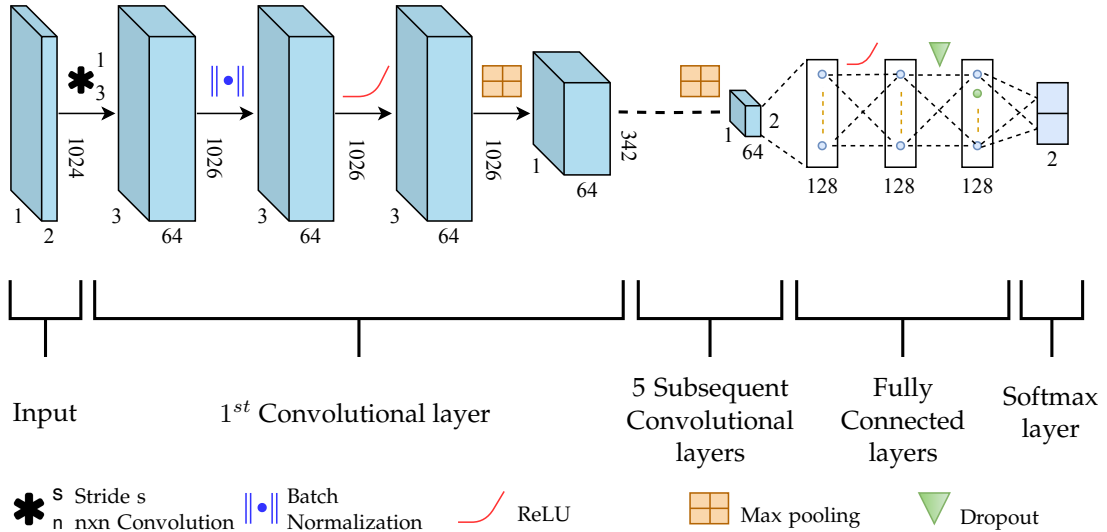


Figure 2.10: Architecture of a convolutional neural network, here for a binary classification task of signal modulation recognition. Each block comprises of a set of feature maps that contain units of similar weights.

Fig. 2.10 shows an architecture of CNN for classifying signals based on their modulation scheme. Here, we consider a binary classification task with the two classes being SC and OFDM. The input to our model is a raw signal with 1024 In-phase (I) and Quadrature (Q) samples each. The input data is usually normalized to zero-mean and unit variance so that the model is easier to train and takes less convergence time. In our case, we only normalize the data to unit variance as normalizing to zero-mean causes the variance to be very close to zero leading to data distortion. The processed data is then inputted to the convolution layer for feature extraction.

Convolution Block

In the convolution block, we slide a $(n \times n)$ filter over the input signal and take a dot product between the filter and input values along the way. The sliding pattern is decided by the stride parameter, if it is set to one, we move the filter over the input one-column at a time in the horizontal direction and one row at a time in the vertical direction. The transformed data is called a feature map and a set of feature maps are used to learn different features from the input data. An illustration of the convolution operation is shown in Fig. 2.11. For an input x with dimensions (N, C_{in}, H, W) , the output y of the convolution operation for the i^{th} batch and the j^{th} output channel can be given as

$$y(N_i, C_{out_j}) = bias(C_{out_j}) + \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, k) \otimes x(N_i, k), \quad (2.24)$$

where N is the batch size and C , H and W denote the number of channels, height, and width of the data respectively. The *weight* term denotes the filter values that are shared over all the input space. Finally, the *bias* term is a constant that denotes an additional set of weights used in the network to help it learn efficiently for a given set of inputs.

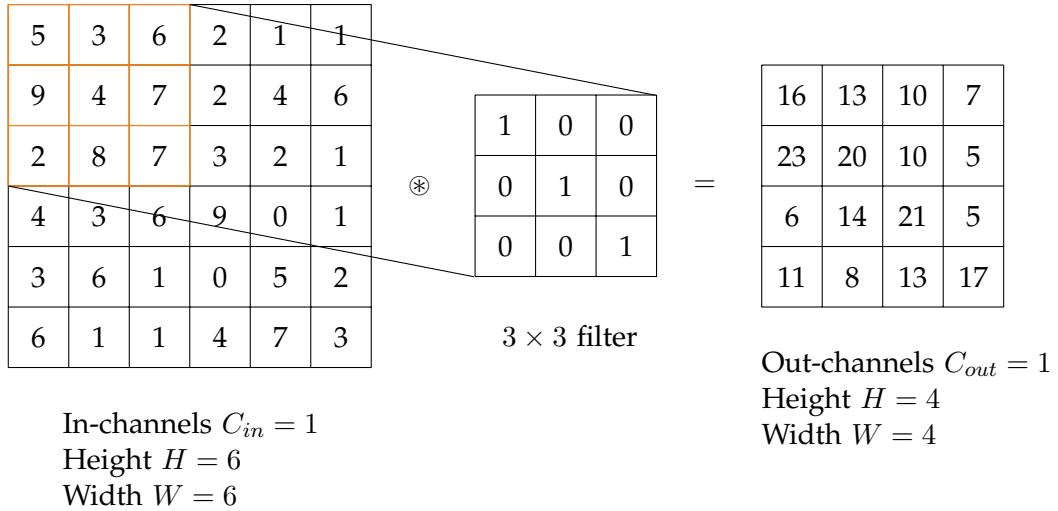


Figure 2.11: Illustration of the convolution operation.

Batch Normalization

We have seen earlier that pre-processing our input data makes our network easier and faster to train. Without pre-processing, the distribution of our train and test set may vary leading to degraded model performance. This effect of varying input data distribution is called *covariate shift* [35]. Even after standardizing our data, the model is still prone to errors because the hidden layers are learning parameters that are constantly changing during model training. This leads to a varying distribution of data in the hidden layers and leads to the so-called *internal covariate shift* [36]. To tackle this issue, *batch normalization* (BN) is used. BN standardizes the data of a mini-batch

consisting of n number of input samples. Specifically, we subtract the mean of mini-batch samples from the input sample and divide this result by the standard deviation of the mini-batch samples. Mathematically, the standardized input sample \hat{x}_i can be represented as

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \quad (2.25)$$

where μ_B and σ_B denote the mean and standard deviation of the mini-batch samples and ϵ is a small value ($\sim 10^{-8}$) that is added to the denominator to prevent the zero-division error in case the standard deviation goes to zero. The BN equation for the standardized input can now be given as

$$BN(x_i) = \gamma \hat{x}_i + \beta, \quad (2.26)$$

where γ and β are the scale and shift parameters. These parameters can be tuned by the network to have data distributions other than the gaussian one as set by the BN layer if they help it learn better. Also, a nice property of BN is that it can act as an identity function if $\gamma = \sqrt{\sigma_B^2}$ and $\beta = \mu_B$ and thus the network can undo the work done by the BN layer. Using BN has several advantages including speeding up the network training, tackling the problem of internal covariate shift, and providing some level of regularization to the network by introducing noise.

Activation and Pooling Layers

Activation functions are used to transform the output of a particular node such as it lies between a specific range such as (0,1) for *sigmoid* activation and (-1,1) in case of *tanh* activation. It is responsible for deciding which neuron should be or shouldn't be activated for a certain prediction. The usage of sigmoid and tanh activation functions is usually avoided as they suffer from the problem of *vanishing gradients*. To understand this, we refer to Fig. 2.12 and notice that for large negative and positive output values, the gradients are very small and the network is saturated. These local gradients are multiplied with the backpropagated gradients during the learning process making them very small and leading to the issue of vanishing gradients mentioned earlier. To solve this issue, the Rectified Linear Unit (ReLU) [37] activation function is used. Using ReLU, all the negative values of the output node are forced to zero and the positive values are kept the same. Mathematically, we can represent the ReLU function for an input x as

$$f(x) = \max(0, x). \quad (2.27)$$

We can notice from Fig. 2.12 that for the ReLU function, there is no saturation and for large positive or negative output values, the local gradient values are either zero (not activated) or increase linearly. In practice, this activation function works quite well and is the most widely used.

After the activation layer, generally, a pooling layer is used to downsample the feature maps. This helps the network in learning more generalized and sophisticated features instead of the finer features that may be subject to distortions owing to the shifting and scaling operations on the input data. An added advantage of pooling is

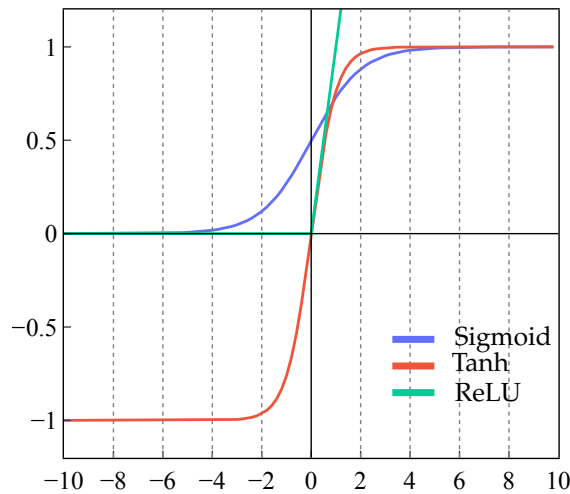


Figure 2.12: Typical activation functions used in a neural network.

that it reduces the number of computations during training by reducing the spatial dimensions of the feature maps. There are two types of pooling operations, namely *max* and *average* pooling. In max pooling, we slide an $(n \times n)$ filter over the feature map and take the maximum value in that filter. Average pooling is similar to max pooling, the only difference being that the average of all values in the filter is taken instead of the maximum value. In the most common form of pooling, a (2×2) filter with a stride of 2 is used. Fig. 2.13 illustrates the application of ReLU activation and max-pooling to the input feature map.

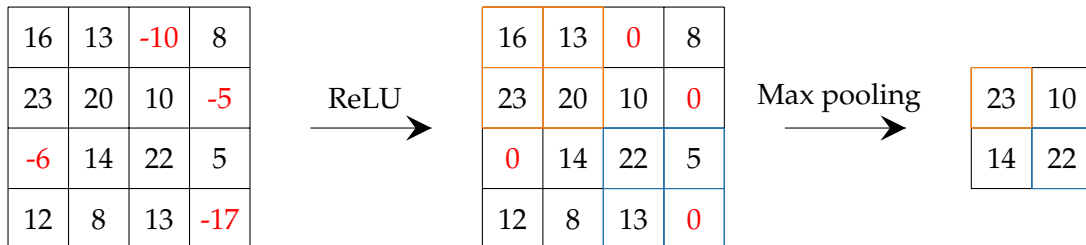


Figure 2.13: Illustration of the application of ReLU activation and max pooling on the input feature map.

Dropout

When we train large neural networks on relatively small datasets, the network can easily overfit, meaning it adapts to the training set so perfectly that it loses its generalization capability and produce poor results when tested on a different dataset. To prevent the network from overfitting, we need a certain level of regularization that will help the network learn better. This is provided by a technique called *dropout* [35]. Dropout is a simple yet effective regularization technique that discards nodes along

with their input and output connections in a particular layer with a probability $(1-p)$ or retains them with a probability p . This helps us in training different network architectures in parallel as the number of nodes per layer changes dynamically. Moreover, the network randomly chooses the nodes to retain which forces them to learn independent of other nodes and thus produce better representations of input data and provide better generalization on the test set. Dropout is generally implemented in the fully connected layers of the neural net.

Fully Connected and Softmax layers

After extracting high-level features from the convolution layers, these features are input to the Fully Connected (FC) layers of the neural network for classification. FC layers contain nodes that connect to all other nodes in the next and previous layers in the neural network and the output of these layers are activated by the softmax layer which gives the predicted probabilities for all the classes. The penultimate FC layer of the neural network that provides the raw model predictions to the softmax layer is called the *logits layer*. For j classes, the softmax probability S of a logit \hat{y}_i can be given as

$$S(\hat{y}_i) = \frac{e^{\hat{y}_i}}{\sum_j e^{\hat{y}_j}}, \quad (2.28)$$

where an exponent of the input logit is normalized by the sum of exponents of all the logits so that the sum of all class probabilities equals one.

Model Training

Now that we are familiar with all the components of a neural network, we can start training it. Training the neural network comprises of three main phases, namely *forward*, *backward* and the *update* phase. During the forward pass, the data propagates through the convolution layers and the features are extracted. These features are used for classification by the FC layers and the softmax layer gives the class probabilities. The quality of these model predictions is then determined using a *loss function*. If the model prediction is close to the actual prediction, the loss will be minimized and vice-versa. Generally, the cross-entropy loss is used for multi-class classification and can be represented as

$$\mathcal{L}_i = - \sum_i^C y_i \log(S(\hat{y}_i)) \quad (2.29)$$

for a particular class where C denotes the number of classes, $S(\hat{y}_i)$ denotes the softmax probability for an input logit \hat{y}_i as given in Equation 2.28. The goal of the network is to optimize this loss function during the learning process. This is done by calculating the *gradients* of the loss function wrt all the weights in the network and updating the network parameters using these gradients. Gradients are nothing but partial derivatives that give us the influence of network weights/parameters on the output and are calculated during the backward pass of the network.

Fig. 2.14 shows how the gradients are calculated using *backpropagation*. The weights x and y are input to a network layer that applies a function f on them and outputs z . During this forward pass, the local gradients $\frac{\partial z}{\partial x}$ and $\frac{\partial z}{\partial y}$ are calculated

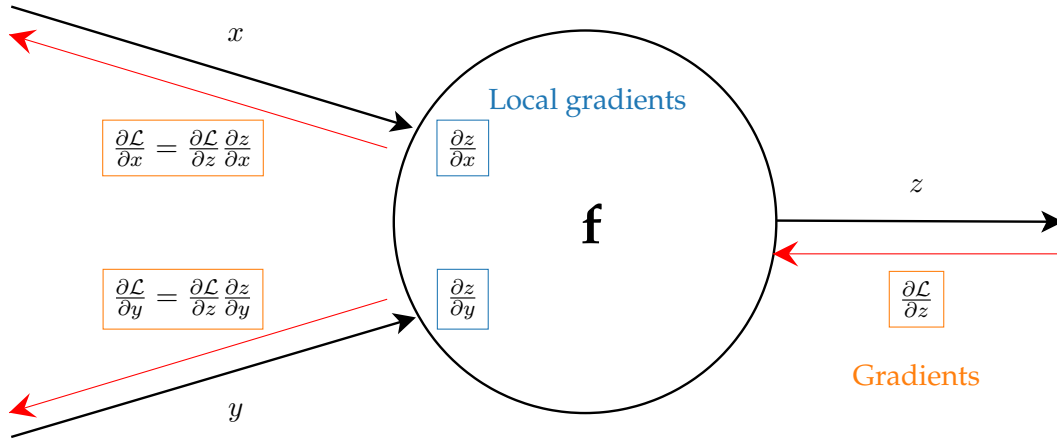


Figure 2.14: Calculation of gradients using the backpropagation algorithm.

and stored in the memory and the loss is computed based on the model output. During the backward pass, the gradient of loss wrt to the output weight z is calculated and backpropagated. This gradient in turn is multiplied with the local gradients of weights x and y to get the gradient of loss wrt these inputs. This process of moving backward across the network and calculating gradients using the chain rule is called *backpropagation*.

Once our gradients have been calculated, the last step is to update the network parameters to optimize our loss function. The most commonly used optimization algorithm is *gradient descent*. Using this algorithm, we can update the network parameters using the formula

$$weights+ = -learning_rate + weights_grad, \quad (2.30)$$

where $weights_grad$ are the gradients that we computed during the backward pass and give us the direction where the loss increases. As we have to minimize the loss, we include the negative sign in the formula. The *learning_rate* is a network hyperparameter that gives the amount by which we should move in the direction given by the gradients so that we can reach the global optimum.

We have now seen all the stages of the training process. The forward, backward, and update pass are performed iteratively until our network loss is optimized. Many more sophisticated optimization algorithms help the network converge faster and with more efficiency but we will not delve into their details here and instead refer the reader to [38].

2.2.1.3 Residual Neural Networks

One would normally assume that increasing the number of layers in CNN would lead to an increase in the model performance because more robust and complex features can be extracted. Surprisingly, this is not the case, in fact as we make our networks

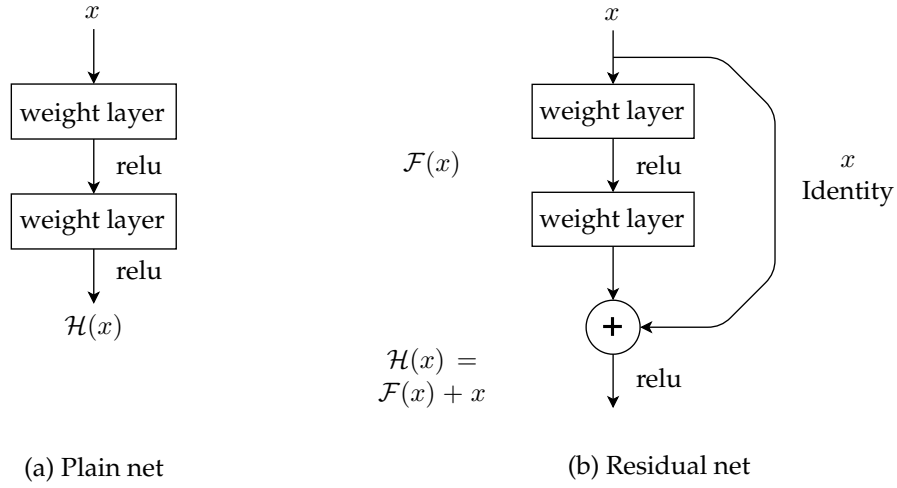


Figure 2.15: Comparison of a vanilla net and residual net building block.

deeper, the accuracy saturates and then degrades rapidly. This is because deeper networks are harder to optimize and suffer from the problem of *vanishing* gradients wherein the backpropagated gradients become very small having a negligible effect on network parameters. ResNets [25] address these issues and lets us implement deeper networks of more than a hundred layers and achieve a significant increase in the accuracy when compared to the *VGG net* [39] which is a deeper version of CNN. This is accomplished by the *shortcut* or *skip* connections present in the ResNet.

Fig. 2.15 shows a two-layer plain neural net and residual net. In a plain net, we transform the input x to a new mapping $\mathcal{H}(x)$ after applying the transformation and activation functions. On the other hand, in a ResNet, instead of transforming the input x directly, we compute a quantity, say $\mathcal{F}(x)$, to add to this input to get the mapping $\mathcal{H}(x)$. Owing to this method, all the information of input x is not discarded during the transformation. During the backward pass, the gradients are distributed equally to the weight layers and the input x . Firstly, this solves the issue of vanishing gradients that we discussed earlier, and secondly, we can do our computations in a single convolution layer close to our input and the other layers learn to add to its outputs leading to time-saving. Intuitively, the shortcut connection is just an identity mapping on top of which the output of the layers is added and this makes the optimization process easier.

Fig. 2.16 shows the architecture of a 34-layer ResNet for a binary classification task. Additionally, the ResNet includes a batch normalization layer after every convolution layer (not shown in the figure) and no dropout is used. The legend conveys the operation carried out at each layer along with the feature maps and stride. The shortcuts indicated by the solid lines denote identity mapping when the input and output dimensions are the same and the dotted lines denote the identity mapping padded with zeros to match the increased dimension at the output. These skip connections help the gradients backpropagate from the last layer, usually a softmax to the first convolution layer of the network, here a 7×7 conv layer. Hence, we can train a

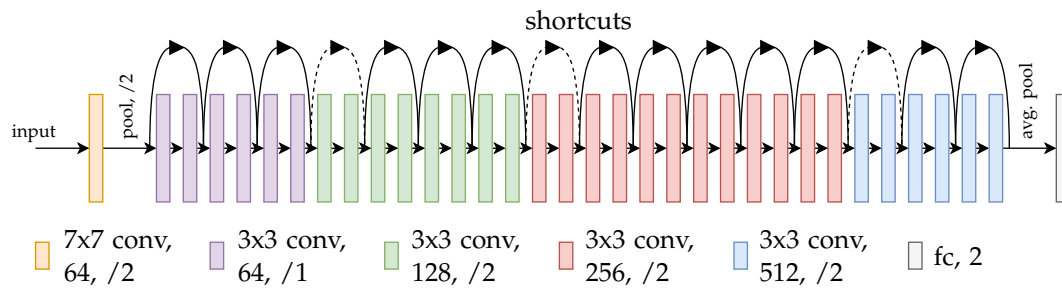


Figure 2.16: Architecture of a 34 layer residual network for a binary classification task.

layer very close to the input and the following deeper layers can just learn to add to its outputs to optimize the loss and model the data effectively.

Model Training

ResNet is trained in a way that is similar to the CNN training with some minor modifications. The network weights are initialized using enhanced *xavier initialization* [40]. The forward and the backward pass remain the same whereas stochastic gradient descent with *momentum* [38] update is used to update the network parameters.

2.2.2 Algorithms for Exploratory Analysis

Exploratory Data Analysis (EDA) is an important task for understanding data before performing predictive analysis using ML algorithms. It gives us insights into the data structure, patterns, and helps in identifying similarities between features. Further, it helps in detecting potential anomalies in data. All this is accomplished through a set of visualization tools, one of which we will be discussing in this section. This section details Self-Organizing Maps (SOMs) for EDA.

2.2.2.1 Self-Organizing Maps

SOMs [41] are an unsupervised machine learning algorithm used for dimensionality reduction and clustering tasks. They can be thought of as a single layer feed-forward network comprising of only an input and output layer that map high-dimensional inputs to a low-dimension (usually 2D) representation, referred to as a *map*. Fig. 5.1 shows a 4×4 SOM that highlights the mapping of an input vector to the 2D output map. They differ from other ANNs in the sense that they employ *competitive learning* instead of error-correcting learning i.e. the output nodes compete against each other to represent the input vector. SOMs also maintains the topological structure of the inputs which makes them a great tool for data visualization in a low dimensional space.

SOM Training

The training of a SOM consists of several steps that need to be repeated over a large number of iterations. First, the weights of the output nodes are randomly initialized. Input is then randomly sampled from the training set and its distance is computed

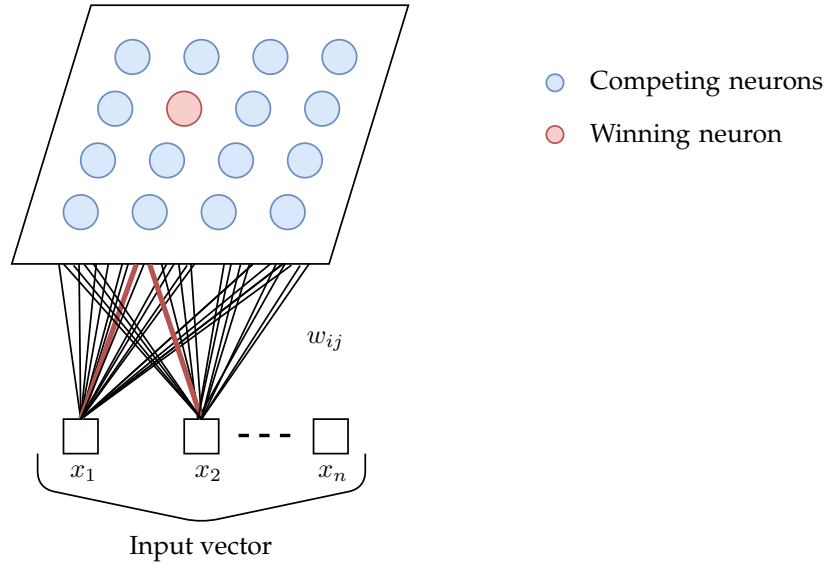


Figure 2.17: A 4x4 SOM that highlights a winning neuron (BMU) for a particular input that is randomly sampled from the training set.

with the weight vector of all output nodes. After examining all the nodes, the node with the minimum distance is chosen and is referred to as the Best Matching Unit (BMU). The most commonly used distance metric is the euclidean distance and the BMU is chosen using the following distance formula

$$BMU = \arg \min_j \sqrt{\sum_{i=1}^n (x_i - w_{ij})^2}, \quad (2.31)$$

where x denotes the input vector and w_j denotes the weight of the j^{th} output node. Once the winning node is found, its weights are updated to make them more like the input vector using the given formula

$$w_j(t+1) = w_j(t) + h_{cj}(x - w_j(t)), \quad (2.32)$$

where h_{cj} denotes the gaussian neighbourhood function of the BMU, c and t denotes the current iteration. The neighbourhood function describes the influence of the training instance on the nodes surrounding the BMU and can be given as

$$h_{cj} = \alpha(t) \cdot \exp\left(-\frac{\|r_c - r_j\|^2}{2\sigma^2(t)}\right), \quad (2.33)$$

where r denotes the node coordinates on the map, $\alpha(t)$ denotes the learning rate and $\sigma(t)$ denotes the neighbourhood radius. The neighbours near the BMU experience greater alteration in weights as compared to the farther neighbours. Both the learning

rate and the neighbourhood function are decreasing functions of time and monotonically decay according to the given formula

$$\alpha(t) = \alpha(0) \left(\frac{\alpha(T)}{\alpha(0)} \right)^{t/T}, \text{ and} \quad (2.34)$$

$$\sigma(t) = \sigma(0) \left(\frac{\sigma(T)}{\sigma(0)} \right)^{t/T}, \quad (2.35)$$

where T is the training length.

The process of identifying BMUs and its neighbours and updating their weights is repeated for all the inputs in the training set for a large number of iterations until the output map stabilizes and no further changes can be observed in the representation. The output of SOM can be visualized using a U-matrix which is a grayscale image that presents the clusters formed by closely space nodes using light colors and distinct separations are depicted by darker colors. The coloring scheme is based on the euclidean distance between the nodes.

2.2.3 Transfer Learning

TL [42] is a method of utilizing a model pre-trained on a particular task for a new but related task. The intuition of TL comes from human learning, we never learn everything from scratch but use our experience to tackle situations that may that relate to past occurrences. The key motivation for TL, especially in the context of deep learning is two-fold: utilizing less training data and saving computation time. It is not easy to obtain large datasets for supervised learning tasks as labelling data is not a trivial task. TL allows us to use comparatively smaller datasets and as we are transferring the knowledge from a pre-trained model, the newer model takes lesser time to train.

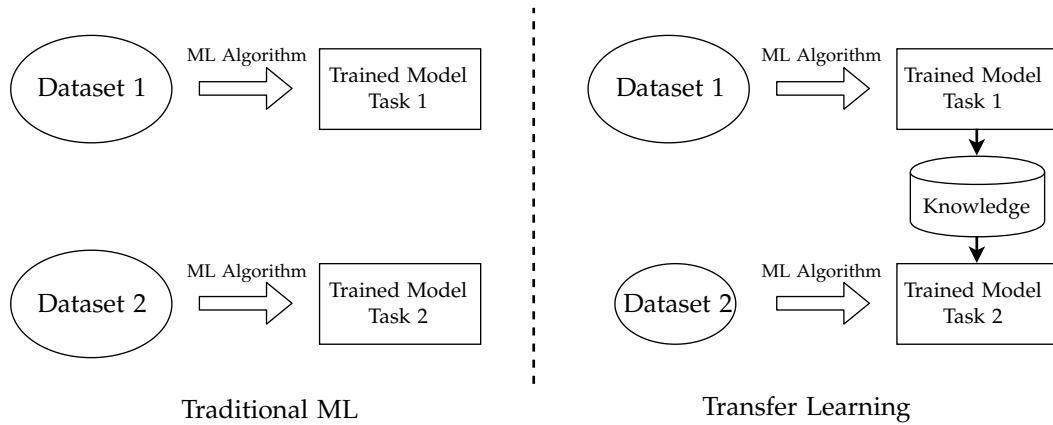


Figure 2.18: An illustration of the comparison between traditional ML and TL techniques.

Fig. 2.18 shows a comparison between traditional ML and TL techniques. While traditional ML requires training models independently for separate tasks, TL is able to

utilize a pre-trained model and transfer its knowledge to the new *task 2* which requires a reduced *dataset 2* for training the model.

Formal Definition

The concept of TL is formally defined in [43] using *domains*, *tasks* and *marginal probability distributions*. A domain \mathcal{D} consists of two components: the feature space \mathcal{X} and probability distribution $P(X)$, where X is a training sample given as $X = \{x_1, \dots, x_n\}, x_i \in \mathcal{X}$. Given a specific domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task consists of two components: a prediction function $f(\cdot)$ and a target space \mathcal{Y} (denoted as $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$). For an instance x , the prediction function $f(x)$ can be defined in probabilistic term $P(y|x)$, where y is the target vector given as $y = \{y_1, \dots, y_n\}, y_i \in \mathcal{Y}$. Given these notations, a formal definition of TL can now be formulated.

Given a source domain \mathcal{D}_s and learning task \mathcal{T}_s , a learning task \mathcal{T}_t and a domain task \mathcal{D}_t , the objective of TL is to learn the conditional probability $P(Y_t|X_t)$ in \mathcal{D}_t with the information gained from \mathcal{D}_s where $\mathcal{D}_s \neq \mathcal{D}_t$, or $\mathcal{T}_s \neq \mathcal{T}_t$.

2.2.3.1 Deep Transfer Learning

Deep learning models have achieved great success in classification tasks but usually require a large amount of labelled data which is not feasible to gather for all tasks. TL helps alleviate this issue and lets us train models using fewer data. Usually, two techniques are used to achieve this in the context of deep learning: *freezing* and *fine-tuning* of network layers. We discuss these techniques in the following paragraphs.

Off-The-Shelf Pre-Trained Models

Deep learning networks have hidden layers between the input and output layers to extract complex features from the input data. The idea is to use these weighted layers of the pre-trained network and use them to extract features for the new task without altering the network weights (the layers are frozen) during the training process. These extracted features can be input to a shallow classifier to get the final predictions for the new task. Fig. 2.19 shows how this knowledge transfer is accomplished.

Fine-Tuning Pre-Trained Models

Deep neural networks are very configurable. Instead of freezing the network layers, we can also fine-tune them during training on the target data. We start with the pre-trained model weights and gradually alter them to suit our current task. This allows us to utilise the overall knowledge of the network. Generally, this approach is used when a comparatively larger training set is available.

Additionally, a hybrid approach with both freezing and fine-tuning the network layers can be used. This can be accomplished by setting varying learning rates while training specific layers. We can use a lower learning rate for layers where we want minimal weight altering and higher learning rates where we want a significant alteration.

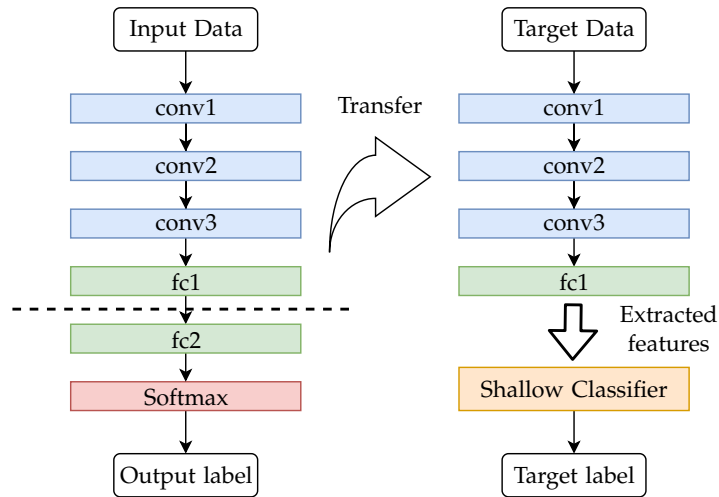


Figure 2.19: TL using off-the-shelf pre-trained models as feature extractors.

2.2.4 Model Interpretation

Deep learning models are black box models that extract complex feature representations which are difficult to interpret. On the other hand, machine learning models make predictions based on the given input features which make them relatively easy to interpret. One way of interpreting these models is to use the concept of *shapely values* [44] from game theory.

In terms of game theory, shapely values are used to assign payouts to players based on their contribution to the total payout. In terms of machine learning, these players can be thought of as features that contribute to the model prediction. Shapely value can be defined as the average marginal contribution of a feature to the model output when all possible feature coalitions are considered.

