Radar Intra-Pulse Modulation Classification Using Convolutional Neural Networks

MSc(Eng) Dissertation

Presented by:
Jean Swart

Prepared for:
A. K. Mishra
Dept. of Electrical and Electronics Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Master of Science degree in Radar and Electronic Defence

5 June 2019

Declaration

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own.

This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Signature: Signed by candidate

Date: 5 June 2019

Abstract

This dissertation presents a detailed investigation into the classification of radar intra-pulse modulation schemes. Recent years have seen increased waveform diversity in radar systems which, while making many aspects of pulse analysis more challenging, have presented new opportunities and features for the field of classification. This dissertation aims to address the increasing difficulty of pulse classification through the use of modern machine learning techniques - more specifically, by utilising convolutional neural networks.

A wide range of modulation schemes was considered and simulated with realistic imperfections to create a dataset that was as representative of real-world scenarios as possible. Data representations of varying levels of abstraction were analysed in order to investigate the effects of data formatting on the performance of various classifiers. A classifier which made use of manual feature extraction was evaluated against a series of convolutional neural network classifiers in order to establish whether improvements in classification accuracy and throughput could be realised. This study also presents research into the viability of classifying data that has been degraded by real transmitter and channel effects using classifiers trained entirely on simulated data. The operation of the tested classifiers is analysed, and parallels are drawn between the feature extraction steps in convolutional neural networks and conventional signal features.

The primary research questions in this study are whether machine learning approaches are able to improve on non-machine learning based classification techniques, and which data representations are best suited to convolutional neural network based classification.

Classifiers were tested across 28 classes of modulation, with signal-to-noise ratios uniformly distributed between -5 dB and 20 dB. It was found that substantial performance and stability improvements could be achieved when convolutional neural networks were used over the tested non-machine learning based classification technique. The most promising classifier made use of time-frequency representations as an input, and was able to achieve a classification accuracy of 98%, while exhibiting extreme robustness against noise and pulse imperfections.

Acknowledgements

This dissertation owes its existence to a number of individuals and organisations who assisted me in my endeavours throughout the project.

Firstly, my utmost thanks are owed to my managers and colleagues at Peralex Electronics for their funding contributions towards this research and their invaluable technical knowledge and support.

I would like to extend my thanks to my supervisor, Amit Mishra, for his advice, his extensive support as a source of knowledge on machine learning and radar technologies, and for assisting me in navigating the nuances of academia over the course of this project.

My thanks go out to my friends, family and loved ones. In particular, Anne Swart, Ryno Swart and Lauren Ellwood, all of whom assisted greatly in my proofreading and kept me engaged by maintaining an active interest in my research.

Thank you to all of the staff at the University of Cape Town, particularly in the faculty of Engineering and the Built Environment. I have formed strong relationships with a number of members in the faculty and, even though this dissertation marks the end of a chapter in my academic career, I am looking forward to working with them more in the future.

Finally, I thank you, the reader, for taking the time to read this dissertation. I hope you find the methods employed, results obtained, and observations made useful, insightful and inspiring.

Contents

List of Figures

List of Tables

List of Abbreviations

CFO ..... Carrier Frequency Offset
CNN ..... Convolutional Neural Network
CWD ..... Choi-Williams Distribution
ECCM ..... Electronic Counter-Countermeasures
ELINT ..... Electronic Intelligence
ESM ..... Electronic Support Measures
FFT ..... Fast Fourier Transform
FM ..... Frequency Modulation
FPGA ..... Field-Programmable Gate Array
GPU ..... Graphics Processing Unit
IQ ..... In-phase/Quadrature
LFM ..... Linear Frequency Modulation
LPD ..... Low Probability of Detection
LPI ..... Low Probability of Interception
LPR ..... Low Probability of Recognition
MSE ..... Mean Squared Error
PCA ..... Principal Component Analysis
PM ..... Phase Modulation
PRF ..... Pulse Repetition Frequency
PRI ..... Pulse Repetition Interval
PSR ..... Peak-to-Sidelobe Ratio
PW ..... Pulse Width
ReLU ..... Rectified Linear Unit
SNR ..... Signal-to-Noise Ratio
SPWVD ..... Smoothed Pseudo Wigner-Ville Distribution
TFR ..... Time-Frequency Resolution
WVD ..... Wigner-Ville Distribution
ZAM-GTFR .... Generalized Time-Frequency Distribution of Zhao, Atlas and Marks

Chapter 1

Introduction

1.1 Background to the Study

With advancements in signal processing techniques and continual improvements in the field of electronic defence, recent years have seen increased amounts of research and interest in low probability of intercept (LPI) and low probability of recognition (LPR) radar systems. LPI radars typically make use of longer, lower energy pulses than older, non-LPI radars, and both LPI and LPR radars tend to employ complex intra-pulse modulation schemes. These complex modulation schemes have created new opportunities to achieve improved radar performance while also reducing the detectability of radar emissions by third parties.

While these intra-pulse modulation schemes have clear benefits for radar manufacturers and users, the increased waveform diversity that they bring also presents new opportunities in the fields of pulse classification, emitter classification and electronic intelligence. The radar waveform and inter-pulse parameters (pulse width, pulse frequency, pulse repetition interval) are essential features for classifiers and electronic intelligence systems. While the majority of classification is performed using these parameters, the modulation scheme can, if extracted properly, assist in the classification of radars with similar inter-pulse parameters. The modulation scheme can also provide an indication of the range resolution of the radar and assist in the fingerprinting of specific emitters. Unfortunately, while it may be trivial to tell if the waveform is modulated or not or to distinguish between simple types of modulation, performing accurate modulation classification is a complex task. Many intra-pulse modulation techniques are too similar to be distinguished even by trained operators when real-world pulse imperfections are taken into account, and human-driven feature extraction becomes impractical and inefficient at low signal-to-noise ratios (SNR) and real-time data rates.

Convolutional neural networks (CNNs) present a new way to classify these intra-pulse modulation schemes. The ability to perform automatic feature extraction without human intervention allows for more complex data representations to be considered and adds the potential for lower-level features to be extracted from the signals. The rising popularity of low-cost hardware acceleration platforms and their compatibility with machine learning frameworks also opens up new possibilities for real-time classification.

1.2 Objectives of this Study

This study seeks to investigate the effectiveness of convolutional neural networks when applied to radar intra-pulse modulation classification and to produce a set of working models capable of classifying various modulation schemes across different data representations with a high degree of accuracy. It also aims to identify the most effective data representation for the input of a convolutional neural network.

The problem of signal classification in low signal-to-noise ratio environments is addressed, and other real-world problems such as carrier frequency offset (CFO) and pulse detection imperfections are accounted for.

The study also seeks to quantify whether convolutional neural networks can produce results better than or comparable to certain non-machine learning based classification approaches. The specific classification approaches under investigation are discussed in Section 3.6 of this study. The question of whether any performance improvements obtained are worth the loss in algorithmic transparency is also investigated.

Finally, this study aims to explain the operation of the convolutional neural network after training and draw parallels between the various extracted features and visual features observed through common signal processing techniques.

1.3 Scope and Limitations

This study includes research on various intra-pulse modulation techniques and the benefits and drawbacks of each. The scope of the classification task is limited to a finite set of intra-pulse modulation schemes and orders, listed below:

- Linear Frequency Modulation (LFM) (Fixed bandwidth)
- Barker Codes (Order 2, 3, 4, 5, 7, 11, 13)
- Frank Codes (Order 2, 4, 6, 8)
- P1 Codes (Order 2, 4, 6, 8)
- P2 Codes (Order 2, 4, 6, 8)
- P3 Codes (Order 4, 16, 36, 64)
- P4 Codes (Order 4, 16, 36, 64)

The choice of a fixed bandwidth LFM waveform is related to the fact that the goal of the study is to create a classification system, not a measurement system. LFM waveforms are easily measurable through common signal processing techniques such as instantaneous frequency analysis, and have been thoroughly investigated in prior literature.

The scope of this study covers the research of various signal representations in order to determine which representations are best suited for use as inputs to a convolutional neural network. Investigating data representations is important as different representations have different processing requirements, some of which may be less feasible for real-world or real-time operation than others.

Convolutional neural network design is researched, and a set of fully operational convolutional neural networks capable of intra-pulse modulation classification is developed, trained and tested. This study does not cover the low-level implementation of the convolutional neural networks, and instead makes use of open source machine learning frameworks and libraries. Non-machine learning based classification approaches are also researched, and a single example of such an approach is implemented and evaluated. As previously stated, the focus of this study is on the applicability of convolutional neural networks to intra-pulse modulation classification - not the re-implementation and evaluation of existing classification techniques. One of the developed CNN models is also analysed in order to attempt to draw parallels between the features extracted by the model and visual features observed using regular signal processing techniques.

Due to the classified nature of military and commercial radar waveforms and the sheer amount of data required for the training and testing of machine learning systems, it is not feasible to produce a dataset from real-world recordings. Instead, signals are simulated using a radar simulator developed at Peralex. Care is taken to ensure that the simulated signals are valid and contain a variety of realistic pulse imperfections to ensure that the classification process is as true to life as possible. The simulated data is used to perform a real-world, SDR-based experiment, which is used to validate a number of the simulated results

Real-time processing of radar signals is not attempted, and no time constraints on training and classification are imposed. However, the classification speed (in pulses per second) of different models is investigated and used to draw information on the feasibility of these models and data representations for real-world applications.

1.4 Report Outline

This study begins by presenting a summary of the current state of the fields related to intra-pulse modulation classification in the form of a **literature review**. The literature review covers a wide range of topics related to this study, from the principles of radar to the various approaches associated with modulation classification.

An overview of the research undertaken during this study is provided in the **methodology** section. This section serves to address issues related to the validity and integrity of the study and provide an overview of the various tasks undertaken and research questions to be answered during the study.

Following the methodology section, the implementation of the various modules and submodules designed for this study is detailed in the **implementation** section. This section provides technical information on the manner in which specific tasks were completed and expands on the software created to support this study.

An unbiased summary of the results of the various classifiers and experiments detailed in the implementation section is presented in the **results** section. While this section provides a breakdown of the raw results of the various tests and a cursory analysis thereof, the **discussion** section serves to draw meaning from these results and analyse them further.

Finally, the research questions proposed during the methodology section are answered in the **conclusions** section, and recommendations for future research are made in the **recommendations** section.

Chapter 2

Literature Review

2.1 Radar Principles

In their simplest form, radar systems operate by transmitting an electromagnetic wave and monitoring the amount of time that it takes for the wave to reflect off an object and travel back to the receiver. This reflection is known as the echo, and objects that are to be monitored are called targets. In reality, many different measurements can be extracted from these echoes, and there are many complex factors to consider when designing and using a radar system. This section aims to guide readers who may be unfamiliar with radar systems through their most basic principles.

The electromagnetic wave used by the radar can be emitted continuously or in pulses. These two transmission schemes allow radar systems to be broadly separated into two distinct types - continuous wave (CW) and pulsed radars.

When designing a pulsed radar, the pulse width (τ) and the pulse repetition interval (PRI) must be chosen. The pulse width is the duration of the pulse, while the pulse repetition interval is the time between successive pulses. The PRI is often represented in its inverse form as the pulse repetition frequency (PRF).

$$PRF = \frac{1}{PRI} \tag{2.1}$$

Figure 2.1 shows a typical transmit/receive cycle of a pulsed radar.

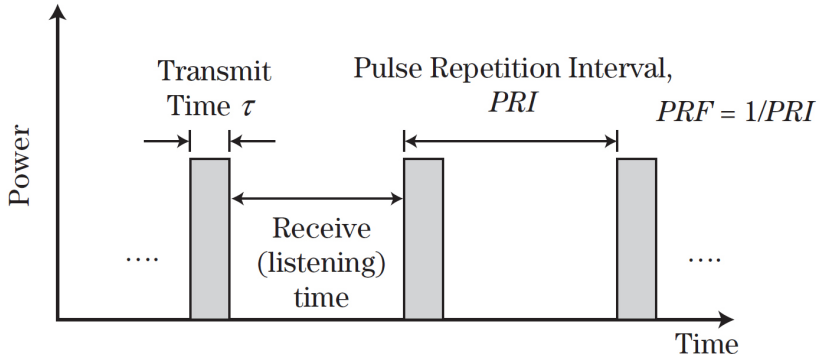


Figure 2.1: *Pulsed radar waveform [1]*

The pulse repetition frequency generally dictates the number of pulses that will be reflected off the target during the time that the antenna beam is aligned with the target. This time is known as the dwell time, t_d. Coherent pulsed radars are able to combine multiple returns from the same target into the equivalent of a single return with N times greater SNR than a regular radar return. This process is known as coherent integration, where N is the number of coherently integrated pulses.

An issue presents itself when an echo arrives at the receiver after one or more pulse repetition intervals have elapsed. When this occurs, it is no longer obvious which transmitted pulse corresponds to which echo. This issue is known as a range ambiguity, and it is for this reason that the PRF of a radar dictates the maximum unambiguous range of the radar, R_{ua}.

$$R_{ua} = \frac{c \times PRI}{2} = \frac{c}{2 \times PRF} \tag{2.2}$$

The PRF of the radar also dictates the blind velocity of the radar, v_b, which occurs when the Doppler shift of a target is equal to an integer multiple of the the pulse repetition frequency. Such a target will appear stationary to the radar, and may be suppressed if the radar employs any static clutter suppression algorithms.

$$v_b = \frac{k \times \lambda \times PRF}{2} \qquad k \in \mathbb{Z} \tag{2.3}$$

Equations 2.2 and 2.3 clearly present a design trade-off. Utilising a higher PRF will result in a higher blind velocity, but a lower maximum unambiguous range, while using a lower PRF will increase the maximum unambiguous range but result in lower blind velocities. This trade-off can be resolved by switching between multiple PRFs during operation. This is known as PRF staggering. Consider a set of predefined PRFs:

$$\{PRF_0,\ PRF_1,\ PRF_2\ \ldots\ PRF_{N-1}\}$$

For this set of PRFs, the maximum unambiguous range is limited by the maximum PRF, while the blind velocity is now determined by the least common multiple (LCM) of the PRFs. The greatest benefit to blind velocity is realised when the PRFs are coprime.

$$\begin{aligned} R_{ua} &= \frac{c}{2 \times \mathtt{max}\,(PRF)} \\ v_b &= \frac{k \times \lambda \times \mathtt{lcm}\,(PRF)}{2} \qquad k \in \mathbb{Z} \end{aligned} \tag{2.4}$$

The number of PRF staggering levels or the staggering pattern can be measured and used as a feature for emitter classification.

The strength of a return in a pulsed radar depends on a number of other parameters. These are quantified using the radar range equation (RRE), shown in Equation 2.5 for the monostatic case.

$$SNR = \frac{P_t G^2 \lambda^2 \sigma N}{(4\pi)^3 R^4 k_B T_0 F B} \tag{2.5}$$

The quantities in equation 2.5 are:

- P_t - Power of the transmitter in watts
- G - Gain of the transmit/receive antenna
- λ - Wavelength of the transmitted signal
- σ - Radar cross section (RCS) of the target
- N - Number of coherently integrated pulses
- R - Range to the target in metres
- k_B - Boltzmann constant
- T_0 - Ambient/reference temperature
- F - Noise figure
- B - Signal bandwidth

After the echo of the transmitted signal has been received, it is passed through a matched filter in order to maximise the signal-to-noise ratio. The SNR at the output of the matched filter depends solely on the signal energy (E) and the power spectral density of the noise (N_0).

$$SNR_{mf} = \frac{2E}{N_0} \tag{2.6}$$

The width of the matched filter output is dependent on the pulse duration, τ.

$$T_{mf} = 2\tau \tag{2.7}$$

The fact that the SNR at the output of the matched filter is dependent on the signal energy leads to the conclusion that in order to maximise SNR, the energy of the pulse should be maximised. This can be achieved by using higher power transmitters or longer pulses, as shown in Equation 2.8.

$$E = \int_0^\tau |x(t)|^2 \, dt \tag{2.8}$$

Higher power transmitters are more challenging to produce, more expensive to operate and easier to detect, whether it be by friendly or hostile platforms. These factors suggest that using longer pulses is the most effective way to increase the energy content of the pulse. However, as shown in Equation 2.7, longer pulses produce wider matched filter outputs, leading to issues in environments where multiple targets may be present at once, as the return from one target may mask the presence of another. The property of how well a radar is able to resolve multiple targets in range is known as the range resolution of the radar. In order to decouple pulse width from the width of the matched filter output and in turn range resolution, the concept of pulse compression was conceived.

2.2 Pulse Compression

Pulse compression involves applying a form of modulation to pulses in order to decouple the pulse width from the width of the matched filter output, which often has the effect of improving range resolution or peak-to-sidelobe ratio. The choice of modulation scheme for pulse compression is a design decision, and, when combined with the inter-pulse measurements of the radar, can be used as a feature for emitter identification and classification. Modulating individual pulses is also known as intra-pulse modulation, while modulation that is applied across multiple pulses is known as inter-pulse modulation.

Intra-pulse modulation can take on many forms, some of which are more suited to certain applications than others. These modulation types can be loosely divided into frequency modulation, binary phase modulation and polyphase modulation.

In general, frequency modulation schemes seek to minimise the main-lobe width after pulse compression and increase the tolerance to Doppler mismatching during the matched filter process. On the other hand, binary and polyphase modulation techniques seek to maximise the peak-to-sidelobe ratio of the output of the pulse compression stage [2].

2.2.1 Frequency Modulation

Frequency modulation involves applying a variation to the instantaneous frequency of the pulse over the pulse width. This approach allows for both the bandwidth and duration of the pulse to be chosen independently, thus overcoming the issue in unmodulated pulses whereby the range resolution is directly proportional to the pulse duration. Having an independently selectable bandwidth is an advantage of frequency modulation, and is not possible for phase-coded waveforms. These waveforms have a fixed bandwidth determined by the coding scheme in use.

The most common form of frequency modulation is linear frequency modulation (LFM), also known as a "chirp". The frequency of a LFM pulse at any point in time is defined in Equation 2.9, where f_0 and f_1 are the start and end frequencies of the pulse, k is the chirp rate and τ is the pulse width. The chirp rate is defined as the rate of change of the frequency of the pulse in $\frac{Hz}{s}$.

$$\begin{aligned} f(t) &= f_0 + kt \\ k &= \frac{f_1 - f_0}{\tau} \end{aligned} \tag{2.9}$$

Figures 2.2 and 2.3 show the frequency and amplitude over time of a LFM pulse with $f_0 = 0$ Hz and $f_1 = 1$ MHz. This frequency range was selected purely for visualisation.

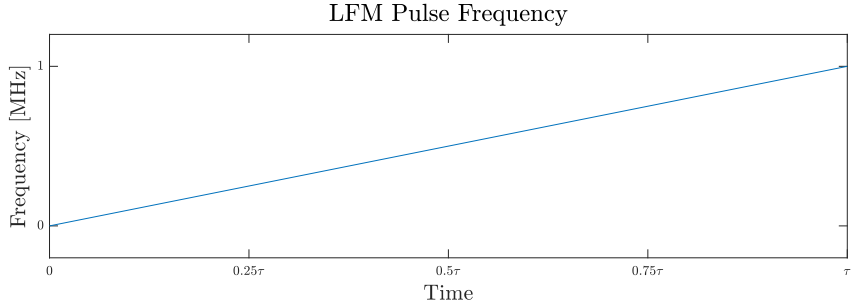


Figure 2.2: *Changing frequency in a LFM pulse*

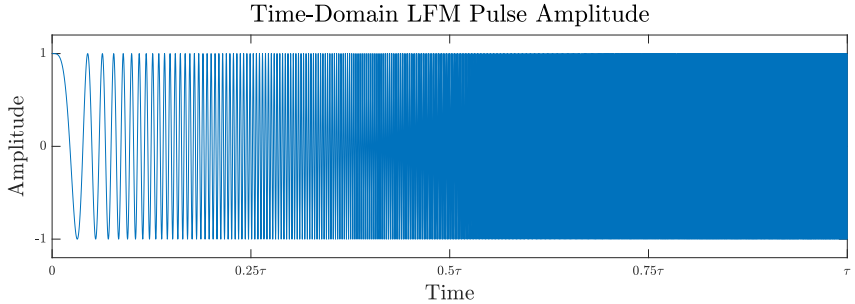


Figure 2.3: *Time-domain representation of a LFM pulse*

2.2.2 Binary Phase Modulation

Binary phase modulation is a form of phase modulation where the phase shifts are either 0 or π. A phase shift of 0 is often represented as "+1", and a phase shift of π is often represented as "-1". This convention is due to the fact that applying a phase shift of π radians is equivalent to multiplying the signal by -1.

Barker codes are the most commonly used form of binary phase modulation and have ideal autocorrelation properties. That is, a N^{th} order Barker code (or a Barker code of length N) modulated pulse will have a peak-to-sidelobe ratio (PSR) of $N:1$. Table 2.1 shows the list of all known Barker codes and their PSRs.

Table 2.1: *Table of known Barker codes*

N	Code		PSR$_{dB}$
2	+1 −1	+1 +1	−6
3	+1 +1 −1	+1 +1 −1	−9.5
4	+1 +1 −1 +1	+1 +1 +1 −1	−12
5	+1 +1 +1 −1 +1		−14
7	+1 +1 +1 −1 −1 +1 −1		−16.9
11	+1 +1 +1 −1 −1 −1 +1 −1 −1 +1 −1		−20.8
13	+1 +1 +1 +1 +1 −1 −1 +1 +1 −1 +1 −1 +1		−22.3

2.2.3 Polyphase Modulation

Polyphase modulation techniques can have any number of arbitrarily-valued phase shifts. The number of phase shifts is typically referred to as the order of the modulation scheme. A number of polyphase coding techniques exist, with many of them being designed to optimise certain properties of pulses.

One of the most common polyphase modulation schemes is the Frank code, formulated by Robert L. Frank in 1963 [3]. The construction of a N^{th} order Frank code first requires the pulse to be split into N groups. Each group is then split into N sub-pulses, and a phase shift is applied to each sub-pulse.

The phase shift applied to each sub-pulse is an integer multiple of the fundamental phase difference for a N^{th} order Frank code, shown in Equation 2.10.

$$\Delta\phi = \frac{2\pi}{N} \tag{2.10}$$

The phase shift applied to sub-pulse j in group i can be computed through Equation 2.11.

$$\phi_{i,j} = \frac{2\pi}{N}(i-1)(j-1) \tag{2.11}$$

Equation 2.11 was originally represented by Frank in the form of the matrix shown in Equation 2.12.

$$\phi = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 2 & \dots & (N-1) \\ 0 & 2 & 4 & \dots & 2(N-1) \\ 0 & 3 & 6 & \dots & 3(N-1) \\ 0 & \vdots & \vdots & \ddots & \vdots \\ 0 & (N-1) & 2(N-1) & \dots & (N-1)^2 \end{pmatrix} \Delta\phi \tag{2.12}$$

A N^{th} order Frank code is able to achieve a compression ratio of N^2 and a pulse bandwidth of $\frac{N^2}{\tau}$.