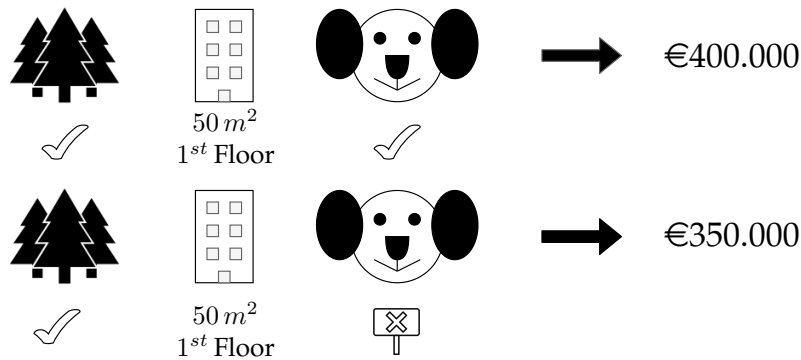


Figure 2.20: Illustration of the effect of a feature *dog-allowed* on the model output when added to a coalition of *forest-nearby*, *1st floor* and *area-50*.

Fig. 2.20 shows the contribution of a feature *dog-allowed* on the house price. When a coalition of *forest-nearby*, *1st floor* and *area-50* is taken, we calculate the house price

with and without the addition of *dog-allowed* feature. The difference between these prices gives is the marginal contribution of the feature *dog-allowed*. In our case, this comes out to be €50.000. We consider all possible feature coalitions shown in Fig. 2.21 and predict the house prices with and without the feature *dog-allowed*. After obtaining all the marginal contributions, their average is taken which gives us the shapely value for the feature under consideration.

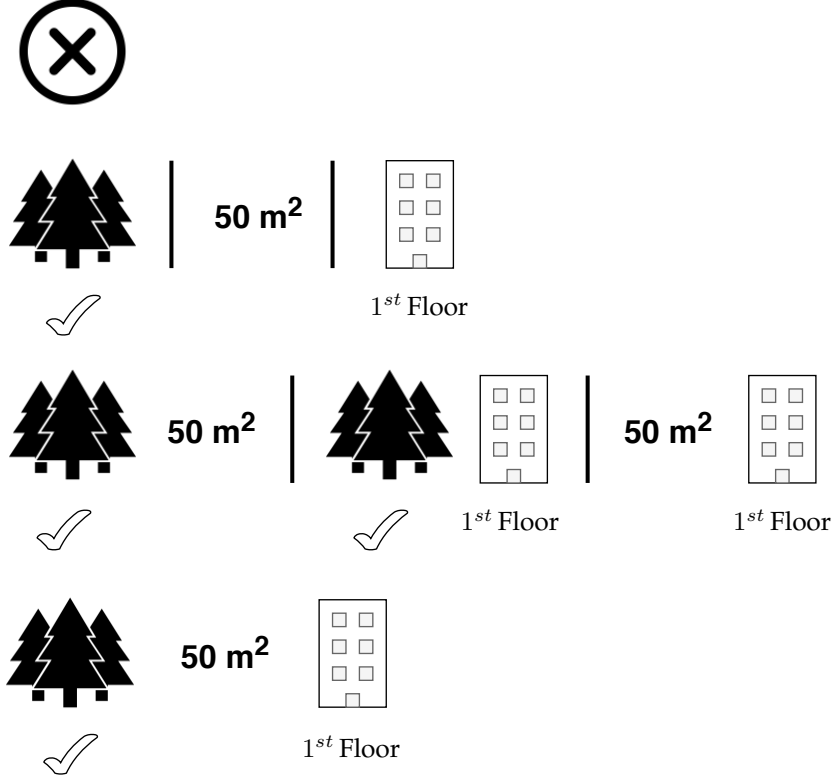


Figure 2.21: All 8 possible feature coalitions when the contribution of feature *dog-allowed* is calculated.

The classic estimation of shapely values involves training a model for all feature subsets $S \subseteq F$ where F is the set of all features. To compute the effect of a feature on the model prediction, a model $f_{S \cup \{i\}}$ is trained with that feature present and a model f_S is trained with that feature absent. The predictions of the two models are compared for the current input $f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)$, where x_S represents the values of input features present in the subset S . These differences are computed for all subsets $S \subseteq F \setminus \{i\}$. Finally, the weighted average of all the possible differences is taken and shapely values are computed using the given formula

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)]. \quad (2.36)$$

2.2.4.1 SHapely Additive exPlanation (SHAP)

SHAP [45] is based on shapely values and quantifies the contribution of a feature to the model prediction for a single instance under observation. It is called an additive explanation as the contribution of features, when added to the average model prediction, give the actual model prediction. Fig. 2.22 shows how the features contribute to the final model output based on the SHAP values. SHAP specifies the explanation as a linear model given as

$$g(z') = \phi_0 + \sum_{j=1}^M \phi_j z'_j, \quad (2.37)$$

where g is the explanation model, M is the maximum coalition size, $\phi_j \in \mathbb{R}$ is the shapely value for the feature j , and $z' \in \{0, 1\}^M$ is a vector of simplified features where 1 denotes that the corresponding feature is present and 0 denotes the absence of a feature.

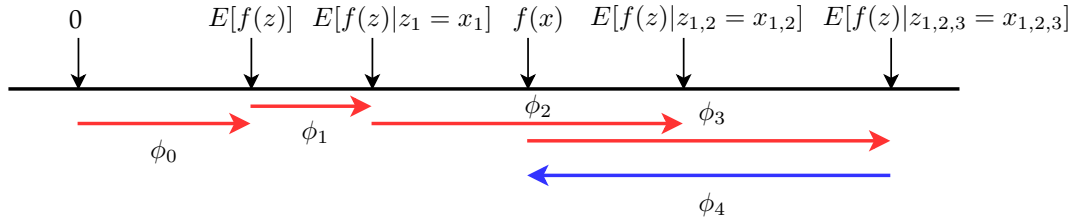


Figure 2.22: SHAP values of features contributing to the model output. The values explain how to get from a base value $E[f(z)]$ where no features are known to the final output $f(x)$.

Kernel SHAP

The kernel-based estimation approach can be used to approximate the shapely values using a lesser number of evaluations. The kernel SHAP involves the following steps:

- Sample K random coalitions $z'_k \in \{0, 1\}^M$ where $k \in \{1 \dots K\}$ and 1 denotes feature presence and 0 denotes feature absence.
- Map the coalitions to the feature space using a function h_x that maps 1's to the feature values in the given instance x , and 0's to the feature values of the randomly sampled instance from the dataset. This function can be represented as

$$h_x(z') = z \quad \text{where } h_x : \{0, 1\}^M \mapsto \mathbb{R}^p. \quad (2.38)$$

- Get predictions on each converted z'_k using the model f : $f(h_x(z'_k))$.
- Next, the weights are computed for all z'_k using the SHAP kernel. The largest weights are assigned to coalitions having the least and the most 1's. This is because when only a single feature is present in the coalition, its effect can be on the output can be known and when all but one feature in a coalition is present,

we can know about the effect of that particular feature on the output. The SHAP kernel complies with the Shapley weighting and is represented as

$$\pi_x(z') = \frac{(M-1)}{(M - \text{choose } |z'|)(M - |z'|)}, \quad (2.39)$$

where M is the maximum coalition size and $|z'|$ the number of present features in the instance z' . The special property of this kernel weighting method is that it yields the shapely values when used in linear regression models.

- Our final linear regression model can be represented as shown in Equation 2.37. We fit this model to our data and the obtained model coefficients ϕ_k give us the shapely values.

Tree SHAP

The kernel SHAP approximation works for all models but it's complexity and time computation exceed with the increasing number of features because the number of coalition possibilities increases massively. Thus, the Tree SHAP [46] method was developed for tree-based machine learning models such as decision trees, random forest, and XGBoost, etc. This allows the computation of SHAP values in polynomial time while also producing additional feature attributions.

If the computational complexity is ignored, the SHAP value can be computed by substituting the value function $f_S(x_S)$ in Equation 2.36 by the conditional expectation $E[f(x)|x_S]$ instead of the marginal expectation. If the subset S comprises of all features, the expected prediction for an instance x can be given by the node where x lies. If S is empty, the expected prediction is the average of the predictions of all leaf nodes. Lastly, if S consists of some of the features, we take the mean of the predictions of the terminal nodes whose decision path does not contradict the values in x_S . The node predictions are weighted by the number of training instances in the node. The issue is that this computation has to be done for all the feature subsets which take exponential time. This is where the Tree SHAP method comes handy.

The intuition is that the algorithm recursively keeps track of what proportion of all the possible subsets reach the leaf nodes. A naive way to do this is to keep a track of how many subsets pass down each branch of the tree. But this combines subsets of different sizes and prevents their proper weighting because weighting is dependent on the number of subset features as seen in Equation 2.36. To address this issue, all possible subset sizes need to be tracked recursively. This increases the complexity of this algorithm. We do not go into further details of the algorithm here and refer the interested reader to [46].

2.2.5 Open Set Classification

Deep learning networks are usually closed set networks i.e. they can only make predictions for target classes they are trained on. If an unknown input is presented to the network, it is easily fooled into predicting an incorrect output class. Thus, the deployment of these networks in a dynamic real-world environment is challenging.

We cannot train the network for all the unknown classes and hence the need arises for an *open set classification* wherein the network can predict the unknown classes.

A naive way to do this is by *thresholding* the class probabilities output by the last layer (usually softmax) of the network. This only works under the assumption that the network outputs low-class probabilities for all the known classes. Prior work has already shown that it is possible to produce data that can easily fool the network in producing incorrect results [47, 48]. Thus, thresholding is not sufficient for our task. One of the main components of the problem is the softmax layer due to its closed nature, a solution is proposed in [49] that replaces the softmax with an *OpenMax* layer that also incorporates the likelihood of the unknown class.

Identifying an unknown class requires the use of meta-recognition algorithms [50, 51] to study network scores and analyze their distribution using the Extreme Value Theory (EVT). These distributions are shown to follow the Weibull distribution and the EVT based meta-recognition algorithm for OpenMax is given in Alg. 1. The first step is to extract the prediction scores from the penultimate layer of the network. These extracted scores are averaged per class for all the correctly classified outputs. This gives us Mean Allocation Vectors (MAVs) for all the known classes. The next step is to compute the distance between the MAV and the training samples corresponding to the same class to find out the deviation of the samples from the mean. The distance metric used here is the euclidean-cosine distance which is a weighted combination of euclidean and cosine distance given as

$$dist(A, B) = \frac{\sqrt{\sum_{i=1}^n (A_i - B_i)^2}}{200} + \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}. \quad (2.40)$$

These distances are used to generate the Weibull model for the training classes. Specifically, the FitHigh function from libMR [52] library is used that gives the parameter ρ_i which is used to estimate the probability of the input being an outlier to the i^{th} class.

Algorithm 1 EVT Meta-Recognition Calibration for the Open Set Deep Networks, with per class Weibull fit to η largest distance to mean activation vector. Return libMR models ρ_j which includes parameters τ_i for shifting the data as well as the Weibull shape and scale parameters: κ_i, λ_i .

Require: FitHigh function from libMR

Require: Activation levels in the penultimate network layer $v(x) = v_1(x) \dots v_N(x)$

Require: For each class j let $S_{i,j} = v_j(x_{i,j})$ for each correctly classified training example $x_{i,j}$.

1: **for** $j = 1 \dots N$ **do**

2: **Compute mean AV**, $\mu_j = \text{mean}_i(S_{i,j})$

3: **EVT Fit** $\rho_j = (\tau_j, \kappa_j, \lambda_j) = \text{FitHigh}(\|\hat{S}_j - \mu_j\|, \eta)$

4: **end for**

5: **Return** means μ_j and libMR models ρ_j

Algorithm 2 OpenMax probability estimation with rejection of unknown or uncertain inputs.

Require: Activation vector for $v(x) = v_1(x), \dots, v_N(x)$

Require: means μ_j and libMR models $\rho_j = (\tau_i, \lambda_i, \kappa_i)$

Require: α , the number of "top" classes to revise

1: Let $s(i) = \text{argsort}(v_j(x))$; Let $w_j = 1$

2: **for** $i = 1, \dots, \alpha$ **do**

3: $\omega_{s(i)}(x) = 1 - \frac{\alpha-i}{\alpha} e^{-\left(\frac{\|x - \tau_{s(i)}\|}{\lambda_{s(i)}}\right)^{\kappa_{s(i)}}}$

4: **end for**

5: Revise activation vector $\hat{v}(x) = v(x) \circ \omega(x)$

6: Define $\hat{v}_0(x) = \sum_i v_i(x)(1 - \omega_i(x))$.

7:

$$\hat{P}(y = j|x) = \frac{e^{\hat{v}_j(x)}}{\sum_{i=0}^N e^{\hat{v}_i(x)}} \quad (2.41)$$

8: Let $y^* = \text{argmax}_j P(y = j|x)$

9: Reject input if $y^* == 0$ or $P(y = y^*|x) < \epsilon$

Once we have the MAVs and Weibull models for all the classes, we can estimate the OpenMax probability for the test input as per Alg. 2. The Weibull Cumulative Distribution Function (CDF) is used to estimate all the probabilities and can be given as

$$S = 1 - \exp \left[- \left(\frac{\|x - \tau\|}{\lambda} \right)^{\kappa} \right], \quad (2.42)$$

where x denotes the features of the test sample and τ, λ, κ are the Weibull parameters. Weights are introduced to scale the Weibull CDF probability (line 3 of Alg. 2) for the top-ranked α classes as they produce the most meaningful probabilities for the EVT function. We then compute the revised activation vector (probabilities) with the modified top scores. Now that we have the probabilities of all the known classes, the probability of the unknown (here, class 0) can be indirectly calculated by summing the probabilities unrelated to the known class (line 6 of Alg. 2). This gives us the revised activation vector comprising the unknown class and the OpenMax probability can be computed using 2.41. If the OpenMax probability for the unknown class is the highest, the input is explicitly rejected. Finally, OpenMax also accounts for rejecting the unknown and uncertain classes based on thresholding (line 9 of Alg. 2).

LITERATURE REVIEW AND STATE OF ART

The task of modulation classification pertains to the detection of unknown modulation of the received signal which helps in decoding it. Moreover, it enables interference monitoring and paves way for opportunistic routing. It has also been widely implemented in military applications to detect jamming signals and identify enemy transmissions. The most recent implementation of these algorithms can be seen in cognitive radios for effective spectrum monitoring and in Multiple Input Multiple Output (MIMO) systems to increase the transmission capacity. Our work deals with the task of AMC which deals with the automatic detection of the incoming signals. The AMC task can be achieved via three approaches widely studied in the literature, namely, likelihood-based, feature-based and neural network-based. We briefly review the work in likelihood-based approaches and then focus our attention mainly on the feature-based and neural network domain as it relates closely to our work.

Likelihood-Based Approaches

The LB approach is a multiple composite hypothesis testing problem whose solution depends on the modeling of unknown parameters [53]. Here, each hypothesis accounts for a modulation class, and the likelihood of each modulation hypothesis is evaluated with the received signal samples. Then, a pairwise comparison of the hypothesis is performed to make a classification decision. Usually, this decision is based on the Likelihood Ratio Test (LRT) which requires thresholding. A more intuitive way is to compute the maximum likelihood amongst all the candidate hypotheses [54].

The LRT works well when all the channel parameters are known but this is generally not the case and we may encounter unknown parameters that hinder the computation of the likelihood function. Thus, variations of the LRT are suggested in the literature. In particular, the Average Likelihood Ratio Test (ALRT) can overcome the issue of unknown parameters by replacing them with the integrals over all their possible values [5]. This approach becomes computationally expensive with the increase in the number of unknown parameters. An alternative approach, the Generalized Likelihood Ratio Test (GLRT), on the other hand, replaces the unknowns with their maximum likelihood estimates over a possible set of intervals [7]. GLRT cannot distinguish between signals having nested constellations e.g. Binary Phase Shift Keying (BPSK) and Quadrature Phase Shift Keying (QPSK). This issue arises due to the estimator bias for high-order modulations caused due to maximisation over signal symbols in the likelihood function. A combination of the ALRT and GLRT approach known as the Hybrid Likelihood Ratio Test (HLRT) [7] is used to overcome this bias by the averaging over the transmitted symbols during the computation of the likelihood

function. Some other modifications of these approaches such as quasi-ALRT [55] and quasi-HLRT [56] are also studied in the literature.

The LB methods produce optimal results when the channel parameters can be estimated accurately [54]. But estimation of parameters is a resource-intensive process [55] and thus a suboptimal approach with lower complexity is preferred. This has lead the community towards finding other solutions, one of which involves the use of signal features and is discussed ahead. For further details on the LB approach, we refer the reader to the comprehensive analysis in [53, 57].

Feature-Based Approaches

The FB approaches involve the crafting of signal features from the received signal samples [58]. The objective is to find distinct features that enable us to distinguish between different modulation signals. The extracted features can be used for classification either using decision-theoretic approaches or neural networks [59]. The FB approaches are more viable than the previously discussed LB approaches as they do not require the estimation of channel parameters and low computational power is needed for their extraction.

Research in modulation classification dates back nearly three decades. The most prominent work in this aspect has been done by Azzouz and Nandi [20, 58, 60], and their seminal publications introduced different spectral features extracted from the signals which have been widely adopted in the literature. In our work, we present these spectral features, which are based on the instantaneous amplitude, phase, and frequency of the intercepted signal, in Section 4.2. Further, Soliman and Hsue's work on classification using moments [13] provides the basis for statistical features that can be derived from the signals. Their work shows that for the M -ary PSK signals, the n^{th} even moment of the signal phase is a monotonically increasing function of M and can hence be used for their classification.

Most of the work in early publications dealt with the simple analog and digital modulation schemes. The modern communication systems involve higher-order modulation signals and have to deal with noise which presents the need for more robust features. Swami and Sadler [61] suggested the use of higher-order statistical features i.e. the fourth-order cumulants for the classification task due to their robustness to the frequency and phase offsets and additive noise. Further modifications to these cumulants have been presented in the work of Wu *et al.* [62] to account for the impact of multipath fading channels. Moreover, Orlic and Dukic [63] adopt higher-order cumulants up to the sixth-order to enhance the feasibility of AMC algorithms in the real-world scenarios. Many other features based on spectral correlation [64], cyclic temporal cumulants [65] and wavelet transform [66] are studied in the literature but are not considered in our work.

Several classification algorithms have been developed to use the extracted set of features for AMC. The early work of Azzouz and Nandi [60] considered hierarchical decision trees where the features and suitable thresholds were manually chosen at each node such that optimal correct decisions are made. Furthermore, their later work [20] was one of the first publications that suggested the use of ANNs for modulation classification. They consider a 2 layer MLP which takes all the extracted features as inputs and outputs the prediction classes without the use of manual thresholding.

The use of MLP with efficient training techniques [67] has also been explored. Later works in AMC have utilized machine learning algorithms such as Support Vector Machines (SVMs) for the featurized data [15, 16, 68, 69] with considerable success. Recently, boosting algorithms such as XGBoost have also been considered for the FB classification [70] and have shown to outperform SVMs [71].

Considering the vast amount of literature available for the AMC task employing signal features, it would also be worthwhile to identify features that lead to a particular modulation prediction. To the best of our knowledge, no prior work has dealt with the aspect of interpreting classifier outputs for finding prominent features in the signal modulations under test. Our work contributes to the understanding of classifier predictions and aids in the selection of the most significant features.

Neural Network-Based Approaches

Feature-based classifiers produce optimal results when the features are crafted carefully. Even DNNs have been used for AMC using the extracted features [72] and have shown to perform well under fading channels [73] if the selected set of features exhibit statistical relevance to the modulation class. Despite the success achieved by FB approaches, there remains the issue of handcrafting features that requires specific domain knowledge. This has propelled the research community to look towards deep learning networks that have the capability of extracting complex features from the raw signal samples [24, 74–83] and use them for classification.

A seminal publication by Timothy *et al.* [74] proposed the first use of CNNs for classifying modulations using the raw signal samples generated by GNU radio [84]. Their later publication [24] introduced a richer OTA dataset consisting of a wider range of modulations seen in the real-world environment. In [77], a modified CNN approach was proposed to correct the signal distortions that may occur in the wireless channel. Further, due to CNNs impressive performance on image data, conversion of the raw signal samples to constellation diagrams has also been suggested [85, 86]. Moreover, an approach utilizing a combination of raw I/Q signals and constellation images as inputs to hierarchical CNNs has been proposed to classify higher-order QAM signals [82]. An effort to analyze and show the robustness of CNN features has also been made in [87]. For further improvements in the classification performance, deeper variants of convolutional networks, such as ResNets, have also been studied [24, 75, 81, 88].

Apart from CNNs, Recurrent Neural Network (RNN) based classifiers such as Gated Recurrent Unit (GRU) [89] have been used to enhance performance [90]. In [79], another RNN-based classifier, Long Short Term Memory (LSTM) network is investigated for modulation classification and its utility for a variable symbol rate scenario is studied. Further, work in [81] explores the use of LSTM networks and presents algorithms to reduce the training time.

Usually, it may be the case that enough data or computing power is not available for training the neural networks. Hence, transfer learning approaches to alleviate this problem have been studied in the literature. In [78], a CNN based classifier that deals with long signal samples are proposed, and transfer learning is introduced to fine-tune pre-trained network for use in different environments or with diverse modulation schemes. Transfer learning has also been used to fine-tune networks

trained on synthetic signal data for testing with the OTA signal data resulting in a notable increase in the accuracy [24]. Recently, authors in [88] have studied the effect of knowledge transfer between environments of varying SNRs and have reported encouraging results.

At this point, we have covered a vast amount of related work around AMC and refer the reader to some comprehensive survey publications [57, 91, 92] covering most of the FB approaches. Though a vast volume of literature is available, most papers till recent base their results on simulated synthetic data [74, 76–78, 81, 82, 87]. Only very few authors validate their work on OTA transmissions using real radio hardware. One seminal publication in this regard is the work of Timothy *et al.* [24] that considers OTA transmissions via USRP Software Defined Radio (SDR) platform. Llenas *et al.* evaluate a SVM-based classifier under thorough simulations of noise, interference and channel models but only consider a single OTA scenario [93]. A more comprehensive analysis of the AMC task is presented in [15] where OTA transmissions with heterogeneous transceiver hardware and co-channel interference are considered but only a SVM is used for classification. None of the works involving deep learning algorithms address the issues of co-channel interference, hardware heterogeneity, or mismatches between training and testing data in their OTA evaluations.

METHODOLOGY

This chapter details our AMC evaluation methodology on the available RF signal dataset presented in Section 4.1. The dataset featurization and pre-processing are detailed in Section 4.2 and 4.3, respectively, to prepare data for use with the ML models. Next, Section 4.4 presents our choice of ML algorithms for the AMC task. Further, Section 4.5 details our extensive evaluation on the data to extract meaningful results for our work. These evaluations are performed across different scenarios explained in Section 4.6 to confirm the robustness of our results.

4.1 AVAILABLE RF SIGNAL DATASET

The RF signal dataset consists of signals recorded under several different configurations. The configuration parameters saved for each signal segment are listed in Table 4.1. Based on these parameters, four different configurations are considered: *baseline*, *parameter sweep*, *interference sweep* and *mixed parameter*.

Baseline Configuration. In the baseline configuration, the most benign radio environment is considered. The transmitter and the receiver are of the highest quality and the signals are recorded at an SNR of 20 dB for every permutation of *tx_mod_type* and *tx_mod*. The measurement parameters are listed in Table 4.2.

Parameter	Description
<i>receiver</i>	Receiving device
<i>transmitter</i>	Transmitting device
<i>tx_mod_type</i>	Modulation type of the transmitter
<i>tx_mod</i>	Modulation scheme of the transmitter
<i>snr</i>	SNR at the receiver
<i>interferer</i>	Interfering device
<i>intf_mod_type</i>	Modulation type of the interferer
<i>intf_mod</i>	Modulation scheme of the interferer
<i>sir</i>	SIR at the receiver
<i>connection</i>	Connection between the radios (default: OTA)
<i>cfo</i>	CFO of the transmitter (default: 0 ppm)
<i>symbols</i>	IQ symbols per signal segment (default: 50000)

Table 4.1: Measurement configuration parameters saved for each signal segment.

Parameter	Description
<i>receiver</i>	SA
<i>transmitter</i>	VSG
<i>tx_mod_type</i>	{SC, OFDM}
<i>tx_mod</i>	{BPSK, QPSK, 16-QAM, 64-QAM}
<i>snr</i>	20 dB
<i>interferer</i>	None

Table 4.2: Measurement parameters for the baseline configuration.

Parameter Sweep Configuration. We sweep across all parameter values for each measurement configuration in order to facilitate a sensitivity analysis of the AMC performance. The considered values are presented in Table 4.3. For all the configurations, we record signals for every permutation of *tx_mod_type* and *tx_mod*.

Parameter	Description
<i>receiver+transmitter</i>	{{SA, VSG}, {SDR, SDR}}
<i>tx_mod_type</i>	{SC, OFDM}
<i>tx_mod</i>	{BPSK, QPSK, 16-QAM, 64-QAM}
<i>snr</i>	{0 dB, 5 dB, 10 dB, 15 dB, 20 dB}
<i>interferer</i>	None
<i>connection</i>	{OTA, cable}
<i>cfo</i>	{None, [-1, 1] ppm, [-5, 5] ppm}
<i>symbols</i>	50000

Table 4.3: Measurement parameters saved for the parameter sweep configuration.

Interference Sweep Configuration. This configuration involves recording of signals in a controlled interference environment. SDRs are used as the transmitter and receiver while Vector Signal Generator (VSG) is used to produce the interference. The signals are transmitted at SNR 10 dB and 20 dB and recorded for every permutation of *tx_mod_type* and *tx_mod* for all *intf_mod_type* + *intf_mod* and SIR values stated in Table 4.4.

Mixed Parameter Configuration. This configuration tries to capture the true essence of a real-world scenario. Consequently, the signals are recorded under random variations of *snr*, *sir* and *cfo* given in Table 4.5 for every combination of *tx_mod_type* + *tx_mod* and *intf_mod_type* + *intf_mod*.

The resulting dataset is saved as binary files and comprises of ~ 730 k signal segments. This data is further processed and used for the AMC task.

Parameter	Description
<i>receiver</i>	SDR
<i>transmitter</i>	SDR
<i>tx_mod_type</i>	{SC, OFDM}
<i>tx_mod</i>	{BPSK, QPSK, 16-QAM, 64-QAM}
<i>snr</i>	{0 dB, 10 dB}
<i>interferer</i>	VSG
<i>intf_mod_type+intf_mod</i>	{{SC BPSK}, {SC 16-QAM}, {OFDM 64-QAM}}
<i>sir</i>	{5 dB, 10 dB, 15 dB, 20 dB, 25 dB}

Table 4.4: Measurement parameters saved for the interference sweep configuration.

Parameter	Description
<i>receiver</i>	SDR
<i>transmitter</i>	SDR
<i>tx_mod_type</i>	{SC, OFDM}
<i>tx_mod</i>	{BPSK, QPSK, 16-QAM, 64-QAM}
<i>snr</i>	[0, 20] dB
<i>interferer</i>	VSG
<i>intf_mod_type</i>	{SC, OFDM}
<i>intf_mod</i>	{BPSK, QPSK, 16-QAM, 64-QAM}
<i>sir</i>	[5, 25] dB
<i>cfo</i>	[-1, 1] ppm

Table 4.5: Measurement parameters saved for the mixed parameter configuration.

4.2 FEATURIZATION

The performance of feature-based classifiers such as XGBoost (*cf.* Section 2.2.1.1) relies on the features extracted from the raw signal samples. These features are chosen such that they encapsulate the information regarding all the signal modulations such that high accuracy can be achieved. This section presents the spectral and statistical features considered for the AMC task and explains how they are derived from the raw I/Q samples of each signal segment. Being consistent with the prior literature, most of the definitions and variables are adopted from [20, 58, 91] with occasional adjustments indicated where applicable.

4.2.1 Nomenclature

The first step is to introduce the notations that we will be using throughout the following sections. We define y as a signal segment, received and sampled with a period of T_s as described in Section 2.1.4. The I/Q sample at time step $n = 1, \dots, N$ is

$$y(nT_s) = y[n] = y_I[n] + jy_Q[n] \quad (4.1)$$

with its in-phase and quadrature component $y_I[n]$ and $y_Q[n]$, respectively. The complex conjugate of $y[n]$ is denoted by

$$y^*[n] = y_I[n] - jy_Q[n]. \quad (4.2)$$

Amplitude Definitions

The instantaneous amplitude of the signal is

$$A[n] = \sqrt{y_I^2[n] + y_Q^2[n]} \quad (4.3)$$

with its mean value of

$$\mu_A = \frac{1}{N} \sum_{n=1}^N A[n]. \quad (4.4)$$

We define

$$A_N[n] = \frac{A[n]}{\mu_A} \quad (4.5)$$

as the normalized instantaneous amplitude and

$$A_{CN}[n] = \frac{A[n] - \mu_A}{\mu_A} \quad (4.6)$$

as the centered and normalized instantaneous amplitude. Following this notation, we define the set of *non-weak* samples of first degree,

$$S_{nw} = \{y[n] | A_N[n] > A_t; n = 1, \dots, N\}, \quad (4.7)$$

as those I/Q samples whose normalized instantaneous amplitude is larger than the threshold value A_t . The cardinality of the set is

$$N_c = |S_{nw}|. \quad (4.8)$$

Similarly, we define the set of non-weak samples of second degree,

$$S_{nw,2} = \{y[n] | (y[n] \in S_{nw}) \wedge (y[n-1] \in S_{nw}); n = 1, \dots, N\}, \quad (4.9)$$

as the subset of non-weak samples whose preceding sample is also non-weak. The set likewise has the cardinality

$$N_{c,2} = |S_{nw,2}|. \quad (4.10)$$

We note that the authors in [58] do not draw this distinction and simply define the *non-weak segment* to be the samples in S_{nw} .

Phase Definitions

In terms of the signal phase,

$$\phi[n] = \text{atan2} \left(\frac{y_Q[n]}{y_I[n]} \right) \quad (4.11)$$

is the instantaneous phase of the signal segment y at time step n . Similar to the amplitude, we continue by defining a set of normalized phase and frequency values. In the definition of the nonlinear component of the unwrapped instantaneous phase,

$$\phi_{NL}[n] \stackrel{\text{here}}{=} \text{unwrap}(\phi[n]), \quad (4.12)$$

the authors in [58] first remove the linear phase component prior to unwrapping to account for the phase added by the carrier frequency. Since we consider baseband instead of passband signals, we assume the linear trend in the phase to be eliminated by the CFO correction. Based on this definition,

$$\phi_{CNL}[n] = \phi_{NL}[n] - \frac{1}{N_c} \sum_{S_{nw}} \phi_{NL}[n] \quad (4.13)$$

is the centered nonlinear instantaneous phase, whereas

$$\phi_C[n] = \phi[n] - \frac{1}{N_c} \sum_{S_{nw}} \phi[n] \quad (4.14)$$

is the centered phase without unwrapping.

Frequency Definitions

Based on the phase value of each sample, the instantaneous frequency

$$f[n] = \frac{1}{2\pi} \frac{\phi_{NL}[n] - \phi_{NL}[n-1]}{T_s} \quad (4.15)$$

is evaluated as the numerical differentiation of the phase at each time step. With the mean of the instantaneous frequency

$$\mu_f = \frac{1}{N_{c,2}} \sum_{S_{nw,2}} f[n] \quad (4.16)$$

defined over the set of non-weak samples of second degree,

$$f_{CN}[n] = \frac{f[n] - \mu_f}{f_s} \quad (4.17)$$

is the centered instantaneous frequency normalized by the sampling frequency $f_s = 1/T_s$. Different from the authors in [20], we normalize the centered frequency by the sampling frequency f_s in lieu of the unknown symbol rate.

Statistical Definitions

The p_{th} moment of the signal segment y is defined as

$$M_{pq} = \mathbb{E} \left[(y[n])^{p-q} (y^*[n])^p \right] \quad (4.18)$$

and can be estimated by the sample moment

$$\hat{M}_{pq} = \frac{1}{N} \sum_n (y[n])^{p-q} (y^*[n])^p. \quad (4.19)$$

The sample moments are only asymptotically unbiased estimators of the real moments [94]. However, since the sample size N is large in our case, the assumption $\hat{M}_{pq} \approx M_{pq}$ is justified.

The joint cumulant of the n random variables X_1, \dots, X_n is defined by the moments-to-cumulants formula [94]

$$\text{cum}(X_1, \dots, X_n) = \sum_{\pi} (|\pi| - 1)! (-1)^{|\pi|-1} \prod_{B \in \pi} \mathbb{E} \left[\prod_{i \in B} X_i \right], \quad (4.20)$$

where π runs through the list of all partitions of X_1, \dots, X_n , $|\pi|$ is the number of parts in a partition and B runs through the parts of the partition π .

The n^{th} order cumulants of signal y are thus defined as

$$C_{nm} = \text{cum}(\underbrace{y, \dots, y}_{n-m \text{ times}}, \underbrace{y^*, \dots, y^*}_{m \text{ times}}). \quad (4.21)$$

The cumulants can be estimated by \hat{C}_{nm} through substitution of the true expectations in (4.20) with the sample estimators in (4.19).

4.2.2 Feature Definitions

Using the nomenclature introduced in Section 4.2.1, Table 4.6 lists the features extracted from the signal segments. Each feature evaluates to a single rational number per signal segment so that the process of featurization can be understood as a mapping from the N -dimensional complex signal sample space to the k -dimensional feature space, $k = 30$ being the total number of extracted features. We subdivide the feature set into spectral features, number features, and statistical features.

Spectral Features

The following subset of spectral features is taken from Azzouz and Nandi's seminal publication on modulation classification [20] and is widely adopted in the literature today. In their work, the authors have demonstrated the capability of these features to differentiate between low-order amplitude, phase, and frequency-shift keying modulations. The authors recommend a threshold range of $A_t \in [0.858, 1]$ for analog modulation and $A_t \in [0.9, 1.05]$ for digital modulation for the definition on non-weak samples. In accordance with their experiments we pick $A_t = 1$. In a few cases, the names of the variables have been altered to avoid confusion with other features added to the set.

#	Feature	Description
(f1)	$\gamma_{1,max}$	Max. DFT value of centered-normalized instantaneous amplitude
(f2)	σ_{aa}	Std. dev. of absolute centered-normalized instantaneous amplitude
(f3)	σ_a	Std. dev. of centered-normalized instantaneous amplitude
(f4)	μ_{42}^a	Kurtosis of centered-normalized instantaneous amplitude
(f5)	$\sigma_{ap,CNL}$	Std. dev. of absolute centered nonlinear instantaneous phase
(f6)	$\sigma_{dp,CNL}$	Std. dev. of centered nonlinear direct instantaneous phase
(f7)	σ_{af}	Std. dev. of absolute centered-normalized instantaneous frequency
(f8)	μ_{42}	Kurtosis of centered-normalized instantaneous frequency
(f9)	SNR	SNR estimate
(f10)	$\gamma_{2,max}$	Max. DFT value of second power centered-normalized instantaneous amplitude
(f11)	$\gamma_{4,max}$	Max. DFT value of fourth power centered-normalized instantaneous amplitude
(f12)	$\sigma_{ap,C}$	Std. dev. of absolute centered instantaneous phase
(f13)	$\sigma_{dp,C}$	Std. dev. of centered direct instantaneous phase
(f14)	Ψ_{max}	Max. value of auto-correlation function
(f15)	σ_R	Std. dev. of spectrum
(f16)	μ_{42}^R	Kurtosis of spectrum
(f17)	$PAPR$	Peak-to-average power ratio
(a) Spectral features		
#	Feature	Description
(f18)	N_c	Number of non-weak samples of first degree
(f19)	$N_{c,2}$	Number of non-weak samples of second degree
(f20)	ZC	Total number of zero-crossings
(b) Number features		
#	Feature	Description
(f21)	\hat{C}_{20}	Second-order cumulant
(f22)	\hat{C}_{21}	Second-order cumulant
(f23)	$ \tilde{C}_{40} $	} Normalized, noise corrected fourth-order cumulants
(f24)	$ \tilde{C}_{41} $	
(f25)	\tilde{C}_{42}	
(f26)	$ \tilde{C}_{60} $	} Normalized, noise corrected sixth-order cumulants
(f27)	$ \tilde{C}_{61} $	
(f28)	$ \tilde{C}_{62} $	
(f29)	\tilde{C}_{63}	
(f30)	μ_{42}	Ratio of the fourth-order even moment to the second order moment
(c) Statistical features		

Table 4.6: Features extracted from the recorded RF signal segments.

The first feature is defined as

$$\gamma_{1,max} = \max |DFT(A_{CN})|^2 / N_s, \quad (f1)$$

which is the maximum value of the spectral power density of the normalized and centered instantaneous amplitude using the DFT of length N_s . In our experiments, we use $N_s = 1000$.

Secondly, the standard deviation of the absolute value of the normalized and centered instantaneous amplitude is defined as

$$\sigma_{aa} = \sqrt{\frac{1}{N} \left(\sum_{n=1}^N A_{CN}^2[n] \right) - \left(\frac{1}{N} \sum_{n=1}^N |A_{CN}[n]| \right)^2}. \quad (f2)$$

Including only the non-weak segment of the signal,

$$\sigma_a = \sqrt{\frac{1}{N_c} \left(\sum_{S_{nw}} A_{CN}^2[n] \right) - \left(\frac{1}{N_c} \sum_{S_{nw}} |A_{CN}[n]| \right)^2} \quad (f3)$$

is the standard deviation of the normalized and centered instantaneous amplitude. The kurtosis of the normalized and centered instantaneous amplitude is further defined as

$$\mu_{42}^a = \frac{\frac{1}{N} \sum_n A_{CN}^4[n]}{\left(\frac{1}{N} \sum_n A_{CN}^2[n] \right)^2}. \quad (f4)$$

All following phase-dependent features discard I/Q samples below a certain normalized amplitude threshold A_t due to their high susceptibility to noise by only evaluating the non-weak samples of first degree as defined in (4.7).

The first phase feature, $\sigma_{ap,CNL}$, is defined as

$$\sigma_{ap,CNL} = \sqrt{\frac{1}{N_c} \left(\sum_{S_{nw}} \phi_{CNL}^2[n] \right) - \left(\frac{1}{N_c} \sum_{S_{nw}} |\phi_{CNL}[n]| \right)^2} \quad (f5)$$

and represents the standard deviation if the absolute value of the centered nonlinear instantaneous phase. Similarly,

$$\sigma_{dp,CNL} = \sqrt{\frac{1}{N_c} \left(\sum_{S_{nw}} \phi_{CNL}^2[n] \right) - \left(\frac{1}{N_c} \sum_{S_{nw}} \phi_{CNL}[n] \right)^2} \quad (f6)$$

is the standard deviation of the centered nonlinear direct instantaneous phase.

For frequency-dependent features, our definitions deviate slightly from the original ones, which uses the same definition of non-weak samples for phase and frequency features. Since the instantaneous frequency also depends on the phase of the

previous samples, it is a sensible choice to only consider non-weak samples of second degree and thus discard those I/Q samples with a weak preceding sample.

Given this distinction,

$$\sigma_{af} = \sqrt{\frac{1}{N_{c,2}} \left(\sum_{S_{nw,2}} f_{CN}^2[n] \right) - \left(\frac{1}{N_{c,2}} \sum_{S_{nw,2}} |f_{CN}[n]| \right)^2} \quad (f7)$$

is the standard deviation of the absolute value of the normalized and centered instantaneous frequency and

$$\mu_{42}^f = \frac{\frac{1}{N_{c,2}} \sum_{S_{nw,2}} f_{CN}^4[n]}{\left(\frac{1}{N_{c,2}} \sum_{S_{nw,2}} f_{CN}^2[n] \right)^2} \quad (f8)$$

is the estimated kurtosis of the normalized and centered instantaneous frequency.

Expanding the original set of features from the seminal work, we further introduce the following features less frequently used in literature [91].

The receiver's estimate of the SNR, denoted by

$$S\hat{N}R = 10 \log \left(\frac{\hat{P}_S}{\hat{P}_N} \right). \quad (f9)$$

While the SNR itself is independent of the modulation scheme, we use it as a side information for the classifier because we assume some of the other features to be SNR-dependent.

Analogously to feature (f1), the maximum value of the spectral power density of the k -th power of the normalized and centered instantaneous amplitude is defined as

$$\gamma_{2,max} = \max |DFT(A_{CN}^2)|^2 / N_s \quad (f10)$$

for $k = 2$ and

$$\gamma_{4,max} = \max |DFT(A_{CN}^4)|^2 / N_s \quad (f11)$$

for $k = 4$.

Omitting the unwrap operation in (f5),

$$\sigma_{ap,C} = \sqrt{\frac{1}{N_c} \left(\sum_{S_{nw}} \phi_C^2[n] \right) - \left(\frac{1}{N_c} \sum_{S_{nw}} |\phi_C[n]| \right)^2} \quad (f12)$$

is the standard deviation of the absolute value of the centered instantaneous phase. Similarly,

$$\sigma_{dp,C} = \sqrt{\frac{1}{N_c} \left(\sum_{S_{nw}} \phi_C^2[n] \right) - \left(\frac{1}{N_c} \sum_{S_{nw}} \phi_C[n] \right)^2} \quad (f13)$$

is the standard deviation of the centered direct instantaneous phase.

With the auto-correlation function of the received signal, we also use its maximum value

$$\Psi_{max} = \max [\Psi_{yy}[n]] \quad (\text{f14})$$

as an additional feature.

Given the power spectrum $R[k] = PS_{rms}[k]$ of the signal with a fast fourier transform (FFT) length of $N = 1000$,

$$\sigma_r = \text{std} [R[k]] \quad (\text{f15})$$

denotes the standard deviation of the spectrum and

$$\mu_{42}^R = \frac{\mathbb{E}[R^4[k]]}{(\mathbb{E}[R^2[k]])^2} \quad (\text{f16})$$

is the estimated kurtosis of the spectrum.

Finally,

$$PAPR = \frac{\max_n [|A[n]|^2]}{\frac{1}{N} \sum_{n=1}^N |A[n]|^2} \quad (\text{f17})$$

is the peak-to-average power ratio.

Number Features

In addition to the spectral features, we add a small number of count-based features. The number of non-weak samples of the first degree is defined as

$$N_c = |S_{nw}| \quad (\text{f18})$$

and the number of non-weak samples of second degree is defined as

$$N_{c,2} = |S_{nw,2}|. \quad (\text{f19})$$

The total number of zero-crossings is defined by

$$ZC = ZC_I + ZC_Q \quad (\text{f20})$$

as the sum of zero-crossings of the in-phase component $y_I[n]$ and the quadrature component $y_Q[n]$. All three features above have to be normalized with respect to the number of recorded I/Q samples N .

Statistical Features

Based on the moments-to-cumulants formula given in (4.20) and the sample moments from (4.19), we introduce a number of statistical features adopted from [61, 63].

The estimated second-order cumulants are defined as

$$\hat{C}_{20} = \hat{M}_{20} = \left| \frac{1}{N} \sum_n y^2[n] \right| \quad (\text{f21})$$

and

$$\hat{C}_{21} = \hat{M}_{21} = \frac{1}{N} \sum_n |y[n]|^2. \quad (\text{f22})$$

Since cumulants of independent variables are additive and the cumulants of larger than second order of gaussian processes are zero [94], the high-order cumulants defined in (4.21) are relatively resilient to noise and can be leveraged as features even at low SNR [61]. The following features are normalized with respect to the signal energy to avoid scaling problems for different power levels. In the presence of noise, the correction term has to exclude the variance of the noise σ_N^2 so that the cumulant is normalized with respect to the power of the transmitted signal.

The normalized, estimated fourth-order cumulants are defined as

$$|\tilde{C}_{40}| = \left| \frac{\hat{C}_{40}}{(\hat{C}_{21} - \sigma_N^2)^2} \right|, \quad (\text{f23})$$

$$|\tilde{C}_{41}| = \left| \frac{\hat{C}_{41}}{(\hat{C}_{21} - \sigma_N^2)^2} \right|, \quad (\text{f24})$$

$$\tilde{C}_{42} = \frac{\hat{C}_{42}}{(\hat{C}_{21} - \sigma_N^2)^2}. \quad (\text{f25})$$

The normalized, estimated sixth-order cumulants are defined as

$$|\tilde{C}_{60}| = \left| \frac{\hat{C}_{60}}{(\hat{C}_{21} - \sigma_N^2)^3} \right|, \quad (\text{f26})$$

$$|\tilde{C}_{61}| = \left| \frac{\hat{C}_{61}}{(\hat{C}_{21} - \sigma_N^2)^3} \right|, \quad (\text{f27})$$

$$|\tilde{C}_{62}| = \left| \frac{\hat{C}_{62}}{(\hat{C}_{21} - \sigma_N^2)^3} \right|, \quad (\text{f28})$$

$$\tilde{C}_{63} = \frac{\hat{C}_{63}}{(\hat{C}_{21} - \sigma_N^2)^3}. \quad (\text{f29})$$

In our experiments, we use the estimate of SNR from (f9)

$$S\hat{N}R \approx \frac{\hat{C}_{21} - \sigma_N^2}{\sigma_N^2} \quad (\text{4.22})$$

to calculate the correction term for the fourth-order cumulants

$$\tilde{C}_{4k} = \tilde{C}_{4k} \left(\frac{1 + S\hat{N}R}{\hat{C}_{21} S\hat{N}R} \right)^2, \quad (\text{4.23})$$

and the sixth-order cumulants

$$\tilde{C}_{6k} = \tilde{C}_{6k} \left(\frac{1 + S\hat{N}R}{\hat{C}_{21}S\hat{N}R} \right)^3. \quad (4.24)$$

Finally, the ratio of the fourth-order even moment to the second order moment is defined as

$$\nu_{42} = \frac{M_{42}}{M_{21}}. \quad (\text{f30})$$

4.3 DATA PRE-PROCESSING

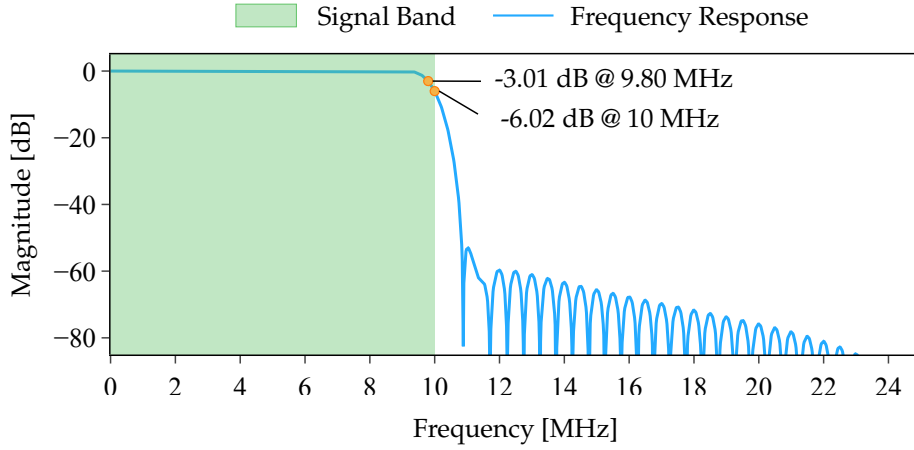
The recorded signals need to be processed so that they can be used as inputs for our DL algorithms. The first step is to attenuate the noise generated outside of the signal bandwidth of interest due to the oversampling of the input signal. This noise is attenuated using a linear phase finite impulse response filter whose magnitude and phase response are shown in Fig. 4.1. The characteristic markers show the filter cut-off frequency. Due to the linear phase and the resulting constant group delay, the signal in the bandwidth of interest is not distorted.

After filtering, the signal is split into 10 subsequences of 1ms each. The first and the last segments are discarded to avoid the recordings contaminated by the transient effects of the receiver or the filter. Each of the remaining subsequences represents one signal segment that can be used as an input for our learning algorithms and is saved in the signal dataset. At this point, each signal segment comprises 50,000 I/Q samples. In most real-world scenarios, we would want to work with even smaller signal recordings for the reasons of faster computation times giving us real-time predictions. For this purpose, we subsample signals of 1024 I/Q samples from the original recordings. Consequently, this also results in a larger signal dataset which is often a requirement for the DL algorithms. The last step in pre-preprocessing involves scaling the signal segments to zero-mean and unit variance. This is an important step as it allows our DL networks to learn efficiently while consuming less time. Finally, these scaled signal segments having 1024 I/Q samples per segment form our processed dataset that can be used to train our neural networks. Table 4.7 lists the configurations of the processed datasets.

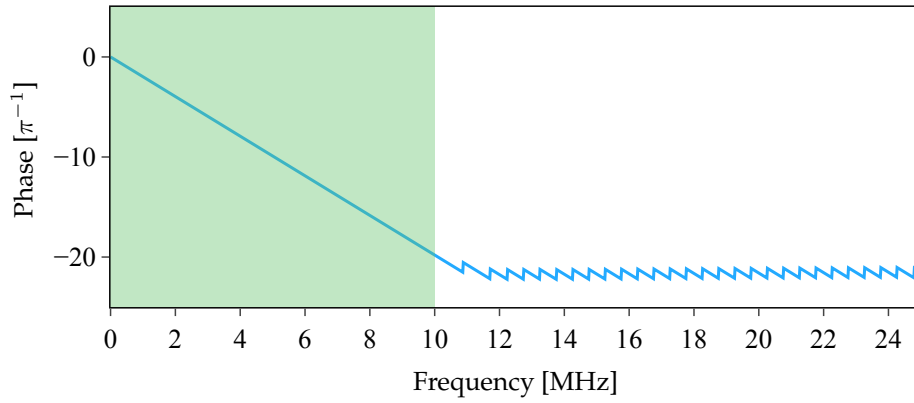
4.4 EMPLOYED MACHINE LEARNING ALGORITHMS

We have previously described the classification algorithms in Section 2.2.1. In this section, we briefly discuss our network architectures and detail the hyperparameters that configure them. Our choice of classifiers include XGBoost (Section 2.2.1.1), CNN (Section 2.2.1.2) and ResNet (Section 2.2.1.3). XGBoost is the most prominent classifier in ML and we use it as our baseline. To improve on its performance, we explore the DL networks of CNN and ResNets.

We optimize the hyperparameters of the XGBoost classifier for our signal data using the *gridCV* [95] search method. The optimized parameters are listed in Table 4.8. The *learning_rate* shrinks the feature weights at each round to prevent overfitting, *num_boost_round* specifies the maximum number of training epochs, *max_depth* pa-



(a) Magnitude Response



(b) Phase Response

Figure 4.1: Frequency response for positive frequencies of the low pass filter used for baseband filtering during pre-processing.

parameter gives the maximum depth of our decision tree, *gamma* controls overfitting, *col-sample_bytree* gives the subsampling ratio of features when a new tree is constructed, *min_child_weight* gives the minimum weight of the leaf node below which further partition is not possible, *objective* specifies the learning objective (softmax in our case), and *early_stopping_rounds* helps in stopping the algorithm if the loss has not improved for a number of iterations denoted by it.

Next, we delve into our DL algorithms. Our CNN model (*cf.* Fig. 2.10) consists of 6-2D convolution layers and 3 FC layers. Additionally, each FC layer is followed by a ReLU activation and dropout layer (not shown in figure). All the network parameters are listed in Table 4.9 and their significance is detailed in Section 2.2.1.2.

We can further improve on the performance of CNNs using ResNet architectures. This helps in scaling the number of layers in an efficient manner. We employ a ResNet

Tx	Rx	Impairment	Interfering Modulation	Number of Samples (\sim)
VSG	SA	Noise	-	6.1M
SDR	SDR	Noise	-	6.1M
VSG	SA	CFO (1 ppm)	-	6.1M
VSG	SA	CFO (5 ppm)	-	6.1M
SDR	SDR	VSG	{SC BPSK, SC 16-QAM, OFDM 64-QAM}	4.5M
SDR	SDR	VSG	{SC, OFDM} \times {BPSK, QPSK, 16-QAM, 64-QAM}	2.8M

Table 4.7: Configuration of the processed datasets where each signal segment consists of 1024 I/Q samples.

Parameter	XGB
<i>learning_rate</i>	0.2
<i>num_boost_round</i>	200
<i>max_depth</i>	18
<i>gamma</i>	0.4
<i>colsample_bytree</i>	0.3
<i>min_child_weight</i>	3
<i>objective</i>	multi:softprob
<i>early_stopping_rounds</i>	10

Table 4.8: Hyperparameters for XGBoost (cf. Section 2.2.1.1).

Parameter	CNN
<i>learning_rate</i>	0.01
<i>momentum</i>	0.9
<i>epochs</i>	30
<i>batch_size</i>	512
<i>out_channels</i>	64
<i>kernel_size</i>	3 \times 3
<i>stride</i>	1
<i>pool_size</i>	3 \times 3
<i>dropout</i>	0.5
<i>out_features</i> (FC layer)	128

Table 4.9: Hyperparameters for CNN (cf. Section 2.2.1.2).

architecture of 101 layers which is more complex than the 34 layer architecture shown in Fig. 2.16. We show the network parameters for the convolution layers in ResNet-101 in Table 4.10. Other learning parameters such as *learning_rate*, *momentum*, *epochs*, *batch_size*, and *out_features* are kept similar to CNN, see Table 4.9.

4.5 EVALUATION PROCEDURE

In this section, we present our evaluation approach for the performance of AMC task on the RF signal datasets described and pre-processed in Sections 4.1 and 4.3 respectively, using the DL algorithms detailed in Section 4.4. We perform an extensive evaluation of the available datasets starting with exploratory analysis to visualize distinct patterns. This is followed by computing the overall classification accuracy of our networks. Consequently, a comparative analysis of our chosen networks is carried out. Up to this point, the evaluations are performed on matching training and testing data. To explore the implications of having mismatched data, we compute

Layer Name	Parameters
<i>conv1</i>	$7 \times 7, 64$, stride 2
	3×3 max pool, stride 2
<i>conv2_x</i>	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
<i>conv3_x</i>	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
<i>conv4_x</i>	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$
<i>conv5_x</i>	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1 avg. pool, stride 1

Table 4.10: Hyperparameters for the convolutional layers of ResNet-101 (*cf.* Section 2.2.1.3).

generalization accuracy and try to improve on this through the application of transfer learning.

Once we have all these results, we delve deeper into explaining the predictions of our classifiers. We perform sequential testing to identify which signal segments are incorrectly predicted. Next, we interpret the effect of features on our model predictions. Finally, we analyze other datasets using our evaluation methodology to confirm the robustness of our results.

For all the considered evaluation procedures, we focus on an *eight-class classification* problem with 4 modulations considered for each of SC and OFDM modulation schemes. It is also worth mentioning that our datasets are balanced i.e. they consist of an equal number of signals from each modulation class. This generally helps in better performance for our networks. The following subsections elaborate on our evaluation process.

4.5.1 Exploratory Data Analysis

Before evaluating the classification performance of our networks, we would like to find any underlying distributions and patterns that may be present in our data. This gives us a first visual impression on how the data is structured and further information about data similarities and outliers can be identified. For this purpose, we utilize the algorithm introduced in Section 2.2.2.

High dimensional data can be represented in a low dimensional space (usually 2D) using the SOM algorithm (*cf.* Section 2.2.2.1). To this end, we create a 32×32 SOM to represent 5000 randomly sampled signal samples in a 2D space such that they can

be interpreted visually. The input for SOM are the 128 features extracted from CNN using our dataset recorded under the baseline configuration. We train the SOM for 10,000 iterations with learning rate $\alpha = 0.3$ and neighbourhood function parameter $\sigma = 16$. We implement SOM in the pytorch framework [96] and utilize batch training which makes our implementation easily scalable.

4.5.2 Evaluation of Overall Classification Accuracy

To get a comprehensive view of the performance of our neural networks on the AMC task, we require a metric that describes their performance under the presence of all variations of the radio environment. For our work, we refer to this metric as *overall accuracy*. The following paragraphs give a detailed explanation for evaluating this metric.

The first step is to choose an appropriate dataset from the available sets described in Section 4.1. For our evaluation, we consider datasets according to all scenarios detailed in Section 4.6. The chosen set consists of raw I/Q signals encapsulating all the 8 modulations and spanning a wide range of SNR from 0-20 dB. This dataset is pre-processed as described in Section 4.3 and split into 3 sets of train (75%), validation (5%) and test (20%). We train our CNN on data from the training set, as explained in Section 2.2.1.2. The validation and test sets are used to analyze the performance of our network during and after the training process respectively.

Once our model is trained, we use it to make predictions on the unseen data from the test set. These output predictions are the modulation schemes that are compared against the true modulations and consequently, the overall accuracy is given as the percentage of signal segments correctly identified based on their modulation scheme.

4.5.3 Evaluation for Comparative Analysis

With the recent advances in the field of DL, many state-of-the-art algorithms have been developed. Depending on our use case, we can choose from a wide range of these algorithms. Our choice of DL networks include CNN and ResNet-101. The objective is to present a fair comparison between the performance of these networks and gain insights for selecting models based on a given scenario.

For this evaluation, we compute the *overall accuracy* metric for both the networks under consideration. We follow the same methodology as stated in the previous section and the ResNets are trained, validated, and tested in the same fashion as CNNs. Again, this analysis is performed for all scenarios detailed in Section 4.6.

4.5.4 Evaluation of Generalization Accuracy

Our previous evaluations considered training and test sets taken from the same dataset. This may suffice in a controlled environment but the real-world environment poses challenges to this approach. We can only train our networks for a limited number of modulations and we may encounter different modulations during the test phase. Thus, we need to analyse the network performance when training and test sets are misaligned. We achieve this by computing the *generalization accuracy* metric.

. We define the generalization accuracy of the model as the percentage of correctly identified signals during the test based on the modulation scheme when the training set consisted of only a fraction of variations present at the testing time. The training and testing sets are subject to the evaluation scenarios described in Section 4.6. The evaluation procedure though remains similar to the one used for overall accuracy.

4.5.5 Evaluation using Transfer Learning

Usually, models trained on signal segments recorded from a particular environment do not generalize well when tested on signals recorded from other environments. But it would be very useful if these pre-trained models could be reused because it would save training time and resources. Most importantly, we wouldn't require to gather datasets for every environment which is not a trivial task. All these benefits can be reaped by using the concept of *transfer learning* (refer Section 2.2.3).

For evaluation using transfer learning, we train our model on a training set that captures only a specific variation of the radio environment. We use this trained model as a base to train shallow classifiers on training sets capturing other variations in the radio environment, e.g., we train a model on signals recorded under no interference and use this model to train a classifier for signals recorded under interference conditions using only a little amount of training data for the interference environment.

The pre-trained model can be utilized for the new task in two ways: freezing and fine-tuning (see Section 2.2.3.1). We use a hybrid approach of freezing and fine-tuning to optimize for the best results. We train on the new dataset for 15 epochs. From epoch 0 to 4, the feature extractor (pre-trained network) is frozen except for the batch normalization layers. The batch normalization layers and the parameters of the classifier are trained as a single parameter group with a learning rate of $\alpha = 1e - 2$. From epoch 5 to 9, the last two layer groups of the pre-trained network are unfrozen (here, the convolution layers) and added to the optimizer with a new parameter group with $\alpha = 1e - 4$ (while $\alpha = 1e - 3$ for the first parameter group in the optimizer). Eventually, from epoch 10, all the remaining layer groups of the pre-trained network are unfrozen and added to the optimizer as a third parameter group. From epoch 10, the parameters of the pre-trained network are trained with $\alpha = 1e - 5$ while those of the classifier are trained with $\alpha = 1e - 4$.

We employ the transfer learning approach for scenarios covered in Section 4.6, specifically in cases where the generalization accuracy of the classifier is very poor.

4.5.6 Sequential Testing

During pre-processing, we subsample our signal segments to have 1024 I/Q samples each whereas, the original signal recording consists of 50,000 I/Q samples each (refer Section 4.3). The goal of sequential testing is to identify portions of the original signal recording that are incorrectly predicted by our network.

A signal comprised of 50,000 I/Q samples can be split into 48 signals of length 1024. Thus every set of 48 signal samples in our processed datasets denote one original signal recording. For this particular analysis, we only consider train and test sets from the interference scenario (discussed in Section 4.6) as it is the most challenging radio environment. We use a ResNet-101 for our analysis and follow the training procedure

detailed in Section 2.2.1.2. The trained network is used to predict the modulations for the unseen test signals. Once we have the modulation predictions, we combine the signals back to their original form and take note of the segments with incorrect predictions. The next step is to count the incorrect predictions at each of these 48 signal segments for all the considered modulation schemes. Finally, we normalize these incorrect predictions and visually present them as stacked bars (representing modulation schemes) for each of the 48 segments. This gives us an insight into the signal portions that are most susceptible to an erroneous prediction by our network.

4.5.7 Evaluation of Feature Importance

Ideally, we would want to interpret the predictions of our DL networks. In most of the cases, this is difficult to achieve because these networks are treated as black-box models. Our case is similar, whereby features extracted from the signal data are represented in the form of weight values that are difficult to interpret. On the other hand, ML algorithms are interpretable because they take as input a known set of features whose contribution to the model predictions can be calculated.

Consequently, we use the XGBoost algorithm (*cf.* Section 2.2.1.1) and interpret its predictions. As XGBoost takes as input, the featurized data, we hand-craft 30 unique features as detailed in Section 4.2 for each signal segment in our dataset. This featurized data is used to train our model and the output predictions are explained with SHAP (*cf.* Section 2.2.4.1). Particularly, we use an efficient variation of SHAP known as TreeSHAP (*cf.* Section 2.2.4.1) for our task.

This gives us the contributions (shapely values) of all signal features to the modulation prediction. This allows us to understand the importance of each feature for a particular modulation scheme and paves the way for efficient feature selection to get the best results in an optimized manner.

4.5.8 Evaluation over Multiple Datasets

Apart from analysing our own RF signal dataset, we also analyse the *deepsig* dataset [24] that is available online¹. This dataset includes both synthetically simulated channel effects and over-the-air recordings of 24 digital and analog modulation types which are listed in Table 4.11. The dataset is comprised of ~ 2.5 M signal segments as complex floating points, each 1024 samples long and spanning across SNR in the range $[-20, 30]$ dB.

Modulation Type	Modulation classes
Analog	{AM-SSB-SC, AM-SSB-WC, AM-DSB-SC, AM-DSB-WC, FM}
Digital	{OOK, 4ASK, 8ASK, BPSK, QPSK, 8PSK, 16PSK, 32PSK, 16APSK, 32APSK, 64APSK, 128APSK, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, GMSK, OQPSK}

Table 4.11: The set of 24 modulations present in the *deepsig* dataset.

¹http://opendata.deepsig.io/datasets/2018.01/2018.01.OSC.0001_1024x2M.h5.tar.gz

We consider two variations of the dataset: the first variation consists of the complete set of 24 modulation schemes which is a comparatively difficult set for our classification problem. The second set consists of only some selected digital modulations, namely, BPSK, QPSK, and variations of QAM. This set is a simpler version and also lets us make a direct comparison with our original dataset. Additionally, we also featurize the complete dataset with our set of 30 chosen features listed in Table 4.6 to make its analysis with the XGBoost algorithm possible.

We compare the performance of all our classifiers (Section 2.2.1) on these datasets and verify their robustness against the results obtained on our original dataset. Moreover, we use the transfer learning approach to transfer knowledge of the model trained on the second variation of the deepsig data to our dataset recorded under the influence of noise and vice-versa. This gives us an idea if knowledge transfer between two similar but differently recorded datasets results in acceptable performance.

4.6 EVALUATION SCENARIOS

To get a comprehensive view of our classifier performance, we need to test it under a variety of scenarios that we may encounter in the real-world environment. To this extent, we consider the following evaluation scenarios based on the recording configurations presented in Section 4.1.