The phase transitions for a Frank code of order 8 are shown in Figure 2.4.

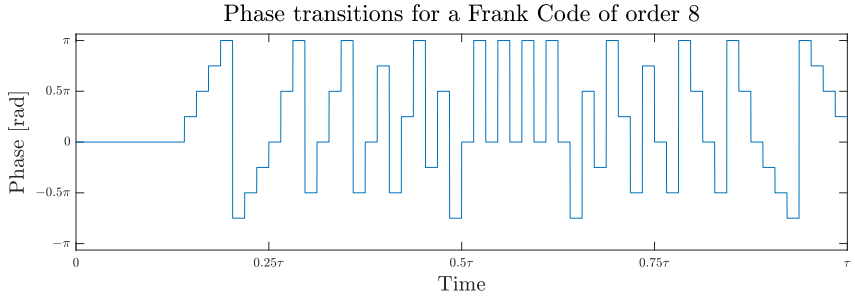


Figure 2.4: *Frank code phase plot ($N = 8$) [2]*

The P1 and P2 phase coding schemes are based off the Frank code and share the same compression ratio and pulse bandwidth properties. These techniques were proposed by B.L. Lewis and F.F. Kretschmer, Jr. in 1981 as adaptations of the Frank code that are more compatible with digital signal processing techniques where signals are bandlimited prior to sampling [4]. It should be noted that in the context of modern signal processing, any sampled signal is inadvertently band-limited due to the presence of the anti-aliasing filter in the sampler.

Any signal with an infinite frequency response will be affected by bandlimiting effects, as the truncation of the signal bandwidth results in a slight mismatch between the transmitted and received pulses. Frank codes are even more strongly affected by bandlimiting effects due to the order of the phase transitions in the code. Limiting the bandwidth of the signal results in a "smoothing" effect, whereby the abruptness at

which phase transitions can be made is reduced. This smoothing effect has the least impact on the edges of the Frank code, where phase transitions are small, and the most effect on the centre of the code, where the phase transitions are at their largest. This has the effect of attenuating the centre frequencies of the Frank code, which strongly reduces the peak-to-sidelobe ratio from its nominal value [4]. The P1 and P2 codes were designed to reduce the effects of bandlimiting by rearranging the phase transitions of the Frank code.

P1 codes consist of the same structure and phase shifts as the Frank code, with the exception that the phase transitions are rearranged such that the largest phase transitions occur at the beginning and end of the code and decrease progressively towards the centre of the code.

P2 codes consist of the same phase increments as the P1 code with a different starting phase to ensure that each of the N groups is symmetric about zero phase. Low autocorrelation sidelobes are not guaranteed for odd values of N, so the P2 code is only valid for even values of N.

The phase shift for the i^{th} sub-pulse of the j^{th} group for the P1 and P2 codes are shown in Equations 2.13 and 2.14 respectively.

$$\phi_{i,j} = -\frac{\pi}{N}\left[N - (2j-1)\right]\left[(j-1)\,N + (i-1)\right] \tag{2.13}$$

$$\phi_{i,j} = \frac{\pi}{2N}\left[N + 1 - 2i\right]\left[N + 1 - 2j\right] \tag{2.14}$$

The phase transitions for 8^{th} order P1 and P2 codes are shown in Figures 2.5 and 2.6 respectively.

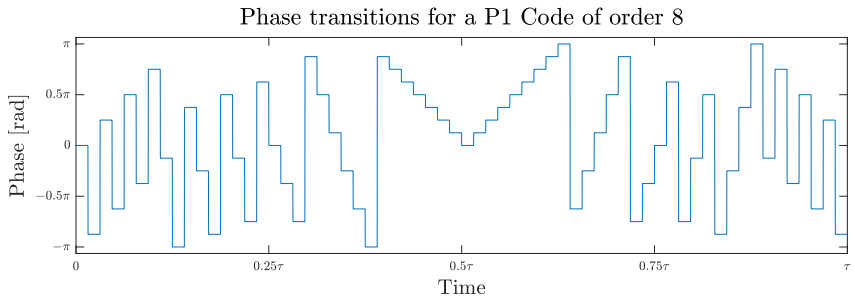


Figure 2.5: *P1 code phase plot ($N = 8$) [2]*

All of the aforementioned polyphase coding techniques are limited to code orders that are a perfect square. This is due to there being N groups and N sub-pulses, leading to a total of N^2 sub-pulses.

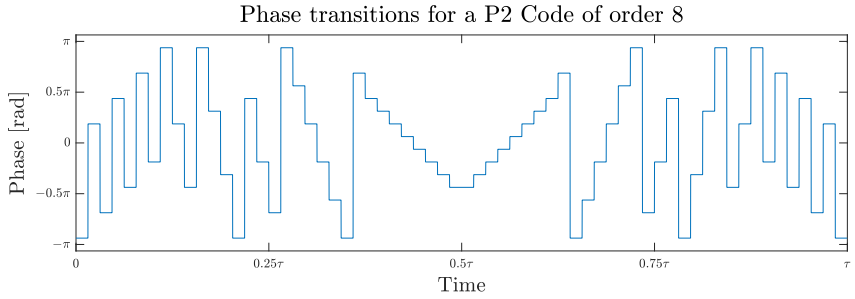


Figure 2.6: *P2 code phase plot ($N = 8$) [2]*

The P3 and P4 codes do not subdivide pulses into groups and sub-pulses and thus support arbitrary numbers of phase transitions. These techniques were designed by Lewis and Kretschmer as Doppler mismatch tolerant alternatives to other polyphase coding techniques [5].

Doppler mismatch tolerance refers to the degree to which a radar waveform is able to withstand a frequency offset between the transmitted and received signals in the matched filtering stage. This frequency offset is usually a result of a Doppler shift caused by a moving target. With this in mind, while the P3 code suffers from performance degradation if the signal is bandlimited prior to sampling, it is able to achieve a high degree of Doppler mismatch tolerance. Once again, the performance degradation due to bandlimiting arises from the fact that the phase transitions of the P3 code are at their largest in the centre of the code, as was the case for the Frank code.

Equation 2.15 shows the phase shift for the i^{th} sub-pulse in a P3 code.

$$\phi_i = \frac{\pi}{N}(i-1)^2 \tag{2.15}$$

P4 codes are able to maintain a high degree of Doppler mismatch tolerance while limiting the effects that bandlimiting has on the performance of the code. This is achieved by rearranging the P3 code to create the generalised form shown in Equation 2.16, where the phase transitions are largest at the edges of the code and smallest in the centre of the code.

$$\phi_i = \left[\frac{\pi}{N}(i-1)^2\right] - \pi(i-1) \tag{2.16}$$

Plots of the phase transitions for 64^{th} order P3 and P4 codes are shown in Figures 2.7 and 2.8.

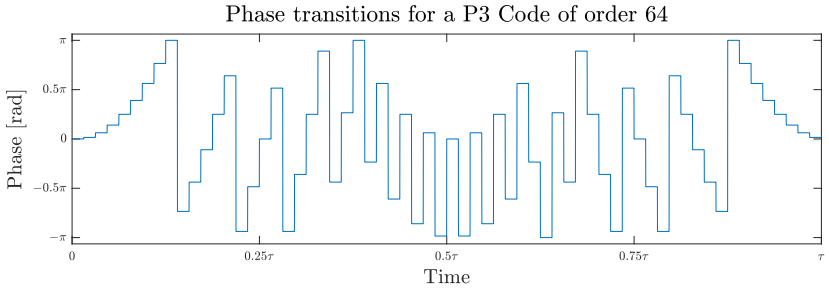


Figure 2.7: *P3 code phase plot ($N = 64$) [2]*

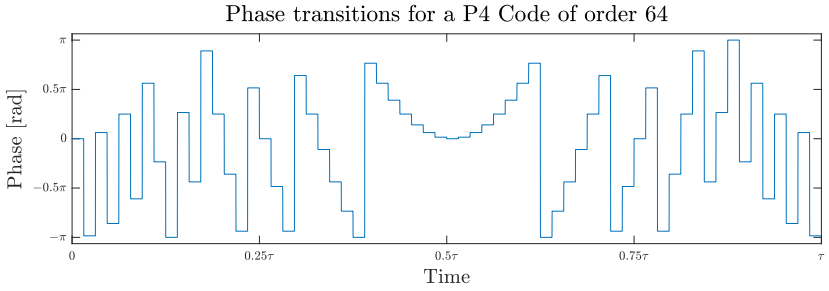


Figure 2.8: *P4 code phase plot ($N = 64$) [2]*

2.3 Signal Representations

2.3.1 Data Processing Inequality

The data processing inequality is fundamental to the field of information theory. In essence, it states that the use of local data processing on a system or piece of data cannot increase the information content of that data [6].

This is an important concept to bear in mind when considering the signal representations under discussion in this section. While different representations may appear to be better at "extracting" different types of information, no pre-processing step is capable of creating information that is not already present in the raw data. This is the motivation behind investigating classifiers capable of operating on extremely low-level data representations, as pre-processing can only result in a net decrease of information content.

2.3.2 Complex Samples

2.3.2.1 Real and Analytic Signals

The Fourier transform, $X(f)$, of any purely real signal, $x(t)$, has the property of conjugate symmetry. This means that its negative frequency components mirror its positive frequency components in a conjugate form, as per Equation 2.17.

$$X(-f) = X^*(f) \tag{2.17}$$

The Nyquist theorem defines the minimum frequency at which purely real signals can be sampled in order to avoid aliasing, f_{Nyq}, as twice the bandwidth of a band-limited signal, B.

$$f_{Nyq} \geq 2B \tag{2.18}$$

It is possible to eliminate the negative frequency component of a real signal by converting it into its analytic representation using the Hilbert transform. The Hilbert transform is best defined in the frequency domain as imparting a phase shift of 90 degrees to each frequency component of a signal. The Hilbert transform, $\hat{X}(f)$, of a frequency domain signal, $X(f)$, is given in Equation 2.19 [7].

$$\hat{X}(f) = \begin{cases} e^{-j\pi/2} X(f) & \text{for } f > 0 \\ e^{+j\pi/2} X(f) & \text{for } f < 0 \end{cases} \tag{2.19}$$

The equivalent time-domain transform is given in Equation 2.20.

$$\hat{x}(t) = \frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{x(\tau)}{t - \tau} d\tau \tag{2.20}$$

The analytic representation of a signal, $\psi(t)$, is defined in Equation 2.21, where $x(t)$ is the real signal and $\hat{x}(t)$ is the Hilbert transform of the real signal.

$$\psi(t) = x(t) + j\hat{x}(t) \tag{2.21}$$

This analytic signal has twice the positive frequency components of the original signal, and none of the negative frequency components. This is illustrated in Figure 2.9.

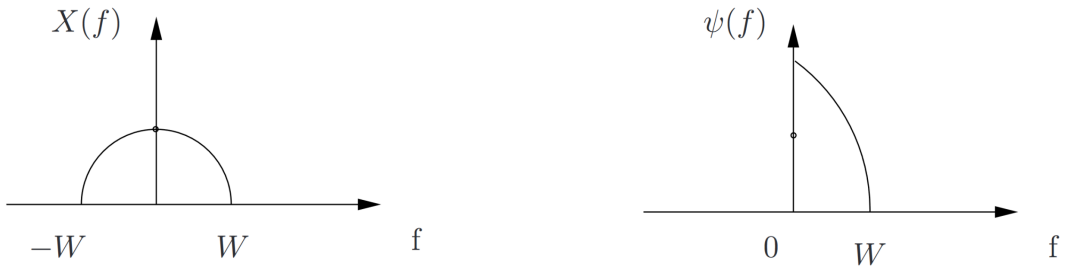


Figure 2.9: *Frequency domain representations of real and analytic signals [7]*

Converting the real signal to its analytic representation allows the signal to be unambiguously sampled at half the sample rate of the real signal, due to the truncation of all negative frequency components. This results in an effective lowering of the Nyquist criterion, where band-limited signals only need to be sampled at a rate equivalent to their bandwidth, B, at the expense of having to store twice the amount of data.

$$f_{Nyq} \geq B \tag{2.22}$$

In addition to reducing the sampling requirements of signals, the analytic signal allows for the effects of amplitude modulation and phase modulation to be observed independently using the complex envelope. The complex envelope is simply defined as the magnitude of the analytic signal, shown in Equation 2.23.

$$\mid \psi(t) \mid = \sqrt{x(t)^2 + \hat{x}(t)^2} \tag{2.23}$$

2.3.2.2 Superheterodyne Receivers

Many high fidelity modern receivers make use of superheterodyne architectures and digital down converters in order to translate the radio frequency (RF) signal into a complex baseband form. A typical block diagram of this architecture is shown in Figure 2.10.

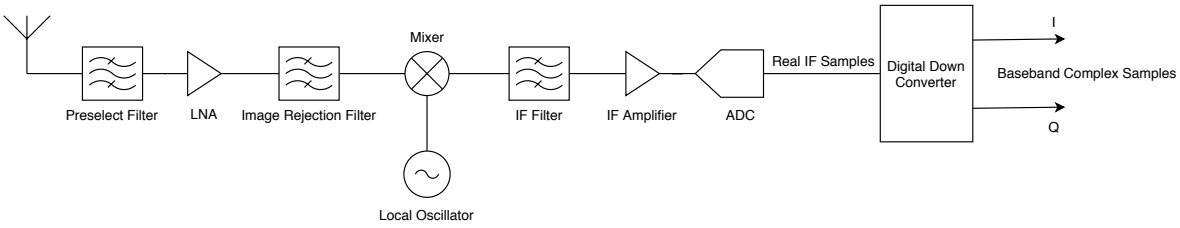


Figure 2.10: *Superheterodyne Receiver block diagram*

This architecture first makes use of a mixing stage to translate the signal at RF to a fixed intermediate frequency (IF) to alleviate sampling requirements. The IF signal is then sampled by an analog to digital converter (ADC), and is finally translated to complex baseband form for analysis using a digital down converter (DDC). This results in a single complex value for each sample consisting of in-phase (I) and quadrature (Q) components.

This is the most basic form in which data can be represented for a classifier. As this is a time-domain representation of the signal, it requires the classifier to extract any spectral data internally. It is also worth noting that the raw complex time data is usually processed into one of the other representations in this section before analysis. This raw data is extremely inefficient for humans to analyse without other tools or transformations.

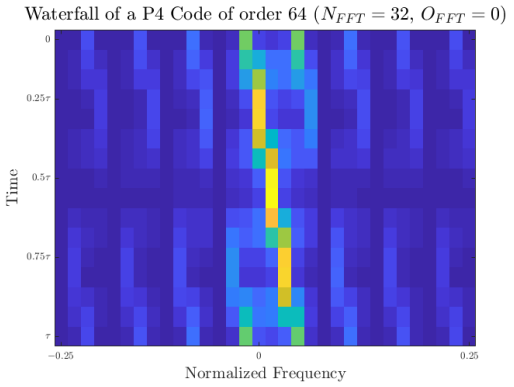
2.3.3 Time-Frequency Distributions

2.3.3.1 Waterfall Diagram

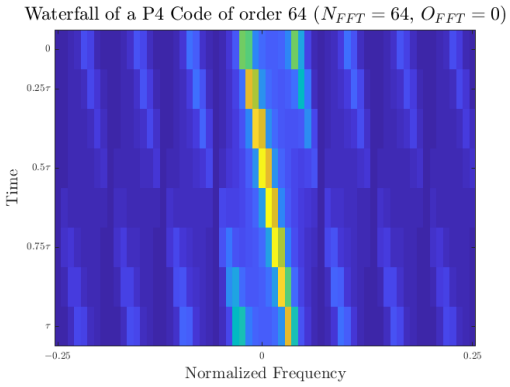
A waterfall diagram, also known as a spectrogram, is the most common form of time-frequency distribution, and is produced by vertically concatenating a number of sequential fast Fourier transforms (FFTs) of segments of the signal.

The choice of the length of the FFT is a critical factor that determines the time and frequency resolutions of the waterfall diagram. Using long FFTs will result in good frequency resolution but poor time resolution, while using short FFTs will provide poor frequency resolution but improved time resolution. The choice of spacing between FFTs is also important. It is possible to overlap FFTs in order to achieve a degree of improved time resolution, although doing so will "smear" the signal across the time-domain.

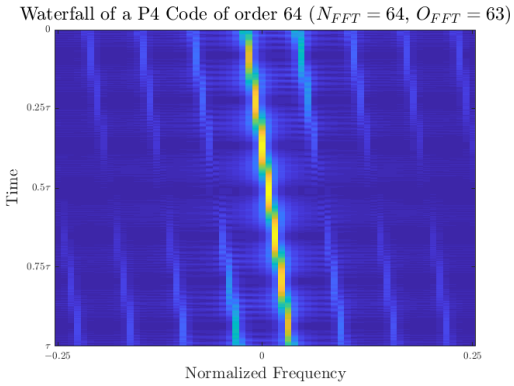
Various waterfall plots for a P4 code of order 64 are shown in Figure 2.11. Figures 2.11a and 2.11b show the trade-off between time and frequency resolution while Figures 2.11c and 2.11d illustrate time-domain "smearing" due to large amounts of overlap.



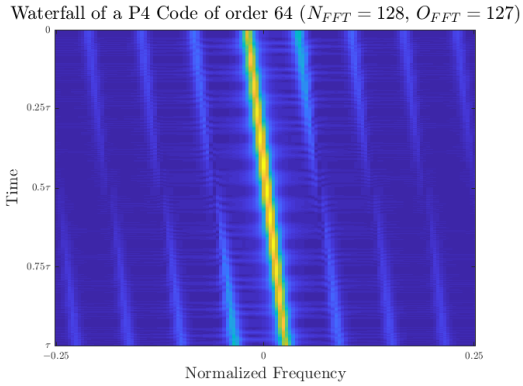
(a) *FFT length 32, no overlap*



(b) *FFT length 64, no overlap*



(c) *FFT length 64, 63-point overlap*



(d) *FFT length 128, 127-point overlap*

Figure 2.11: *Waterfall plots with varying parameters for a* 64^{th} *order P4 code*

2.3.3.2 Wigner-Ville Distribution

The Wigner-Ville distribution is an alternative to a waterfall diagram that does not come with the cost of a trade-off between time and frequency resolutions. The Wigner-Ville distribution, $W_x(t, f)$, for a signal $x(t)$ is given in Equation 2.24.

$$W_x(t, f) = \int_{-\infty}^{\infty} x\left(t + \frac{\tau}{2}\right) x^*\left(t - \frac{\tau}{2}\right) e^{-j2\pi f\tau} d\tau \tag{2.24}$$

This technique has the disadvantage of introducing cross-terms in the time-frequency plane when the signal has multiple frequency components. This is due to the quadratic nature of the Wigner-Ville distribution and can be effectively countered through the use of a smoothing kernel [8]. Time-frequency distributions that seek to minimise the effects of cross-terms in the Wigner-Ville distribution are often known as reduced-interference distributions (RIDs).

The resultant distribution of applying a smoothing kernel to a Wigner-Ville distribution is known as a smoothed pseudo-Wigner-Ville distribution (SPWVD), and has been successfully used for high-resolution, multi-component time-frequency analysis [9].

The Wigner-Ville distribution of a P4 code of order 64 is shown in Figure 2.12.

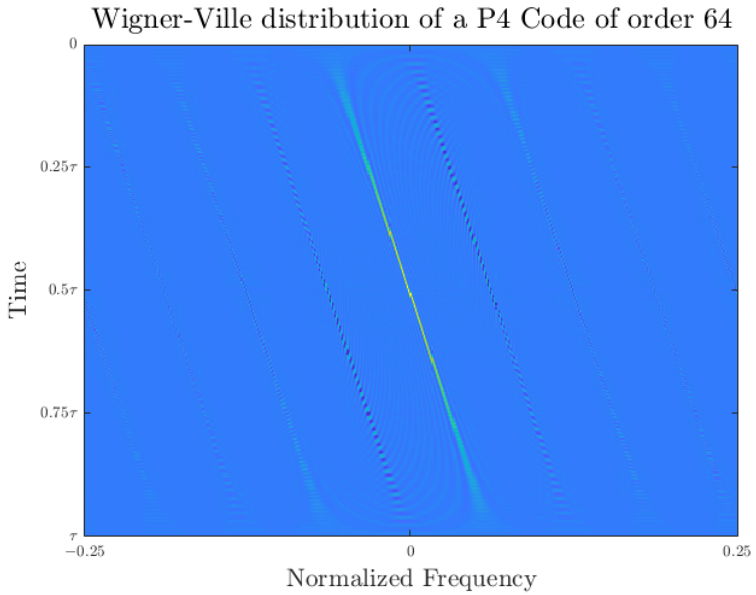


Figure 2.12: *Wigner-Ville distribution of a* 64^{th} *order P4 code*

2.3.3.3 Choi-Williams Distribution

The Choi-Williams distribution [8] is one of a number of reduced interference distributions that fall within Cohen's class [10] of distribution functions. This distribution seeks to reduce the cross-terms present in the Wigner-Ville distribution while maintaining a high degree of temporal and spectral resolution and good mathematical properties. It is able to achieve this through the use of an exponential kernel function.

Equation 2.25 shows the general form of a Cohen class distribution, which can be interpreted as the two-dimensional inverse Fourier transform of the product of a kernel and the ambiguity function of the signal. Equation 2.26 shows the ambiguity function for a signal $x(t)$, and Equation 2.27 shows the kernel function for the Choi-Williams distribution [11], [8]. These equations together form the Choi-Williams distribution.

$$C_x(t,f) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} A_x(\eta,\tau)\,\Phi(\eta,\tau)\,e^{j2\pi(\eta t-\tau f)}d\eta d\tau \tag{2.25}$$

$$A_x(\eta,\tau) = \int_{-\infty}^{\infty} x\left(t+\frac{\tau}{2}\right)x^*\left(t-\frac{\tau}{2}\right)e^{-j2\pi t\eta}dt \tag{2.26}$$

$$\Phi(\eta,\tau) = e^{-\alpha(\eta\tau)^2} \tag{2.27}$$

The term α also allows the user to control the degree of smoothing by the kernel. The Choi-Williams distribution is able to achieve partial attenuation of cross-terms while maintaining the marginal properties required for it to be an energy distribution. It should be noted, however, that the Choi-Williams distribution is only able to filter out cross-terms arising from components that differ in both time and frequency from other components.

The marginal properties state that an integral over frequency for a particular time instant should be equal to the instantaneous power of the signal and that the integral over time for a particular frequency should be equal to the spectral energy density. These properties together state that the double integral over time and frequency of the distribution will result in the energy of the signal [12].