4.6.1 Baseline Scenario

The baseline scenario is based on the recordings as per the baseline configuration described in Section 4.1. As already stated, this is the most benign radio environment with no interference and a high SNR of 20 dB. These recorded signals result in a dataset of ~ 1.2 M samples after pre-processing as detailed in Section 4.3. The performance of the classifiers on this scenario gives us an upper bound for the maximum achievable accuracy. Consequently, this data allows us to perform exploratory data analysis using SOM because the CNN extracted features are quite robust and show distinct patterns for the eight modulation schemes under consideration.

4.6.2 Sensitivity Analysis

The baseline scenario serves as a reference for our further evaluation. The objective is the understand and analyze the classifier performance under the influence of varying radio environments, hardware, and training parameters. The datasets used for this evaluation are based on signals recorded under the parameter sweep configuration described in Section 4.1. The following paragraphs describe the effect of each parameter keeping all the other factors constant.

Influence of Noise. Using the dataset of signals recorded using VSG as transmitter, Signal Analyzer (SA) as receiver and spanning across SNR levels of $\text{SNR} \in \{0 \text{ dB}, 5 \text{ dB}, 10 \text{ dB}, 15 \text{ dB}, 20 \text{ dB}\}$ (see Table 4.7), we calculate the overall classification accuracy, and generalization accuracy for the classifier. We also make a comparison between different classifiers to improve the overall accuracy. Lastly, we use the transfer learning approach to enhance the generalization accuracy by utilizing the knowledge of a

model trained on signals having a particular SNR value to train models for signals with other varying SNR levels. Here each dataset with signals recorded under a certain SNR consists of $\sim 1.2\text{M}$ samples.

Influence of Transceiver Hardware. Our prior evaluations are done using signals recorded by high-quality lab equipment. We now analyze the influence of low-grade recording equipment (SDR) on the overall accuracy of the classifier and compare it against other classifiers. We further compute the generalization accuracy where the training and test sets are misaligned and comprise of signals recorded using heterogeneous hardware. We also improve the performance of the model on misaligned sets using the transfer learning approach. This involves fine-tuning the model trained on signals from particular hardware for signals using different hardware and vice-versa. The datasets for this scenario are listed in Table 4.7 and consist of $\sim 6.1\text{M}$ samples each, spanning across all SNRs.

Influence of CFO. The baseline scenario assumes complete frequency synchronization between the transmitter and the receiver. This is usually not the case and frequency drift is common due to the misalignment of the transmitter and receiver clocks. Moreover, the doppler effect worsens the situation when mobility is taken into account. This drift is called the CFO. We study the overall accuracy of the classifier under the influence of CFO levels uniformly varying between ± 1 ppm and ± 5 ppm. We also perform a comparative analysis between different classifiers for a comprehensive analysis. Finally, we compute the generalization accuracy using a combination of misaligned training and test sets where the training set can be one that has no CFO, or CFO ± 1 ppm, or CFO ± 5 ppm. The details for the utilized datasets are presented in Table 4.7.

Influence of Length of Signal Segments. As evident from our data pre-processing approach in Section 4.3, we sample the signals to have 1024 I/Q samples per signal segment. In practice, we can sample the signals to have lesser or more samples depending on the use case and the trade-off between the training time and accuracy. For this particular analysis, we sample the signals to have 256, 512 and 2048 I/Q samples per segment from the dataset of recordings impaired by noise and having VSG as the transmitter and SA as the receiver (see Table 4.7). We compute the overall accuracy of the classifier for all these variations of signal lengths. The datasets having signals of length 256, 512 and 2048 contain $\sim 25\text{M}$, $\sim 12\text{M}$, and $\sim 3\text{M}$ signals respectively.

Influence of Size of Training Data. We know that DL networks perform well with more training data. But a large amount of training data may not be always readily available. Thus, the objective of this evaluation is to analyse the effect of the size of training data on the accuracy of the classifier using the dataset with recordings impaired by noise and VSG as a transmitter (see Table 4.7). We train our classifier with 5 different data splits starting at 15% and ending at 75% with increments of 15% in between. The classifier is then tested against the remaining test set split and the overall accuracy is reported.

Influence of Unknown Signals. We may encounter many different signals in a real-world environment which may consequently have different modulations. It is not possible to train our classifier with all these modulation signals due to the reasons of data unavailability and limited computing resources. This poses a serious problem for the classifier as it will always predict the incoming unknown signal incorrectly. We can tackle this issue with the use of open-set classification (*cf.* Section 2.2.5) which involves the addition of an extra output class to the classifier during the prediction phase that allows the unknown signals to be predicted as this new class instead of mixing with the other modulation classes. For this analysis, we train our model on the signals recorded under the baseline configuration (*cf.* Section 4.1) and test it with signals of 19 unknown modulations taken from the deepsig data (*cf.* Table 4.11) along with the signals taken from our original test set. Finally, we report the overall accuracy of the classifier.

4.6.3 Interference Scenario

The most prominent issue for a real-world radio system is the presence of interference in the environment. This can be due to the presence of other radio systems in the vicinity transmitting signals at the same frequency and time. This interference hinders the performance of the classifier and poses a tough challenge to the AMC task.

In this scenario, we study the impact of modulated Co-channel Interference (CCI) on the accuracy of the classifier. Our dataset consists of signals recorded based on the interference configuration described in Section 4.1 and impaired with 3 different interference signals, namely, SC BPSK, SC 16-QAM, OFDM 64-QAM (see Table 4.7). We make the AMC task more challenging by using SDRs as the transmitter and receiver and high-grade lab hardware as the interferer. We analyze the impact of all 3 modulated CCI on the overall accuracy of the classifier across a variety of SIRs in the range of {5 dB, 10 dB, 15 dB, 20 dB} at SNR=10 dB. Further, we make a comparative analysis of various classifiers to get the best performance for this task. We also study the generalization accuracy for the classifier where the training set consists of non-interfered data and the testing is done on the interfered data. Finally, we optimize the results for the generalization accuracy using the transfer learning approach to fine-tune a model pre-trained on the non-interfered data by utilizing some data from the interfered training set and test it against the interfered test set.

4.6.4 Real-World Scenario

Up to now, all the discussed scenarios considered impairments due to a standalone parameter. In a realistic environment, these impairments do not occur one-by-one in a controlled fashion. Thus, we need to simulate an environment that captures all the rich variations discussed so far to get a comprehensive view of the classifier performance. Our dataset comprises of the non-interfered data recorded using SDRs and the data interfered by all the 8 modulations (see Table 4.7). We compute the overall accuracy of the classifier to study how well it generalizes to the heterogeneous set.

RESULTS

This chapter presents the results of AMC performance following the methodology described in the previous chapter. In Section 5.1, we present the results of the baseline scenario evaluation. Section 5.2 discusses the results of sensitivity analysis followed by the examination for the interference and real world scenarios in Sections 5.3 and 5.4, respectively. Section 5.5 presents the results from our sequential testing methodology. Further, Section 5.6 presents our observations from the examination of model predictions. Lastly, we discuss the results from our evaluations on other modulation datasets in Section 5.7.

5.1 BASELINE SCENARIO

This section presents the results of our deep neural networks for the baseline scenario described in Section 4.6.1. First, we explore the representation of CNN derived features in a low-dimension feature space. Second, we analyse the performance of different DL models.

5.1.1 Exploratory Data Analysis

All raw data needs to be analysed before it can be used for prediction tasks. This gives us an intuition on how the data looks like in visual space and helps us detecting anomalies and make meaningful assumptions before presenting any results. Here we analyse the features generated by our CNN for a *eight-class* classification problem.

Low-Dimensional Data Representation. Fig. 5.1 shows a SOM generated from the CNN-extracted features for the baseline scenario. The 3D surface plot is generated by interpolating the U-matrix of a 32-by-32 SOM and visualizes the euclidean distance between feature maps. The scatter plot comprises of 5000 random points sampled from the dataset. These points have been extrapolated in height for better visualization. The color of the points depicts the modulation scheme and their opacity denotes the extent of activation. Furthermore, the darker points show that multiple samples overlap in a certain region and are very similar to each other in the feature space.

Fig. 5.1 shows the SOM generated using the complete dataset of SC and OFDM modulations. The map shows distinct separations between all modulation schemes. Moreover, the SC and OFDM modulation nodes lie on opposite ends of the map which shows a clear separation between the two modulation types. Among the SC modulation, BPSK, QPSK, and QAM nodes show a large relative distance between them except for 16-QAM and 64-QAM, which lie rather close but still form distinct

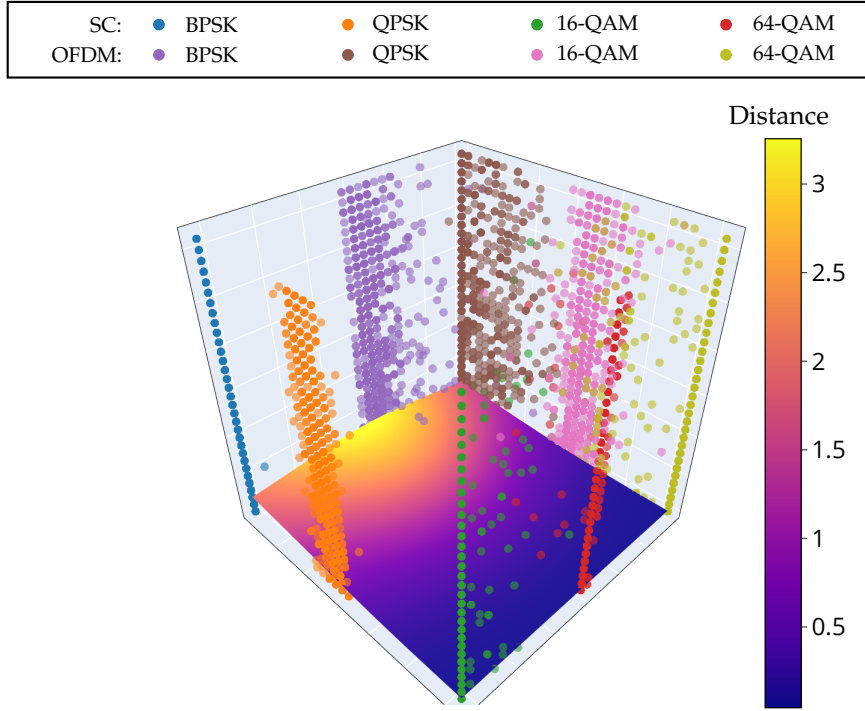


Figure 5.1: Topological SOM based on the features extracted from CNN for the baseline scenario. The colored circles represent the winning nodes per modulation scheme and the surface map depicts their separation distance. Lighter spots indicate large relative distances between the nodes.

clusters. Only some nodes of 16-QAM stray away from their cluster and blend with the 64-QAM nodes. For the OFDM modulation, we see that BPSK and QPSK nodes show the most separation amongst them whereas the 16-QAM and 64-QAM nodes are closely spaced and blend to some extent.

From these observations, we deduce that a relatively simple deep neural network like CNN can differentiate between the modulation schemes with good precision. The next section gives proof of this with the evaluation of model metrics like confusion matrix and accuracy.

5.1.2 Evaluation of Model Performance

This section presents the quantitative results of our evaluation. We first present the results of our custom CNN model for an *eight-class* classification problem. Further, we compare the performance of our DL model against novel feature-based ML algorithms.

Fig. 5.2 presents the accuracy of the CNN model on the baseline scenario. It is clearly evident from the confusion matrix that the model achieves perfect accuracy for all the modulation classes barring slight confusions amongst the OFDM QAM

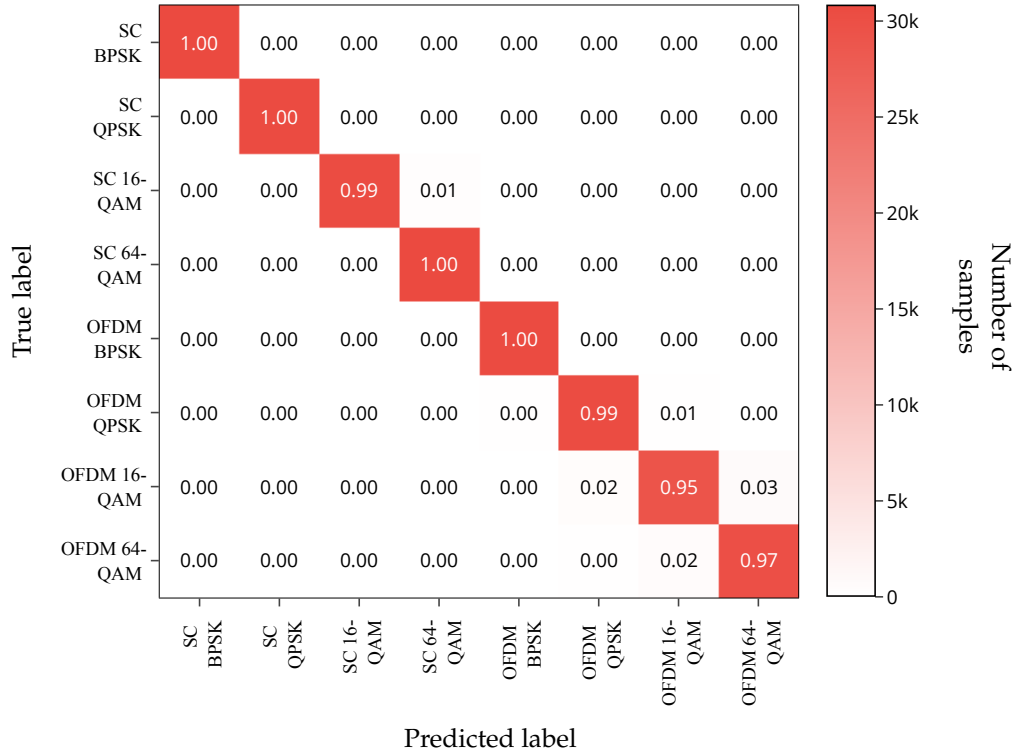


Figure 5.2: Confusion matrix of CNN for the baseline scenario. The numbers represent the accuracy of predicted class and the color intensity depicts the number of samples.

scenarios. This verifies our earlier observations from the exploratory data analysis (*cf.* Section 5.1.1).

Comparative Analysis. In the modern approach, DL models are used to generate their own complex set of features from raw data. On the other hand, in the traditional approach, ML models use hand-crafted features obtained from the raw data to perform the classification tasks. Here we provide a comparison between these two approaches. We take into consideration various ML models, namely SVM, kernelized SVM, Random Forest and ANN. A comprehensive feature set X comprising of spectral and statistical signal features is created for the evaluation of these models.

Table 5.1 shows the accuracy obtained by the classifiers on the baseline scenario. We observe that the CNN gives significant improvement over all other classifiers resulting in almost perfect accuracy. Random forests can produce better results in comparison with SVM and its kernelized variant (poly) but still fare worse than the CNN. The reason for this is their inability to distinguish between the OFDM modulation schemes.

Fig. 5.3 shows the confusion matrix for SVM classifier on the baseline scenario using feature set X . It can be observed that the classifier can disambiguate between the SC modulation schemes but fails in the case of the OFDM modulations. Consequently, the prediction accuracy for OFDM schemes lies between 56-71%.

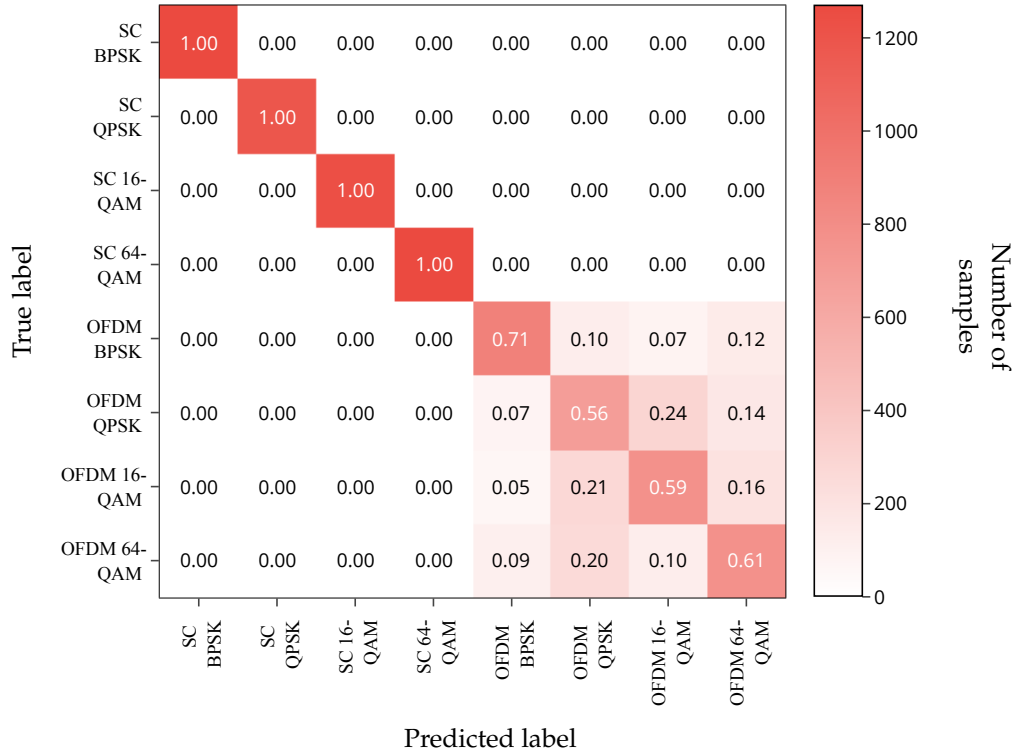


Figure 5.3: Confusion matrix of SVM classifier for the baseline scenario using feature set X . The numbers represent the accuracy of predicted class and the color intensity depicts the number of samples.

From the results of this section, we can conclude that feature-based ML algorithms are not sufficient to classify between the OFDM modulation schemes even under the highest integrity conditions. We can overcome this flaw using a fairly simple CNN model. Thus, from here on, we present the results obtained using DL algorithms.

5.2 SENSITIVITY ANALYSIS

This section presents the results of model performance under the influence of noise, hardware homogeneity, CFO, a variation of the signal segment and sample lengths, and under the presence of unknown signals as described in Section 4.6.2 of the previous chapter.

Classifier	SVM	POLY	FOREST	ANN	CNN
Accuracy	80.7%	85.6%	94.3%	79.9%	98.7%

Table 5.1: Accuracy of classifiers across all eight modulation schemes for the baseline scenario.

5.2.1 Effect of Noise

For evaluating the effect of noise on the AMC performance, we consider the dataset described in Section 4.6.2 with signals of $\text{SNR} \in \{0\text{dB}, 5\text{dB}, 10\text{dB}, 15\text{dB}, 20\text{dB}\}$ and all the parameters set to the baseline scenario.

Overall Accuracy. Table 5.2 shows the overall accuracy of CNN per SNR class when trained on signal segments spanning the whole range of possible SNR values. The results shows that the model is able to achieve perfect accuracy for SNR above 15 dB. At 10 dB SNR, the accuracy drops by $\sim 4\%$ to $\sim 94\%$ which is still an acceptable result. Below 10 dB SNR, the accuracy plummets to $\sim 83\%$ and $\sim 61\%$ for SNR of 5 dB and 0 dB respectively. The source of this confusion at 0 dB SNR is represented by Fig. 5.4. We observe that there is still a distinct separation between the SC and OFDM modulation schemes, though within them all the modulations confuse with each other. Only BPSK and to some extent QPSK signals of SC modulation are clearly distinguished from the other modulations. The main source of errors are the QAM signals in the SC case as they become completely inseparable. For the OFDM case, we notice that the pairs of (BPSK, QPSK) and (16-QAM, 64-QAM) confuse amongst each other resulting in accuracies ranging between $\sim 51\%$ and $\sim 57\%$ only.

SNR	0 dB	5 dB	10 dB	15 dB	20 dB	0-20 dB
Accuracy	60.8%	83.0%	94.1%	98.6%	99.4%	87.2%

Table 5.2: CNN accuracy on a dataset of signals with varying SNR levels, after training with all SNR levels.

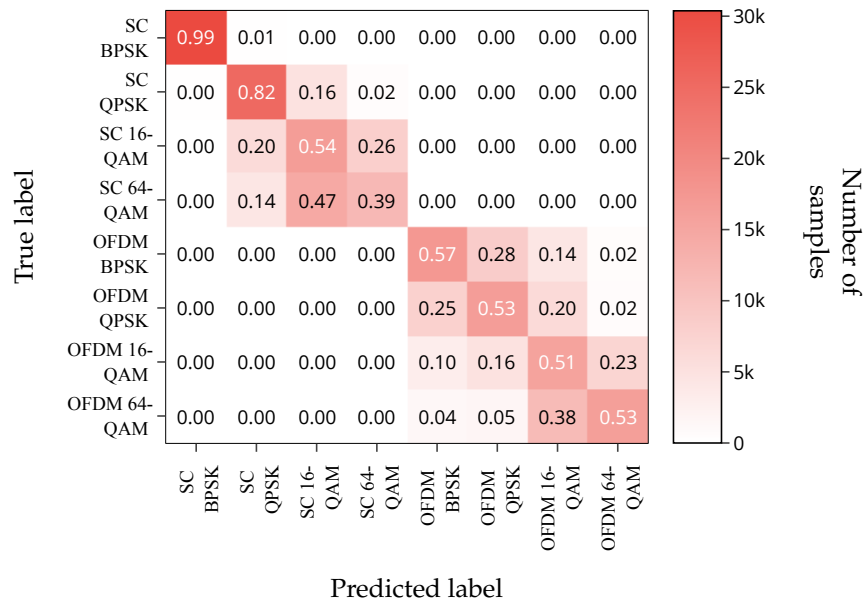


Figure 5.4: Confusion matrix of CNN evaluated for signals at SNR=0 dB, after training with all SNR levels.

Comparative Analysis. Since a simple CNN does not perform well for low SNR signals, we decide to use a more sophisticated CNN with increased depth called ResNet-101.

Fig. 5.5 shows how DL models compare against each other at different SNR levels when trained with a dataset comprising of all SNR values. We notice that ResNet enhances the accuracy significantly ($\sim 15\%$) at lower SNRs. For SNR exceeding 10 dB, both models perform well with the ResNet model achieving perfect classification accuracy. These results affirm that complex models can be deployed to tackle the effect of noise with high efficiency.

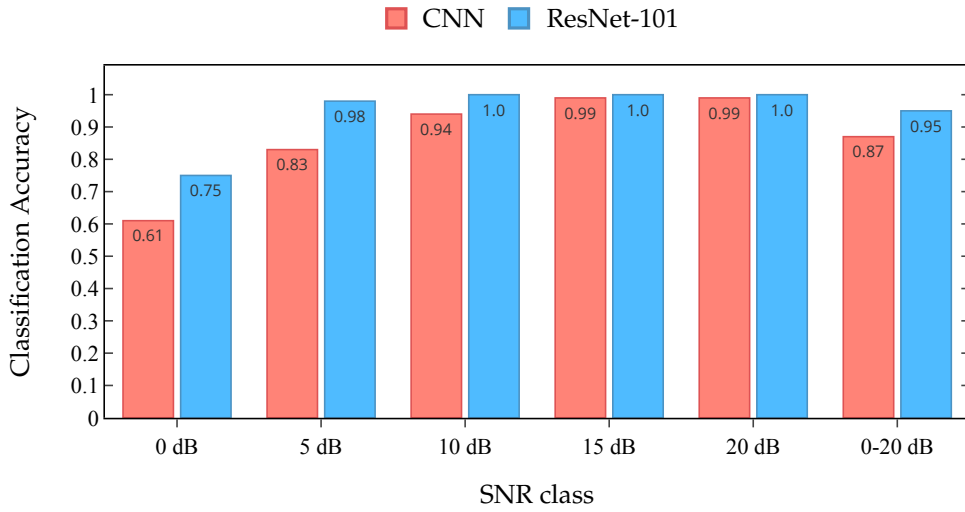


Figure 5.5: Model comparison for different SNR classes, when trained on a dataset with all SNRs.

Generalization Accuracy. Usually, a lot of data is needed for training deep neural networks. In case of insufficient data, we may need to deploy models that are trained on similar data. If these models can perform well, we would save a lot of training time and computation resources. Thus, in this section, we analyse how a model generalizes for varying SNR levels if it is trained only with a particular level.

Table 5.3 shows the accuracy obtained when the CNN model is trained with only 10 dB SNR signals and the testing is done on a dataset with varying SNR levels. We observe that the model performs well for higher SNRs, reaching an accuracy of $\sim 95\%$. Although, at low SNR values, the performance deteriorates significantly and reaches to a point at 0 dB where the predictions are no better than a random guess.

These results conclude that CNNs can generalize higher SNR signals efficiently. This behaviour arises because signals of higher integrity contain distinguishable features which are suppressed by noise at lower SNRs. Moreover, we can say with some certainty that the complex features derived by CNN are independent of SNR, provided a certain level of signal integrity is maintained.

SNR	0 dB	5 dB	10 dB	15 dB	20 dB	0-20 dB
Accuracy	24.2%	69.9%	91.3%	94.5%	95.2%	75.0%

Table 5.3: Generalization accuracy of CNN on a dataset with varying SNR levels, after training with data at SNR=10 dB

Transfer Learning. We have already seen that models trained for a particular environment do not generalize well for other varying conditions. This limitation of the model can be mitigated by using the transfer learning (TL) approach which allows us to leverage knowledge from pre-trained models and build new ones without the requirement of huge amount of data and time.

Fig. 5.6 shows a heatmap which presents the accuracy of CNN when it is trained individually on datasets of varying SNR levels and the models obtained are fine tuned via the TL approach to evaluate their performance on datasets of varying SNR levels. From the results, we discern that the model trained at 10 dB SNR generalizes well for all other SNR classes producing more than double the accuracy ($\sim 55\%$) at 0 dB SNR when compared with the generalization accuracy obtained without the TL approach (cf. Table 5.3). We also observe that when models trained on datasets with higher SNR are transferred to successive lower SNR datasets, the performance degrades and hence we also notice here that model pre-trained at 20 dB SNR results in lowest accuracies for the 0 and 5 dB SNR case. For SNR exceeding 10 dB, all models barring the one at 0 dB achieve decent accuracies ($> 85\%$).

These results show that the TL approach is successful in producing efficient models that can generalize to different but related environments with the added advantages in terms of resource consumption.

5.2.2 Effect of Hardware Heterogeneity

We analyse the effect of hardware heterogeneity on the AMC performance in this section. Specifically, we use the dataset described in Section 4.6.2 where the samples are generated via both laboratory-grade hardware and SDR and span the complete range of SNR with the parameters set to baseline scenario.

Overall Accuracy. Fig. 5.7 presents a comparison of model accuracy for different transceivers. We find a very slight difference between the results for the two hardware set-ups. As one would surmise, the model performs better if high-grade lab equipment is used in comparison to a SDR. At lower SNRs, we get an improvement of $\sim 4\%$ and as the signal integrity increases, the results for both configurations become similar. Hence, we can conclude that the use of SDR does not hinder the model performance.

Comparative Analysis. Fig. 5.8 presents the comparison of CNN and ResNet accuracy across all SNR levels when heterogeneous transceivers are considered. As per our intuition, ResNet performs better than CNN across all SNR values. For high-grade lab hardware, we see a significant increase in accuracy ($\sim 15\%$) at low SNR values. Similar results are seen when SDR is used, with the exception at 0 dB SNR where we do not notice significant change ($\sim 3\%$). For SNR exceeding 10 dB, both the

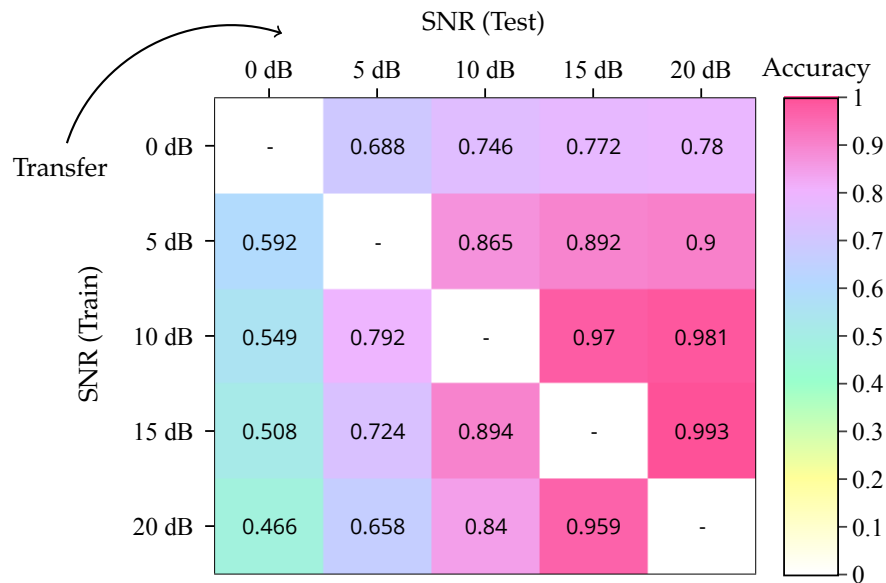


Figure 5.6: Heatmap conveying the accuracy of CNN when transfer learning approach is used across all SNR levels.

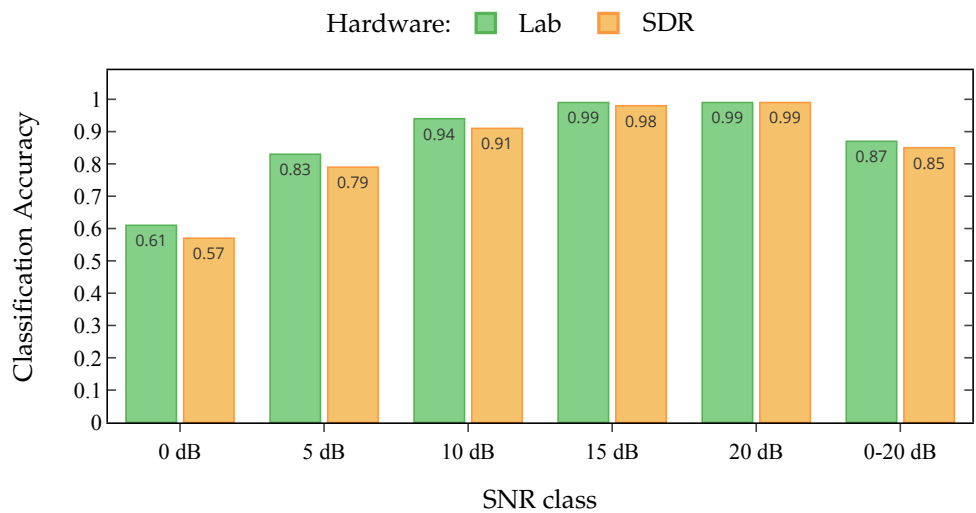


Figure 5.7: Classification accuracy of CNN on the homogeneous hardware datasets across all SNR levels.

models produce good accuracy ($> 90\%$) with ResNet model achieving near perfect results for most cases.

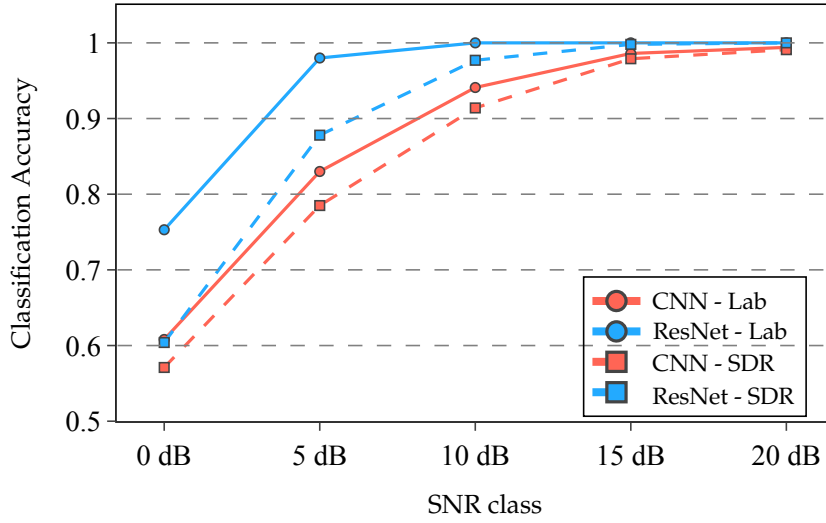


Figure 5.8: Comparison of CNN and ResNet accuracy across all SNR levels when datasets from heterogeneous hardware are considered.

Generalization Accuracy. Fig. 5.9 shows the accuracy of CNN when a mismatch between the training and testing sets is considered owing to the different hardware configurations. Contrary to the results seen in Fig. 5.7, employing heterogeneous hardware datasets for training and testing the model lead to a highly adverse effect on the classification performance. In particular, when the model is trained on the lab-grade hardware dataset and the testing is performed on the SDR dataset, the overall accuracy spanning the complete SNR range does not exceed 70%. For the reverse configuration, the results show further performance degradation with the overall accuracy hovering around 50%. In general, the better performance of the first configuration can be owed to the high quality signals generated by the lab hardware which lets the model extract complex and finer features which are robust to hardware inconsistencies to some extent.

Transfer Learning. Fig. 5.10 presents the comparison of accuracy and training time of CNN for different learning approaches. For traditional learning, the model is trained from scratch on a specific hardware dataset and testing is done on that same dataset. On the other hand, in TL, a model pre-trained on a specific hardware set is fine-tuned for testing on a different hardware set. Fig 5.10a shows the case when training and testing is done on high-grade lab generated signal sets and TL employs the model pre-trained on a SDR dataset. We notice that the TL approach results in slight degradation in the overall accuracy ($\sim 5\%$) but takes only one-tenth of training time in comparison to the traditional approach. Thus, we see a trade-off between the accuracy obtained and time taken for training but we save a lot of precious resources in comparison to little performance loss. Likewise, Fig 5.10b shows the results of training and testing on the SDR dataset where the TL approach utilizes a model pre-trained on the lab

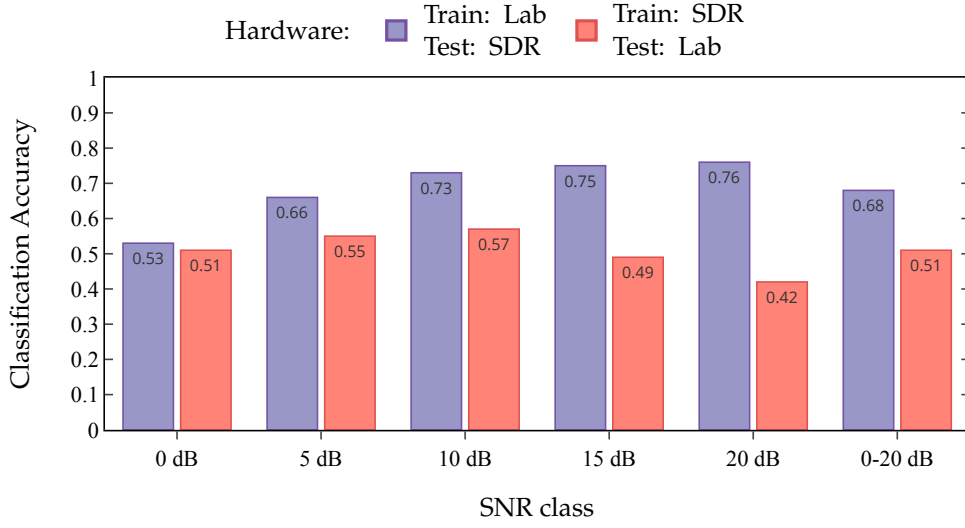


Figure 5.9: Generalization accuracy of CNN across all SNR levels when heterogeneous hardware datasets are considered.

hardware dataset. We see results similar to the alternate case seen earlier and notice a small drop in accuracy using TL while saving significant amount of training time in comparison to the traditional learning.

Furthermore, TL also provides significant improvements over the generalization accuracy, specifically, for testing performed on the lab and SDR datasets, we see an $\sim 30\%$ and $\sim 12\%$ increase in the overall accuracy respectively (*cf.* Fig. 5.9). The only caveat is that we still need, albeit small, a training dataset for the environment where we intend to test.

5.2.3 Effect of Carrier Frequency Offset

The effect of CFO on the AMC performance is analysed in this section. For this purpose, we utilize the dataset described in Section 4.6.2, comprising of signal segments spanning the full SNR range while the CFO of the transmitter is toggled between the interval of ± 1 ppm and ± 5 ppm.

Overall Accuracy. Fig. 5.11 shows the accuracy of CNN when both training and testing are conducted on datasets with similar CFO impairment. We notice that the results for both CFO ranges are almost identical with very slight differences ($\sim 1\%$). Moreover, the performance is nearly akin to the case where CFO has no influence, see CNN accuracy in Fig. 5.5. Thus, we can deduce that frequency mismatches in transceivers have no or very little detrimental effect on the AMC performance within our observed range of upto ± 5 ppm.

Comparative Analysis. Fig. 5.12 shows a comparison of accuracy for the CNN and ResNet models when they are trained and tested on datasets impaired by CFO

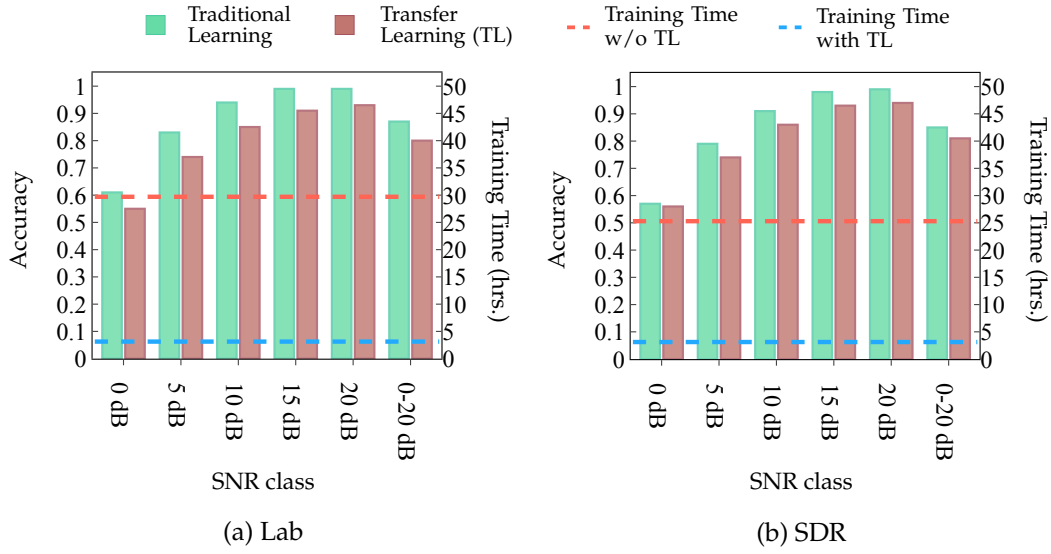


Figure 5.10: Comparison of CNN accuracy and training time for different learning approaches under distinct hardware datasets across all SNR levels.

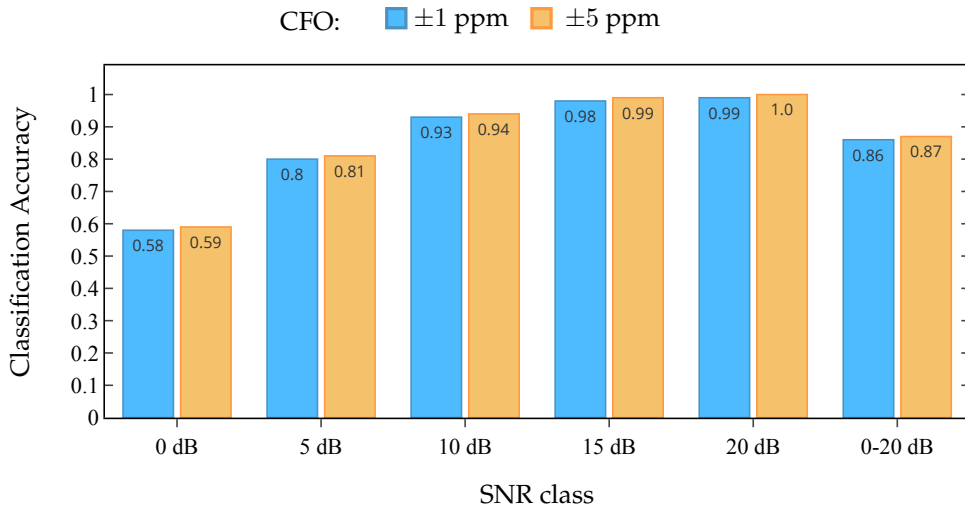


Figure 5.11: Classification accuracy of CNN across all SNR levels on the homogeneous hardware and CFO datasets.

spanning the complete range SNR values. For both the scenarios seen in Fig. 5.12a and 5.12b, we observe that the ResNet model provides a significant boost in the accuracy, specifically at lower SNR where the increase is around 12% - 18%. With increasing SNR, both the models perform well for the observed scenarios achieving near-perfect accuracy above 10 dB SNR. As a consequence of these results, we see an increase in the overall accuracy of around 7% - 9%. Hence, we can deduce that complex

models outperform simpler models for our task but at the cost of increased resource consumption.

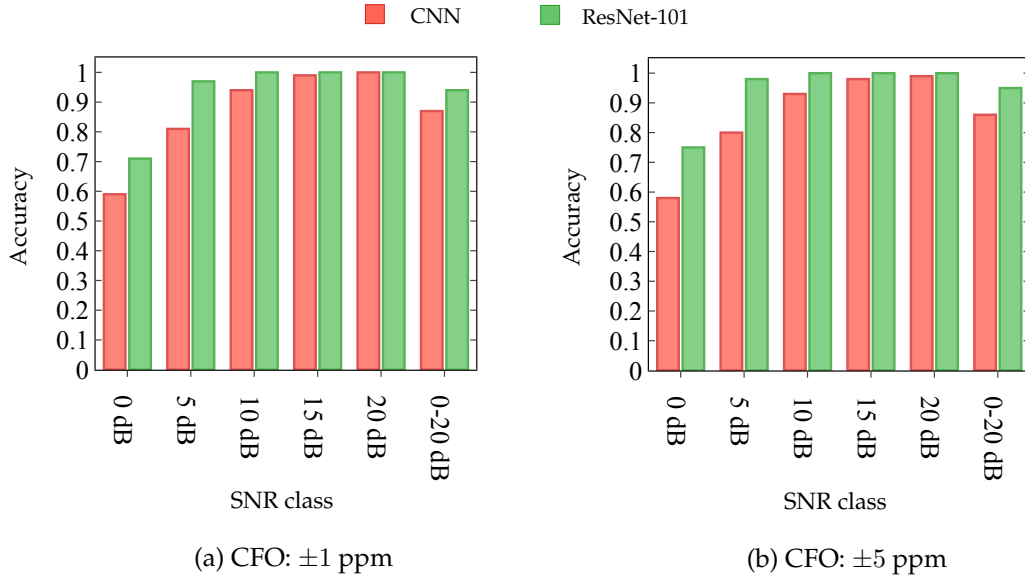


Figure 5.12: Comparison of CNN and ResNet performance on datasets impaired by varying CFO across all SNR levels.

Generalization Accuracy. Fig. 5.13 shows the generalization accuracy of CNN when the train and test sets are taken from heterogeneous datasets of CFO impairment over the complete range of SNR. We notice that there is no degradation in the model performance under all considered scenarios when compared to the case where homogeneous datasets are utilized (*cf.* Fig. 5.11). The worst accuracy ($\sim 60\%$) is recorded at 0 dB SNR and improves continually for higher SNR values recording near perfect classification accuracy at 15-20 dB SNR. These results indicate that DL models are immune to frequency offsets between the transmitter and receiver under our observed set of range.

5.2.4 Effect of Length of Signal Segments

The performance of DL networks is dependent upon the amount and quality of training data one can gather. The estimation of sample size for optimal classification has been vastly researched in the community [97–99]. In this section, we analyse the effect of the variation of segment length per sample on the model performance. For our purpose, we consider the dataset described in Section 4.6.2 across the full range of SNR. Keeping all the other parameters constant, the number of I/Q samples per signal segment are varied.

Fig. 5.14 shows the effect of varying segment lengths on the CNN accuracy at all SNR levels. It is evident from the graph that increasing the number of I/Q samples results in enhanced model performance. We achieve an approximate 5 – 8% improvement every time the segment length is doubled till we reach 10 dB SNR. Above this SNR range, the model accuracy begins to converge and becomes almost

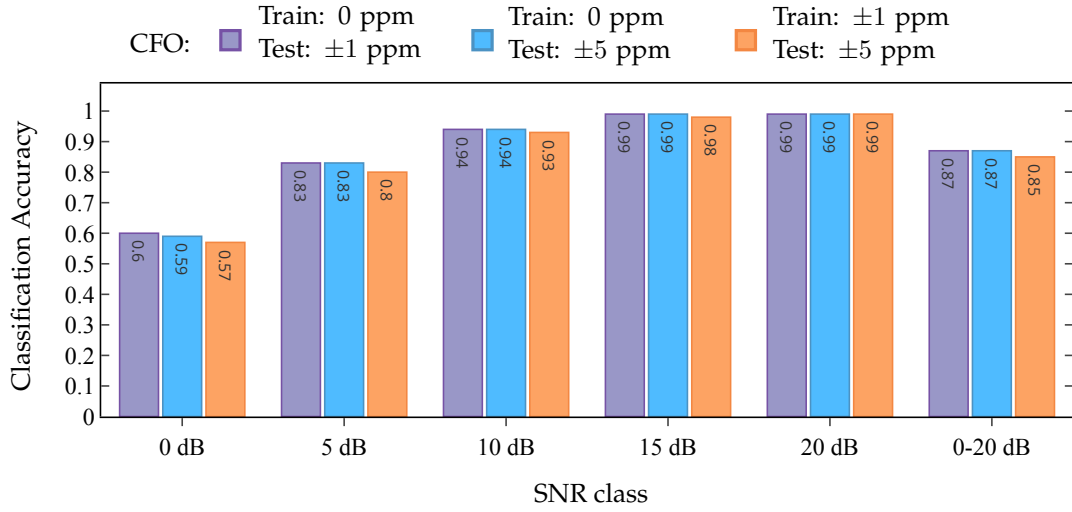


Figure 5.13: Generalization accuracy of CNN across all SNR levels on datasets impaired by CFO.

similar at 20 dB SNR. The only exception to this is the segment length of 256 which can reach only $\sim 92\%$ accuracy at the highest signal integrity. These results conclude that a segment length of more than 512 would be optimal to get maximum efficiency from the model.

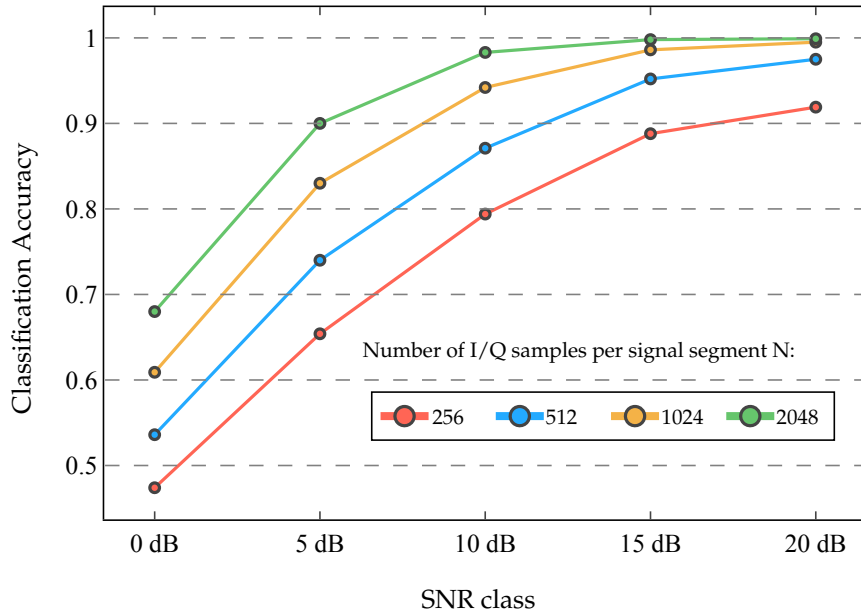


Figure 5.14: Classification accuracy of CNN with the variation of segment length across all SNR levels.

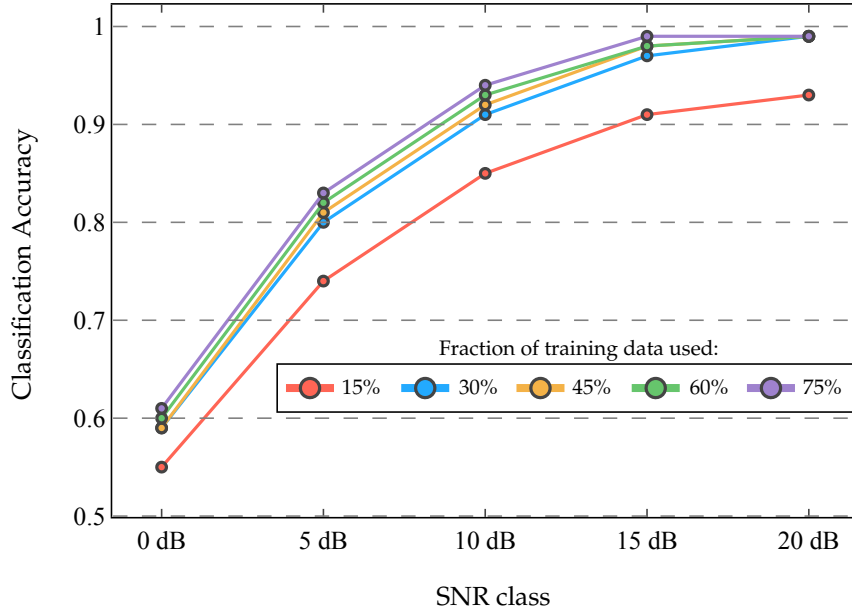


Figure 5.15: Classification accuracy of CNN with the variation of considered training size across all SNR levels.

5.2.5 Effect of Size of Training Data

The general assumption is that deep neural networks require huge amounts of training data. This is particularly true in cases where very deep networks are considered. This section gives an insight about the network performance when smaller training sets are considered. To study the effect of size of training data on the AMC performance, we consider the dataset described in Section 4.6.2 incorporating data from the complete SNR range. For this analysis, all the parameters are set to the baseline scenario and the training sample size is varied.

Fig. 5.15 shows the impact of varying training set sizes on the CNN performance. Interestingly, we only notice a significant drop in accuracy (5 – 9%) across all SNRs when 15% training set is used. For all the other fractions of training data, the difference in CNN accuracy varies between 1 – 4% which is surprisingly minimal. Thus, for this particular case, even a 30% training set can produce optimal results with the benefits of faster network training and lower resource consumption. Still, these results should be taken with a pinch of salt because the considered dataset is recorded via high quality equipment and the real-world signals present a different challenge which may lead to the requirement of significantly larger training data.

5.2.6 Effect of Unknown Signals

The real-world consists of many different modulations but it is not possible to train models for all of them due to limitations of gathering enough training data and predicting the nature of the environment. Thus, our model should be able to differentiate between the trained and unknown modulations. The dataset for this evaluation is

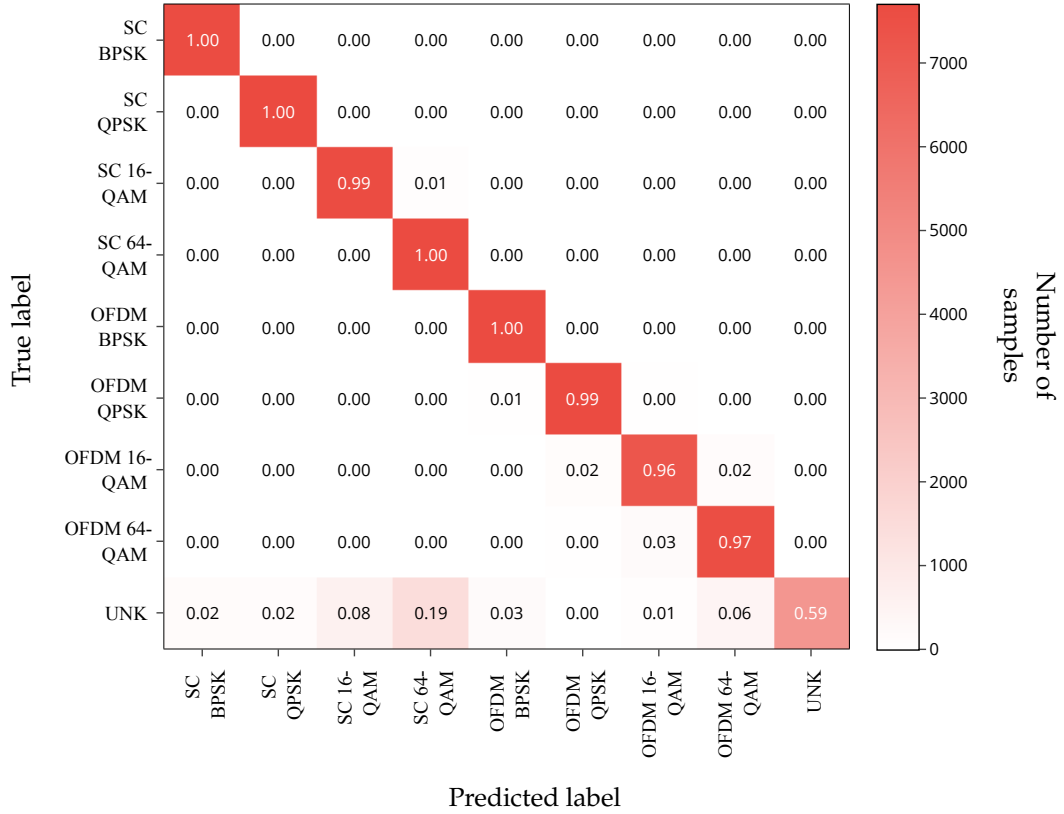


Figure 5.16: Confusion matrix of CNN evaluated on a dataset comprising signals of unknown modulations at SNR in the range of ± 20 dB.

described in Section 4.6.2, where all the parameters are set to the baseline scenario and signal segments of SNR 20 dB are incorporated during the training phase and the testing phase consists of signals of unknown and known modulations having SNR in the range of $[-20, 30]$ dB.

Fig. 5.16 shows the accuracy of CNN for each modulation class when the dataset consists of signals with unknown modulations at SNR in the range of ± 20 dB. Our results show that $\sim 60\%$ of the signals with unknown modulations are predicted correctly and assigned to the *UNK* class while $\sim 20\%$ of them are confused with the 64-QAM modulated signals. This is expected behaviour because the unknown signals consist of varying and higher-order QAM modulated signals which are known to confuse at lower SNR values (*cf.* Fig. 5.4). As for the rest of the modulation classes, the model achieves near-perfect accuracy as also seen in the baseline scenario (*cf.* Fig. 5.2).

5.3 INTERFERENCE SCENARIO

This section details the analysis of AMC performance for the interference scenario as described in Section 5.3. First, we present the overall accuracy of the model on the CCI modulated datasets. Second, we demonstrate the generalization accuracy of the

model when trained on the interference-free data and tested on the interference data. Lastly, we show the results of employing the transfer learning approach to improve the generalization accuracy of the model.

Overall Accuracy. Fig. 5.17 shows the classification accuracy of CNN on datasets impaired by modulated CCI for the complete range of Signal-to-Interference Ratio (SIR) values at 10 dB SNR when both the training and test sets are taken from the same dataset. We observe that all considered modulated interferences cause noticeable degradation ($\sim 12\%$) in the performance of the model (*cf.* Fig. 5.8). Moreover, the model achieves similar overall accuracy under the influence of all interfering signals except 16-QAM modulated interference at lower SIR values where it produces slightly better results. Interestingly, the model is unable to predict signals which are identical to the interfering signals. This can be verified from the confusion matrix for the SC BPSK modulated interference case shown in Fig. 5.18. We can deduce that the SC BPSK signals are the main source of errors and confuse with all the other modulation signals. This is because all signals show characteristics similar to the interfering signals due to their superimposition with them. The other source of confusion are the 16-QAM and 64-QAM signals which is rather expected under imperfect channel conditions (*cf.* Fig. 5.4).

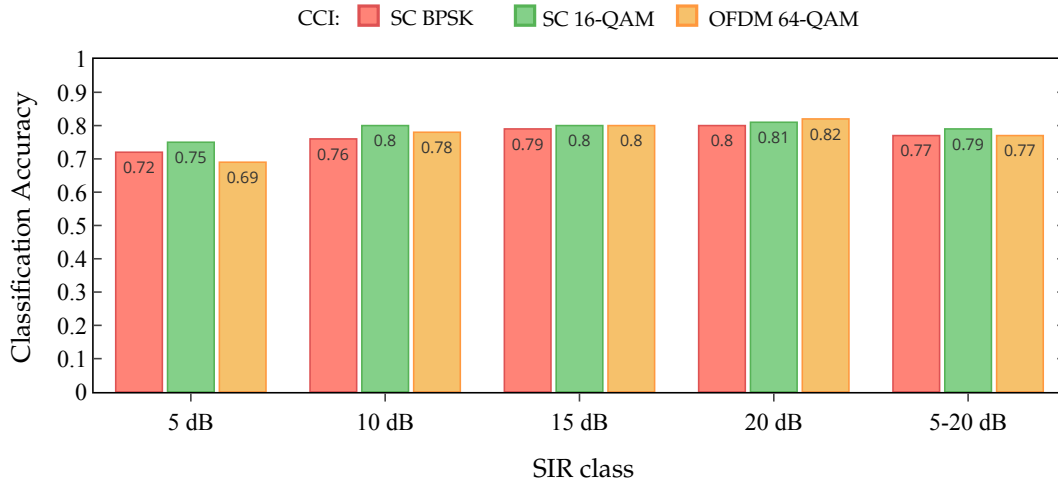


Figure 5.17: Overall accuracy of CNN on the homogeneous modulated CCI datasets across all SIR levels at SNR=10 dB.

Comparative Analysis. Since a simple CNN shows a significant reduction in the model accuracy in the interference scenario, we analyse a more complex ResNet model to see if it can produce better results.

Fig. 5.19 shows the comparison between the accuracy obtained with CNN and ResNet at all SIR levels at 10 dB SNR, when training and testing is done on the same modulated CCI dataset. We notice that ResNet gives a notable increase in the accuracy ($\sim 6 - 10\%$) for all the interference cases, where the maximum improvement is seen for the SC BPSK modulated CCI and the least for SC 16-QAM modulated CCI.

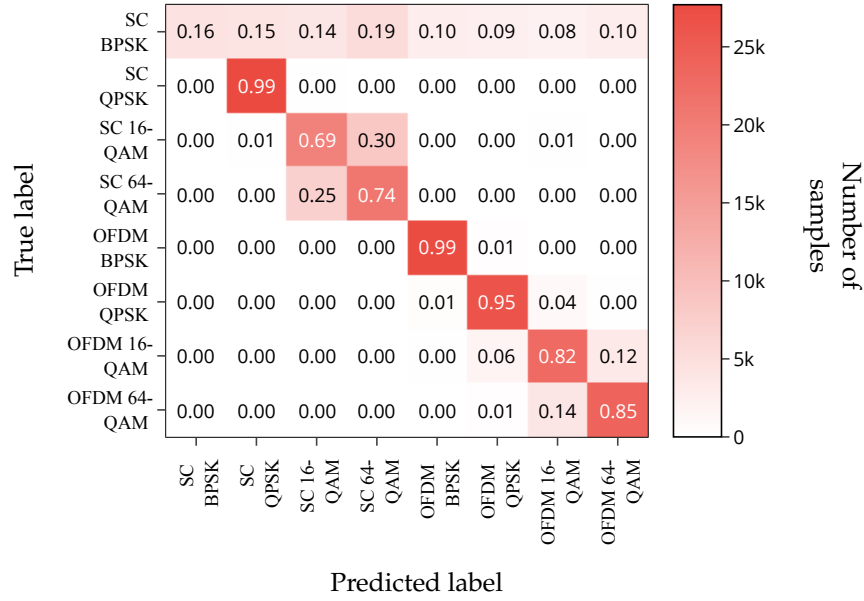


Figure 5.18: Confusion matrix of CNN evaluated across all SIR levels at SNR=10 dB, when trained and tested on the SC BPSK modulated interference dataset.

Though there is a significant performance boost, the ResNet model also fails to predict modulations that are identical to the interfering modulation. Instead, it helps mitigate the confusion caused amongst the 16-QAM and 64-QAM signals.

Generalization Accuracy. Fig. 5.20 presents the accuracy of CNN when it is trained on the interference-free dataset and tested on the dataset impaired by modulated CCI. The results show a drastic decrease in the model performance across all SIR levels. At 5 dB SIR, the accuracy drops by nearly 50% (*cf.* Fig. 5.17) when compared to the

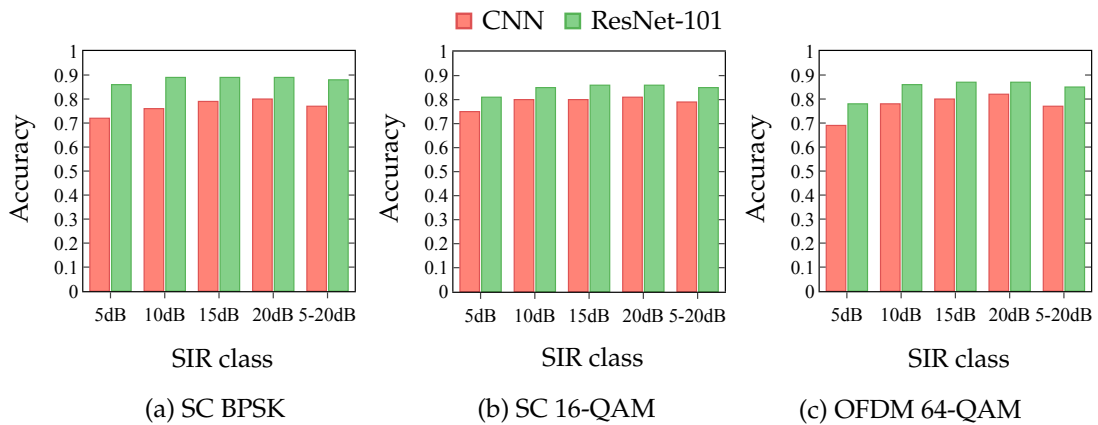


Figure 5.19: Comparison of CNN and ResNet performance for different modulated CCI across all SIR levels at SNR=10 dB.

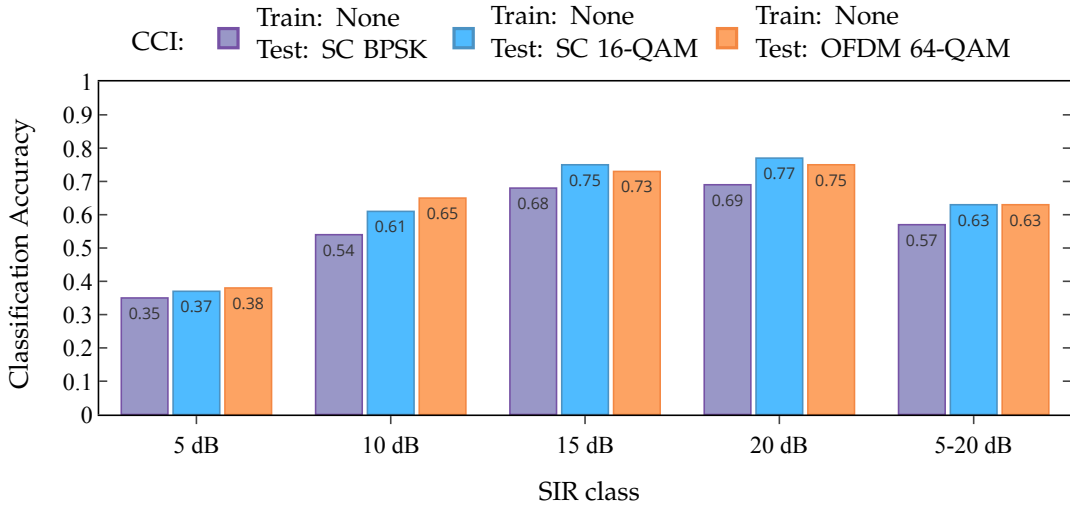


Figure 5.20: Generalization accuracy of CNN across all SIR levels at SNR=10 dB when heterogeneous train and test datasets are considered.

results achieved when homogeneous modulated CCI datasets are used. For higher SIR levels, we notice moderately better performance with the accuracy reaching 77% and 75% at 20 dB SIR for the case of SC 16-QAM and OFDM 64-QAM interferences respectively. On the whole, the worst performance is seen under the influence of interfering SC BPSK signals where the overall accuracy does not exceed 60%. These results emphasize that models trained on high integrity and interference-free lab generated signals cannot be deployed in the real-world scenarios due to the inherent presence of interference there. However, we can overcome this limitation by using the transfer learning approach, the results for which are discussed in the following paragraphs.