These relations are shown in Equations 2.28, 2.29 and 2.30 for a signal $x(t)$ and a time-frequency distribution $C(t,f)$ [12].

Time marginal property (instantaneous power):

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} C(t,f)\ df = |x(t)|^2 \tag{2.28}$$

Frequency marginal property (spectral energy density):

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} P(t,f)\ dt = |X(f)|^2 \tag{2.29}$$

Energy distribution property:

$$\frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} P(t,f)\ df dt = E_x \tag{2.30}$$

The fact that the Choi-Williams distribution can satisfy these marginal properties while still able to attenuate cross-terms makes it a common choice for time-frequency analysis, and it has been successfully used as the input to a convolutional neural network for intra-pulse modulation classification [13].

The Choi-Williams distribution of a P4 code of order 64 is shown in Figure 2.13.

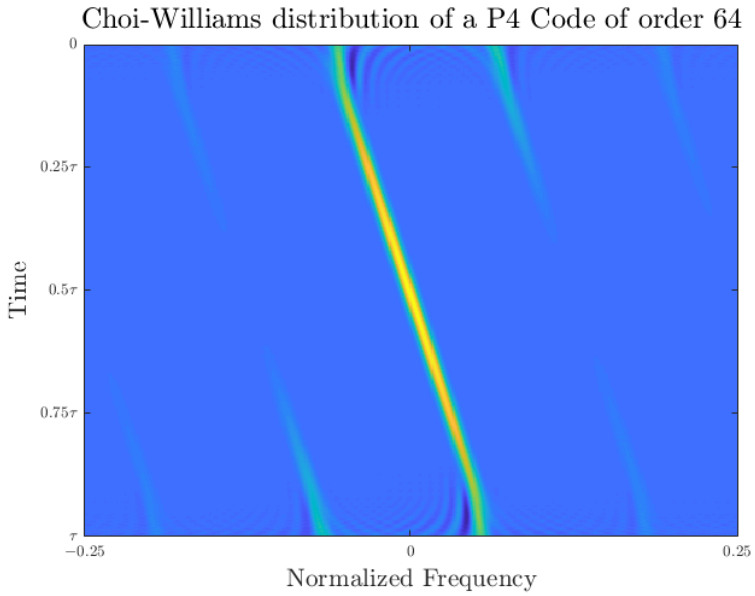


Figure 2.13: *Choi-Williams distribution of a* 64^{th} *order P4 code*

2.3.3.4 ZAM-GTFR

The generalised time-frequency distribution of Zhao, Atlas and Marks (ZAM-GTFR) [14] is also a member of Cohen's class of distribution functions.

The primary advantage of this distribution is the ability to completely remove cross-terms between components at identical frequencies due to its "cone-shaped" kernel function [15]. However, it is not able to remove cross-terms arising from components occurring at the same time.

The kernel of the ZAM-GTFR is given in Equation 2.31. This can be used in conjunction with the general form of a Cohen class distribution shown in Equation 2.25 to compute the ZAM-GTFR.

$$\Phi(\eta, \tau) = sinc(\pi\eta\tau)e^{-2\pi\alpha\tau^2} \tag{2.31}$$

The $e^{-2\pi\alpha\tau^2}$ term acts as an exponential windowing function for the kernel, noted as $p(\tau)$ in papers by authors such as such as Oh and Marks [15].

The ZAM-GTFR of a P4 code of order 64 is shown in Figure 2.14.

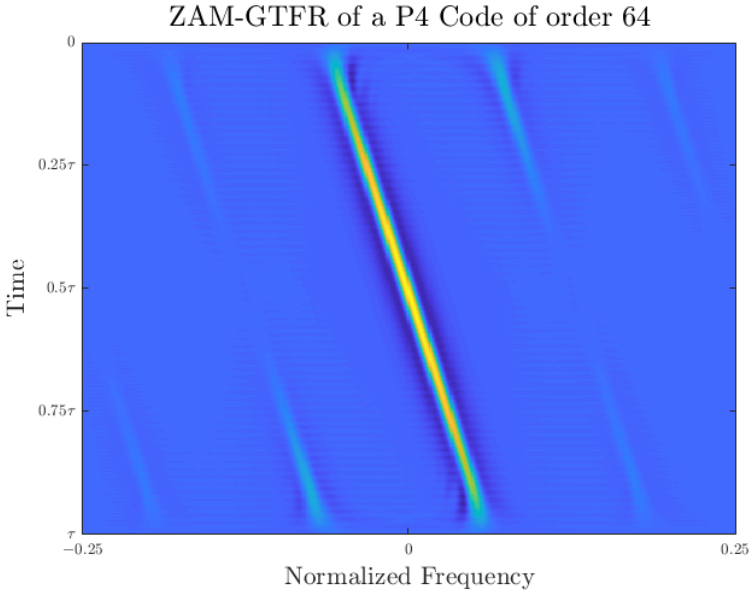


Figure 2.14: *ZAM-GTFR of a* 64^{th} *order P4 code*

2.3.4 Instantaneous Measurements

Raw IQ data can also be represented through a set of instantaneous measurements. The term "instantaneous measurements" in this context refers to a series of time-domain data that represents a property of the series measured on a per-sample basis.

The most relevant measurements for waveform classification are instantaneous magnitude, instantaneous phase and instantaneous frequency.

2.3.4.1 Instantaneous Magnitude

The instantaneous magnitude of a complex-valued signal can be obtained by simply calculating the absolute value of each sample in the signal. This was described in Section 2.3.2.1 as the complex envelope of the signal.

The instantaneous magnitude of a signal x for each sample n is given by Equation 2.32. The instantaneous magnitude for a noiseless P4 coded pulse of order 64 is superimposed with its real component in Figure 2.15.

$$|x[n]| = \sqrt{\Re(x[n])^2 + \Im(x[n])^2} \tag{2.32}$$

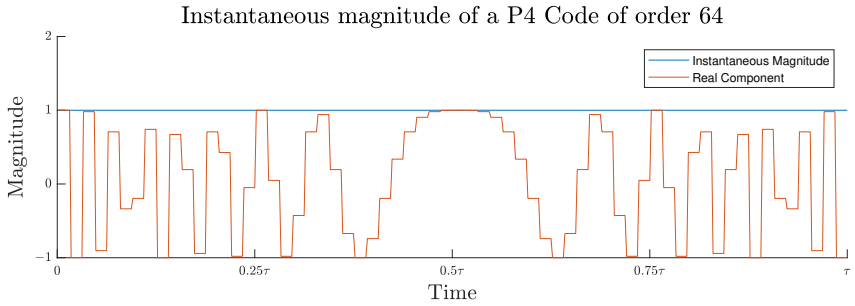


Figure 2.15: *Instantaneous magnitude of a P4 code of order 64*

2.3.4.2 Instantaneous Phase

The instantaneous phase of a complex-valued signal can be obtained by observing the argument (complex angle) of each sample in the signal.

Equation 2.33 shows the instantaneous phase for the n^{th} sample of a signal x. The instantaneous phase of a noiseless P4 coded pulse of order 64 is shown in Figure 2.16.

$$\phi[n] = arg(x[n]) \tag{2.33}$$

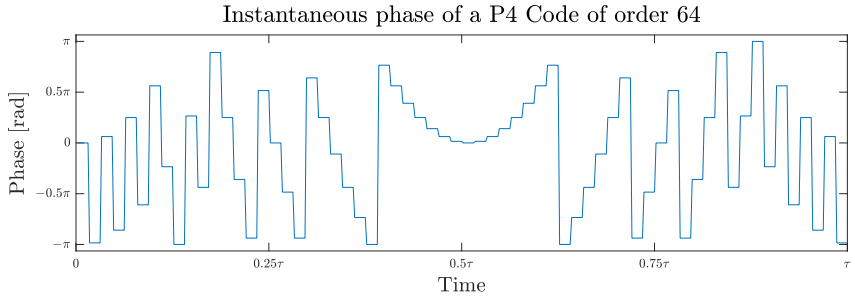


Figure 2.16: *Instantaneous phase of a P4 code of order 64*

2.3.4.3 Instantaneous Frequency

The instantaneous frequency of a signal can be extracted by observing the phase progression between successive samples in the signal, and is given in Equation 2.34.

$$f[n] = arg\left(x[n]\, x^*[n-1]\right) \tag{2.34}$$

The instantaneous frequency of each sample is directly proportional to the magnitude of the phase change between the sample and the preceding sample. Phase changes of π radians will result in the maximum attainable instantaneous frequency of $\frac{f_s}{2}$.

The instantaneous frequency for a noiseless P4 coded pulse of order 64 is shown in Figure 2.17. Note the presence of "spikes" in the plot at the positions of phase transitions in the P4 order 64 code. As stated, these spikes have frequencies proportional to the phase difference between the samples on either side of the transition.

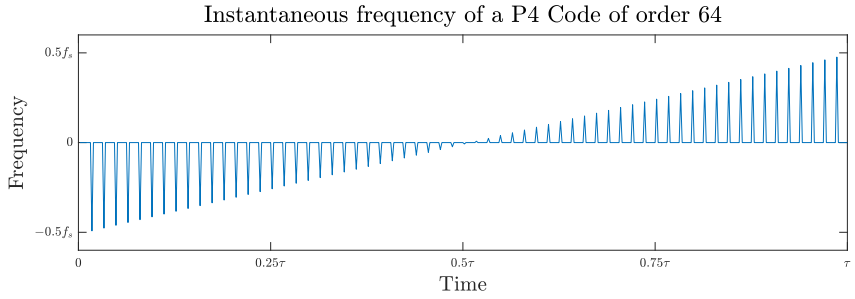


Figure 2.17: *Instantaneous frequency of a P4 code of order 64*

These instantaneous frequency spikes are easily extractable using peak-detection algorithms, and their frequency and location can be a useful feature for classifiers.

2.4 Machine Learning

2.4.1 Naïve Bayes Classifier

Naïve Bayes classifiers are based on Bayes' theorem and operate under the "naïve" assumption that all features are statistically independent from one another [16]. Assume that a classification problem consists of a set of K different classes, c_1 to c_K.

$$C = \{c_1, c_2, c_3, \ldots, c_K\} = \{c_k\} \text{ where } \{k \in \mathbb{N} : k \leq K\} \tag{2.35}$$

For the same problem, let each observation be comprised of a number of features, x_1 to x_n, where n is the total number of features.

$$X = \{x_1, x_2, x_3, \ldots, x_n\} = \{x_i\} \text{ where } \{i \in \mathbb{N} : i \leq n\} \tag{2.36}$$

Bayes' theorem in the context of such a problem is shown in Equation 2.37.

$$P(c_k \mid X) = \frac{P(X \mid c_k)\, P(X)}{P(X)} \tag{2.37}$$

The following quantities are defined in Equation 2.37:

- $P(c_k \mid X)$ is the **posterior** probability of the class c_k given the set of features, X.
- $P(X \mid c_k)$ is the **likelihood**, or the probability of observing X given the class, c_k.
- $P(c_k)$ is the **prior** probability of the class c_j.
- $P(X)$ is the **evidence**, which is the prior probability of the data, X.

The denominator of Equation 2.37 is not dependent on c_k and is given for a particular observation, making it effectively constant.

The numerator can be broken down using the definition of conditional probability and the assumption of conditional independence into the form shown in Equation 2.38.

$$P(c_k \mid X) \propto P(c_k) \prod_{i=1}^{n} P(x_i \mid c_k) \tag{2.38}$$

Equation 2.38 is represented as a proportion instead of an equality as the scaling factor of $\frac{1}{P(X)}$ is not shown. This constant of proportionality is not strictly needed for the construction of a classifier from the presented model.

In order to construct a classifier from the model in Equation 2.38, a decision rule is needed. A common decision rule is the "maximum a posteriori" (MAP) [16] rule, which simply picks the hypothesis that results in the highest posterior probability given the feature vector. Equation 2.39 shows a naïve Bayes classifier operating using the MAP decision rule, where a class label c is assigned based on the class corresponding to the maximum posterior probability.

$$c = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; P\left(c_k\right) \prod_{i=1}^{n} P\left(x_i \mid c_k\right) \tag{2.39}$$

The prior probabilities for each class are easily calculated through analysis of the training set. Let N_k be the number of samples of the class c_k in the dataset, and let N be the total number of samples in the dataset. The prior probability for a class c_k is then given by Equation 2.40.

$$P\left(c_k\right) = \frac{N_k}{N} \tag{2.40}$$

The final remaining parameter to be computed or estimated are the likelihoods of the features for the given class, $P\left(x_i \mid c_k\right)$. In most cases, this is achieved by assuming that the features follow a particular distribution, the most common of which is the normal distribution. Naïve Bayes classifiers that operate under the assumption that all features follow a normal distribution are known as Gaussian naïve Bayes classifiers.

The use of Naïve Bayes classifiers in the field of radar analysis is explored further in Section 2.5.2.

2.4.2 Nearest Neighbour Classifiers

The nearest neighbour (NN) rule is one of the simplest procedures that can be used for classification problems. In essence, it classifies a sample based on its position in the feature space, and the label of its nearest neighbour in the feature space [16].

As defined when discussing the Naïve Bayes classifier, consider a set of classes, C, and a set of features making up a single observation, X.

$$C = \{c_1, c_2, c_3, \ldots, c_K\} = \{c_k\} \text{ where } \{k \in \mathbb{N} : k \leq K\} \tag{2.35}$$

$$X = \{x_1, x_2, x_3, \ldots, x_n\} = \{x_i\} \text{ where } \{i \in \mathbb{N} : i \leq n\} \tag{2.36}$$

Each observation in the training set of size N is represented by a pattern, p_j, which is simply a combination of its feature vector and class. The set of all patterns in the training set is given in Equation 2.41.

$$P = \{p_1, p_2, p_3, \ldots, p_N\} = \{p_j\} \text{ where } \{j \in \mathbb{N} : j \leq N\} \tag{2.41}$$

The definition of the pattern p_j for observation j is shown in Equation 2.42.

$$p_j = (X_j, c_j) = (x_1, x_2, \ldots, x_n, c_j) \tag{2.42}$$

In a nearest neighbour classifier, the class label of the test pattern is assigned based on the class label of the training pattern that is nearest in the feature space. Consider a two-dimensional classification problem ($n = 2$) as shown in Figure 2.18, where 6 training patterns have been observed for each of the 3 distinct classes ($K = 3$, $N = 18$), and a single test pattern (represented as a black $\times$) is under analysis.

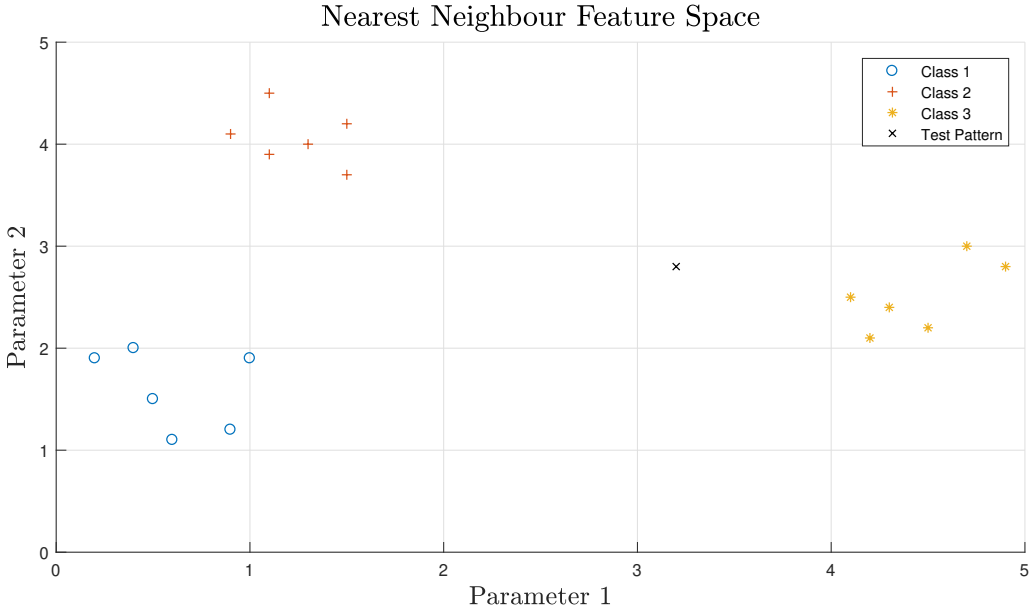


Figure 2.18: *Nearest neighbour example scenario*

In order to determine which of the training patterns is closest to the test pattern, the Euclidean distance between the test pattern T and every other pattern p_j is calculated.

$$d(T, p_j) = \sqrt{(T[1] - p_j[1])^2 + (T[2] - p_j[2])^2} \tag{2.43}$$

For the example in Figure 2.18, the nearest neighbour to the test point T is the leftmost point of the class 3 cluster. This means that test pattern is taken to belong to class 3.

In practice, more than a single nearest neighbour is generally used to perform classification. This approach is known as the k-Nearest Neighbour (k-NN) algorithm, where k is the number of nearest neighbours that are considered for in the classification step. With this approach, the class that makes up the majority of the k nearest neighbours is taken as the decision for the test pattern.

Using multiple nearest neighbours makes the classifier more robust, and can reject outliers in the training patterns. The exact number of nearest neighbours is often determined by experimentation as the value that results in the lowest classification error.

The use of nearest neighbour-based approaches in the field of radar analysis and classification is discussed in Section 2.5.3.

2.4.3 Artificial Neural Networks

Artificial neural networks are generally used as a method to approximate functions that translate an arbitrary number of inputs to an arbitrary number of outputs. They are inspired by neural networks in the human brain and the fully connected nature of each neuron to its adjacent neurons.

Figure 2.19 shows the structure of a simple three-layer neural network; consisting of input, hidden and output layers.

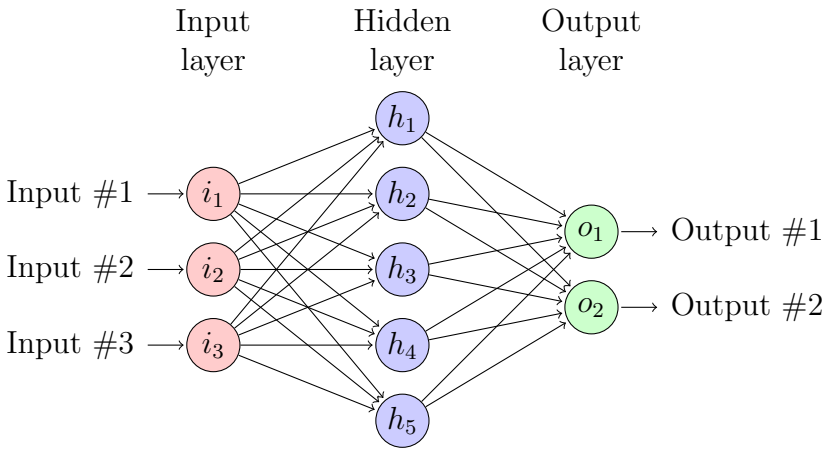


Figure 2.19: *Simple three-layer artificial neural network*

The three layers are linked by means of connections, represented as arrows in Figure 2.19.

Each connection has an associated weight that dictates how much of an influence it holds over the neuron to which it is connected.

The input layer is the interface between the user and the neural network. In a classification problem, the values assigned to the input nodes are generally features extracted from the data to be classified.

The hidden layers are where the inputs are translated into outputs by propagating the input values through the weighting matrix and activation functions of each node.

Finally, the output layer is where the results of the function or classification can be observed. For classification purposes, these outputs tend to be in "one-hot" format. This means that instead of a single neuron with multiple states, multiple neurons with binary states are used. For example, if a classification process resulted in an input being classified as the 5^{th} class of 8 possible classes, the output layer would have 8 neurons with the states as shown in Equation 2.44.

$$\texttt{one\_hot}(5, 8) = [0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0] \tag{2.44}$$

In order to produce meaningful results, the neural network must first be trained. Training is performed using a set of input data and the "true" or desired output data. These are normally referred to as the training data and the labels.

The weights in the network are first initialised, usually randomly between -1 and 1, and a piece of data is passed through the network. The output of the network at this point will be meaningless, but by comparing it with the label for the data, an error can be computed. This error is then propagated backwards through the hidden layers, and the error at each neuron is computed. These errors, known as "deltas", are used to determine how the weight of each connection should change in preparation for the next training step to minimise the error at the output (defined by a loss function). This process is repeated until the loss decreases to an acceptable value or starts to converge. This technique is known as the backward propagation of errors, or backpropagation [17].

2.4.4 Convolutional Neural Networks

The convolutional neural network is a technique that allows for spatial relationships in data to be preserved and for features to be automatically extracted for classification. This makes CNNs highly suited to image classification, where strong spatial relationships exist on both dimensions of an image.

2.4.4.1 Convolution

As the name suggests, the most important parts of a convolutional neural network are the convolution layers. These layers operate by sliding a small matrix of weights across a matrix (often an image) to produce a feature map. This matrix of weights is known as a kernel. At each step, an elementwise multiplication is performed between each element in the kernel and the element in the input matrix that it overlaps. The sum of the multiplications is then computed and forms a value in the feature map that corresponds to the position of the kernel. The kernel is then shifted to the next position, and the process is repeated.

Consider the following examples, where a matrix A is convolved by the kernel K to produce the result C.

$$A = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} \qquad K = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$C = A * K = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} * \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 3 \\ 2 & 3 & 3 \\ 3 & 2 & 2 \end{pmatrix}$$

Position $[1, 1]$ of the result, C, can be obtained by centring the kernel over position $[2, 2]$ of A and calculating the sum of an elementwise multiplication between the overlapped portion of A and K.

$$\begin{aligned} C[2,2] &= (1 \cdot 0) + (0 \cdot 1) + (1 \cdot 0) + (0 \cdot 1) + (0 \cdot 0) + (1 \cdot 1) + (1 \cdot 0) + (1 \cdot 1) + (1 \cdot 0) \\ &= 0 + 0 + 0 + 0 + 0 + 1 + 0 + 1 + 0 \\ &= 2 \end{aligned}$$

Multiple kernels can be used on a single layer, which in turn will produce multiple feature maps.

2.4.4.2 Rectified Linear Unit

The rectified linear unit, or ReLU, is an activation function designed to introduce non-linearity into the network. It is an elementwise operation that replaces all negative values with zero and leaves positive values unchanged.