Transfer Learning. Fig. 5.21 shows the comparison of CNN accuracy and training time for the two learning approaches, one where the model is built from scratch on the interfered dataset (*traditional learning*) and the other one where the model is built upon an existing model pre-trained on the interference-free dataset (*transfer learning*). The results are evaluated across all SIR levels at 10 dB SNR and the test sets are taken from the interfered datasets. We observe that the TL approach can produce accuracies that are similar to the traditional approach across all modulated interferences while reducing the model training time. For SC BPSK interference, the performance of TL approach lags behind the traditional approach at 5 dB SIR by $\sim 10\%$ while producing near identical results at higher SIR values and reducing the training time by nearly four times. For SC 16-QAM modulated interference, we see similar or even slightly better performance across all SIR classes resulting in the same overall accuracy for both the approaches. It is also interesting to note that we do not get much reduction in the training time with the TL approach in this case. The case of SC OFDM 64-QAM is alike the SC BPSK case and a huge improvement is seen in terms of training time while achieving at par accuracy when compared to the traditional method.

Furthermore, utilization of TL leads to significant improvements in the overall accuracy ($\sim 15 - 20\%$) over the generalization accuracy (*cf.* Fig. 5.20), though some

amount of data from the testing environment is still required for this. It is evident from these results that TL should be employed when models need to be generalized across varying environments.

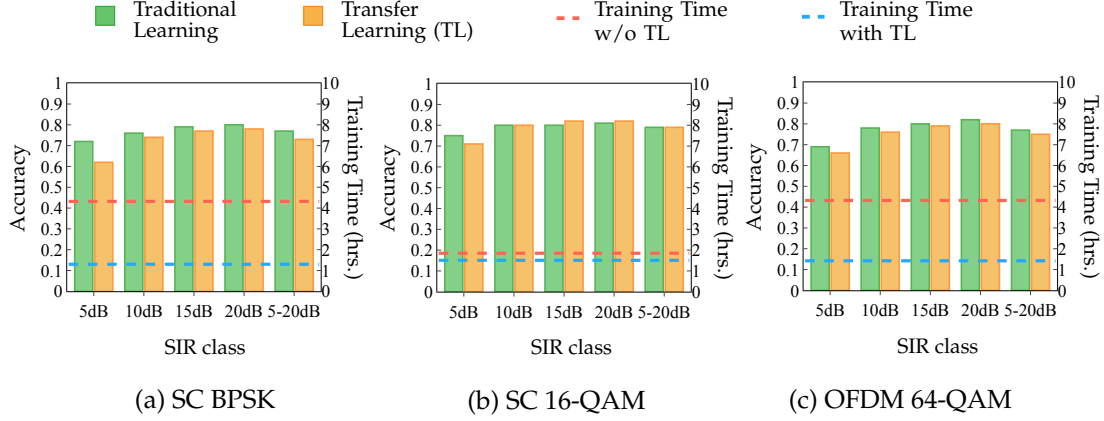


Figure 5.21: Comparison of CNN accuracy and training time for different learning approaches under distinct modulated CCI across all SIR levels at SNR=10 dB.

5.4 REAL-WORLD SCENARIO

To conclude our scenario-based evaluation, this section presents the results of analyzing the real-world scenario as described in Section 4.6.4, where signals subject to varying impairments are considered. This case presents the most challenging test to the AMC task and we use the overall accuracy to describe the performance of our model under these conditions.

Overall Accuracy. Table 5.4 lists the classification accuracy of CNN for the dataset of signals recorded across all SNR and SIR levels, modulation schemes, and interference modulations. As per intuition, an overall accuracy of only $\sim 72\%$ is achieved by the model which is comparatively lower as compared to the accuracy achieved in all the scenarios considered earlier. The performance improves with increasing signal integrity and reaches $\sim 80\%$ and beyond at higher SNRs.

SNR	0 dB	5 dB	10 dB	15 dB	20 dB	0-20 dB
Accuracy	51.7%	64.2%	74.2%	80.0%	90.1%	71.8%

Table 5.4: Classification accuracy of CNN in a real-world scenario.

To further study the source of confusion, Fig. 5.22 shows a confusion matrix of CNN across all SNR levels for the real-world scenario. We notice that from the SC modulations, 16-QAM and 64-QAM modulations confuse amongst each other leading to lower prediction accuracy whereas BPSK and QPSK have a high prediction accuracy reaching $\sim 80\%$. From the OFDM modulations, we observe that BPSK and 64-QAM modulations are classified with stronger confidence and the 16-QAM

modulation is again a source of error and is misclassified as 64-QAM modulation. Apart from this, the OFDM QPSK modulation is also confused with OFDM 16-QAM modulation. These results convey that the higher-order QAM modulations are the main sources of misclassification for CNN in the real-world scenario.

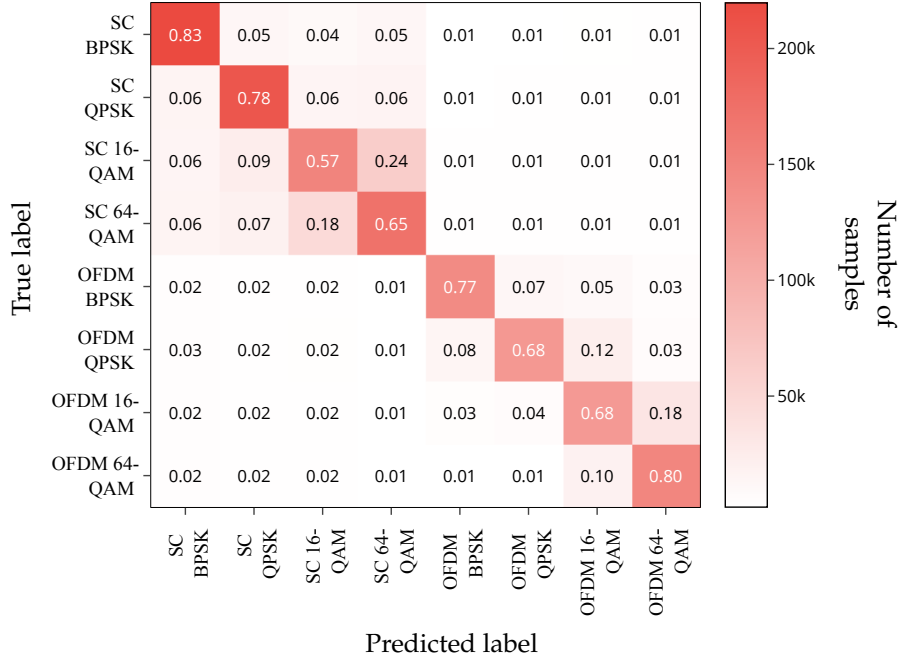


Figure 5.22: Confusion matrix of CNN evaluated across all SNR levels for the real-world scenario.

5.5 SEQUENTIAL TESTING

The signals recorded under harsh interference conditions cannot be decoded properly, but some information may still be extracted from the signal where the damage is minimal. We use sequential testing as a methodology to identify portions of recordings that are most susceptible to this interference. The dataset used for this task is described in Section 5.3, where the signals are impaired by modulated CCI and span across the whole range of SIR values at SNR=10 dB. We use the ResNet-101 network for our evaluation as detailed in Section 4.5.6.

Fig. 5.23 shows the normalized incorrect predictions for our eight modulation schemes, at all portions of the signal recordings, when the dataset interfered by SC BPSK modulation is considered. The highlighted bars convey erroneous predictions for the SC BPSK modulated signals that largely dominate at all the signal segments. From this dominant behaviour, we can conclude that the interfering signals mask similar signals present in the environment by making other signals more like it. This confuses the classifier and leads to random predictions of other modulations. Also, it is worthwhile to note the consistency of incorrect predictions amongst all the segments as it points to the robustness of the classifier.

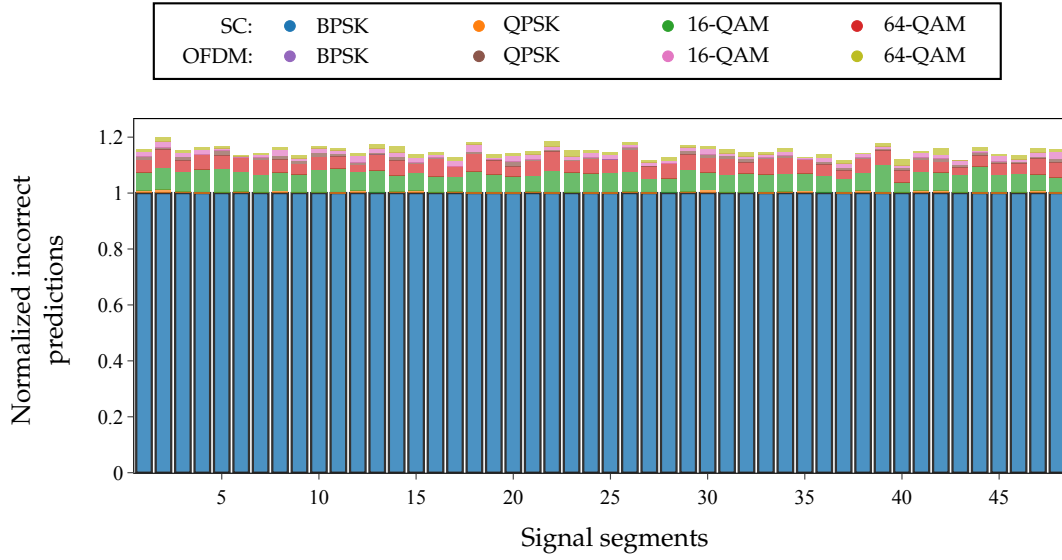


Figure 5.23: Normalized incorrect predictions from the ResNet-101 model at different portions of the signal segments taken from the SC BPSK interfered dataset spanning across all SIR values at 10 dB SNR.

To confirm our results, we also show the confusion matrices in Fig. 5.24 to illustrate the ResNet performance on datasets impaired with CCI modulated by SC 16-QAM and OFDM 64-QAM at SNR level of 10 dB. We observe a similar behaviour,

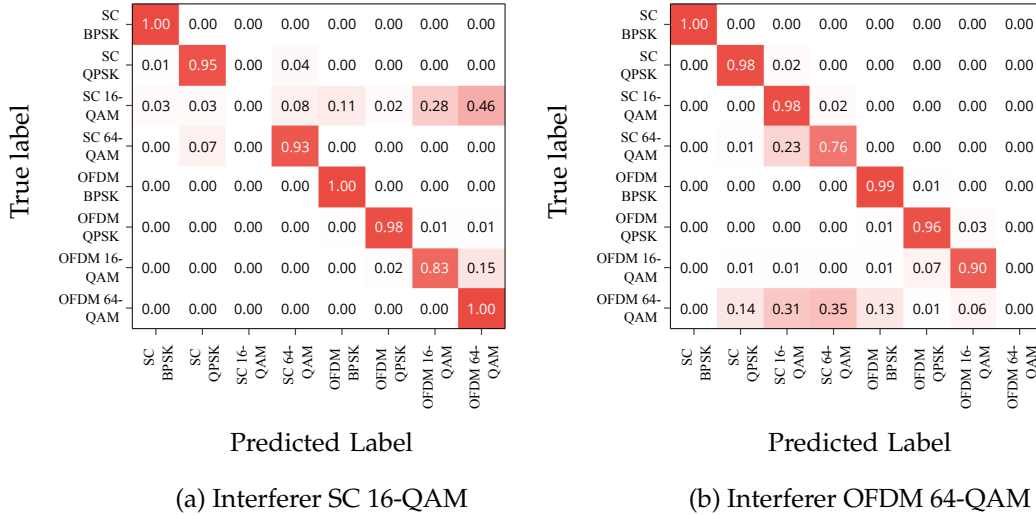


Figure 5.24: Confusion matrix for ResNet-101 performance on the interfered dataset spanning across all SIR levels at 10 dB SNR.

with the interfering signals hampering the predictions of signals having modulation similar to them. Under the influence of SC 16-QAM interferer, the classifier confuses the SC 16-QAM modulated signals with OFDM 16 and 64-QAM modulated signals. Similarly, under the influence of OFDM 64-QAM interferer, none of the OFDM 64-QAM signals are correctly identified and are confused with the SC 16 and 64-QAM modulated signals. Apart from this, all the other modulations are predicted with high accuracy.

From our sequential analysis, we can deduce that all the signal segments in a recording have an equally probable chance of misclassification by the classifier. Moreover, signals that closely relate to the interfering signals are the most susceptible to erroneous predictions.

5.6 EXPLAINING MODEL PREDICTIONS

The DL networks used for our evaluations up to now are treated as black-box models. This implies that their predictions are hard to explain due to the complexity of interpreting the extracted features. We can, however, explain the predictions of any model using SHAP values (*cf.* Section 2.2.4.1) if we have a rich feature set for the classification task. Consequently, we use the dataset described in Section 4.6.2 and spanning across all SNRs. We featurize this set as detailed in Section 4.2 and use the XGBoost algorithm for interpreting the predictions (*cf.* Section 4.5.7).

In this section, we first present the results of the feature-based evaluation with the XGBoost network using the metric of overall accuracy. We then compare its performance against the DL networks which only utilize the raw data. Further, we present the results of interpreting the network predictions with the help of feature-importance, decision, and dependence plots provided by SHAP.

5.6.1 Feature-Based Classification

Overall Accuracy. Table 5.5 shows the classification accuracy of XGBoost on the featurized set spanning the whole range of possible SNR values. The results show that the classifier achieves a high overall accuracy of $\sim 96\%$. Moreover, it performs well under harsh conditions with 0 dB SNR, resulting in $\sim 90\%$ prediction accuracy. Subsequently, for higher SNRs, the performance is even better and an accuracy of more than $\sim 95\%$ is reached.

To analyze the source of error in the XGBoost predictions, we show its confusion matrix in Fig. 5.25 that is evaluated on the featurized set covering the complete range of SNR values. We notice that only the OFDM BPSK and QPSK modulated signals confuse amongst each other whereas all the other modulations are predicted with near-perfect accuracy.

SNR	0 dB	5 dB	10 dB	15 dB	20 dB	0-20 dB
Accuracy	89.2%	95.6%	96.9%	99.4%	99.6%	96.1%

Table 5.5: Classification accuracy of XGBoost on the featurized set across all SNRs.

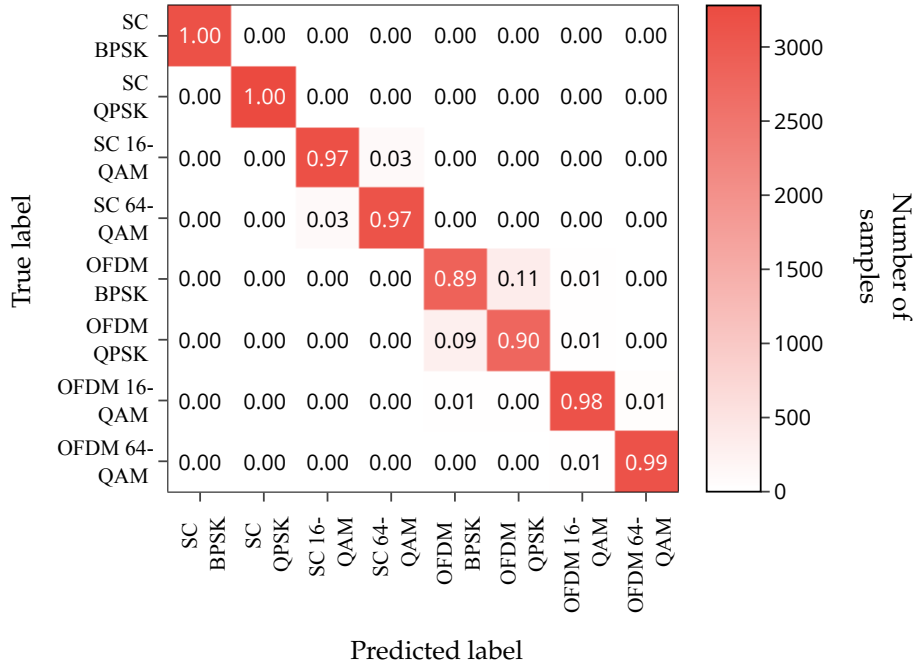


Figure 5.25: Confusion matrix of XGBoost evaluated across all SNR levels for the featurized set.

Comparative Analysis. Fig. 5.26 shows a comparison of the accuracy obtained with XGBoost, CNN and ResNet when trained on signal segments across all the possible SNR values. Amongst all the considered networks, CNN produces the least accuracy at all SNRs. Barring the 0 dB SNR case where the XGBoost produces significant accuracy improvement, the ResNet performs slightly better for all the remaining SNRs. Due to an average high accuracy, the XGBoost results in better overall accuracy across the complete set of SNR values. XGBoost performs well due to the robustness of features extracted using signals comprising 10000 I/Q samples each whereas the CNN and ResNet use raw signal dataset wherein each raw signal consists of only 1024 I/Q samples which are significantly smaller and more suited to real-world applications (*cf.* Section A.2.3 in Appendix).

5.6.2 Model Interpretation

Now that we have our trained classifier, we interpret its outputs and present the results in this section using the plots described in the following paragraphs.

Feature Importance Plot. The feature importance plot gives the importance of a feature for the prediction of a particular class using the mean of the SHAP values for all the observations of that class. Fig. 5.27 shows the feature importance plot for the top 20 features in our set that contribute to the modulation predictions. It is evident from the plot that the fourth and sixth-order cumulants (\tilde{C}_{42} , $|\tilde{C}_{40}|$ and

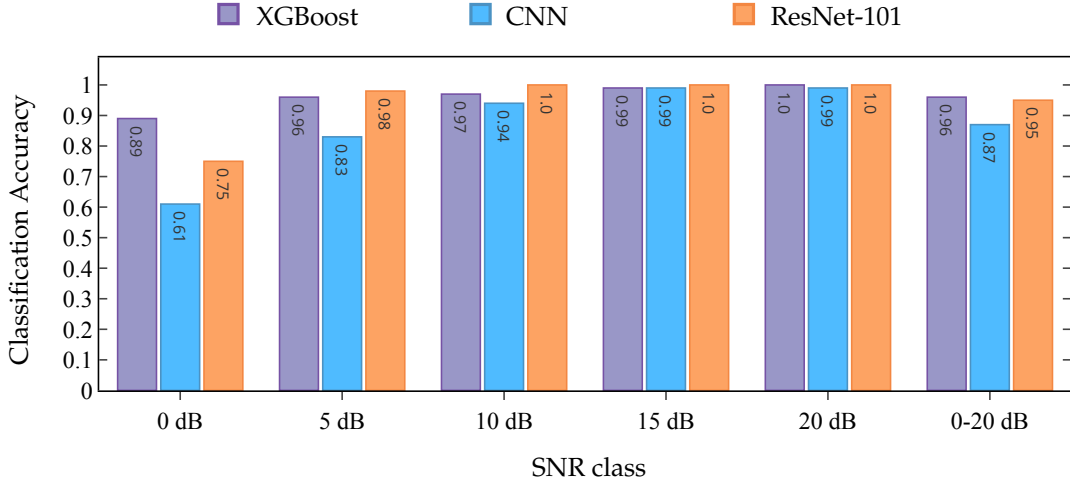


Figure 5.26: Comparison of the classification accuracy of XGBoost on the featurized set against CNN and ResNet-101 evaluated on the raw I/Q dataset across all SNR levels.

\tilde{C}_{63}) play a notable role in the prediction of the SC modulation schemes while the kurtosis of spectrum and centered-normalized instantaneous amplitude (μ_{42}^R and μ_{42}^a) are the most significant features for the prediction of all the OFDM schemes. Other important features that help in the prediction of both the SC and OFDM modulation types are max value of auto-correlation function and the standard deviation of centered-normalized instantaneous amplitude (Ψ_{max} and σ_a). The remaining features are mostly responsible for identifying individual modulation schemes.

Decision Plot. A decision plot is used to explain how a model arrives at a particular prediction. It does so by laying down the contribution of features in terms of their importance for predicting the modulation. Fig. 5.29 shows decision plots illustrating the contribution of the top 10 features for the prediction of three distinct observations in the test set. Fig 5.28a presents the decision plot for a correct prediction of SC QPSK modulation whereby the fourth and sixth-order cumulant features (\tilde{C}_{42} and \tilde{C}_{63}) have a stronger positive impact on the model output. Likewise, we show a decision plot for the OFDM 16-QAM modulation in Fig. 5.28b where the most influential features are the kurtosis and standard deviation of spectrum (μ_{42}^R and σ_R). Lastly, we look into the case of incorrect model output in Fig. 5.28c to understand the features leading to such behaviour. We notice that the amplitude features ($\gamma_{4,max}$, σ_{aa} , and σ_a) lead to the confusion between OFDM BPSK and QPSK modulations.

Dependence Plot. A dependence plot is a scatter plot that is used to analyze the effect of a single feature on the model prediction. As we have already observed in Section 5.6.2, the fourth-order cumulant, \tilde{C}_{42} and the kurtosis of spectrum, μ_{42}^R is the most significant feature for the prediction of SC and OFDM modulations respectively, therefore we analyze their effect on all the considered modulations using dependence plots.

Figs. 5.29a - 5.29d show the influence of \tilde{C}_{42} on the prediction of SC modulation schemes where each dot in the plot represents a sample in our test set and its color the value of the feature that interacts closely with \tilde{C}_{42} . The x and y axis present the feature and its contribution (shapely value) respectively. We observe that distinct values of the cumulant feature lead to distinct modulation outputs, for instance, if the value of \tilde{C}_{42} is greater than 2, the output is BPSK, else if the value is near to 1, a prediction of QPSK is highly probable. Similarly, if the value of \tilde{C}_{42} is around 0, the prediction is likely to be 16-QAM and if the value falls below zero, the model will usually predict a 64-QAM modulation. In most cases, the cumulant feature may be enough to reach an output for the SC modulations, but when it fails to do so, other features come into play. Our plots show the interaction of the \tilde{C}_{42} with one such prominent feature to make the identification possible in case it is not sufficient alone.

Figs. 5.29e - 5.29h show the effect of μ_{42}^R on the prediction of OFDM schemes. We notice that the plots for BPSK and QPSK schemes are almost identical, suggesting that the kurtosis feature alone cannot distinguish between the two modulations. This is also one of the reasons why our classifier confused between these two modulations as seen in Section 5.6.1. Nevertheless, we observe that μ_{42}^R has a strong interaction with $\gamma_{4,max}$ and $|\tilde{C}_{40}|$ which helps it to identify the BPSK and QPSK modulations, respectively. On the other hand, it can be seen from Figs. 5.29g and 5.29h, that the

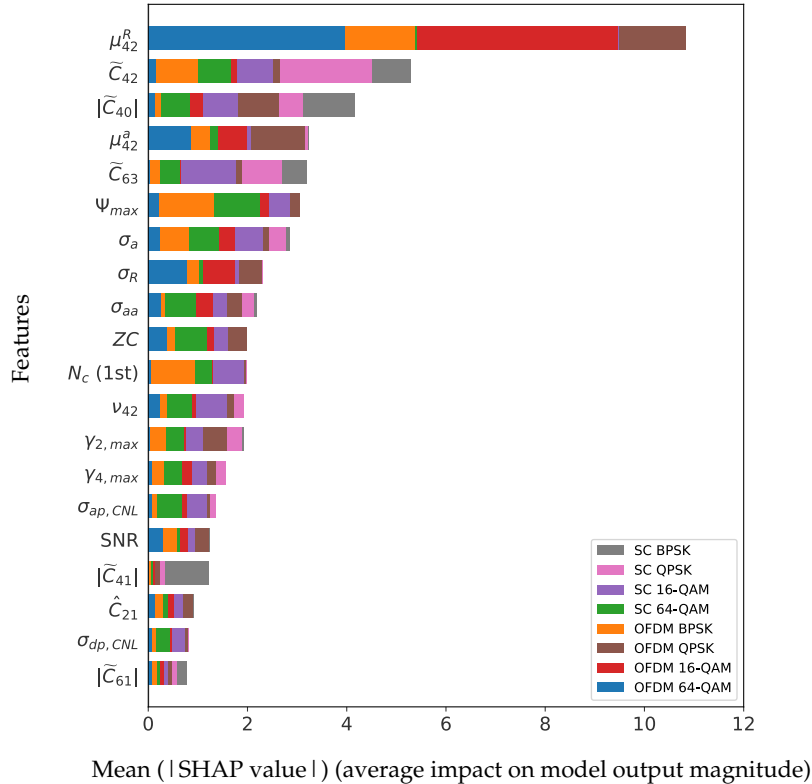
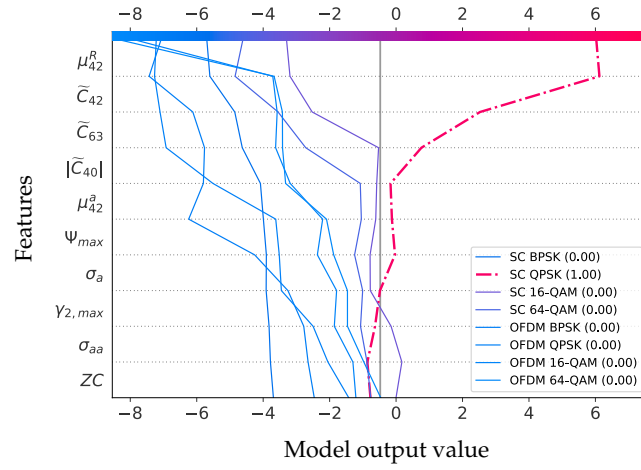
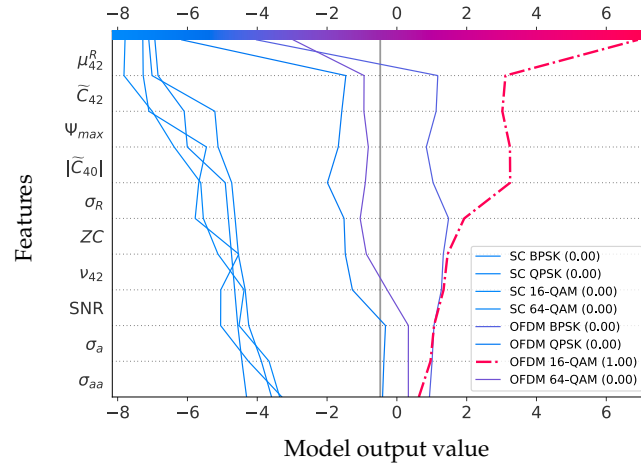


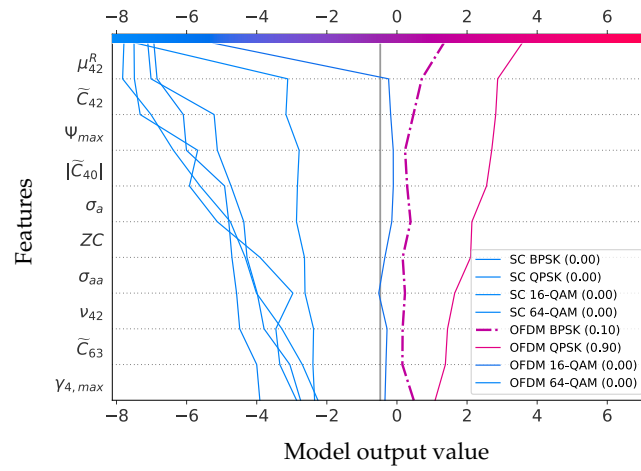
Figure 5.27: Feature importance plot showing the top 20 features and their average impact on the model output magnitude for all the observations in the test set.



(a) Correct SC QPSK prediction



(b) Correct OFDM 16-QAM prediction



(c) Incorrect OFDM QPSK prediction

Figure 5.28: Decision plots illustrating the contribution of the top 10 features for the prediction of different samples in the test set.

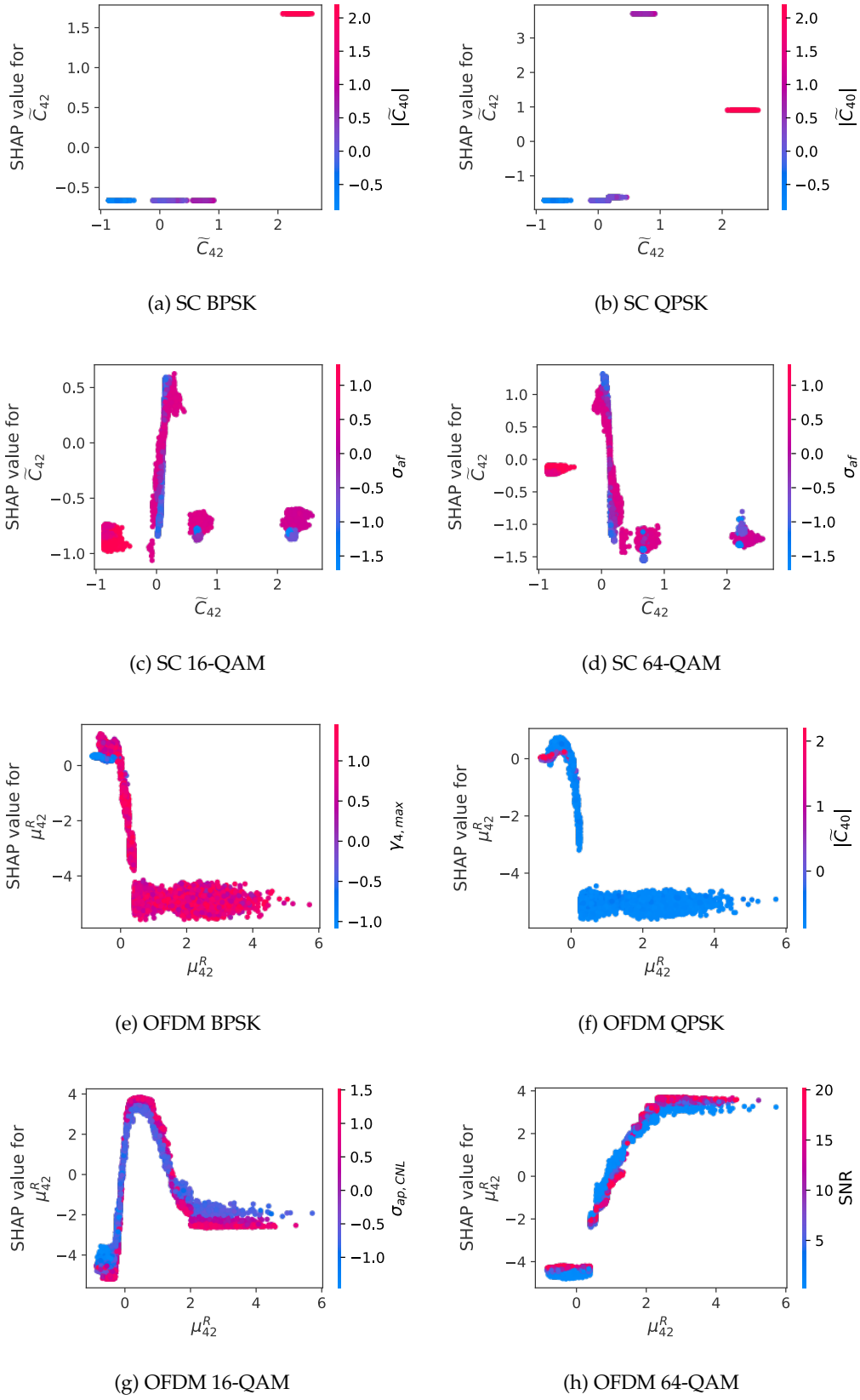


Figure 5.29: Dependence plots illustrating the effect of the single most important feature and its interaction with a related feature for the prediction of different modulation schemes.

kurtosis feature is able to distinguish between the 16-QAM and 64-QAM modulations on its own. Particularly, for a positive value less than 2, the prediction is 16-QAM, else 64-QAM.