The ReLU activation function is given in Equation 2.45 and plotted in Figure 2.20.

$$R(x) = max(0, x) \tag{2.45}$$

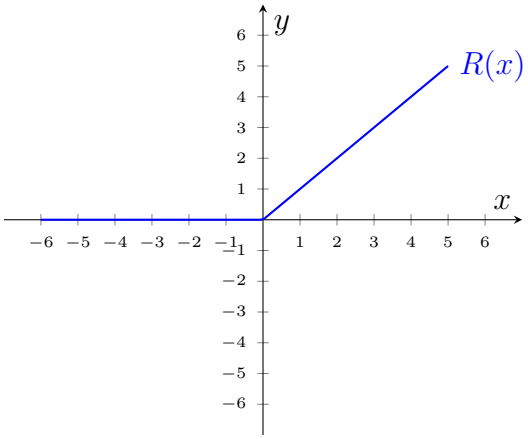


Figure 2.20: *Plot of the ReLU activation function*

The ReLU activation function results in many nodes being deactivated (having an output of zero). This is a property of the ReLU activation function known as sparse activation [18]. The ReLU function is easy to compute and so lends itself to the processing of extremely large amounts of data.

Krizhevsky, Sutskever, and Hinton [19] made use of the ReLU activation function and showed that a four-layer convolutional neural network using ReLU neurons reached a 25% training error 6 times faster than an identical network with hyperbolic tangent (tanh) neurons.

2.4.4.3 Pooling

It can often be desirable to prune feature maps by reducing their dimension while retaining the most important features. This can be accomplished through pooling, a process whereby a window size and stride are defined and applied to a feature map in order to achieve dimensionality reduction [20].

The size of the window determines the size of the feature map after pooling, much like the size of the kernel determines the size of the output of a convolution layer.

The stride dictates by how many elements the window is shifted between pooling steps. This is often, but not always, equal to one of the dimensions of the window size.

Finally, the pooling strategy determines how the elements contained within the pooling window are combined. The most commonly used form of pooling is known as max pooling, whereby the output of the pooling step is the greatest element of the elements in the pooling window.

An example of max pooling for a 2×2 window and a stride of 2 is shown in Equation 2.46.

$$\texttt{maxpool}\,(C) = \left(\begin{array}{cc|cc} 1 & 2 & 3 & 5 \\ 0 & 1 & 2 & 0 \\ \hline 6 & 0 & 4 & 2 \\ 1 & 2 & 5 & 1 \end{array} \right) = \begin{pmatrix} 2 & 5 \\ 6 & 5 \end{pmatrix} \tag{2.46}$$

2.4.4.4 The Adam Optimisation Algorithm

Optimisation algorithms, or optimisers, are used in machine learning to minimise an error function (the optimisation objective) in an efficient manner. Many well-researched optimisation algorithms exist in literature, but the most popular optimiser in recent deep learning research is the Adam optimiser.

The Adam optimiser was conceived by Kingma and Ba [21] as a means for combining the advantages of the AdaGrad [22] and RMSProp [23] optimisers. The result is an optimiser that is computationally efficient and has low memory requirements, while retaining the benefits of other optimisers such as compatibility with sparse gradients and invariance to gradient rescaling. These properties make the Adam optimiser highly suited to problems and architectures with large amounts of data and parameters.

The Adam optimiser requires manual selection of the learning rate, but a near-optimal value for this parameter can usually be found empirically through repeated tuning and by observing the cross-validation performance.

2.4.4.5 Classification

After convolution and max pooling, the final feature map is flattened and passed into a regular fully connected neural network. This can be seen in Figure 2.21, showing a typical convolutional neural network architecture with the fully connected layers in the final stage.

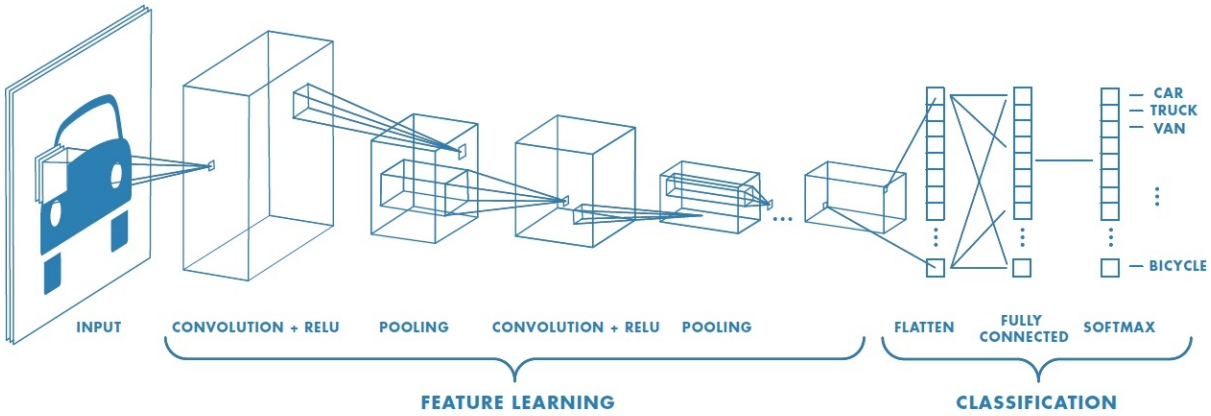


Figure 2.21: *Typical convolutional neural network architecture [24]*

Most classifiers make use of softmax function as the activation function for the output layer. The softmax activation function allows the values in the output layer to take on the form of probabilities, with the sum of the output layer values being equal to 1. The highest valued node in the output layer corresponds to the class that best represents the input image. The probability of the classes can be compared to determine how confident the neural network was of the result and can provide a warning sign for results that should potentially not be trusted.

2.5 Classification Approaches

2.5.1 Decision Tree Approaches

In classification problems where prominent, measurable features exist between classes, it can be beneficial to utilise a decision tree approach instead of resorting to machine learning techniques. Such an approach has the benefit of being transparent and, in many cases, being more straightforward to implement. This approach means that the designer has full control and understanding of every step of the system, which is often more responsible than following a "black-box" methodology. This can be especially favourable in systems where critical decisions need to be made based on the predictions of the classifier.

Wang [25] found that the first and second order phase derivatives (instantaneous frequency and its derivative) of the signal under test contained features that could be used for intra-pulse modulation classification. It was observed that impulses in the instantaneous frequency corresponded to phase shifts in the instantaneous phase and could be used to differentiate between various phase-shift keying techniques. A number of frequency modulation and frequency shift keying techniques were able to be classified using features found in the second order phase derivative of the signal.

Svarc, Brnak and Richterova [26] performed a set of real-world measurement trials and constructed a decision tree-based emitter classifier using a variety of features. The single-valued features extracted from each pulse before classification included carrier frequency, pulse width, bandwidth, autocorrelation width, and time-bandwidth product. The instantaneous amplitude, instantaneous frequency, and autocorrelation of each radar pulse was also computed and resampled to a consistent 1024 samples. The single-valued features were used in the initial comparison with the radar database to reduce the number of possible candidates, while the vector features were used to narrow down the candidates to a specific emitter where possible. It was found that the cross-correlation was the most useful method for matching a pulse to another pulse in the database, while also being the most computationally demanding. While no accuracy metrics were presented, it was found that emitter classification using only simple intra-pulse parameters is highly viable. This paper deals with a different form of classification to the one under investigation in this study, but nonetheless presents a valuable summary of pulse features and their usefulness for classification.

Singh and Rao [27] implemented a real-time LPI radar analysis system capable of accurately measuring various frequency and phase modulation techniques. This implementation made use of a custom digital receiver for sampling, and a combination of FPGA and DSP-based processing for data extraction. Basic pulse parameters such as frequency, amplitude, pulse width, and time of arrival were extracted in real-time using a FPGA-based FFT algorithm, while more complex parameters were extracted in post-processing using DSP processors. It was found that FMCW detection could be performed reliably using the FPGA-based FFT approach, and that the same approach could be used for the coarse detection of phase modulation. It was also shown that the Choi-Williams distribution was effective at removing the cross-terms present in the Wigner-Ville distribution of a signal with multiple frequency components, and that this distribution could be used to accurately measure the parameters of short (32 ns) Frank coded pulses. These extracted parameters can be used to perform a look-up into a radar database in a similar manner to the method proposed by Svarc et al. [26]. While this paper does not detail any classification techniques, it presents a valuable use-case for some of the time-frequency representations under investigation.

Zeng, Zeng, Lu, and Tang [28] were able to achieve a classification accuracy of 90% at low SNR levels (< -2 dB) using a decision tree approach based on the generalised time-frequency representation of Zhao, Atlas and Marks (ZAM-GTFR). A two-dimensional representation of the ZAM-GTFR, known as the "ridge" was computed by plotting the frequency index (column) at which the distribution was at a maximum for each time slice (row). The ZAM-GTFR was cited as being the time-frequency representation of choice due to its ability to strengthen spectral peaks, smooth cross terms

and its support for finite time signals. The features extracted from this representation were found to contain enough information to classify a range of signals. All tested signals were simulated with SNR in the range $[-5,\ 5]$ dB, and the classes under investigation included no modulation, LFM, BPSK (Barker Code), 2FSK, and 4FSK. While the signals classified in this paper lacked realistic imperfections and were limited in terms of the range and complexity of modulation schemes, Zeng et al. proved that decision tree based classification in the face of extremely low SNR is viable.

2.5.2 Naïve Bayes Approaches

Guo and Zhang [29] made use of a Naïve Bayes classifier to perform modulation classification based on features extracted through a combination of short time Fourier transforms (STFTs) and principal component analysis (PCA). STFTs were used as an intermediate representation of the signal, while PCA was used to reduce the dimensionality of the STFTs and to produce a finite-sized feature vector for classification. In total, 13 modulation schemes were selected for classification. These schemes included carrier modulation, Barker codes, linear frequency modulation, Costas codes, and non-linear frequency modulation. The classifier was trained using signals with a fixed SNR of 10 dB, and tested on signals with SNR in the range $[-20, 20]$. Extremely high classification accuracy was achieved on examples with SNR of 0 dB or greater, and the Naïve Bayes classifier in use was shown to outperform a support-vector machine (SVM) trained for the same purpose. The signals generated for classification in this study did not contain any imperfections apart from noise.

Barshan and Eravci [30] tested the ability of a number of classifiers to perform antenna scan type (AST) recognition. These classifiers tested included a Naïve Bayes classifier, a decision tree, an ANN, and a SVM. An antenna scan pattern simulator (ASPS) was developed to generate data for the classification problem in this study. This simulator was capable of simulating the received pulse amplitude sequences for circular, sector, raster, helical and conical scan patterns at varying PRI and SNR values. All tested classifiers, including the Naïve Bayes approach, performed comparably well, achieving classification accuracies of over 80% for SNRs greater than 25 dB. While this is not the classification problem under investigation in this study, it presents a useful comparison between the implementation, complexity, and performance of different classification techniques.

2.5.3 Nearest Neighbour Approaches

Aslam, Zhu and Nandi [31] made use of k-nearest neighbour algorithm to perform signal modulation classification. The modulation schemes classified were BPSK, QPSK, QAM16 and QAM64. Genetic programming was used to intelligently combine features from the fourth and sixth order cumulants into a single pattern for use in the k-NN classifier. This classifier was tested on simulated signals with SNR in the range of $[5, 20]$ dB, and was able to achieve an accuracy of 84% for the worst-case scenario in which 512 signal samples were used at a SNR of 5 dB. The results of this method were compared to other experiments in prior literature, and the developed classifier was found to have better performance and robustness than the majority of other techniques under consideration.

Mishra and Mulgrew [32] used a k-NN classifier in conjunction with PCA in order to classify images produced by a synthetic aperture radar (SAR). PCA was applied to each SAR image in order to reduce the number of observed features, and the extracted data was used as features for the k-NN classifier. This method was able to outperform the conditional Gaussian model based Bayesian classifier trained for the same task by DeVore and O'Sullivan [33]. This paper shows the applicability of k-NN techniques to the field of radar, but is not directly related to the classification problem in this study.

2.5.4 Neural Network Approaches

In 1999, Noone [34] was able to use a 3 layer perceptron with 4 hidden layer neurons to achieve 99.89% classification accuracy on various inter-pulse modulation techniques. Noone was able to overcome some of the shortcomings of traditional methods such as difference time of arrival (DTOA) histogramming using this approach. While this is not the problem to be addressed in this paper, it does demonstrate a potential use for machine learning techniques in the field of ELINT.

Conning and Potgieter [35] utilised various dimensionality reduction techniques and a Fuzzy ARTMAP classifier to perform specific emitter identification (SEI). Three radar systems of the same manufacture were tested in order to determine whether SEI could be performed on radars with identical design parameters. Pulse detection was performed using phase data instead of the common amplitude-based approach, which proved to yield improved accuracy with a higher tolerance to SNR. It is worth noting that only pulses with a SNR above 30 dB were considered for processing. A total of eighteen features were extracted from the instantaneous measurements (amplitude, frequency and phase) of each pulse. The authors observed that a number of these features were closely correlated and likely redundant, and made use of the gamma test [36] to find the subset of the features

that resulted in the least classification uncertainty. Using this dimensionality reduction technique and the Fuzzy ARTMAP classifier, Conning and Potgieter were able to achieve a classification accuracy of approximately 85%. It is interesting to note that by manually selecting their own combination of features through trial and error, the authors were able to improve upon these results by approximately 7%. This research is significant as it shows that SEI can be performed accurately on emitters with identical design parameters, and that the process of dimensionality reduction can improve classification accuracy.

Lunden and Koivunen [13] devised a novel system to classify intra-pulse modulation types using two multi-layer perceptrons to analyse features extracted from the Wigner and Choi-Williams time-frequency distributions. The first perceptron was used to separate polyphase modulated signals from binary phase modulated and LFM signals. If the signal was classified as polyphase, the second perceptron would be used to further refine the classification into the specific polyphase modulation technique. All waveforms were subjected to rigorous preprocessing in the form of carrier removal, sub-pulse rate estimation, sub-pulse rate sampling and manual feature extraction. A classification accuracy of 98% was achieved using this technique, but the amount of preprocessing may limit its potential in real-time applications.

O'Shea [37] applied a convolutional neural network to the problem of radio modulation type classification. Raw IQ data was treated as an input image for the CNN by separating the stream into 128 sample chunks. The in-phase and quadrature components of each chunk were separated and stacked vertically to form a 2×128 pixel input "image". This technique was able to achieve 87.4% classification accuracy for SNRs ranging from -20 dB to 20 dB over 11 modulation techniques. The fact that a number of the modulation techniques were at times indistinguishable from one another coupled with the extreme lower bound on SNR indicates that a much more favourable result could have been achieved under more lenient testing conditions.

Zhang [38] used a convolutional neural network to perform radar intra-pulse modulation classification based on the Choi-Williams time-frequency distribution. A classification accuracy of 93.7% was achieved on 8 different modulation techniques at SNRs as low as -2 dB. The modulation techniques considered were Barker codes, LFM, Costas codes, Frank codes and the T1-T4 polytime codes.

The Choi-Williams time-frequency distribution was used to create a 1024×1024 point image for each signal. These images were denoised and converted to binary images to remove any dependence on image colour and focus solely on the shape of the time-frequency distribution. This is discussed in Section 2.5.5 along with other preprocessing techniques.

2.5.5 Data Preprocessing

As mentioned previously, Zhang [38] used preprocessing to denoise the training images and convert them to binary images. This limited the CNN to training only on the shapes present in the images, while it might have otherwise attempted to extract colour-based features. In Zhang's study, the 1024×1024 point images were down-sampled to 32×32 point images as the last step before classification. This was achieved using nearest neighbour interpolation and allowed key features to be retained while reducing computational load.

The choice of image size when making use any image-based CNN is important as it plays a large part in dictating the number of trainable parameters in the network. In turn, the number of trainable parameters in a CNN is the primary influence over the complexity, training time and feedforward speed of the network.

Svarc et al. [26] made a point of resampling all computed waveforms to 1024 samples in order to facilitate effective comparisons between signals of different lengths. This is important in the context of this study, as most neural network-based architectures consist of a fixed size input while real-world emitters will make use of pulses with different lengths. Resampling is likely easiest way to allow a single classifier to operate on pulses that originated from different emitters or were sampled at different sample rates.

2.5.6 Classification Countermeasures

The classification of radar pulses for ELINT is a heavily researched field, and has led to the research and development of a number effective countermeasures. It is important that the risks of such countermeasures be understood when designing a classification or ELINT system in order to ensure maximum robustness. This study focuses on the implementation of a modulation classifier, which may form a component of an ELINT system. The modulation classifier in itself will not be responsible for ensuring robustness against many of the countermeasures discussed in this section, but they are still highly relevant to the overall field of pulse and emitter classification.

Van der Merwe, du Plessis, Maasdorp, and Cilliers [39] present an excellent summary on the current state of low probability of intercept (LPI), low probability of detection (LPD), and low probability of recognition (LPR) radars, and their potential threats to classifiers. It is suggested that radars may reduce the recognition capability of classifiers through the variation of simple PDW parameters, or by making use of unusual (in the context of radar) waveforms. In general, Van der Merwe et al. propose that there is an inherent trade-off between performance and covertness in any radar system.

Utilising waveforms and frequency bands designed for communication or public broadcast purposes may severely hinder or cause emissions to go unnoticed by classifiers, but will likely result in lower performance than waveforms and modulation schemes designed with radar in mind. Examples of services that may be exploited by a LPR radar include broadcast services such as digital video broadcasting - terrestrial (DVB-T) and frequency modulation (FM) radio, and communication systems such as Global System for Mobile Communications (GSM) and Long-Term Evolution (LTE). Van der Merwe et al. suggest that classification systems may be able to mitigate the effects of LPR radar techniques by adding extra analysis steps such as demodulation and content validation, extending spectrum monitoring to communication and broadcast bands, and analysing pulse parameters such as bandwidth, staggering patterns, and modulation schemes.

The topic of radar-embedded communications has also seen a large amount of research in recent years [40], [41], [42], [43]. This topic aims to combine the functionality of radar and communication systems into a single signal through the development of new intra-pulse modulation techniques, or the adaptation of existing communication techniques. Radar-embedded communication systems may present a challenge to modulation classifiers due to the variation of communication content between pulses. If the type or volume of communication content strongly influences the overall characteristics of the radar waveform, it may be enough to cause uncertainty in classification or ELINT systems that were trained on other pulses from the same emitter.

The generation and use of adversarial examples has been thoroughly investigated as a direct attack on deep neural networks. An adversarial example is any input to a machine learning model that has been designed to cause the the model to make a mistake [44].

Nguyen, Yosinski, and Clune [45] found that evolutionary algorithms or gradient descent could be used to produce examples that have no visual similarities to a target class, but that are classified as the target class by the model under test with extremely high confidence. This idea was approached differently by Su, Vargas, and Sakurai [46], who developed a system for fooling image classification models by modifying a single pixel of an existing image based on differential evolution. This method has the advantage of being less perceptible to humans than other adversarial approaches, where images are sometimes significantly modified with seemingly random artefacts.

Madry, Makelov, Schmidt, Tsipras, and Vladu [47] have carried out research on making deep learning models more resistant to adversarial attacks. It was found that increasing the capacity (number of nodes and layers) of a model that is vulnerable to adversarial attacks results in significantly reduced effectiveness of these attacks. It was also found that incorporating certain adversarial examples into the training set could reduce the overall effectiveness of adversarial approaches on the resultant network.

Chapter 3

Methodology

3.1 Research Type and Motivation

This study focuses on a quantitative research approach in which various signal classification techniques are researched and a selection of implementations are evaluated and compared. A strong emphasis is placed on the implementation and testing of convolutional neural network based classifiers.

The study is motivated by the need to stay abreast of complex and emerging radar technologies and by the desire to explore unusual applications of machine learning techniques so as to better understand their capabilities. It also addresses the practical issue in the field of electronic intelligence whereby the increased waveform diversity observed in LPI and LPR radars is making emitter classification more challenging. The results of this study hold relevance in many fields in the academic, industrial and defence sectors.

3.2 Research Questions

The main research questions for this study are:

1. Can radar intra-pulse modulation schemes be reliably classified without machine learning techniques?
2. Could the application of machine learning improve on non-machine learning based classification techniques?
3. Which data representations are best suited to classification using a convolutional neural network?
4. What level of classification accuracy can be achieved using a fully trained convolutional neural network for each of the approaches explored in the previous question?
5. What is the effect of low (negative) signal-to-noise ratio on the accuracy of the classification techniques?

6. What is the effect of carrier frequency offset on the accuracy of the classification techniques?

7. What is the effect of pulse detection jitter on the accuracy of the classification techniques?

8. Can parallels be drawn between the features extracted by the convolutional neural networks and the features extracted using common visualisation methods?

9. Can classifiers trained on simulated data classify data that has passed through a real-world radio link?

10. Can classification be carried out in real time if a realistic pulse density is assumed?

11. Are the performance improvements obtained from convolutional neural networks worth the loss in transparency from non-machine learning approaches?

Research question 1 is already answered through the review of prior literature in this study. However, none of the non-machine learning based techniques found in prior literature considered as broad a spectrum of modulation schemes as this study. Hence, the results obtained in this study serve to supplement the results obtained by researchers in prior literature.

The primary research questions are "Could the application of machine learning improve on non-machine learning based classification techniques?" and "Which data representations are best suited to classification using a convolutional neural network?" These questions will provide the best guidance as to whether machine learning techniques are a valuable research area in the realm of radar and signal processing, and lay out some basic guidelines for the classification of unusual data representations.

3.3 Sources of Literature and Engagement

The primary sources of literature in this study were academic journals and conference papers. Reputable websites and online sources were also used at the discretion of the author.

The literature considered for this research was not limited to the field of radar waveform classification. Broader concepts from the general fields of radar and machine learning were also explored to provide guidelines for best development and research practices. Care was also taken to engage with a range of material that reflected both historical and cutting-edge approaches to radar waveform classification.

3.4 Collection and Generation of Research Data

The collection of research data was entirely simulation-driven, with experiments designed to evaluate the effectiveness and validity of this simulation-driven approach. This approach was taken due to the engineering effort that would be required to consolidate a library of radar waveforms large enough to train and test classifiers.

As previously mentioned, there is no public knowledge of any radar systems in South Africa that make use of the complex modulation schemes explored in this study. This made the analysis of real-world data largely impossible and reinforced the decision to make use of a simulation-driven approach. The hope is that the techniques arising from this study can be expanded upon and applied to real-world data as a part of future research.

Simulated data was generated by a radar simulator written at Peralex Electronics that is capable of generating all of the necessary radar waveforms with varying levels of signal-to-noise ratio, carrier frequency offset and pulse detection jitter (the temporal accuracy of the pulse detector).

3.5 Extraction of Results

The primary measure of the effectiveness of a classification system is the classification accuracy. This is given as the percentage of examples that are correctly paired with their modulation type.