As a result of this section, we conclude that feature-based classification works remarkably well when the features are chosen carefully but detailed knowledge of signal processing is required. Although for our case XGBoost performed better than the DL networks, this is just due to our choice of data pre-processing and may not be otherwise true. Nonetheless, good performance helps us in identifying feature importance and interpreting the results of our network.

5.7 EVALUATION OVER MULTIPLE DATASETS

This section presents the results of our evaluation on the deepsig dataset as described in Section 4.5.8. The dataset consists of 24 analog and digital modulations spanning across an SNR range of $[-20, 30]$ dB. We calculate the overall classification accuracy of our classifiers on this dataset and present a comparative analysis amongst them. Further, we also compare the robustness of the classifiers by comparing their performance on our RF dataset against the similar modulations available in the deepsig set. Lastly, we present the results of the transfer learning approach amongst the two datasets to understand the generalization accuracy of the models.

Overall Accuracy. Fig. 5.30 shows the comparison of XGBoost, CNN and ResNet accuracy on the deepsig dataset. We notice that for SNR below 0 dB, none of the classifiers perform well and barely reach an accuracy of $\sim 50\%$ at 0 dB. From 0 dB SNR, the accuracy tends to increase till 10 dB SNR, above which it saturates to a steady value. With the increasing complexity of the classifier, the performance improves, and hence both the DL networks perform better than the decision tree classifier, XGBoost. Further, the ResNet betters CNN accuracy above 5 dB SNR and hence is the best performing classifier on this set.

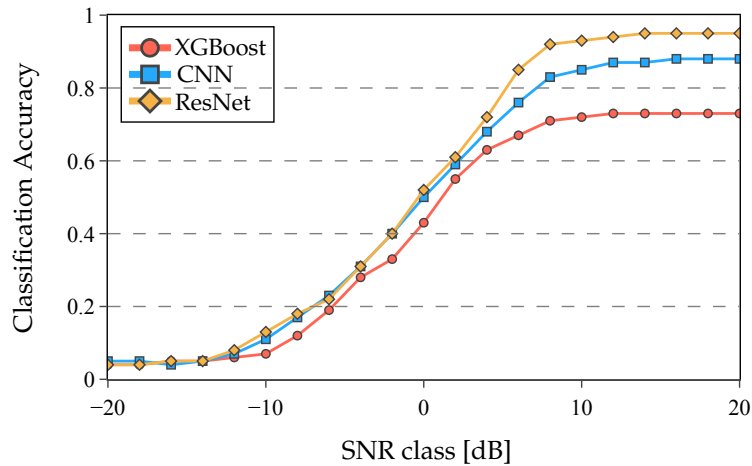


Figure 5.30: Comparison of XGBoost, CNN and ResNet accuracy evaluated on the deepsig dataset consisting of a set of 24 modulations spanning across the SNR in range $[-20, 20]$ dB.

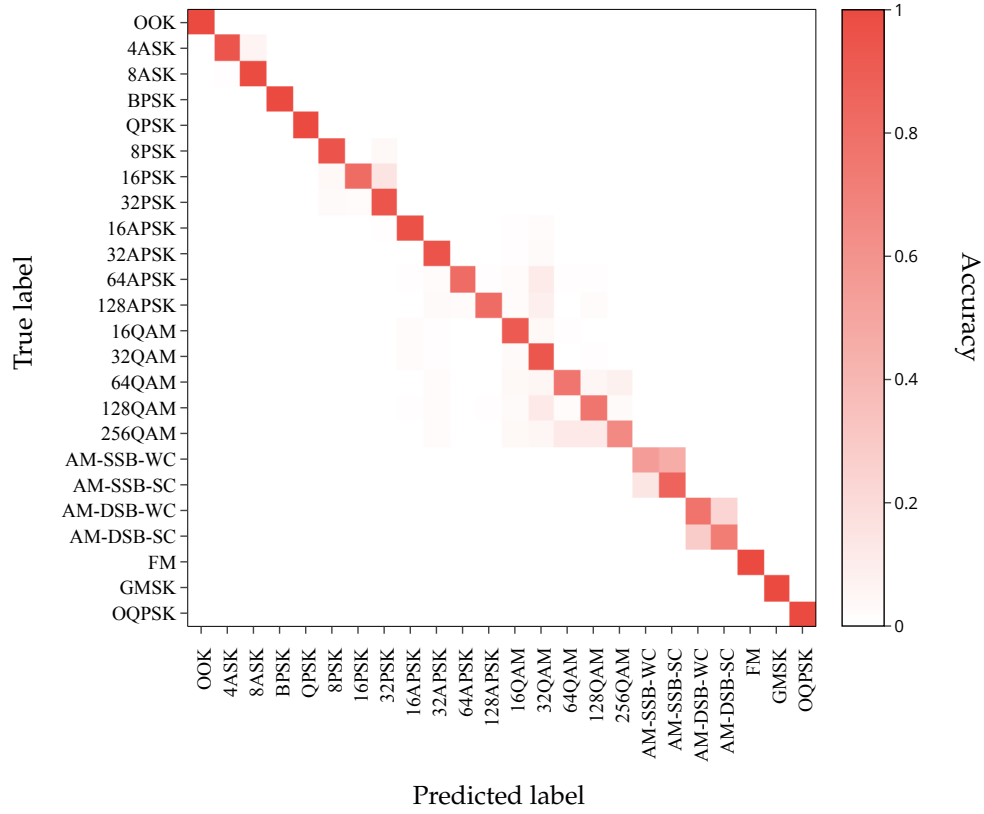
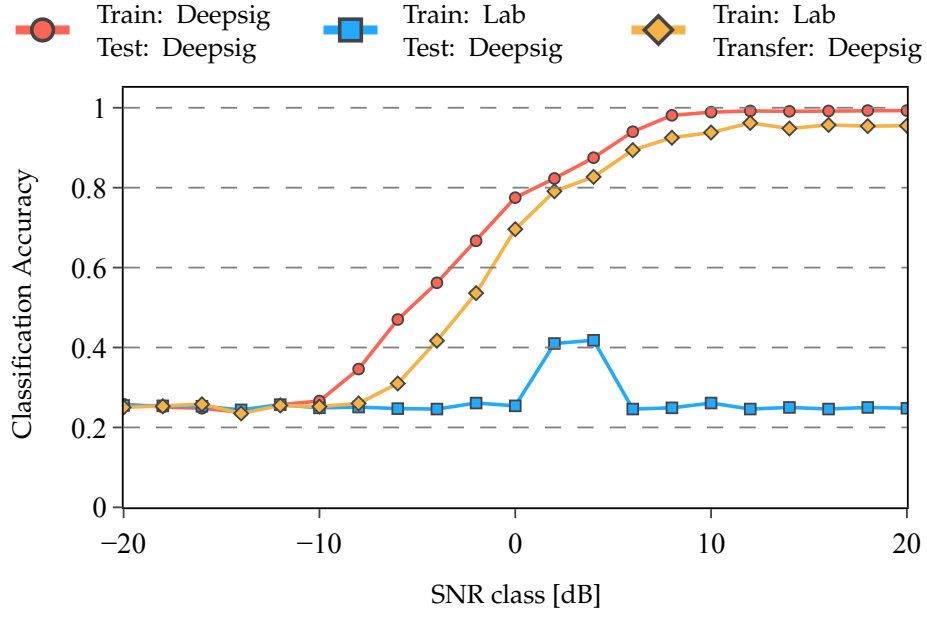


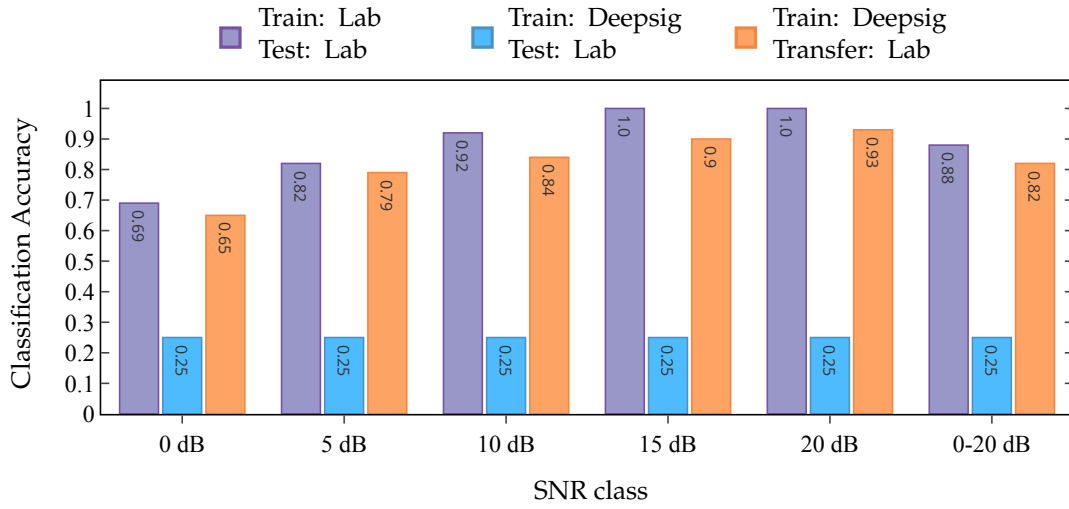
Figure 5.31: Confusion matrix for ResNet-101 evaluated across $\text{SNR} \geq 0$ dB when the train and test sets are taken from the deepsig dataset of 24 modulations.

To understand the source of confusion for the classifiers, we analyze the confusion matrix of our best performing network, ResNet in Fig. 5.31 where the evaluation is performed on the deepsig set encompassing samples with $\text{SNR} \geq 0$ dB. We observe that the classifier confuses between the pair of AM-SSB and AM-DSB modulations. Apart from this, we notice slight confusion between the higher-order QAM and PSK modulations. Most of the other modulations are predicted with high accuracy.

Transfer Learning. Figs. 5.32a and 5.32b show a comparison of CNN accuracy for the transfer learning approach against the traditional learning approaches across all possible SNR values when the train and test sets are taken from our RF set and deepsig set, respectively and vice-versa. We observe a good performance when the aligned train and test sets are used. However, when we use misaligned sets, the model fails to generalize miserably, barely reaching an accuracy of $\sim 30\%$ for both of our testing configurations. The transfer learning approach significantly improves the generalization capability of the model, so much so that it almost performs as well as the case when aligned sets are used. The slight performance drop is made up for in reduction of the training time and the amount of training data required.



(a) Deepsig dataset



(b) Our RF signal dataset

Figure 5.32: Comparison of CNN accuracy for the transfer learning technique against the traditional learning approaches when the train and test sets spanning all SNRs and encompassing four modulations, namely, BPSK, QPSK, 16-QAM and 64-QAM are taken from our RF signal dataset and the deepsig dataset, respectively in (a) and vice-versa in (b).

CONCLUSIONS AND FUTURE WORK

In this thesis, we analyzed the AMC performance under realistic conditions using DL techniques. In particular, we evaluate the performance of CNN and ResNet on datasets comprising raw signal samples recorded using actual radio hardware and impaired with noise, frequency offsets, and interference. We also featurized our datasets to study the AMC performance using a ML classifier, XGBoost. Further, we explained the predictions of the XGBoost classifier and identified the most significant features for each of the considered modulation schemes. Lastly, we analyzed the robustness of our classifiers on other available RF signal datasets. In this chapter, we briefly summarise the main results, key findings, and limitations of our work and discuss the possible future research directions.

Our initial results were based on the baseline scenario consisting of high integrity signals where the CNN achieved near-perfect accuracy, surpassing the performance of traditional ML models. We then performed a sensitivity analysis wherein our classifiers were subject to the test under challenging environments. Our results showed that both CNN and ResNet performed well under the influence of noise and frequency offsets when $\text{SNR} \geq 5$ dB and train and test sets were taken from the same dataset. When this was not the case and misaligned sets were considered, the classifier performance fell significantly showing that the generalization to different environments was not possible. We mitigated this issue with the use of transfer learning techniques in our work, which showed a high improvement in classification accuracy comparable to the use case of aligned sets. In general, ResNet almost always gave a better performance than CNN but more training time and resources were required.

We also analyzed the performance of our classifiers under the presence of interfering signals which resulted in performance degradation as the modulations similar to the interfering modulations were masked. However, we still achieved a good overall accuracy for other modulations. Further, we employed transfer learning to generalize models trained under interference-free conditions to the interference scenario and observed good performance while saving time required in training models from scratch. Finally, we stress-tested our classifier for the real-world scenario consisting of several modulated CCI and noise impairments and the results indicated a further drop in performance with only a few modulations being identified with high probability.

Next, we performed a feature-based classification using the XGBoost algorithm and compared its performance against the DL models. We noticed that it produced at par accuracy when the features were extracted from long signal samples. We further leveraged our feature-based classifier to interpret its predictions and identified the most significant features for the considered modulation schemes. The results showed that the fourth and sixth-order cumulants were responsible for predicting most of the

SC modulation schemes whereas the kurtosis of spectrum differentiated between the OFDM schemes. Moreover, we identified the dependency of these prominent features on other important features such as the standard deviation features of amplitude, frequency, and phase for the AMC task.

To complete our evaluation of the AMC task, we analyzed the robustness of our classifiers on a separately available deepsig dataset consisting of 24 modulation signals recorded using synthetic and real radio hardware. We observed a similar performance from our classifiers above 0 dB SNR, with the ResNet model producing the best results followed by CNN and XGBoost. We furthered our analysis by evaluating the generalization accuracy of CNN using train and test sets from our RF signal dataset and the deepsig dataset, respectively. Our evaluation showed that the model did not generalize well under these circumstances. Hence, we used the transfer learning technique to improve the model generalization and noticed a significant increase in classification accuracy.

Limitations. The primary limitation of our work pertains to the performance of our classifiers in a real-world environment. Although we consider OTA transmissions using real radio hardware, we only consider indoor environments which are not particularly challenging as compared to the outdoor conditions. Other secondary issues pertain to the classifier performance when misaligned sets are used. Although we propose the transfer learning approach to mitigate this issue with success, we cannot use this approach if the training data is very less. Finally, the interpretation of our DL networks is also not possible due to the complexity involved in analyzing the extracted features. However, we do explain, in our work, the predictions of the ML classifier where the input features are already known.

Future Work. To overcome the limitations discussed earlier, future work should investigate the performance of networks on datasets encompassing the conditions of a real wireless communication system. For example, the signals should be recorded using a transmitter and receiver placed sufficiently far to mimic the actual transmissions, and the effects of fading, doppler, noise, and interference should be captured comprehensively. Further, our work only considers the commonly employed DL algorithms for the AMC task and the future work should consider recurrent, generative, and attention models for improving the classification accuracy. Moreover, AMC research should focus on the creation of models that can generalize between environments because of the lack of data availability and computing resources. Another possible research direction is the interpretability of the DL models such that its predictions can be explained and the models can be optimized for achieving the best possible accuracy. Furthermore, since these models have to be deployed in cognitive radios and mobile stations where the resources are highly constricted, future research should focus on optimizing the models for efficiency with less resource consumption. One possible way to achieve this is by model *pruning* which helps in reducing the model size while maintaining almost similar or at par accuracy.

A

APPENDIX

The appendix comprises of the supplementary material to our thesis. We describe our method of storing and accessing the created datasets in Section A.1. Then, in Section A.2, we provide our auxiliary results that further contribute to the AMC task.

A.1 DATASET FILE FORMAT

The RF signal datasets are detailed in Table 4.7 of Chapter 4 and consist of the raw I/Q data samples. Additionally, the deepsig dataset and its featurized version are described in Section 4.5.8. The following sections present the organization of these datasets on the file system and provide the code snippets to access them.

A.1.1 File Structure

Fig. A.1 shows the file structure of our stored datasets. We have two root folders for our RF dataset and the deepsig dataset. The `rf_dataset` comprises three sub-folders based on the recording environments. Further sub-folders are indicative of the configuration parameters in the recording environment and the file names include the modulation type. Additionally, for the interference case, the file names also include the SNR. For the second dataset, `deepsig_dataset`, we have two sub-folders comprising the raw signal and the featurized data.

A.1.2 Accessing the Data

The raw I/Q data is stored in `.h5` and `.hdf5` formats as complex floating point values. Each file consists of three datasets for the raw I/Q samples, class labels and SNR/SIR values where each raw signal is 1024 samples long, class label is one-hot encoded, and the SNR/SIR is an integer. The code to access these datasets is provided in Listings A.1 - A.3.

Further, the featurized deepsig data is stored in `.csv` format where each row represents a signal sample and consists of the 30 extracted features, class label, and SNR. This set can be accessed using the code snippet provided in Listing A.4.

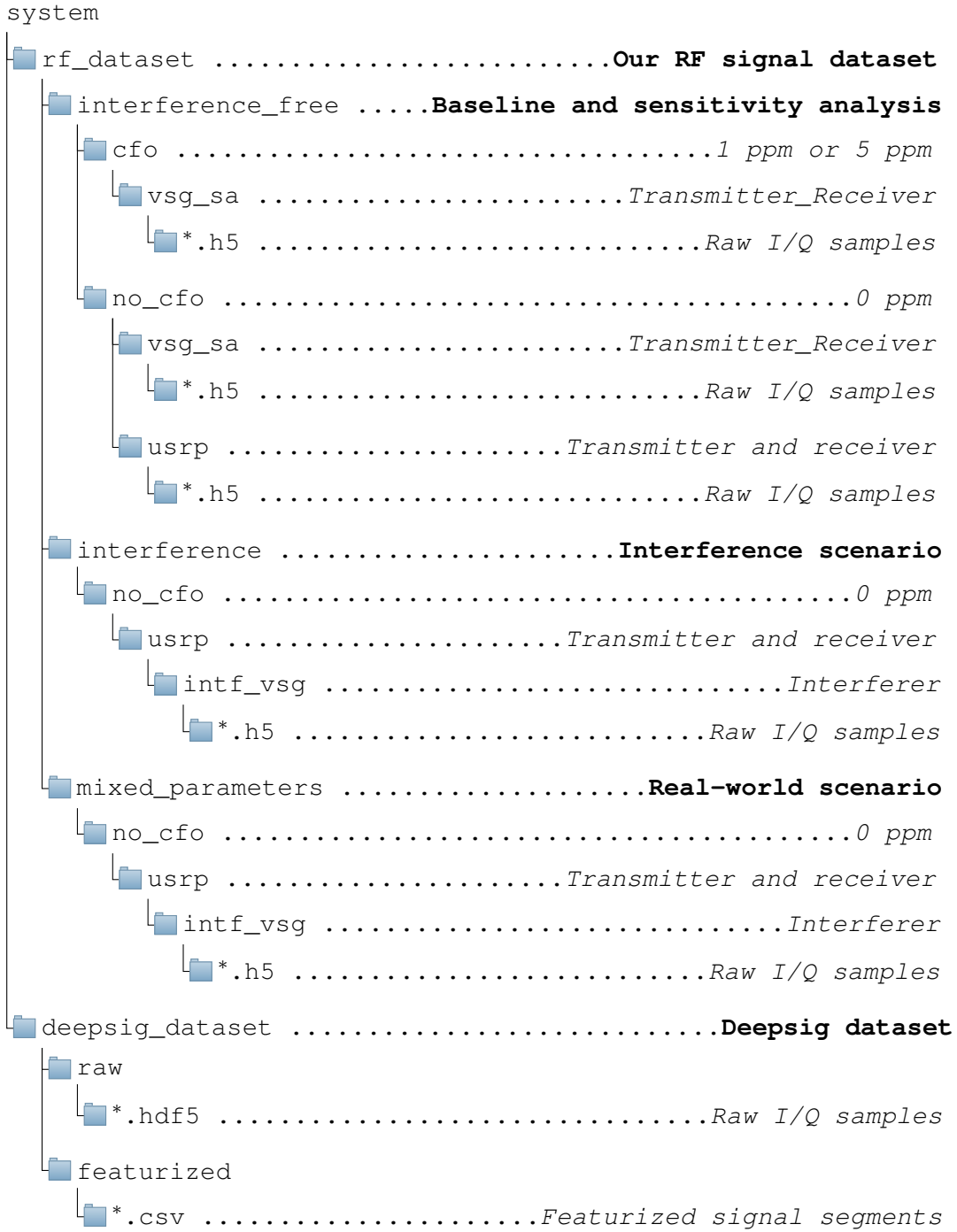


Figure A.1: File structure of the stored RF signal dataset where the sub-folders indicate the configuration parameters and the file names include the modulation type. For the interference case, the file names also include the SNR . Further, the deepsig dataset is stored separately in its raw and featurized form.

```

import h5py as h5

path = "path/to/file"
file = h5.File(path, "r")
raw_iq = file["iq"]
labels = file["labels"]
snrs = file["snrs"]
""" view the number of entries and their shape """
print(raw_iq.shape, labels.shape, snrs.shape)
""" view individual sample """
print("iq: {}".format(raw_iq[0]), "\n",
      "label: {}".format(labels[0]), "\n",
      "snr: {}".format(snrs[0]))

```

Code Listing A.1: Python code snippet to load the signal samples from .h5 files into multidimensional numpy arrays. This code works for all files of the `rf_dataset` except for those in the `interference` folder.

```

import h5py as h5

path = "path/to/file"
file = h5.File(path, "r")
raw_iq = file["iq"]
labels = file["labels"]
sirs = file["sirs"]
""" view the number of entries and their shape """
print(raw_iq.shape, labels.shape, sirs.shape)
""" view individual sample """
print("iq: {}".format(raw_iq[0]), "\n",
      "label: {}".format(labels[0]), "\n",
      "sir: {}".format(sirs[0]))

```

Code Listing A.2: Python code snippet to load the .h5 files in the `interference` folder into multidimensional numpy arrays.

```

import h5py as h5

path = "path/to/file"
file = h5.File(path, "r")
raw_iq = file["X"]
labels = file["Y"]
snrs = file["Z"]
""" view the number of entries and their shape """
print(raw_iq.shape, labels.shape, snrs.shape)
""" view individual sample """
print("iq: {}".format(raw_iq[0]), "\n",
      "label: {}".format(labels[0]), "\n",
      "snr: {}".format(snrs[0]))

```

Code Listing A.3: Python code snippet to load the raw version of the `deepsig` dataset from a .hdf5 file into multidimensional numpy arrays.

```

import csv
import numpy as np
from sklearn import preprocessing

path = "path/to/file"
df = pd.read_csv(path)
df_X = df.drop(['label'], axis=1)
df_y = df[['SNR', 'label']]

features = list(df_X.columns)
""" view the features """
print(features)
scale_features = list(set(features) - set(['SNR']))
""" normalize features """
df_X.loc[:, scale_features] = preprocessing.scale(df_X.loc[:, scale_features
])
""" shuffle the entries """
np.random.seed(4)
idx = np.random.permutation(df_X.index)
""" load into feature and label dataframes """
df_X = df_X.reindex(idx)
df_y = df_y.reindex(idx)
""" view dataframe entries """
print(df_X.head())
print(df_y.head())

```

Code Listing A.4: Python code snippet to load the featurized version of the deepsig dataset from a .csv file into pandas dataframes.

A.2 AUXILIARY RESULTS

This section presents the supplementary results for the AMC task. In Section A.2.1, we present our evaluation using dense neural networks (DNNs). Then, in Section A.2.2, we evaluate the performance of our classifiers for the four class modulation problem consisting the SC modulation schemes only.

A.2.1 Evaluation using DNN

Fig. A.2 shows the confusion matrix of an 8-layer DNN for the baseline scenario. We notice that only the BPSK modulation is predicted correctly from all the SC modulations and the QPSK and QAM modulations confuse amongst themselves. In contrast, from the OFDM schemes, the QAM modulations are predicted with a high accuracy whereas the BPSK and QPSK modulations confuse with each other. Due to all these errors, the overall accuracy of the DNN is $\sim 61\%$ only. Thus, we conclude that the DNN cannot be used for the AMC task as it does not perform well even under the best test conditions.

A.2.2 Evaluation for Four Class Modulation

Fig. A.3 shows the comparison of CNN and ResNet accuracy for the dataset of SC modulations covering a wide range of SNR values. We observe that the ResNet per-

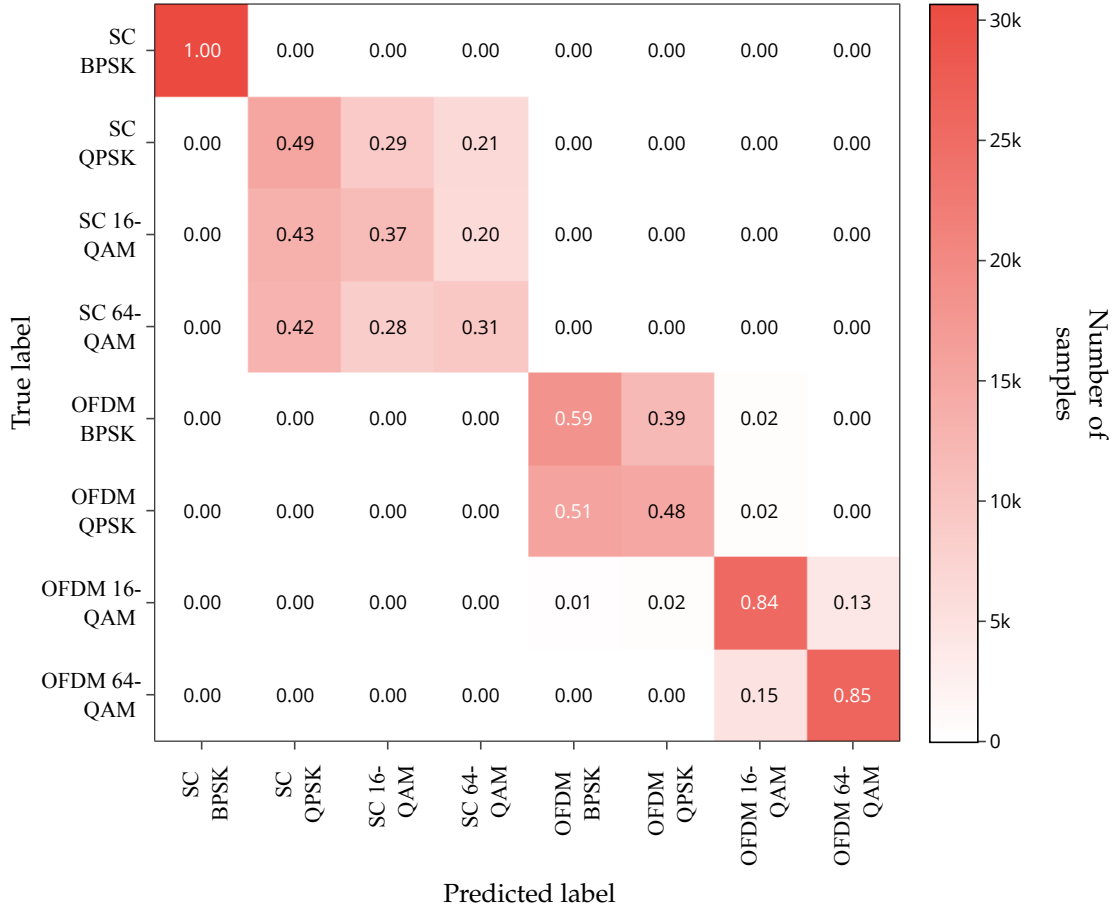


Figure A.2: Confusion matrix of DNN for the baseline scenario.

forms slightly better than CNN at all SNR levels and both the classifiers achieve high accuracies for SNRs above and equal to 5 dB. Only at 0 dB SNR, we see a drop in performance and the accuracy is $\sim 70\%$.

To understand the source of confusion, Figs. A.4 and A.5 show the confusion matrices of CNN and ResNet on the set spanning all SNRs. We observe similar confusions in both cases where the BPSK and QPSK modulations are predicted almost perfectly whereas the two QAM modulations confuse amongst each other leading to a decline in the classifier performance.

A.2.3 XGBoost Performance using Lesser Samples for Featurization

In Section 5.6.1, we had shown that XGBoost performs well for the AMC task under all SNRs and even produces better accuracy than the ResNets. However, we had also mentioned that this was due to the robustness of features extracted using signals consisting of 10000 I/Q samples. In this section, we use features extracted using lesser signal samples to evaluate the XGBoost performance under realistic conditions and make a fair comparison with the DL classifiers.

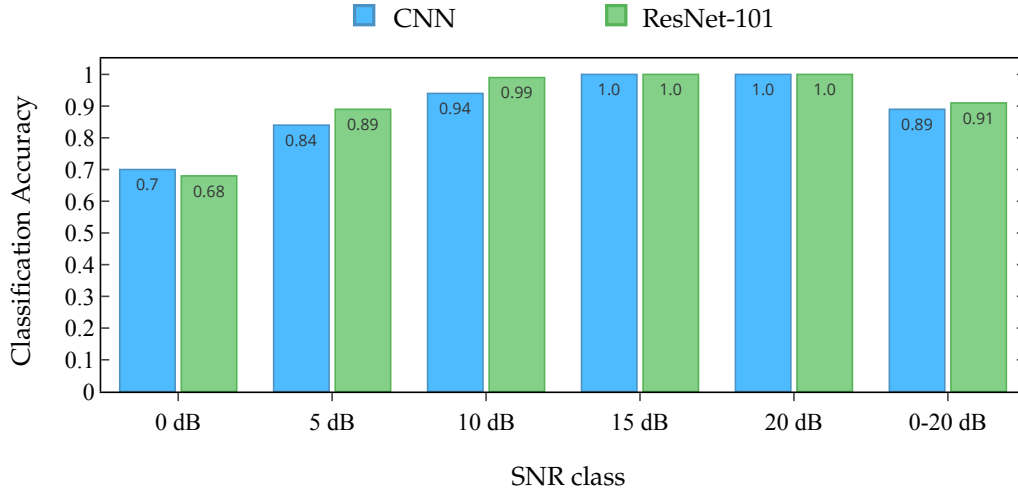


Figure A.3: Comparison of the classification accuracy of CNN and ResNet evaluated on the non-interfered data comprising the SC modulations and spanning across all SNRs.

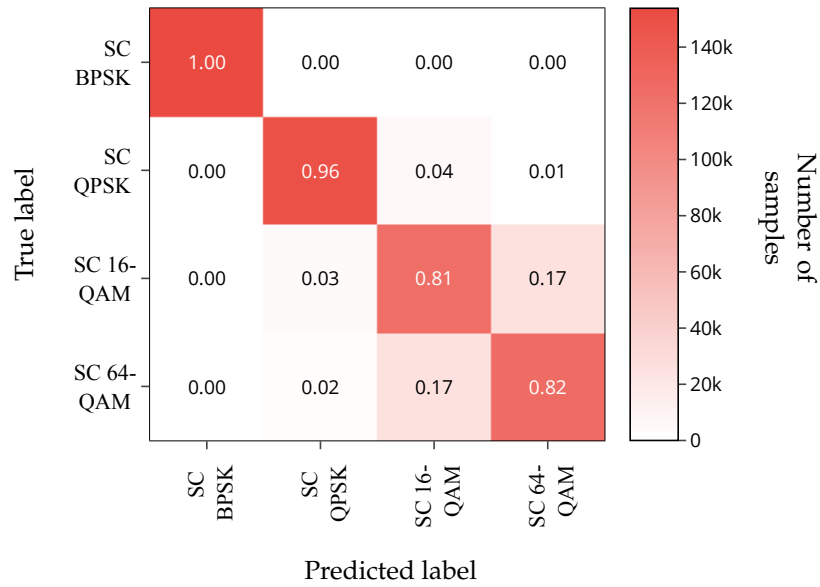


Figure A.4: Confusion matrix of CNN for the evaluation on SC modulation dataset covering the complete range of SNR values.

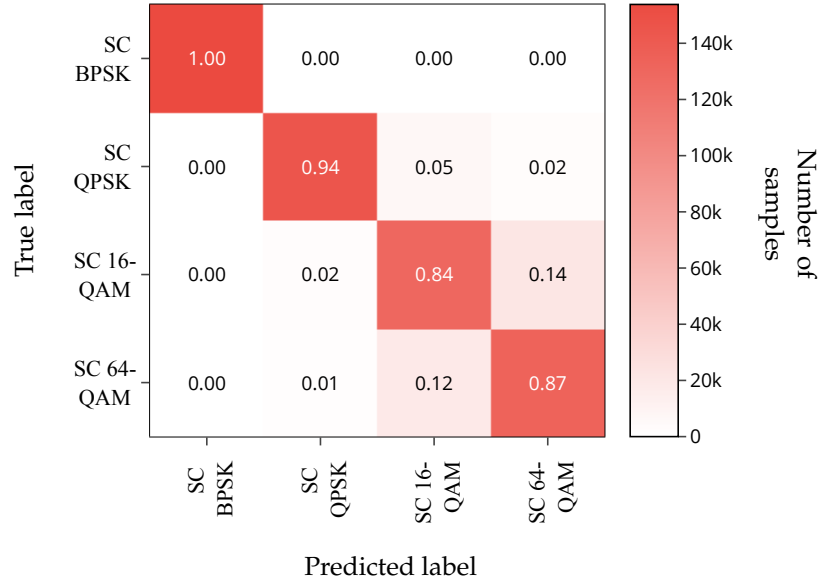


Figure A.5: Confusion matrix of ResNet for the evaluation on SC modulation dataset covering the complete range of SNR values.

Fig. A.6 shows the confusion matrix of XGBoost at SNR=0 dB on the set featurized using the dataset comprising 1024 samples per segment. We notice that XGBoost does not perform well for any modulation class except the SC BPSK modulation. The overall accuracy drops to a mere $\sim 47\%$ from $\sim 89\%$ (*cf.* Table 5.5). This is significantly lesser than both the CNN and ResNet accuracy and confirms that the DL classifiers perform better than XGBoost when similar number of I/Q samples are used.

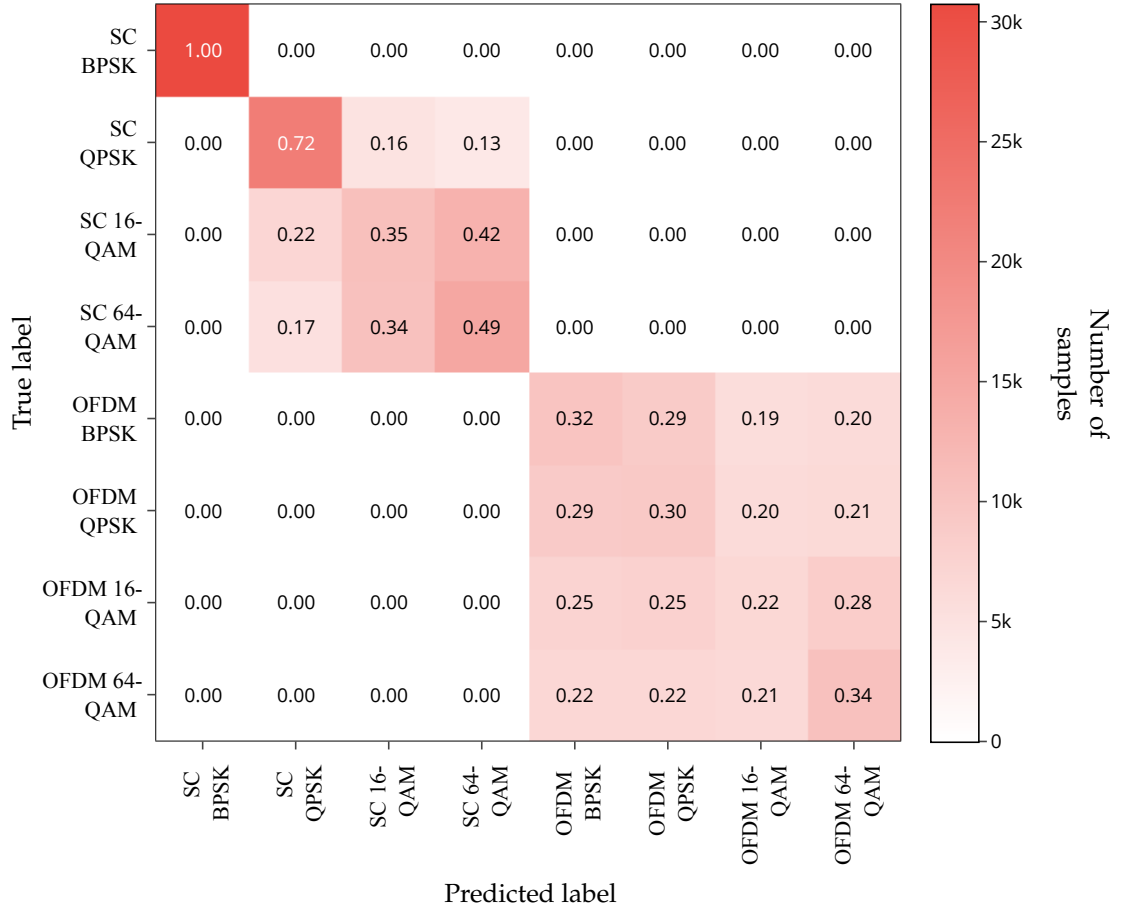


Figure A.6: Confusion matrix of XGBoost at SNR=0 dB on the set featurized using the dataset comprising 1024 samples per segment.

B

ABBREVIATIONS

AMC Automatic Modulation Classification

ALRT Average Likelihood Ratio Test

ANN Artificial Neural Network

BB Baseband

BPSK Binary Phase Shift Keying

BMU Best Matching Unit

CART Classification and Regression Tree

CCI Co-channel Interference

CDF Cumulative Distribution Function

CFO Carrier Frequency Offset

CNN Convolutional Neural Network

DAC Digital-to-Analog Converter

DL Deep Learning

DNN Deep Neural Network

EDA Exploratory Data Analysis

EVT Extreme Value Theory

FB Feature-based

FC Fully Connected

FFT Fast Fourier Transform

GLRT Generalized Likelihood Ratio Test

GRU Gated Recurrent Unit

HLRT Hybrid Likelihood Ratio Test

I In-phase

ICC Inter-carrier Interference
ICI Inter-channel Interference
IoT Internet of Things
IDFT Inverse Discrete Fourier Transform
ISI Intersymbol Interference
LB Likelihood-based
LNA Low Noise Amplifier
LO Local Oscillator
LRT Likelihood Ratio Test
LSTM Long Short Term Memory
M2M Machine-to-Machine
MAV Mean Allocation Vector
MIMO Multiple Input Multiple Output
ML Machine Learning
MLP Multi Layer Perceptron
MR Modulation Recognition
OFDM Orthogonal Frequency Division Multiplexing
OTA Over-the-Air
PB Passband
PSK Phase Shift Keying
Q Quadrature
QAM Quadrature Amplitude Modulation
QPSK Quadrature Phase Shift Keying
RF Radio Frequency
ReLU Rectified Linear Unit
ResNet Residual Network
RNN Recurrent Neural Network
SA Signal Analyzer
SC Single Carrier

SDR Software Defined Radio

SHAP SHapely Additive exPlanation

SIR Signal-to-Interference Ratio

SNR Signal-to-Noise Ratio

SOM Self-Organizing Map

SVM Support Vector Machine

TL Transfer Learning

VSG Vector Signal Generator

LIST OF TABLES

4.1	Measurement configuration parameters saved for each signal segment. . . .	37
4.2	Measurement parameters for the baseline configuration.	38
4.3	Measurement parameters saved for the parameter sweep configuration. . .	38
4.4	Measurement parameters saved for the interference sweep configuration. .	39
4.5	Measurement parameters saved for the mixed parameter configuration. . .	39
4.6	Features extracted from the recorded RF signal segments.	43
4.7	Configuration of the processed datasets where each signal segment consists of 1024 I/Q samples.	50
4.8	Hyperparameters for XGBoost (<i>cf.</i> Section 2.2.1.1).	50
4.9	Hyperparameters for CNN (<i>cf.</i> Section 2.2.1.2).	50
4.10	Hyperparameters for the convolutional layers of ResNet-101 (<i>cf.</i> Section 2.2.1.3).	51
4.11	The set of 24 modulations present in the deepsig dataset.	54
5.1	Accuracy of classifiers across all eight modulation schemes for the baseline scenario.	61
5.2	CNN accuracy on a dataset of signals with varying SNR levels, after training with all SNR levels.	62
5.3	Generalization accuracy of CNN on a dataset with varying SNR levels, after training with data at SNR=10 dB	64
5.4	Classification accuracy of CNN in a real-world scenario.	76
5.5	Classification accuracy of XGBoost on the featurized set across all SNRs. . .	79

LIST OF FIGURES

2.1	Block diagram of SC modulator.	4
2.2	Constellation diagrams of SC modulation schemes of different modulation orders along with their bit mapping sequences where M denotes the number of bits per symbol ¹	5
2.3	Filter responses of a RC filter with different roll-off factors α	6
2.4	Block diagram of OFDM signal generator.	7
2.5	Frequency-Time representation of an OFDM signal.	8
2.6	Typical block diagram of the RF transmit chain.	9
2.7	Typical block diagram of the RF receive chain.	11
2.8	A simplified schematic of the supervised learning technique.	12
2.9	Schematic showing an ensemble of two trees to classify signals based on their modulation scheme.	13
2.10	Architecture of a convolutional neural network, here for a binary classification task of signal modulation recognition. Each block comprises of a set of feature maps that contain units of similar weights.	16
2.11	Illustration of the convolution operation.	17
2.12	Typical activation functions used in a neural network.	19
2.13	Illustration of the application of ReLU activation and max pooling on the input feature map.	19
2.14	Calculation of gradients using the backpropagation algorithm.	21
2.15	Comparison of a vanilla net and residual net building block.	22
2.16	Architecture of a 34 layer residual network for a binary classification task.	23
2.17	A 4x4 SOM that highlights a winning neuron (BMU) for a particular input that is randomly sampled from the training set.	24
2.18	An illustration of the comparison between traditional ML and TL techniques.	25
2.19	TL using off-the-shelf pre-trained models as feature extractors.	27
2.20	Illustration of the effect of a feature <i>dog-allowed</i> on the model output when added to a coalition of <i>forest-nearby</i> , <i>1st floor</i> and <i>area-50</i>	27
2.21	All 8 possible feature coalitions when the contribution of feature <i>dog-allowed</i> is calculated.	28
2.22	SHAP values of features contributing to the model output. The values explain how to get from a base value $E[f(z)]$ where no features are known to the final output $f(x)$	29
4.1	Frequency response for positive frequencies of the low pass filter used for baseband filtering during pre-processing.	49
5.1	Topological SOM based on the features extracted from CNN for the baseline scenario. The colored circles represent the winning nodes per modulation scheme and the surface map depicts their separation distance. Lighter spots indicate large relative distances between the nodes.	59

5.2	Confusion matrix of CNN for the baseline scenario. The numbers represent the accuracy of predicted class and the color intensity depicts the number of samples.	60
5.3	Confusion matrix of SVM classifier for the baseline scenario using feature set X . The numbers represent the accuracy of predicted class and the color intensity depicts the number of samples.	61
5.4	Confusion matrix of CNN evaluated for signals at SNR=0 dB, after training with all SNR levels.	62
5.5	Model comparison for different SNR classes, when trained on a dataset with all SNRs.	63
5.6	Heatmap conveying the accuracy of CNN when transfer learning approach is used across all SNR levels.	65
5.7	Classification accuracy of CNN on the homogeneous hardware datasets across all SNR levels.	65
5.8	Comparison of CNN and ResNet accuracy across all SNR levels when datasets from heterogeneous hardware are considered.	66
5.9	Generalization accuracy of CNN across all SNR levels when heterogeneous hardware datasets are considered.	67
5.10	Comparison of CNN accuracy and training time for different learning approaches under distinct hardware datasets across all SNR levels.	68
5.11	Classification accuracy of CNN across all SNR levels on the homogeneous hardware and CFO datasets.	68
5.12	Comparison of CNN and ResNet performance on datasets impaired by varying CFO across all SNR levels.	69
5.13	Generalization accuracy of CNN across all SNR levels on datasets impaired by CFO.	70
5.14	Classification accuracy of CNN with the variation of segment length across all SNR levels.	70
5.15	Classification accuracy of CNN with the variation of considered training size across all SNR levels.	71
5.16	Confusion matrix of CNN evaluated on a dataset comprising signals of unknown modulations at SNR in the range of ± 20 dB.	72
5.17	Overall accuracy of CNN on the homogeneous modulated CCI datasets across all SIR levels at SNR=10 dB.	73
5.18	Confusion matrix of CNN evaluated across all SIR levels at SNR=10 dB, when trained and tested on the SC BPSK modulated interference dataset.	74
5.19	Comparison of CNN and ResNet performance for different modulated CCI across all SIR levels at SNR=10 dB.	74
5.20	Generalization accuracy of CNN across all SIR levels at SNR=10 dB when heterogeneous train and test datasets are considered.	75
5.21	Comparison of CNN accuracy and training time for different learning approaches under distinct modulated CCI across all SIR levels at SNR=10 dB.	76
5.22	Confusion matrix of CNN evaluated across all SNR levels for the real-world scenario.	77
5.23	Normalized incorrect predictions from the ResNet-101 model at different portions of the signal segments taken from the SC BPSK interfered dataset spanning across all SIR values at 10 dB SNR.	78

5.24	Confusion matrix for ResNet-101 performance on the interfered dataset spanning across all SIR levels at 10 dB SNR.	78
5.25	Confusion matrix of XGBoost evaluated across all SNR levels for the featurized set.	80
5.26	Comparison of the classification accuracy of XGBoost on the featurized set against CNN and ResNet-101 evaluated on the raw I/Q dataset across all SNR levels.	81
5.27	Feature importance plot showing the top 20 features and their average impact on the model output magnitude for all the observations in the test set.	82
5.28	Decision plots illustrating the contribution of the top 10 features for the prediction of different samples in the test set.	83
5.29	Dependence plots illustrating the effect of the single most important feature and its interaction with a related feature for the prediction of different modulation schemes.	84
5.30	Comparison of XGBoost, CNN and ResNet accuracy evaluated on the deepsig dataset consisting of a set of 24 modulations spanning across the SNR in range [-20, 20] dB.	85
5.31	Confusion matrix for ResNet-101 evaluated across SNR ≥ 0 dB when the train and test sets are taken from the deepsig dataset of 24 modulations. . .	86
5.32	Comparison of CNN accuracy for the transfer learning technique against the traditional learning approaches when the train and test sets spanning all SNRs and encompassing four modulations, namely, BPSK, QPSK, 16-QAM and 64-QAM are taken from our RF signal dataset and the deepsig dataset, respectively in (a) and vice-versa in (b).	87
A.1	File structure of the stored RF signal dataset where the sub-folders indicate the configuration parameters and the file names include the modulation type. For the interference case, the file names also include the SNR . Further, the deepsig dataset is stored separately in its raw and featurized form.	91
A.2	Confusion matrix of DNN for the baseline scenario.	94
A.3	Comparison of the classification accuracy of CNN and ResNet evaluated on the non-interfered data comprising the SC modulations and spanning across all SNRs.	95
A.4	Confusion matrix of CNN for the evaluation on SC modulation dataset covering the complete range of SNR values.	95
A.5	Confusion matrix of ResNet for the evaluation on SC modulation dataset covering the complete range of SNR values.	96
A.6	Confusion matrix of XGBoost at SNR=0 dB on the set featurized using the dataset comprising 1024 samples per segment.	97

BIBLIOGRAPHY

- [1] "Cisco Annual Internet Report (2018–2023) White Paper". [Online]. Available: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html> [Last visited: 01-10-2020].
- [2] Y. Shi, K. Davaslioglu, Y. E. Sagduyu, W. C. Headley, M. Fowler, and G. Green, "Deep learning for RF signal classification in unknown and dynamic spectrum environments," *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, Nov 2019. [Online]. Available: <http://dx.doi.org/10.1109/DySPAN.2019.8935684>
- [3] J. A. Sills, "Maximum-likelihood modulation classification for PSK/QAM," *MILCOM 1999. IEEE Military Communications Conference Proceedings (Cat. No.99CH36341)*, vol. 1, pp. 217–220 vol.1, 1999.
- [4] P. C. Sapiano and J. D. Martin, "Maximum likelihood PSK classifier," in *Proceedings of MILCOM '96 IEEE Military Communications Conference*, vol. 3, 1996, pp. 1010–1014 vol.3.
- [5] A. Polydoros and K. Kim, "On the detection and classification of quadrature digital modulations in broad-band noise," *IEEE Transactions on Communications*, vol. 38, no. 8, pp. 1199–1211, 1990.
- [6] B. F. Beidas and C. L. Weber, "Asynchronous classification of MFSK signals using the higher order correlation domain," *IEEE Transactions on Communications*, vol. 46, no. 4, pp. 480–493, 1998.
- [7] P. Panagiotou, A. Anastasopoulos, and A. Polydoros, "Likelihood ratio tests for modulation classification," in *MILCOM 2000 Proceedings. 21st Century Military Communications. Architectures and Technologies for Information Superiority (Cat. No.00CH37155)*, vol. 2, 2000, pp. 670–674 vol.2.
- [8] Liang Hong and K. C. Ho, "Modulation classification of BPSK and QPSK signals using a two element antenna array receiver," in *2001 MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force (Cat. No.01CH37277)*, vol. 1, 2001, pp. 118–122 vol.1.
- [9] S. . Hsue and S. S. Soliman, "Automatic modulation recognition of digitally modulated signals," in *IEEE Military Communications Conference, 'Bridging the Gap. Interoperability, Survivability, Security'*, 1989, pp. 645–649 vol.3.

- [10] L. Hong and K. C. Ho, "Identification of digital modulation types using the wavelet transform," in *MILCOM 1999. IEEE Military Communications. Conference Proceedings (Cat. No.99CH36341)*, vol. 1, 1999, pp. 427–431 vol.1.
- [11] A. Swami and B. M. Sadler, "Hierarchical digital modulation classification using cumulants," *IEEE Transactions on Communications*, vol. 48, no. 3, pp. 416–429, 2000.
- [12] G. Hatzichristos and M. P. Fargues, "A hierarchical approach to the classification of digital modulation types in multipath environments," in *Conference Record of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers (Cat.No.01CH37256)*, vol. 2, 2001, pp. 1494–1498 vol.2.
- [13] S. S. Soliman and S. . Hsue, "Signal classification using statistical moments," *IEEE Transactions on Communications*, vol. 40, no. 5, pp. 908–916, 1992.
- [14] Luo Lichun, "Comments on "Signal classification using statistical moments"," *IEEE Transactions on Communications*, vol. 50, no. 2, pp. 195–, 2002.
- [15] C. d. Vrieze, L. Simić, and P. Mähönen, "The importance of being earnest: Performance of modulation classification for real RF signals," *2018 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, Oct 2018. [Online]. Available: <http://dx.doi.org/10.1109/DySPAN.2018.8610499>
- [16] M. Petrova, P. Mähönen, and A. Osuna, "Multi-class classification of analog and digital signals in cognitive radios using support vector machines," in *2010 7th International Symposium on Wireless Communication Systems*. IEEE, 2010, pp. 986–990.
- [17] L. Mingquan, X. Xianci, and L. Lemin, "AR modeling-based features extraction of multiple signals for modulation recognition," in *ICSP '98. 1998 Fourth International Conference on Signal Processing (Cat. No.98TH8344)*, vol. 2, 1998, pp. 1385–1388 vol.2.
- [18] L. Mingquan, X. Xianci, and L. Leming, "Cyclic spectral features based modulation recognition," in *Proceedings of International Conference on Communication Technology. ICCT '96*, vol. 2, 1996, pp. 792–795 vol.2.
- [19] B. G. Mobasser, "Digital modulation classification using constellation shape," *Signal processing*, vol. 80, no. 2, pp. 251–277, 2000.
- [20] E. E. Azzouz and A. K. Nandi, "Modulation recognition using artificial neural networks," in *Automatic Modulation Recognition of Communication Signals*. Springer, 1996, pp. 132–176.
- [21] K. E. Nolan, L. Doyle, D. O'Mahony, and P. Mackenzie, "Modulation scheme recognition techniques for software radio on a general purpose processor platform," in *Proceedings of the First Joint IEI/IEE Symposium on Telecommunication Systems, Dublin*, 2001.

- [22] S. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification." *IEEE transactions on neural networks*, vol. 3, no. 5, p. 683, 1992.
- [23] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, contour and grouping in computer vision*. Springer, 1999, pp. 319–345.
- [24] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018.
- [25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. [Online]. Available: <http://dx.doi.org/10.1109/cvpr.2016.90>
- [26] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.
- [27] B. Razavi, *Design of integrated circuits for optical communications*. John Wiley & Sons, 2012.
- [28] T. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. USA: Prentice Hall PTR, 2001.
- [29] J. G. Proakis and M. Salehi, *Digital communications*, 4th ed. McGraw-Hill Higher Education, 2001.
- [30] M. Lapan, *Deep Reinforcement Learning Hands-On: Apply Modern RL Methods, with Deep Q-Networks, Value Iteration, Policy Gradients, TRPO, AlphaGo Zero and More*. Packt Publishing, 2018.
- [31] T. Chen and C. Guestrin, "XGBoost," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2016. [Online]. Available: <http://dx.doi.org/10.1145/2939672.2939785>
- [32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [33] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.
- [34] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [35] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.
- [36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

- [37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [38] J. Zhang, "Gradient descent based optimization algorithms for deep learning models training," *arXiv preprint arXiv:1903.03614*, 2019.
- [39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [40] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.
- [41] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.
- [42] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.
- [43] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.
- [44] L. S. Shapley, "A value for n-person games," *Contributions to the Theory of Games*, vol. 2, no. 28, pp. 307–317, 1953.
- [45] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in neural information processing systems*, 2017, pp. 4765–4774.
- [46] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," *arXiv preprint arXiv:1802.03888*, 2018.
- [47] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 427–436.
- [48] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
- [49] A. Bendale and T. E. Boulton, "Towards open set deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1563–1572.
- [50] W. J. Scheirer, A. Rocha, R. J. Michalek, and T. E. Boulton, "Meta-recognition: The theory and practice of recognition score analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 8, pp. 1689–1695, 2011.
- [51] P. Zhang, J. Wang, A. Farhadi, M. Hebert, and D. Parikh, "Predicting failures of vision systems," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3566–3573.
- [52] W. J. Scheirer, A. Rocha, R. Michaels, and T. E. Boulton, "Meta-recognition: The theory and practice of recognition score analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 33, pp. 1689–1695, 2011.

- [53] F. Hameed, O. A. Dobre, and D. C. Popescu, "On the likelihood-based approach to modulation classification," *IEEE Transactions on Wireless Communications*, vol. 8, no. 12, pp. 5884–5892, 2009.
- [54] Z. Zhu and A. K. Nandi, *Automatic modulation classification: principles, algorithms and applications*. John Wiley & Sons, 2015.
- [55] C.-Y. Huan and A. Polydoros, "Likelihood methods for MPSK modulation classification," *IEEE Transactions on Communications*, vol. 43, no. 2/3/4, pp. 1493–1504, 1995.
- [56] A. Abdi, O. A. Dobre, R. Choudhry, Y. Bar-Ness, and W. Su, "Modulation classification in fading channels using antenna arrays," in *IEEE MILCOM 2004. Military Communications Conference, 2004.*, vol. 1. IEEE, 2004, pp. 211–217.
- [57] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "Survey of automatic modulation classification techniques: classical approaches and new trends," *IET communications*, vol. 1, no. 2, pp. 137–156, 2007.
- [58] E. E. Azzouz and A. K. Nandi, *Automatic Modulation Recognition of Communication Signals*. USA: Kluwer Academic Publishers, 1996.
- [59] A. K. Nandi and E. E. Azzouz, "Algorithms for automatic modulation recognition of communication signals," *IEEE Transactions on communications*, vol. 46, no. 4, pp. 431–436, 1998.
- [60] E. E. Azzouz and A. K. Nandi, "Automatic identification of digital modulation types," *Signal Processing*, vol. 47, no. 1, pp. 55–69, 1995.
- [61] A. Swami and B. M. Sadler, "Hierarchical digital modulation classification using cumulants," *IEEE Transactions on communications*, vol. 48, no. 3, pp. 416–429, 2000.
- [62] H.-C. Wu, M. Saquib, and Z. Yun, "Novel automatic modulation classification using cumulant features for communications via multipath channels," *IEEE Transactions on Wireless Communications*, vol. 7, no. 8, pp. 3098–3105, 2008.
- [63] V. D. Orlic and M. L. Dukic, "Automatic modulation classification algorithm using higher-order cumulants under real-world channel conditions," *IEEE Communications Letters*, vol. 13, no. 12, pp. 917–919, 2009.
- [64] W. Gardner, W. Brown, and C.-K. Chen, "Spectral correlation of modulated signals: Part ii-digital modulation," *IEEE Transactions on Communications*, vol. 35, no. 6, pp. 595–601, 1987.
- [65] P. Marchand, C. Le Martret, and J.-L. Lacoume, "Classification of linear modulations by a combination of different orders cyclic cumulants," in *Proceedings of the IEEE Signal Processing Workshop on Higher-Order Statistics*. IEEE, 1997, pp. 47–51.
- [66] K. Ho, W. Prokopiw, and Y. Chan, "Modulation identification of digital signals by the wavelet transform," *IEE Proceedings-Radar, Sonar and Navigation*, vol. 147, no. 4, pp. 169–176, 2000.

- [67] M. D. Wong and A. K. Nandi, "Automatic digital modulation recognition using artificial neural network and genetic algorithm," *Signal Processing*, vol. 84, no. 2, pp. 351–365, 2004.
- [68] Z. Wu, X. Wang, Z. Gao, and G. Ren, "Automatic digital modulation recognition based on support vector machines," in *2005 International Conference on Neural Networks and Brain*, vol. 2. IEEE, 2005, pp. 1025–1028.
- [69] C.-S. Park, J.-H. Choi, S.-P. Nah, W. Jang, and D. Y. Kim, "Automatic modulation recognition of digital signals using wavelet features and svm," in *2008 10th International Conference on Advanced Communication Technology*, vol. 1. IEEE, 2008, pp. 387–390.
- [70] T. Liu, Y. Guan, and Y. Lin, "Research on modulation recognition with ensemble learning," *EURASIP Journal on Wireless Communications and Networking*, vol. 2017, no. 1, p. 179, 2017.
- [71] Z. Zhang, Y. Li, S. Jin, Z. Zhang, H. Wang, L. Qi, and R. Zhou, "Modulation signal recognition based on information entropy and ensemble learning," *Entropy*, vol. 20, no. 3, p. 198, 2018.
- [72] B. Kim, J. Kim, H. Chae, D. Yoon, and J. W. Choi, "Deep neural network-based automatic modulation classification technique," in *2016 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2016, pp. 579–582.
- [73] J. Lee, B. Kim, J. Kim, D. Yoon, and J. W. Choi, "Deep neural network-based blind modulation classification for fading channels," in *2017 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2017, pp. 551–554.
- [74] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *International conference on engineering applications of neural networks*. Springer, 2016, pp. 213–226.
- [75] X. Liu, D. Yang, and A. El Gamal, "Deep neural network architectures for modulation classification," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2017, pp. 915–919.
- [76] S. C. Hauser, W. C. Headley, and A. J. Michaels, "Signal detection effects on deep neural networks utilizing raw iq for modulation classification," in *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*. IEEE, 2017, pp. 121–127.
- [77] K. Yashashwi, A. Sethi, and P. Chaporkar, "A learnable distortion correction module for modulation recognition," *IEEE Wireless Communications Letters*, vol. 8, no. 1, pp. 77–80, 2018.
- [78] F. Meng, P. Chen, L. Wu, and X. Wang, "Automatic modulation classification: A deep learning enabled approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10 760–10 772, 2018.

- [79] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, 2018.
- [80] S. Hu, Y. Pei, P. P. Liang, and Y.-C. Liang, "Deep neural network for robust modulation classification under uncertain noise conditions," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 564–577, 2019.
- [81] S. Ramjee, S. Ju, D. Yang, X. Liu, A. E. Gamal, and Y. C. Eldar, "Fast deep learning for automatic modulation classification," *arXiv preprint arXiv:1901.05850*, 2019.
- [82] Y. Wang, M. Liu, J. Yang, and G. Gui, "Data-driven deep learning for automatic modulation recognition in cognitive radios," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 4074–4077, 2019.
- [83] Y. Wang, J. Yang, M. Liu, and G. Gui, "Lightamc: Lightweight automatic modulation classification via deep learning and compressive sensing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3491–3495, 2020.
- [84] T. J. O'shea and N. West, "Radio machine learning dataset generation with gnu radio," in *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, 2016.
- [85] 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 (WOCC)*. IEEE, 2017, pp. 1–5.
- [86] S. Peng, H. Jiang, H. Wang, H. Alwageed, Y. Zhou, M. M. Sebdani, and Y.-D. Yao, "Modulation classification based on signal constellation diagrams and deep learning," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 3, pp. 718–727, 2018.
- [87] S. Zhou, Z. Yin, Z. Wu, Y. Chen, N. Zhao, and Z. Yang, "A robust modulation classification method using convolutional neural networks," *EURASIP Journal on Advances in Signal Processing*, vol. 2019, no. 1, p. 21, 2019.
- [88] Y. Xu, D. Li, Z. Wang, Q. Guo, and W. Xiang, "A deep learning method based on convolutional neural network for automatic modulation classification of wireless signals," *Wireless Networks*, vol. 25, no. 7, pp. 3735–3746, 2019.
- [89] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.
- [90] D. Hong, Z. Zhang, and X. Xu, "Automatic modulation classification using recurrent neural networks," in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*. IEEE, 2017, pp. 695–700.
- [91] A. Hazza, M. Shoaib, S. A. Alshebeili, and A. Fahad, "An overview of feature-based methods for digital modulation classification," in *2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*. IEEE, 2013, pp. 1–6.

- [92] Z. Yu, "Automatic modulation classification of communication signals," Ph.D. dissertation, New Jersey Institute of Technology, August 2006.
- [93] A. M. Llenas, J. Riihijärvi, and M. Petrova, "Performance evaluation of machine learning based signal classification using statistical and multiscale entropy features," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2017, pp. 1–6.
- [94] D. R. Brillinger, *Time Series*. Society for Industrial and Applied Mathematics, 2001.
- [95] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [96] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- [97] T. Oyedare and J.-M. J. Park, "Estimating the Required Training Dataset Size for Transmitter Classification Using Deep Learning," *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pp. 1–10, 2019.
- [98] J. Cho, K. Lee, E. Shin, G. Choy, and S. Do, "How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?" *arXiv preprint arXiv:1511.06348*, 2015.
- [99] R. L. Figueroa, Q. Zeng-Treitler, S. Kandula, and L. H. Ngo, "Predicting sample size required for classification performance," *BMC Medical Informatics and Decision Making*, vol. 12, pp. 8 – 8, 2012.

Eidesstattliche Versicherung

Statutory Declaration in Lieu of an Oath

Sachdeva, Rachneet Singh

383257

Name, Vorname/Last Name, First Name

Matrikelnummer (freiwillige Angabe)

Matriculation No. (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/
Masterarbeit* mit dem Titel

I hereby declare in lieu of an oath that I have completed the present paper/Bachelor thesis/Master thesis* entitled

Deep Learning Features in RF Signal Classification

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

independently and without illegitimate assistance from third parties (such as academic ghostwriters). I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Aachen, 15.12.2020

Ort, Datum/City, Date


Unterschrift/Signature

*Nichtzutreffendes bitte streichen

*Please delete as appropriate

Belehrung:

Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

Para. 156 StGB (German Criminal Code): False Statutory Declarations

Whoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence

(1) If a person commits one of the offences listed in sections 154 through 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.

(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

I have read and understood the above official notification:

Aachen, 15.12.2020

Ort, Datum/City, Date


Unterschrift/Signature