Where applicable and necessary, the classification accuracy for each of the classification systems is independently evaluated to quantify a number of effects:

- The classification accuracy of each class
- The effect of signal-to-noise ratio on classification accuracy
- The effect of carrier frequency offset on classification accuracy
- The effect of pulse detection jitter on classification accuracy

Wherever possible, the classification speed of the classifier was also measured in order to provide an indication as to how feasible the classifier may be for real-time operation. Note that this measure was not prioritised in the implementation of any of the classifiers in this study, but is rather a by-product of the inherent complexity of the different classifiers.

The amount of preprocessing was also investigated for each classifier and data representation under test. This factor is not as easily measured as the items listed above, but still plays an important role as it dictates the rate at which data can be fed into the classification system.

3.6 Methods, Tools and Timeline

All neural network and machine learning development was carried out in Python 3.6 using the Keras [48] library with the TensorFlow [49] backend. This selection of language and library was chosen as it is fully open-source. TensorFlow and Python also allow for model acceleration using a GPU, which enhances the rate at which classifiers can be trained and tested. It should be reiterated that none of the classifiers in this study were optimised for speed, and that the training time of models is not an indicator of performance or viability.

A large amount of the data pre and postprocessing and manual classifier development was carried out using MATLAB [50] due to its ease of use as a prototyping language and its excellent graphing and plotting capabilities. Most plots and graphs in this study were generated using MATLAB. All time-frequency representations were generated in MATLAB using the Time-Frequency toolbox developed by members of the CNRS and Rice University [51].

All development, testing and analysis was carried out on a desktop computer with the specifications listed below. Care was taken to ensure that the development environment remained consistent throughout the study to ensure that results dependant on timing performance were valid.

- Operating System: Microsoft Windows 10
- Processor: Intel Core i5-2500K (3.3 GHz)
- Graphics Processor: NVIDIA GeForce GTX 1080 Ti
- Memory: 18.0 GB DDR3-1600 MHz

While the structure of this study was outlined in the Plan of Development section of the report, the steps taken in carrying out the study are detailed below.

Firstly, the Peralex Radar Simulator was tested to ensure that it was capable of generating the required pulses and modulation schema. This testing included validating that all modulation schemes are generated correctly, verifying that all noise is decorrelated, and

ensuring that any imperfections added to pulses were valid. The majority of this testing is trivial and is not detailed in this study.

A set of real-world imperfections and parameter ranges was decided upon, and a dataset for the study was generated. The specifics of this dataset are discussed in Section 4 of this report.

After the dataset had been generated, the feasibility of "manual" classification was investigated. In this context, the term "manual" refers to the fact that the classifier did not make use of machine learning techniques. Due to the fact that prior literature already shows that classification using non-machine learning based techniques is possible, and the fact that this primary objective of this study is to evaluate convolutional neural network based approaches, only a single classifier of this nature was investigated.

A classifier was developed in MATLAB that made use of pulse parameters extracted from time-frequency representations of the pulse. Time-frequency representations were chosen for analysis as they provide a convenient means of consolidating temporal and spectral information into a single piece of data for analysis, removing the need for the classifier to maintain a state history and thus reducing complexity. The development of this classifier was accompanied by a brief feasibility study that evaluated the separability of the features extracted from the dataset using principal component analysis.

After the feasibility of a non-machine learning classifier was evaluated, the effectiveness of various machine learning models and data representations was investigated. The first machine learning classifier created for this study was a convolutional neural network that utilised time-frequency representations as inputs.

A large amount of the time taken to build this initial machine learning classifier was spent establishing a framework that could be used across other networks generated as a part of this study to ensure that consistent and fair comparisons between results could be made.

In an attempt to analyse the effects of data representation on model complexity and performance, two additional CNN classifiers were developed. The first of these classifiers utilised a set of instantaneous measurements (instantaneous magnitude, frequency and phase) as an input and the second made use of raw complex time data as an input. Both of the above classifiers aimed to analyse the effects of varying degrees of data preprocessing and abstraction on classification accuracy and feedforward time (and hence data throughput).

Building on the raw complex time data classifier, a system comprising of two separate convolutional neural networks was designed that would perform independent classification on the two components of the signal and combine the results in a post-classification step.

The CNN-based classifiers were analysed further using a separate low-SNR dataset to determine which classifier has the most resilience to noise. An antenna-to-antenna experiment was also designed to test the effectiveness of using the simulation-trained CNN-based classifiers on data that had passed through a real-world radio link.

Some of the final steps taken in this study were to perform analysis on the results of the raw complex time data classifier in order to relate some of the features extracted by the classifier to easily understandable concepts. This was carried out by analysing a number of the convolutional filters in the trained model and attempting to reproduce pulse data based on what the neural network found "optimal" for a certain class. This also corresponds to the research on adversarial techniques as discussed in Section 2.5.6.

3.7 Threats to Validity and Integrity

The primary threats to the validity and integrity of the results of this study stem from the fact that the data to be analysed was completely simulated. If the simulated data is erroneous, then the results may not provide an accurate reflection on the applicability of machine learning techniques to real-world radar waveform classification.

Care was taken to ensure that all simulations were as representative of a real-world environment as possible, and incorporated a suitably diverse set of imperfections. The simulator used was capable of simulating major imperfections like SNR variation, CFO and pulse detection jitter, but was not equipped to simulate complex amplifier-specific effects such as pulse droop, overshoot and ringing. It is the belief of the author that the capabilities of the simulator are of sufficient complexity for the purposes of this study.

The experiments designed to test the effectiveness of the simulation-trained classifiers on more realistic data also serve to validate the results obtained in other experiments in this study. Any discrepancies noted are discussed in Section 6 of this study.

Chapter 4

Implementation

4.1 Dataset Generation

In order to ensure that experiments are valid, care was taken to ensure that the simulated signals making up the dataset were as realistic as possible, which meant incorporating realistic imperfections.

Each modulation scheme was iterated over 500 times, with each iteration carrying a random permutation of signal-to-noise ratio, carrier frequency offset and pulse detection jitter. This resulted in a dataset containing with 14 000 examples uniformly distributed modulation schemes and parameter variations.

The dataset was split into training, cross-validation and testing sets with a 60/20/20% split for all CNN classifiers. This resulted in 8400 signals for the training set, and 2800 signals each for the cross-validation and testing sets. The data indices used for training, cross-validation and testing were saved to ensure that any subsequent analysis would be consistent with the original data split.

The manual (no machine-learning) ridge-based classifier differs from the machine learning classifiers in that it does not make use of a cross-validation set. Instead, the dataset was split into training and test sets with an 80/20% ratio, resulting in 11 200 training signals and 2800 testing signals. This classifier was also tested on a secondary dataset with no pulse detection jitter in order to quantify the effects of this imperfection on the ridge-based classification approach.

4.1.0.1 Pulse Parameters

All pulses were simulated with a fixed pulse width of 40 μs and a sample rate of 25.6 MHz.

The pulse width was fixed at 40 μs as neural networks (both convolutional and regular fully-connected types) typically require a fixed-size input vector. The actual value of 40 μs is rather arbitrary, and was chosen as it equates to 1024 samples at a sample rate of 25.6 MHz, which is a convenient input vector size.

In reality, not all radar emitters will make use of pulses of the same width. Some emitters may even vary their pulse width between pulses or bursts. This implies that pulses would need to be preprocessed (interpolated or decimated) to the correct number of samples before they can be classified using the models developed in this study. This level of preprocessing was not accounted for in this study, and would form part of the future work involved with integrating one or more of the models into a real-world, real-time system.

4.1.1 Signal-to-noise Ratio

Simulating signals at low signal-to-noise ratios is important as many of the modulation schemes under investigation are aimed at low probability of intercept radars, which tend to transmit using very little power.

All signals were simulated with uniformly distributed random signal-to-noise ratios in the range of $[-5, 20]$ dB. This range of signal-to-noise ratios is illustrated in Figures 4.1 and 4.2, which show the instantaneous magnitudes of a Barker 13 pulse with 20 dB and -5 dB SNR respectively.

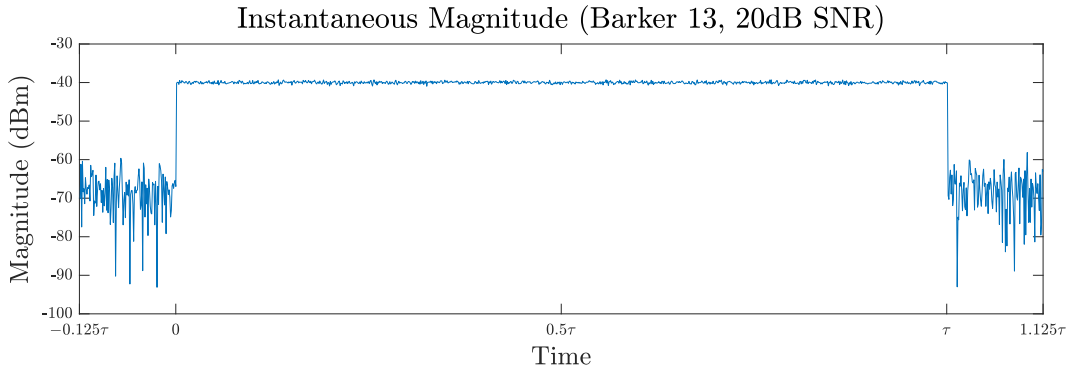


Figure 4.1: *Barker 13 instantaneous magnitude with* 20 *dB SNR*

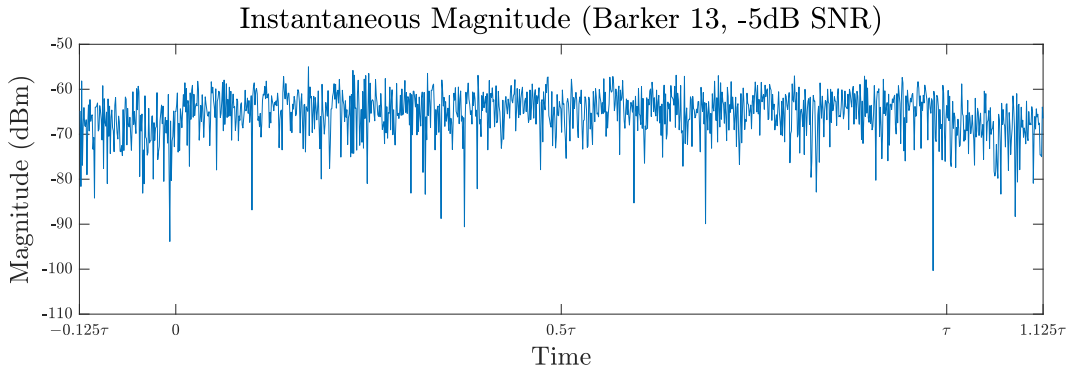


Figure 4.2: *Barker 13 instantaneous magnitude with* -5 *dB SNR*

4.1.2 Carrier Frequency Offset

Radar pulses are generally down-converted to baseband to aid analysis, either through an analogue heterodyning stage or digital postprocessing. However, there is no guarantee that the centre frequency of the downconversion channel will match the centre frequency of the radar pulse. These frequency mismatches are known as carrier frequency offsets. All signals in this study were simulated with uniformly distributed CFO in the range of $\left[-\frac{f_s}{4}, \frac{f_s}{4}\right]$ Hz. This range of CFO is shown in Figures 4.3 and 4.4, which show the instantaneous phases of a Barker 13 pulse with 0 Hz and $\frac{f_s}{4}$ Hz (6.4 MHz in this context) CFO respectively.

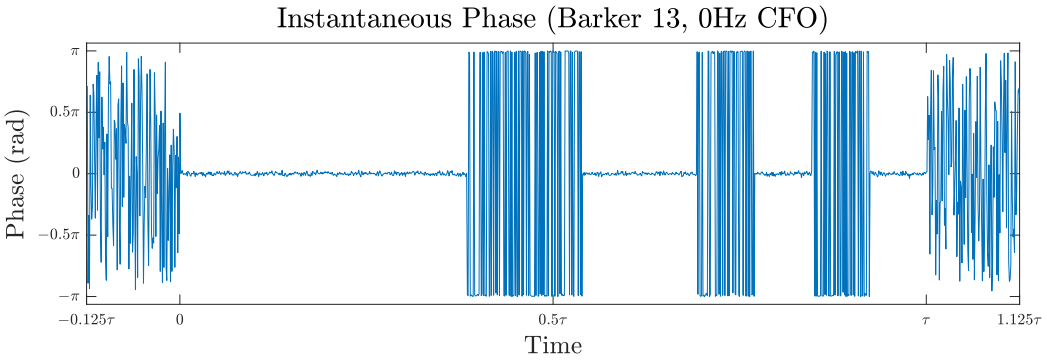


Figure 4.3: *Barker 13 instantaneous phase with no carrier frequency offset*

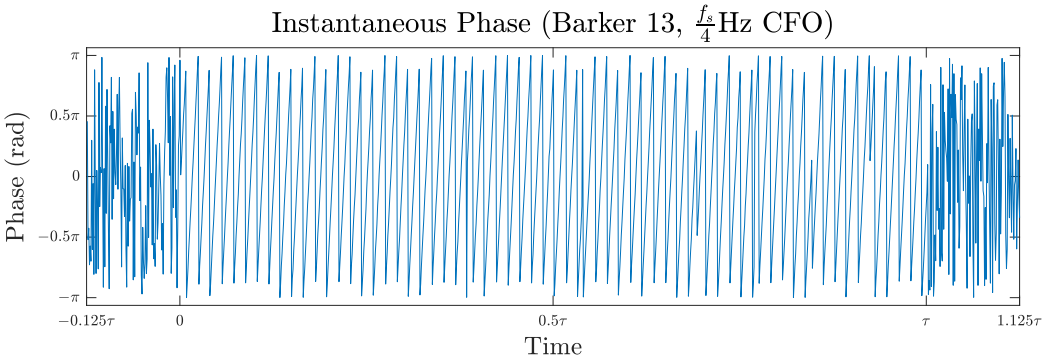


Figure 4.4: *Barker 13 instantaneous phase with $\frac{f_s}{4}$ Hz carrier frequency offset*

4.1.3 Pulse Detection Jitter

Radar pulses need to be detected and extracted before analysis can occur. This extraction can occur in the time or frequency domains, but typically takes the form of a noise-riding thresholding operation.

It can often occur that the pulse is imperfectly captured by the pulse detector, leading to a portion of the pulse being missed or a portion of the noise adjacent to the pulse being

captured. This will occur more often when the SNR of the signal is low and the noise has the potential to exceed the threshold in the level detector.

In order to simulate this, a fixed pulse width of 40 μs was used, and each pulse was padded with 5 μs of noise on either side. For each permutation, a 40 μs long window was randomly positioned within this 50 μs time-series to simulate pulse detection jitter.

This is illustrated in Figure 4.5. The top row of boxes represents the time series generated during simulation, with the red box showing the location of the signal content. Below this, the 40 μs long extraction window is positioned randomly within the time series, which results in a segment of the pulse being extracted along with some noise. This is shown in the bottom row.

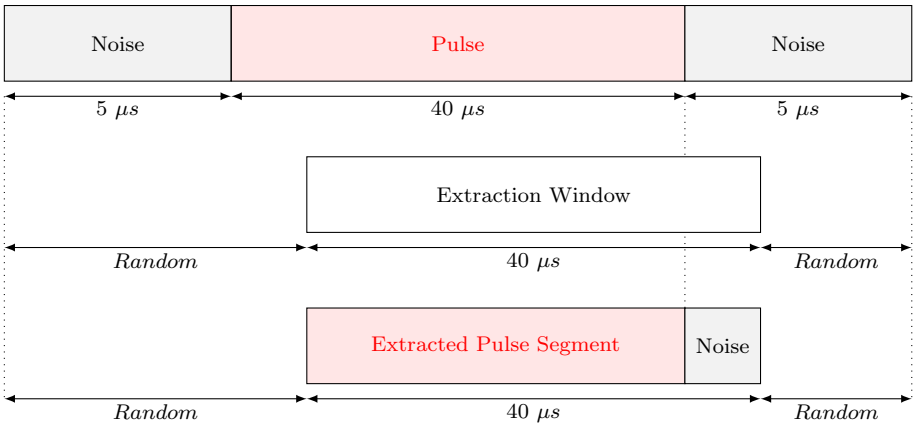


Figure 4.5: *Visualisation of pulse detection jitter*

4.2 MSE-Based Ridge Classifier

Following the work carried out by Zeng et al. [28], a classifier based on the ZAM-GTFR distribution and manual feature extraction was implemented. This classifier was traditional in the sense that it did not make use of machine learning techniques.

4.2.1 Ridge Computation

This classifier makes use of what is defined by Zeng et al. [28] as the "ridge" of the time-frequency distribution. The ridge of the distribution is given as the frequency (or the frequency bin index) at which the distribution is at a maximum for each time slice. For a given `n` by `m` bin time-frequency distribution, `tfr[n, m]`, the ridge is typically extracted using a function call similar to the one shown in Equation 4.1.

```
[max_TFR_values, ridge] = max(tfr, 2)
```
(4.1)

In this line, the `max` function is assumed to take two arguments and return two results. The first argument is the object under test, being the time-frequency distribution, and the second argument is the dimension along which to operate. If each time bin is given by a row in the `tfr` matrix and each frequency bin is given by a column, operating along dimension 2 would result in the computation of the maximum TFR value and corresponding frequency bin index in each time slice. The first result, `max_TFR_values`, provides the maximum value of the distribution for each time slice. The second value provides the frequency bin index at which this maximum value occurred. This array of indices is the ridge of the time-frequency distribution, and will be of dimension $\texttt{n} \times \texttt{1}$.

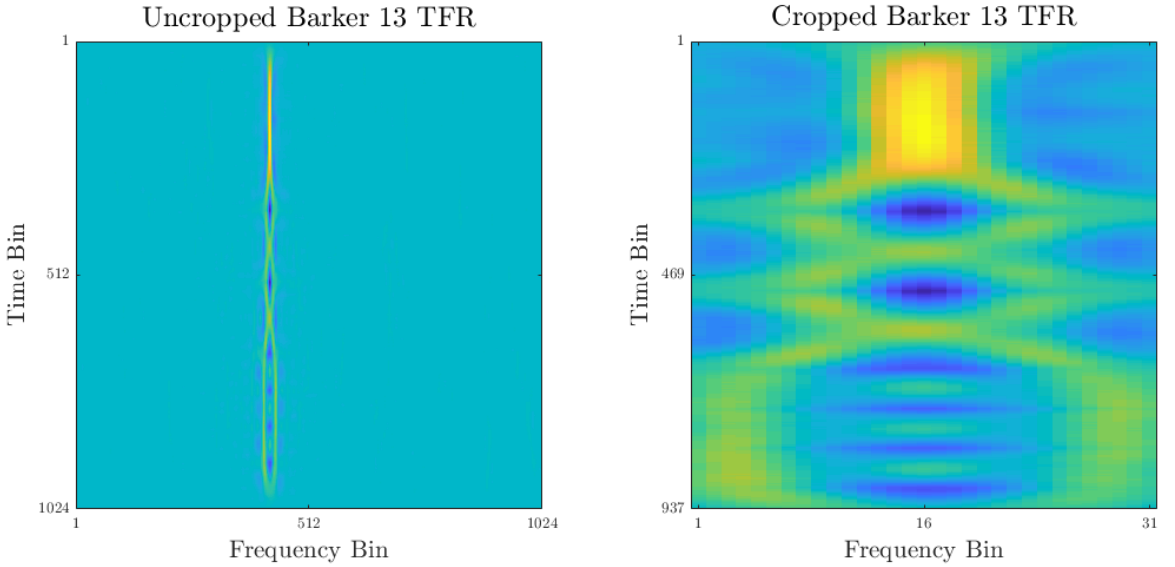
As previously shown by Zeng et al. [28], the ridge provides a two-dimensional representation of the movement in frequency of the largest component in the distribution over time, and contains a number of features useful for classification.

4.2.2 Carrier Frequency Offset Compensation

Before the ridge can be computed, the time-frequency distribution needs to be preprocessed to compensate for the effects of carrier frequency offset. This preprocessing takes the form of a "cropping" operation whereby the size of the time-frequency distribution is reduced until it contained only information relevant to the signal and minimal excess noise. This is accomplished using a threshold, T, to perform signal detection. The threshold is based on the standard deviation, σ, and the mean, μ, of the time-frequency distribution $C_x(t, f)$, and is defined in Equation 4.2. N is the number of standard deviations from the mean that the threshold is set to, and was fixed at 5 for the implementation of this classifier.

$$T = \mu\left(C_x(t, f)\right) + N \times \sigma\left(C_x(t, f)\right) \tag{4.2}$$

The results of the signal cropping operation are shown in Figure 4.6, where a Barker 13 pulse sampled with a carrier frequency offset was detected and isolated in its time-frequency distribution. It can be observed from Figure 4.6a that carrier frequency offset manifests itself in the form of a horizontal shift of the signal content in the TFR. Figure 4.6b shows that the carrier frequency offset was effectively removed from this example by means of the thresholding step defined in Equation 4.2 and the subsequent cropping operation.



(a) *Pre-crop time-frequency distribution* (b) *Post-crop time-frequency distribution*

Figure 4.6: *Time-frequency distribution before and after cropping*

4.2.3 Ridge Preprocessing

Before features can be extracted from the ridge, it needs to be preprocessed into a form suitable for classification. Figure 4.7 shows the ridge of the cropped ZAM-GTFR time-frequency distribution of a Barker 13 pulse before any additional preprocessing is applied.

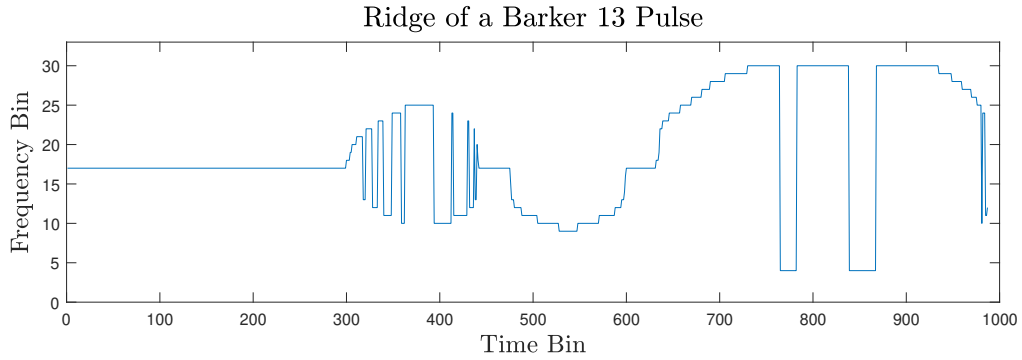


Figure 4.7: *Unprocessed Barker 13 ridge*

Figure 4.7 presents a number of issues. Firstly, the ridge is not zero-centric and thus lacks a convenient reference point for the extraction of various features. Secondly, it appears that the frequency bin values tend to oscillate sharply during steps. Finally, the time and frequency bin values are unnormalised and have no meaning without additional information. These values should be represented in terms of the size of the distribution.

In order to solve the lack of a convenient reference point, the ridge is centred using the modal value or, in cases where there is no clear centre-line, the mean value. The result of this operation is shown in Figure 4.8. Note that in Figure 4.8 the y-axis label has changed from "Frequency Bin" to "Ridge Amplitude". The ridge amplitude in this context is simply defined as the frequency bin number after the ridge has been zero-centred.

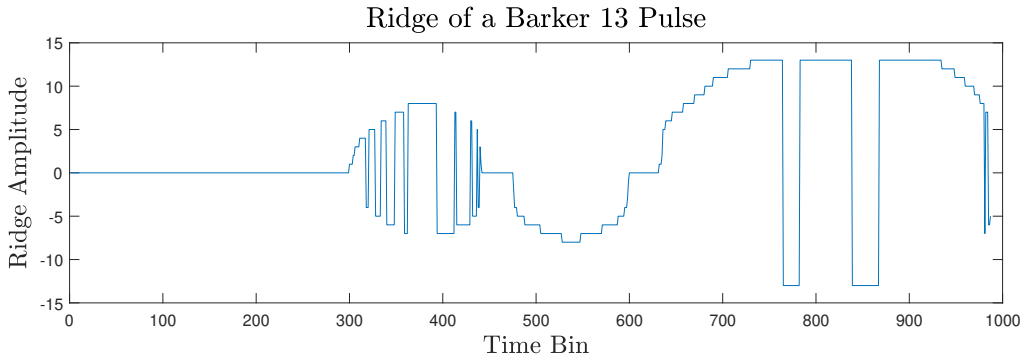


Figure 4.8: *Centred Barker 13 ridge*

The centring step exposes the fact that the oscillations in ridge amplitude occur due to symmetry in the time-frequency distribution. The symmetry in the distribution can be eliminated by only considering the absolute value, or magnitude, of the ridge. This is shown in Figure 4.9.

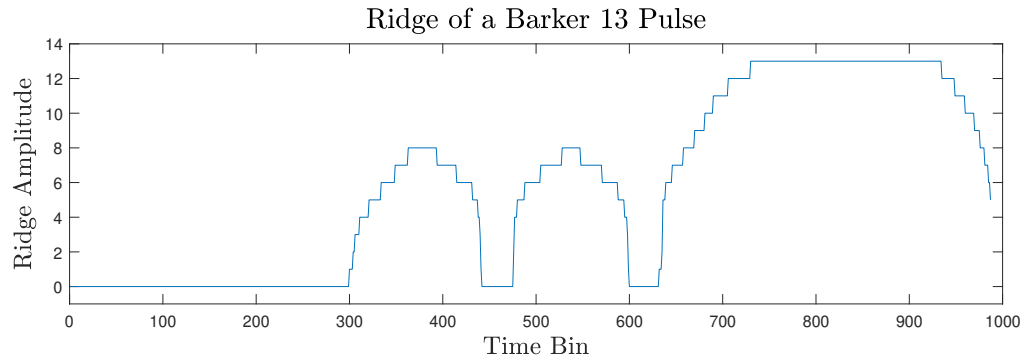


Figure 4.9: *Magnitude of the Barker 13 ridge*

Finally, normalisation is applied to the time and ridge amplitude values in order to facilitate meaningful comparisons between ridges of different classes. The time axis is normalised by the number of time slices in the time-frequency distribution, while the ridge amplitude is normalised by half the number of frequency bins (due to centring and absolute value operations) in the time-frequency distribution. The final ridge after all preprocessing is shown in Figure 4.10.

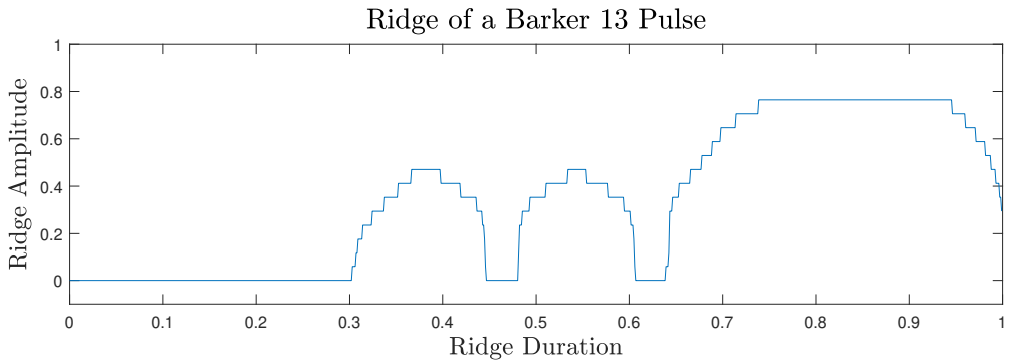


Figure 4.10: *Normalised Barker 13 ridge*

4.2.4 Ridge Feature Extraction

4.2.4.1 Number of Steps

The number of steps in the ridge is an extremely useful parameter for performing coarse separation of classes.

This feature is extracted from each ridge before normalisation using a state machine and a threshold for step detection. The threshold is defined in Equation 4.3 and is based on the mean value of the ridge $R(t)$, and a single bin of hysteresis on either side. This results in two thresholds, an upper threshold that must be exceeded to indicate the start of a step, and a lower threshold that the ridge must fall below to indicate the end of a step.

$$T = \mu(R(t)) \pm 1 \tag{4.3}$$

The upper and lower thresholds are shown in Figure 4.11 by the red and blue lines respectively. The state of the state machine at each step is denoted by the colour of the ridge plot, with red indicating the "on" state and blue indicating the "off" state. The results of this operation indicate that the ridge in this example has three distinct steps.

4.2.4.2 Step Parameters

While the number of steps provides an effective means for performing coarse classification, a method is needed to perform classification between classes with the same number of steps. This is accomplished by extracting parameters for each of the steps in the ridge. More specifically - the start, end, and peak locations, and the peak amplitude.

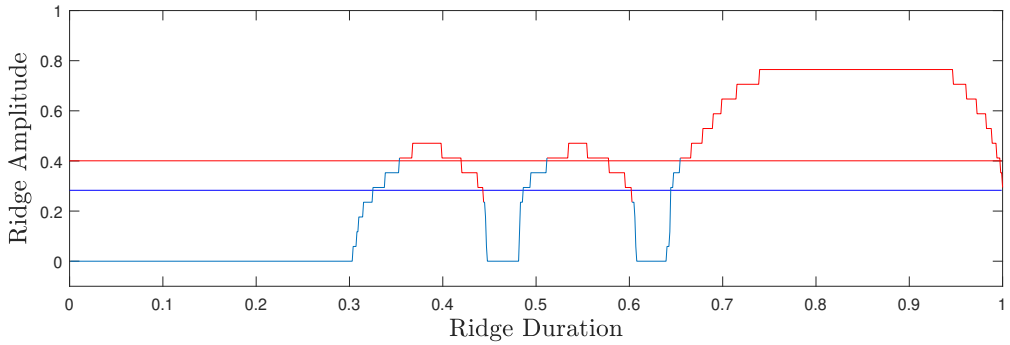


Figure 4.11: *Step thresholds of a Barker 13 ridge*

Figure 4.12 shows how these properties are measured, with the green, red and blue markers showing the start, peak, and end locations for each step. Note that the step start and end locations are recorded as the index after crossing a threshold, and the peak location is recorded as the first occurrence of the largest amplitude of the step.

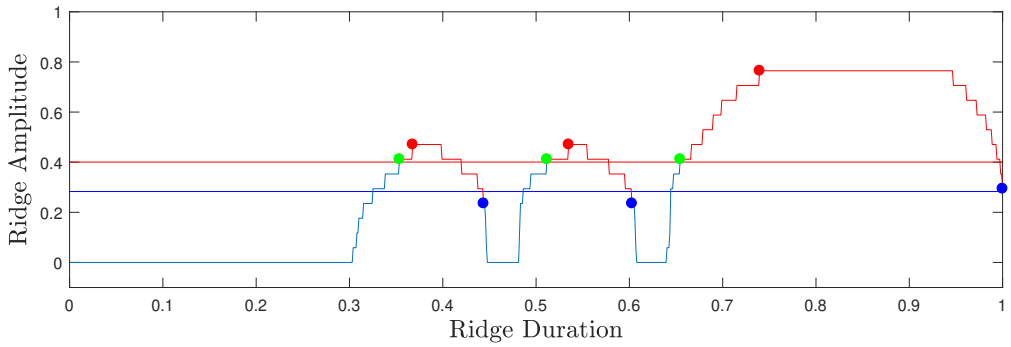


Figure 4.12: *Step start, peak and end locations of a Barker 13 ridge*

4.2.5 Classification Process

Even though the classifier does not make use of machine learning techniques, it is still run through a "training"' process to collect a set of baseline measurements for each class. This training process is comprised of the steps listed below and is repeated for each training example.

1. Load the time-frequency distribution of the training example
2. Compute threshold to extract the signal portion of the time-frequency distribution
3. Crop the time-frequency distribution
4. Extract the ridge of the time-frequency distribution

5. Preprocess the ridge
6. Extract the number of steps and step parameters using a state machine
7. Record the step parameters to a comma separated file based on the class

Due to the deterministic nature of the signals under test, the number of steps is not expected to change between examples of the same class. It is for this reason that each class is treated as having a fixed number of steps. The only time that the number of steps may change in a class is at low SNR if the noise dominates the peak of the ridge, or if the thresholds for step detection (defined in Equation 4.3) fall within the peaks of the noise. These two cases will lead to imperfect CFO compensation and a potentially random frequency offset, or an increased number of "false" steps due to peaks in the noise.

The step parameters for all of the training examples of each class are postprocessed and averaged to provide a set of baselines to which new examples can be compared for classification. It is important to note that the datasets for testing and training were kept separate throughout the course of this study.

The full table of baseline measurements for this experiment was recorded in comma-separated format and can be found in Appendix A. A subset of this dataset containing the number of steps for each class is shown in Table 4.1.

Table 4.1: *Number of steps for different classes*

Class	N	Class	N
FM	2	P1 6	2
Barker 2	1	P1 8	2
Barker 3	1	P2 2	2
Barker 4	2	P2 4	4
Barker 5	2	P2 6	5
Barker 7	3	P2 8	3
Barker 11	3	P3 4	2
Barker 13	3	P3 16	1
Frank 2	1	P3 36	1
Frank 4	1	P3 64	1
Frank 6	1	P4 4	1
Frank 8	1	P4 16	2
P1 2	3	P4 36	2
P1 4	2	P4 64	2

After the baseline data has been generated through the training process, the classifier is switched into classification mode, and the test data can be loaded. The same process as described above is followed to compute the ridge and step parameters for each example of the test dataset. After the step parameters have been computed, the baseline step parameters for all classes with the same number of steps as the example under test are loaded, and all other classes are ruled out of the classification result. This is the coarse classification by means of the number of steps that was discussed earlier.

The feature vector of the example under test is then compared to the baseline feature vector for each possible class, and the mean squared error (MSE) is computed for each comparison. The MSE metric is used as it provides a measure of similarity between the feature vector of the example and the baseline vector of the class under test. The class that results in the lowest MSE is selected as the prediction for the example under test.

The results for each example in this experiment were recorded, and the full set of results was analysed at the end of the classification process to provide performance metrics across the different classes and the range of tested signal-to-noise ratios, carrier frequency offsets, and amounts of pulse detection jitter.

4.2.6 Feature Analysis

Principal component analysis (PCA) was performed in order to reduce the features extracted from the experimental dataset to three dimensions for visualisation. This allows the manner in which different classes clustered together to be observed, providing intuition as to how the classifier may perform on certain classes. The general expectation is that classes that are clearly separable from other classes in the PCA visualisations will have better classification performance than classes that appear closely clustered together.

Due to the fact that the number of features is dependant on the number of steps in the ridge, the experimental dataset was split according to the number of steps for visualisation. Figures 4.13, 4.14, 4.15, 4.16 and 4.17 show the PCA reduced datasets for all classes with 1, 2, 3, 4, and 5 steps respectively.

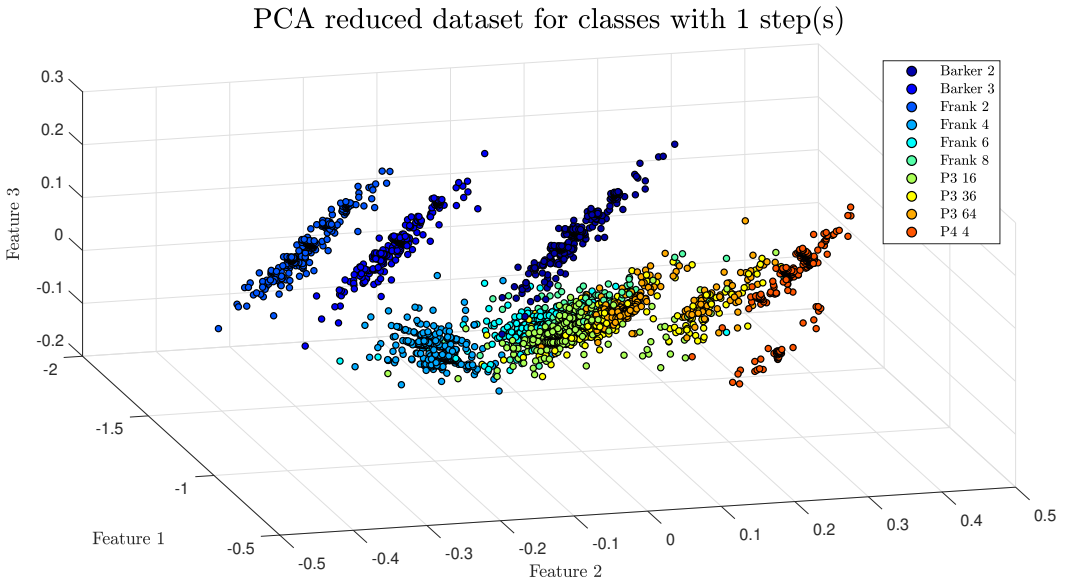


Figure 4.13: *PCA reduced dataset for classes with 1 step*

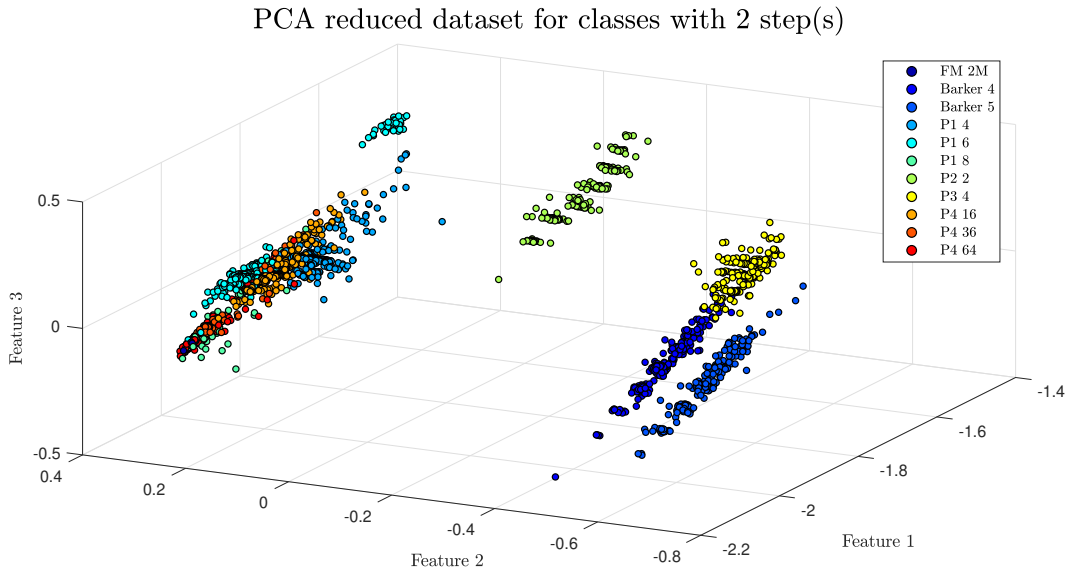


Figure 4.14: *PCA reduced dataset for classes with 2 steps*

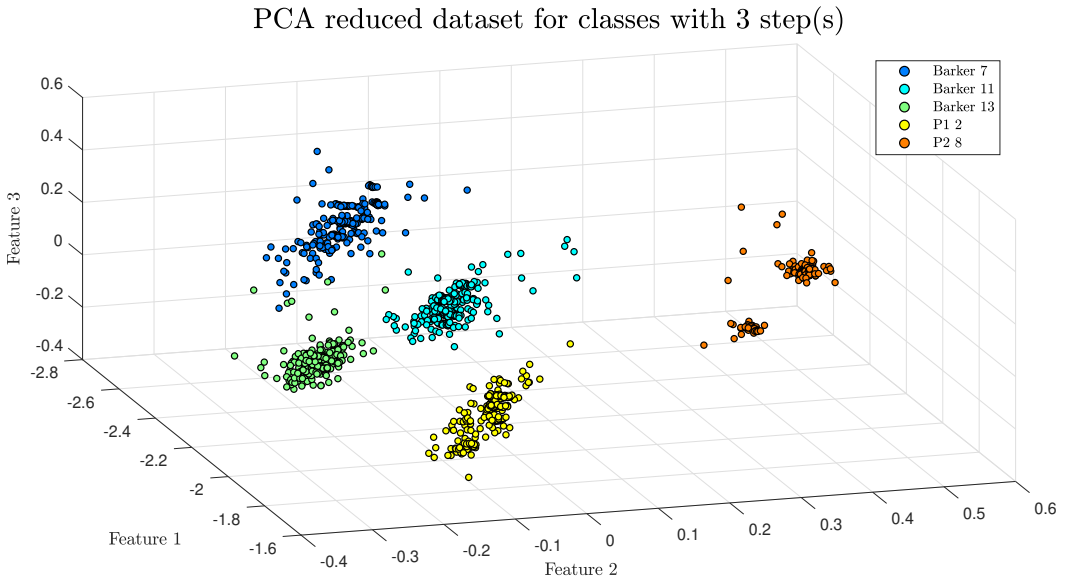


Figure 4.15: *PCA reduced dataset for classes with 3 steps*

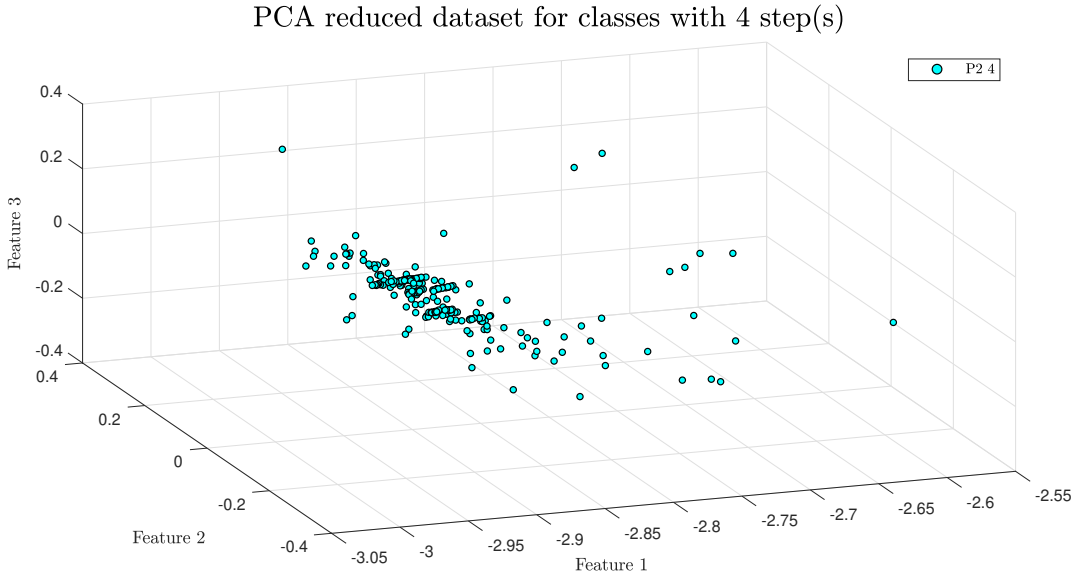


Figure 4.16: *PCA reduced dataset for classes with 4 steps*

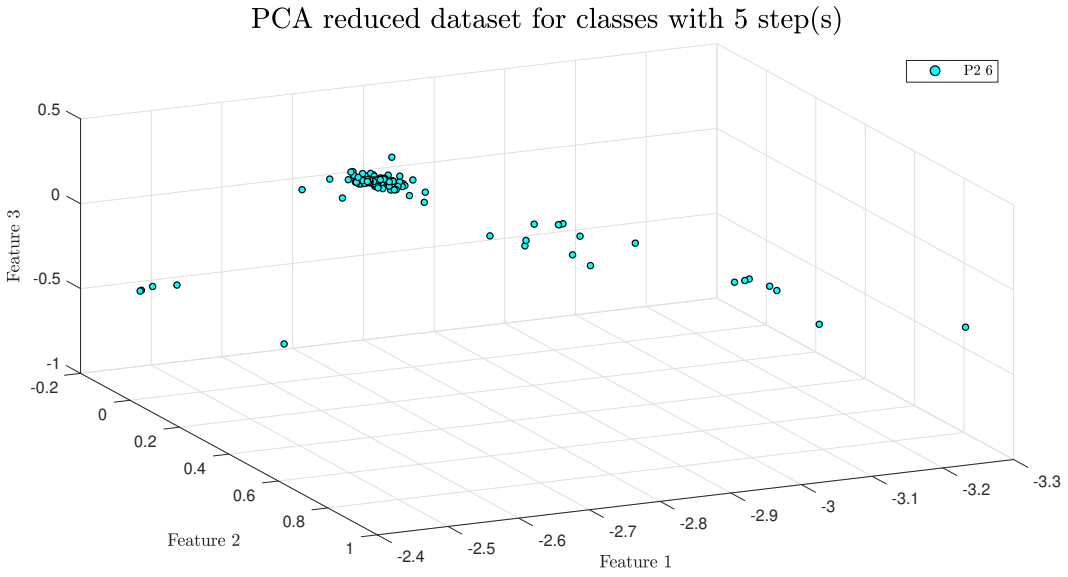


Figure 4.17: *PCA reduced dataset for classes with 5 steps*

These results are best analysed by considering the classes with different numbers of steps independently.

Figure 4.13 shows that the Barker 2, Barker 3, Frank 2, Frank 4, and P4 4 classes cluster in relative isolation compared to the other classes. The clusters for the Frank 6, Frank 8, P3 16, P3 36, and P3 64 clouds all overlap strongly, which implies that these classes may be challenging to discern from one another. It should be noted that all of these classes are high order polyphase codes.

Figure 4.14 shows a similar picture to the classes with a single step. Certain classes (Barker 4, Barker 5, P2 2, and P3 4) are clustered in isolation from other classes, while the other classes (LFM, P1 4, P1 6, P4 16, P4 36, and P4 64) appear to be densely packed and difficult to isolate. Once again, many of the classes that are packed together in this figure are high order polyphase codes.

Figure 4.15 indicates that all classes with 3 steps cluster in isolation from one another, and appear to be easily separable.

There is little use in remarking on the clustering of classes with 4 and 5 steps in Figures 4.16 and 4.17, as there is only a single class in each category.

4.3 TFR-Based CNN Classifier

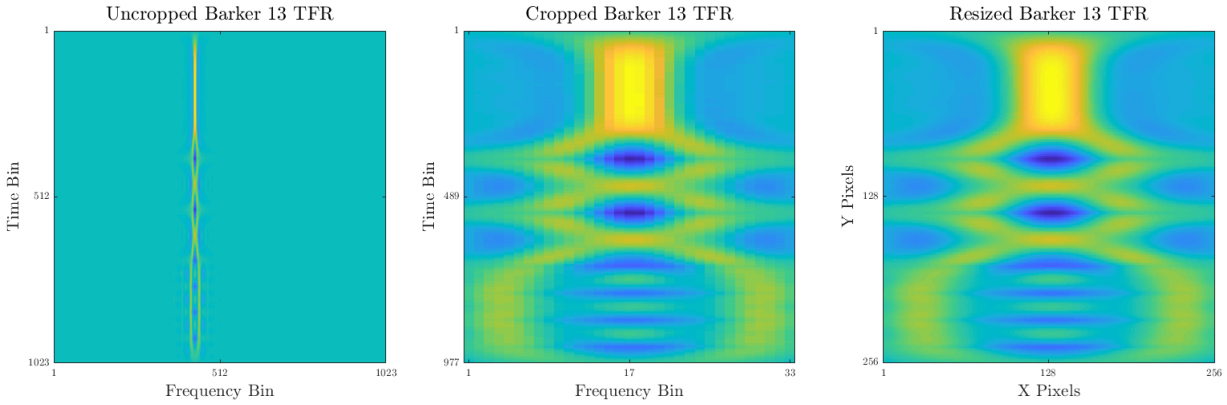
While the MSE-based ridge classifier provides a suitable base for deterministic (non-machine learning) classifiers, its implementation reveals that there is room for improvement on a number of fronts, particularly in the ability to process and generalise to more complex features. For this purpose, a time-frequency representation based (ZAM-GTFR in specific) convolutional neural network classifier was implemented.

4.3.1 TFR Preprocessing

As is the case in the ridge-based classifier, preprocessing is required to transform the time-frequency representations into a format suitable for CNN classification.

This preprocessing also involves cropping the time-frequency representation to contain only the useful parts of the image, using the same operation described in Equation 4.2, with the exception that N is set to 3 for this implementation.

An important property of CNNs is that they typically require fixed-size inputs. The size of the output of the cropping operation will vary based on the signal content, so a resizing operation was carried out to resize all cropped TFRs to 256×256 bins using bicubic interpolation. The result of the cropping and resizing steps are shown in Figures 4.18a, 4.18b and 4.18c.



(a) *Pre-crop TFR* (b) *Post-crop TFR* (c) *Post-resize TFR*

Figure 4.18: *Time-frequency distribution before and after cropping and resizing*

It is important to note that the inputs to the CNN are not RGB images, but intensity maps. Each pixel is a 1-dimensional value in the interval $[0, 1]$. A constant value of 0.5 is subtracted from each image as a final preprocessing step to bring the data into the symmetric interval $[-0.5, 0.5]$. This normalisation serves to allow the CNN to directly analyse image content instead of attempting to resolve scale differences between images.

4.3.2 CNN Model

The model used in this classifier is inspired by the VGG16 CNN created by members of the Visual Geometry Group at the University of Oxford [52]. VGG16 showed that utilising many convolutional layers (for a deeper network) with small filters could result in comparable or superior performance to shallow networks with large convolutional filters.

This ideology is followed in the model summarised in Table 4.2, where "Conv" denotes a convolutional layer and "FC" denotes a fully-connected layer.

Table 4.2: *TFR-based CNN model summary*

Layer Number	Layer Configuration
Input (256×256 intensity map)	
1	5×5 Conv, 16 filters
2×2 Maxpool	
2	3×3 Conv, 32 filters
2×2 Maxpool	
3	3×3 Conv, 64 filters
2×2 Maxpool	
4	3×3 Conv, 128 filters
2×2 Maxpool	
Flatten	
5	FC-2500, 0.5 Dropout
6	FC-200, 0.5 Dropout
Softmax	

The model was trained with the categorical cross-entropy loss function over 25 epochs using the Adam optimiser with a learning rate of 0.00001.

The number of epochs is usually determined empirically as a trade-off between the convergence of the training and validation losses and the potential for overfitting. The presence of dropout in the network reduces the tendency to overfit, so the number of epochs simply becomes a balance between loss convergence and training time. This number differs between all classifiers in this study due to the different data representations and network depths, but they are all determined according to this trade-off.
The Adam optimizer is used in all classifiers in this study due to its computational efficiency and low memory requirements as described in Section 2.4.4.4.

4.4 Instantaneous Measurement CNN Classifier

While the TFR-based CNN classifier makes use of a data representation that is efficient for manual processing and visual information extraction, the instantaneous measurement classifier seeks to explore the classification of a lower level data representation. This representation takes the form of a set of instantaneous magnitude, frequency and phase measurements. This classifier is the first deviation from data representations that are efficient for manual analysis, and aims to determine whether CNNs are able to perform feature extraction in instances where humans may struggle.

4.4.1 Data Preprocessing

The raw complex time data is processed into three vectors of instantaneous magnitude $|x[n]|$, phase $\phi[n]$, and frequency $f[n]$ using Equations 2.32, 2.33, and 2.34.

$$|x[n]| = \sqrt{\Re(x[n])^2 + \Im(x[n])^2} \tag{2.32}$$

$$\phi[n] = arg(x[n]) \tag{2.33}$$

$$f[n] = arg(x[n]\, x^*[n-1]) \tag{2.34}$$

These three vectors are stacked into a two-dimensional matrix in the order of instantaneous magnitude, frequency and then phase. Mean and range normalisation is performed on a per-row basis to translate the entire matrix to the range $[-0.5, 0.5]$. This serves to alleviate the potential effects of scale differences between the three rows. An example of such a matrix after normalisation is shown in Figure 4.19.

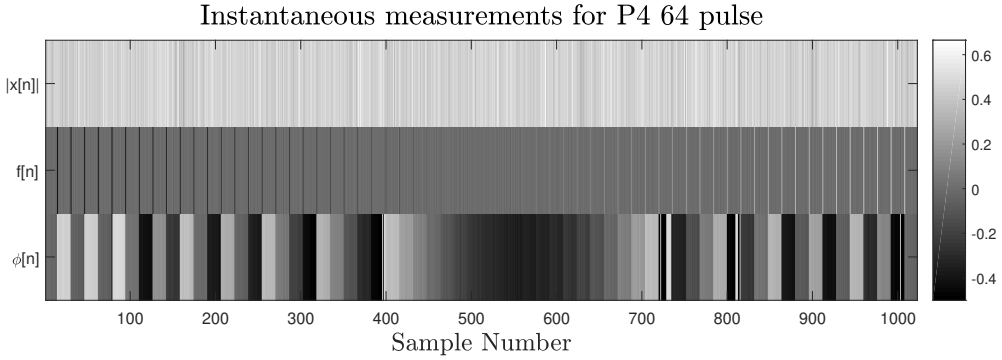


Figure 4.19: *Scaled and normalised instantaneous measurement matrix of a P4 64 pulse*

4.4.2 CNN Model

The model used in the instantaneous measurement classifier is shallower than the TFR-Based classifier model and consists of three convolutional layers before the two fully connected layers. The increased simplicity of the model is possible due to the reduced size of the input "image".

The model implemented is summarised in Table 4.3, where "Conv" layers are convolutional layers and "FC" layers are fully-connected layers.

Table 4.3: *Instantaneous measurement CNN model summary*

Layer Number	Layer Configuration
Input (3×1024 intensity map)	
1	1×32 Conv, 8 filters
1×2 Maxpool	
2	1×64 Conv, 16 filters
1×2 Maxpool	
3	1×128 Conv, 32 filters
1×2 Maxpool	
Flatten	
4	FC-1000, 0.5 Dropout
5	FC-1000, 0.5 Dropout
Softmax	

It was found that restricting the max pooling layers and convolutional layers to only horizontal operation allows for the maximum amount of data to be extracted from the three rows. This has the effect of preserving the information unique to each row throughout the convolution process while still allowing for efficient size reduction across the horizontal (time) dimension.

It should also be noted that the filter kernels are much longer (up to 128 bins) than in the TFR-based CNN classifier in Section 4.3. This change is fitting given the transformation from "image" (TFR) data to a time-series, as it was found that extending the kernels allows temporal relationships between distant samples to be observed within the window of a single kernel. Smaller kernels are inefficient in this context as they are only able to extract relationships between highly localised samples.

The model was trained with the categorical cross-entropy loss function over 350 epochs using the Adam optimiser with a learning rate of 0.00001.

4.5 Raw Complex Time Data CNN Classifier

This classifier takes the concept of exploring lower level data representations even further, to the point where almost no data preprocessing is performed and the raw complex time data of each signal is used as an input to the CNN.

The lack of complex preprocessing makes the data representation well suited to high-throughput classification, but ineffective for analysis by humans. While humans can analyse and understand IQ data, it is highly inefficient compared to the amount of information that can be visually extracted from a TFR or even a frequency spectrum.

4.5.1 Data Preprocessing

The real and imaginary parts of the complex series are separated into two vectors, which are stacked to form the complex time data "image". The only preprocessing applied to the raw complex time data is mean and range normalisation on each row of the input matrix to bring the data into the range $[-0.5, 0.5]$.

Figure 4.20 shows an example of a raw complex time data image after normalisation.

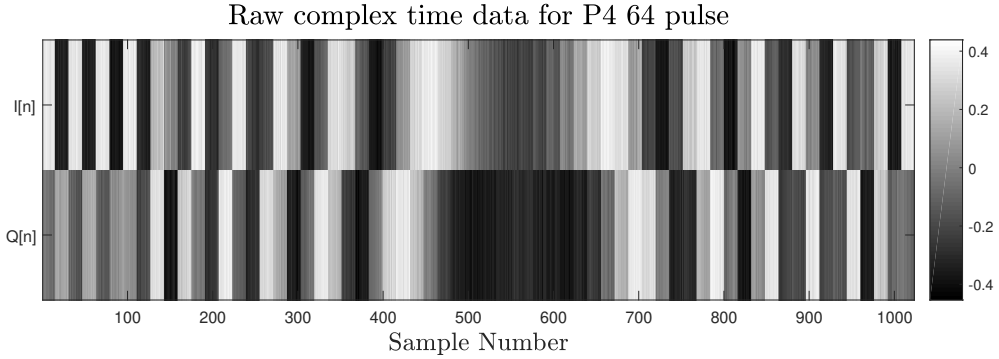


Figure 4.20: *Scaled and normalised complex time data matrix of a P4 64 pulse*

4.5.2 CNN Model

The model used in this classifier is simpler than models used for TFR and instantaneous measurement classification in Sections 4.3 and 4.4. It consists of three convolutional layers with fewer filters than in the instantaneous measurement classifier model and a single fully connected layer.

The model implemented is summarised in Table 4.4, where "Conv" layers are convolutional layers and "FC" layers are fully-connected layers.

Table 4.4: *Complex time data CNN model summary*

Layer Number	Layer Configuration
Input (2×1024 intensity map)	
1	1×32 Conv, 4 filters
1×2 Maxpool	
2	1×64 Conv, 8 filters
1×2 Maxpool	
3	1×128 Conv, 16 filters
1×2 Maxpool	
Flatten	
4	FC-1000, 0.5 Dropout
Softmax	

It should be noted once again that the convolutional and max pooling layers are limited to horizontal operation as described in Section 4.4.2. This has the effect of keeping the rows containing the real and imaginary data intact until they are flattened and fed into the fully connected layer. In keeping the real and imaginary data intact, the phase information between the real and imaginary samples is also preserved throughout the convolution process.

Since this model is similar to the instantaneous measurement model in that it also operates on time-series data, the filter kernels have once again been extended to allow for more efficient extraction of relationships between distant samples.

The model was trained with the categorical cross-entropy loss function over 500 epochs using the Adam optimiser with a learning rate of 0.0001.

4.6 Split I and Q Complex Time Data Classifiers

The raw complex time data classifier utilises both the in-phase (I) and quadrature (Q) components of the complex signal for analysis simultaneously. The idea of performing classification using only a single component (I or Q) is analysed by training two separate classifiers, one for the I component and one for the Q component.

The results of the classifiers are finally fused in order to observe whether a combination of two separate classifiers for I and Q can produce comparable results to a single IQ classifier.

4.6.1 Data Preprocessing

Data is preprocessed in exactly the same manner as for the raw complex time data classifier, as described in Section 4.5.1. The only difference is the fact that the I and Q components of the signal are split into two separate datasets, meaning that the input shape of the classifier is a vector instead of a matrix.

4.6.2 CNN Model

The models for the I and Q classifiers are identical and are extremely similar to the model of the raw complex time data classifier, with the only difference being the size of the input to the network.

These models are summarised in Table 4.5.

Table 4.5: *Split complex time data CNN model summary*

Layer Number	Layer Configuration
Input (1×1024 intensity map)	
1	1×32 Conv, 4 filters
1×2 Maxpool	
2	1×64 Conv, 8 filters
1×2 Maxpool	
3	1×128 Conv, 16 filters
1×2 Maxpool	
Flatten	
4	FC-1000, 0.5 Dropout
Softmax	

The I and Q models were trained with the categorical cross-entropy loss function over 500 epochs using the Adam optimiser with a learning rate of 0.0001.

4.6.3 Modes of Testing

The two split complex time data classifiers can either be tested independently or combined to produce a single set of results.

4.6.3.1 Separate Analysis

In the separate analysis mode of testing, the I and Q classifiers are evaluated separately, and their accuracy is computed using the same metrics as used for all other classifiers in this study.

4.6.3.2 Combined Analysis

In the combined analysis mode of testing, the outputs of the I and Q classifiers are combined using a confidence-based selection method.

This method compares the results produced by the two classifiers and selects the result with the highest confidence value (largest output of the softmax layer). This process is repeated for each example in the test set, the outcomes of which are then compared to the labels of the dataset and used to establish an accuracy metric.

4.7 Low-SNR Analysis

In order to fully evaluate the trained models, a secondary test dataset was generated using an extended SNR range. This experiment aims to test the models until the point of classification break-down in order to better evaluate the resilience of the models and the different data representations to noise.

This extended test dataset is made up of 5600 examples with uniform distributions of modulation type, SNR, CFO and jitter. With the exception of SNR, all parameters and ranges were simulated as described in Section 4.1. The SNR range was extended from the original [-5, 20] dB range to a larger [-20, 20] dB range.

The models were tested on the low-SNR dataset and their relative performance was compared and discussed. The performance differences of the models between the low-SNR dataset and the original dataset were also analysed and discussed.

4.8 Filter Activation Analysis

By following the path of data through a convolutional neural network, it is possible to observe the "activations" of each convolutional filter to the presence of the input data. These activations are simply the output of the convolution operation between the filter and the input data being fed through the filter.

Visualising these activations can provide an indication of the features being extracted by the specific filter in question, and can provide insight into which parts of the input image are deemed important by the CNN.

The first layer of convolutional filters in the raw complex time data classifier was analysed for certain classes in an attempt to draw conclusions as to which features or events from the input data were important to the classification process.

4.9 Complex Time Data Reproduction

The complex time data classifier was further analysed by finding and visualising the input "images" (complex time data) that maximised certain output classes. This experiment forms the precursor to possible research on adversarial techniques targeted at this classifier. The concept of adversarial examples and techniques is discussed in Section 2.5.6.

The resultant signals are compared to reference signals from different classes, and the differences are remarked upon. The complex time data generated through this technique is also visualised using common signal processing transformations in order to provide insights into the important features of each class as observed by the CNN.

4.9.1 Gradient Ascent Input Optimisation

Members of the Keras machine learning framework development team implemented an approach for the above technique that treats it as an optimisation problem [53]. In this problem, the optimisation objective (the loss function) is simply defined as the output of the neural network (the confidence value) corresponding to the class of interest. The input parameter to be varied by the optimisation process is an image with the same dimensions as the input of the neural network.

The input image is varied over a predefined number of iterations, with a step of gradient ascent performed after each iteration. Gradient ascent is used instead of gradient descent as the objective is now to maximise the loss function as it represents the confidence of the neural network that the input image belongs to the class in question. The implementation by the Keras team was originally demonstrated using the VGG16 model, but was adapted for use with raw complex time data classifier in this study. These adaptations included correcting the expected image size and modifying the gradient ascent algorithm to operate on intensity maps instead of a RGB images.

4.9.2 Visual Analysis

Once the complex time data has been extracted using the techniques mentioned above, common signal processing transformations are applied to visually analyse and compare the resultant signal to a reference signal from the dataset. The transformations used for this visual analysis have been used throughout this study and are listed below.

- Instantaneous magnitude
- Instantaneous frequency
- Instantaneous phase
- Time-frequency representation (ZAM-GTFR)

4.10 Antenna to Antenna Experiment

An experiment was performed whereby pulses were transmitted and received over a real-world radio link before being classified. The aim of this experiment was to investigate the effects of such a radio link on the simulated signals and to evaluate the feasibility of applying simulation-trained classifiers to real-world data.

A block diagram of the experiment is shown in Figure 4.21.

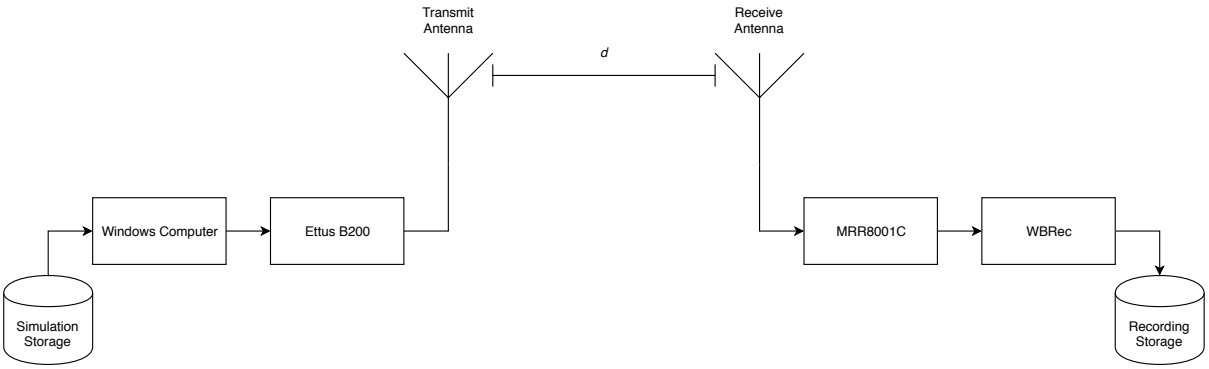


Figure 4.21: *Block diagram of the antenna to antenna experiment configuration*

Simulated pulses were transmitted using an Ettus Universal Software Radio Peripheral (USRP) B200 [54] and subsequently captured using a GEW MRR8001C receiver [55] attached to a wideband recorder (WBRec). The MRR8001C receiver captured the signal with a 20 MHz bandwidth and a sample rate of 25.6 MHz.

The experiment was performed in the section of the UHF band licensed for amateur radio (2300 MHz - 2450 MHz) [56] using off-the-shelf antennas designed for the transmission and reception of Wi-Fi signals.

The antennas were positioned approximately 3 metres apart (constrained by physical equipment locations in the Peralex offices) in order to ensure that the separation between antennas, d, exceeded the far-field distance of the signal, $D_{FarField}$. The far-field distance is given by Equation 4.4, where L is the length of the antenna and λ is the wavelength of the signal.

$$D_{FarField} \geq \frac{2L^2}{\lambda} \tag{4.4}$$

A simple calculation using "worst-case" values for the antenna length (197 mm including the connector and housing) and wavelength (12.2 cm at 2450 MHz) reveals that the far field distance of such a signal is 0.634 m. This shows that an antenna separation of 3 metres is more than sufficient for far-field operation in this context.

A recording of a pulse train for each class was made and a single pulse from each class was manually extracted. This set of pulses was packaged into a dataset compatible with the models designed in this study and run through each model for classification.

Different data representations were analysed visually in order to remark on the effects of the radio link on these representations, and the salient results arising from the analysis of the performance of each model were discussed.

Chapter 5

Results

5.1 MSE-Based Ridge Classifier

5.1.1 Dataset Information

As described in Section 4.1, the ridge-based classifier was trained on a dataset comprised of 11200 simulated signals and tested on 2800 simulated signals. All signals are spread uniformly across all classes and pulse imperfection ranges. It should be noted that this classifier does not make use of a cross-validation set. The extra examples that would form the cross-validation set have instead been used as extra training examples.

The classifier was also tested on a secondary dataset with no pulse detection jitter in order to better quantify the effects of SNR on this classification approach in a jitter-free environment.

5.1.2 Classification Accuracy

The classifier achieved an overall classification accuracy of 70.57% across the 28 classes. The breakdown of classification accuracy per class is given in Table 5.1.

The performance of the classifier can be further visualised through the confusion matrix shown in Figure 5.1, which is also tabulated in Table B.2 in Appendix B. This matrix shows the actual classes (labels) on the x-axis and the predicted classes on the y-axis. Each cell provides the percentage of how many times the actual class on the x-axis was predicted as the class on the y-axis. This representation allows for prediction trends to be observed where one class may be consistently being misclassified (or "confused") for another.

Note that a 29^{th} class has been included in the confusion matrix on the predictions axis. This represents the case in which the classifier was unable to make a valid prediction due to an unexpected number of steps. It can, therefore, be thought of as an "undefined" or "invalid" class.

Table 5.1: *Classification accuracy per class*

Class	*Accuracy* (%)
FM	74.29
Barker 2	96.19
Barker 3	80.37
Barker 4	74.53
Barker 5	62.74
Barker 7	92.86
Barker 11	100.00
Barker 13	91.20
Frank 2	91.18
Frank 4	98.00
Frank 6	73.47
Frank 8	75.86
P1 2	81.31
P1 4	71.03
P1 6	93.14
P1 8	34.94
P2 2	81.18
P2 4	47.53
P2 6	26.00
P2 8	38.89
P3 4	30.85
P3 16	45.28
P3 36	11.45
P3 64	58.47
P4 4	93.68
P4 16	73.87
P4 36	96.94
P4 64	75.24

The effects of signal-to-noise ratio, carrier frequency offset, and pulse detection jitter are shown in Figures 5.2, 5.3, and 5.4 respectively. These figures show that the performance of the classifier tended to decrease as more jitter was introduced into pulses. The classification accuracy begins to decline slightly as more noise is introduced into the system, but SNR is clearly not the performance-limiting factor in this experiment. There was no clear trend between CFO and classification accuracy. This implies that the thresholding and cropping operations that this classifier uses for CFO compensation are performing as intended.

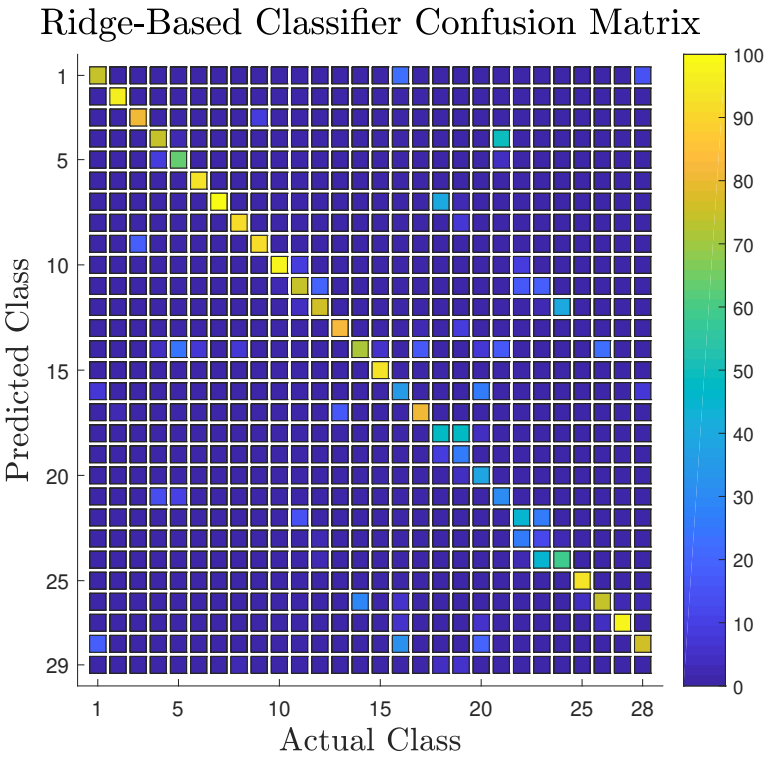


Figure 5.1: *2D confusion matrix for the ridge-based classifier*

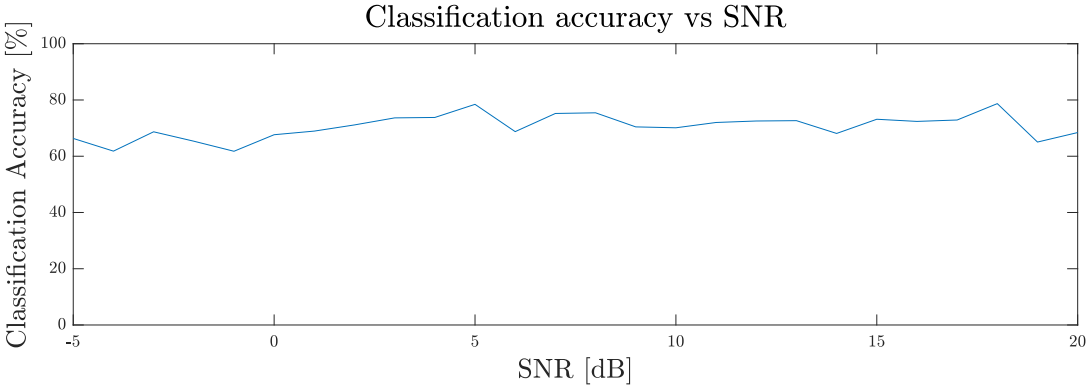


Figure 5.2: *Accuracy vs SNR for the ridge-based classifier*

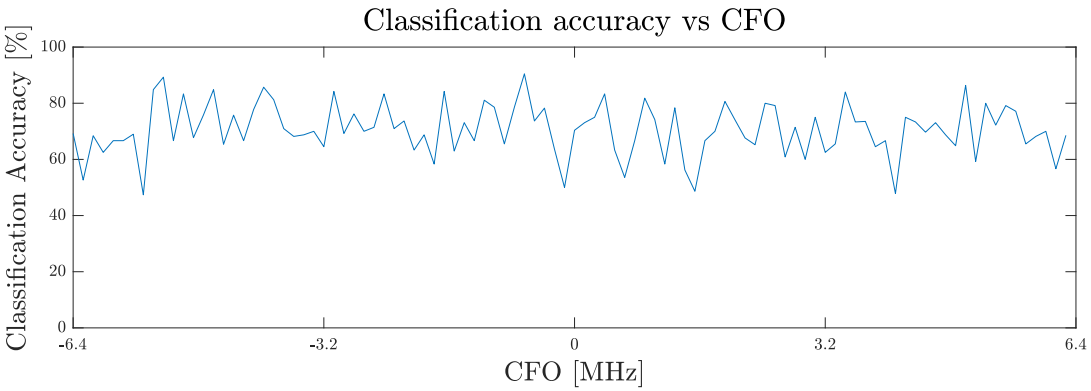


Figure 5.3: *Accuracy vs CFO for the ridge-based classifier*

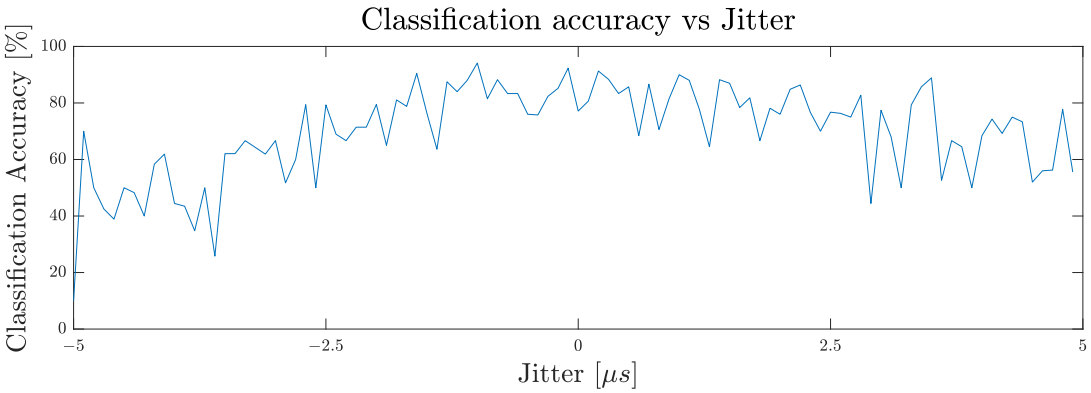


Figure 5.4: *Accuracy vs pulse detection jitter for the ridge-based classifier*

In order to further quantify the effects of pulse detection jitter and SNR, the classifier was run over a secondary training and test dataset that contains no jitter. The classifier achieved an accuracy of 87.40% on this dataset, and showed SNR to be the performance-limiting factor when jitter is not present. This classification accuracy corresponds to the performance observed in the regions of Figure 5.4 where the amount of jitter is minimal. The plot of classification accuracy vs SNR is the salient result from this experiment, and is shown in Figure 5.5.

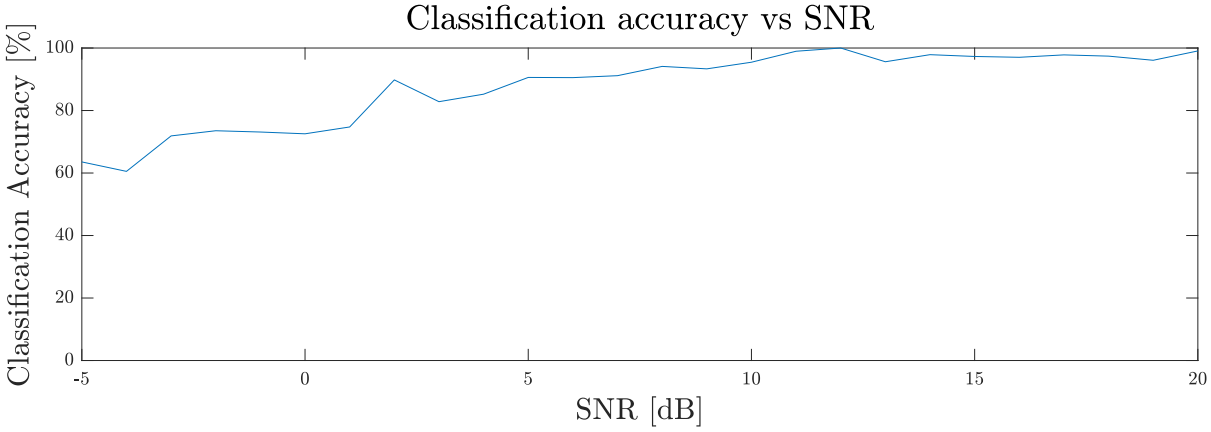


Figure 5.5: *Accuracy vs SNR for the ridge-based classifier with no jitter*

5.1.3 Shortcomings and Limitations

While this classifier performed acceptably, it is not well equipped to deal with LFM signals with varying bandwidths, high order polyphase modulated signals, signals without clean steps in their ridge, and signals containing pulse detection jitter. The performance degradation due to pulse detection jitter is to be expected - jitter shifts the locations of steps within the ridge of the signal, and these step locations are a critical feature for classification. The other limitations are discussed in the coming subsections.

5.1.3.1 Non-Stepped Ridges

Table 5.1 shows that one of the classes on which the classifier performed particularly poorly was P2 order 6. Consider the ridge of a P2 coded pulse of order 6 shown in Figure 5.6.

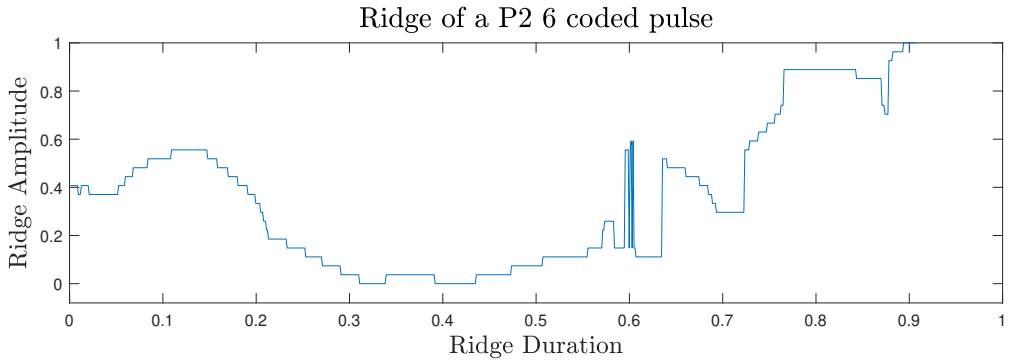


Figure 5.6: *Ridge of a P2 6 coded pulse showing the lack of distinct steps*

The ridge in Figure 5.6 clearly does not contain distinctly separable steps. Due to the fact that the number of steps is crucial to the operation of the classifier, ridges without steps such as the one in Figure 5.6 are ill-suited for classification using this model.

5.1.3.2 LFM and High Order PM Signals

Due to the fact that the ridge plot represents the frequency at which the distribution is at a maximum for each time slice, linear frequency modulated signals will have different step amplitudes based on the modulation bandwidth. This will interfere with classification accuracy if the step amplitude is highly weighted during classification.

Many high order polyphase modulation schemes (particularly P1, P2, P3, and P4) have ridge plots with structures extremely similar to that of a LFM pulse. This is due to the fact that these modulation schemes implement a stepped chirp-like waveform. This similarity makes classifying between different high order polyphase modulated pulses (P3 36, P3 64, and P4 64 for example) and LFM pulses extremely difficult and sometimes impossible when using the ridge-based approach. This is illustrated in Figure 5.7, where the ridge plots of a LFM pulse and a P4 64 pulse are overlaid to show their similarity. Note that these ridge plots appear to change direction half-way through the pulse. This is a side-effect of the preprocessing step described in Section 4.2.3 whereby the absolute value of the ridge is computed in order to solve issues with ridge oscillations due to symmetry in the TFR for some phase coded waveforms.

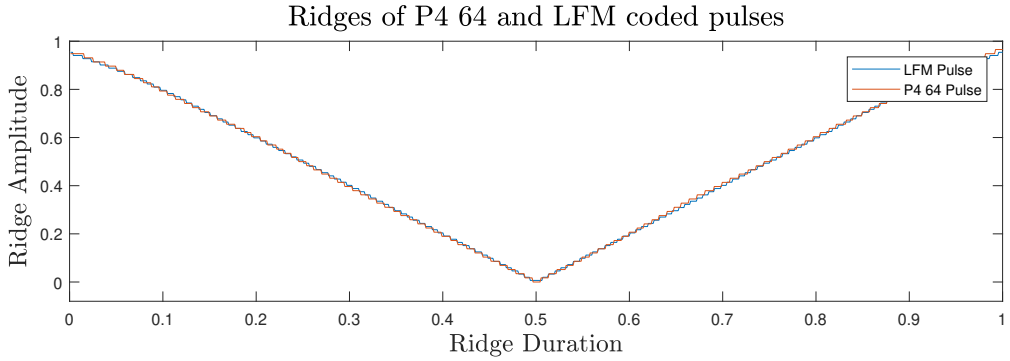


Figure 5.7: *Ridges of LFM and P4 64 pulses*

5.2 TFR-Based CNN Classifier

5.2.1 Dataset Information

As described in Section 4.1, the TFR-Based CNN classifier was trained on a dataset consisting of 8400 simulated signals, with the same uniform distribution across class, SNR, CFO and jitter as used in the ridge-based MSE classifier.

The cross-validation and testing datasets consisted of 2800 examples each.

5.2.2 Classification Accuracy

This classifier achieved an overall classification accuracy of 98.82% across the 28 classes.

The training process took 14.28 minutes, an average of 34.2 seconds per epoch. The learning curves of the classifier are shown in Figure 5.8. The training and validation accuracy appears to have completely converged by 15 epochs, implying that the training could have feasibly been stopped at this point.

The feedforward time of the model was measured in batches of 64 pulses, and the model was found to be able to process approximately 725 pulses per second.

The confusion matrix for the classifier is shown in Figure 5.9, and is tabulated in Table B.3 in Appendix B. Each cell represents the percentage of how many times the actual class on the x-axis was classified as the predicted class on the y-axis.

The effects of SNR, CFO and pulse detection jitter variation can be observed through Figures 5.10, 5.11, and 5.12 respectively. Due to the exceptionally high classification accuracy of the model, it is clear that none of the pulse imperfections had a significant impact on the performance of the model.

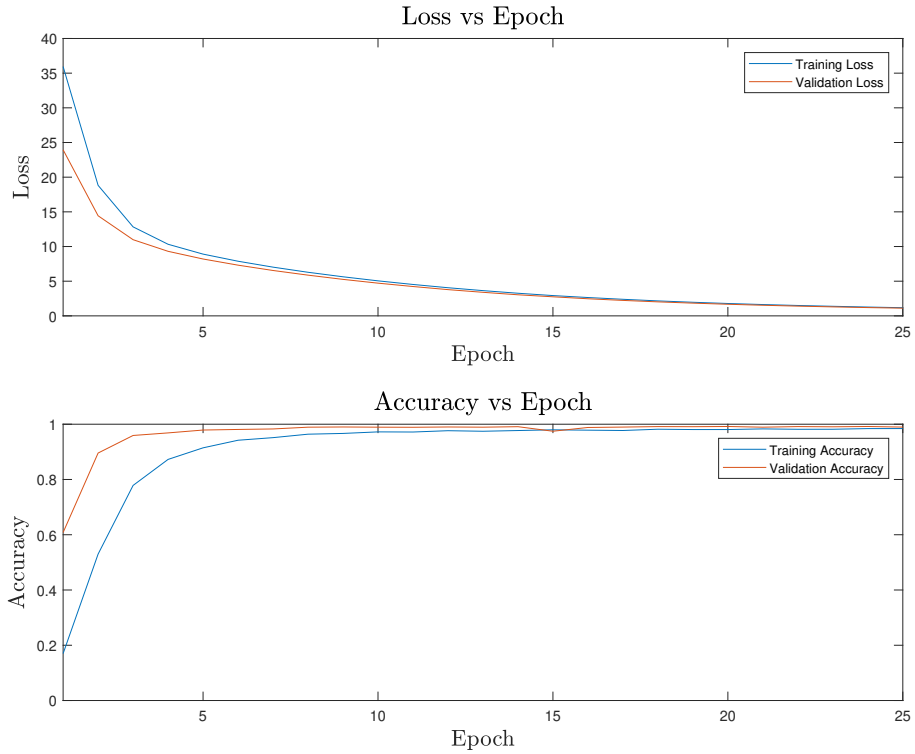


Figure 5.8: *Loss and accuracy learning curves for the TFR-based classifier*

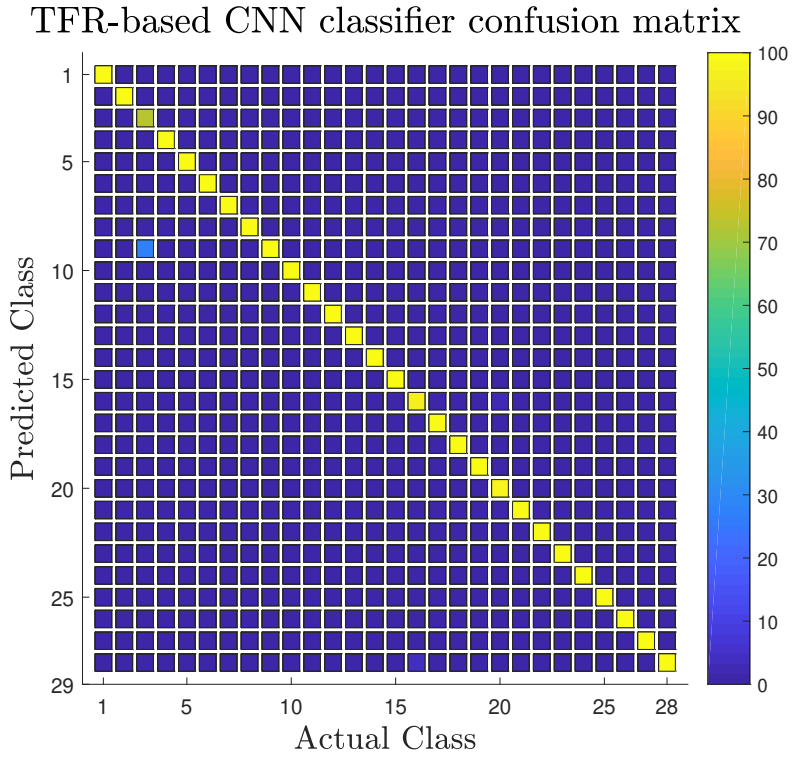


Figure 5.9: *Confusion matrix for the TFR-based CNN classifier*

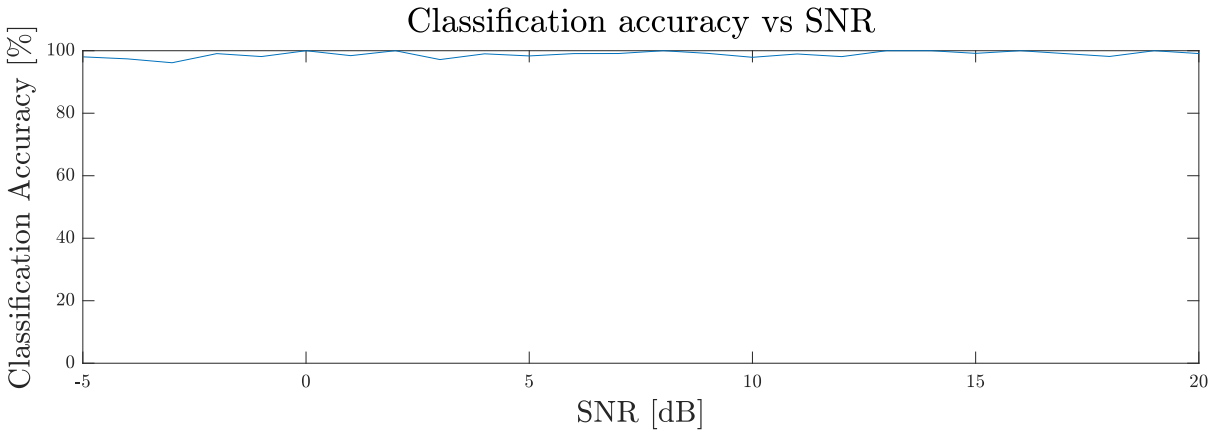


Figure 5.10: *Accuracy vs SNR for the TFR-based classifier*

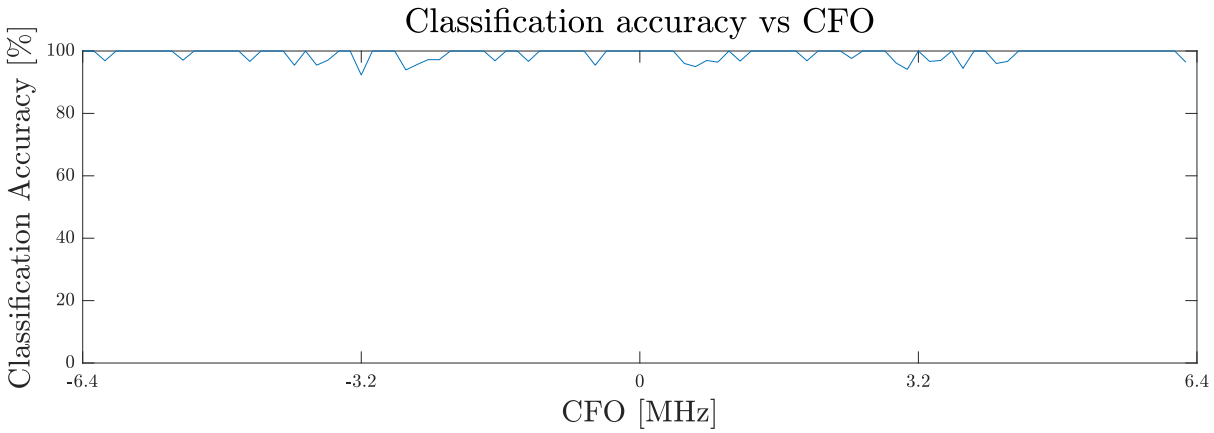


Figure 5.11: *Accuracy vs CFO for the TFR-based classifier*

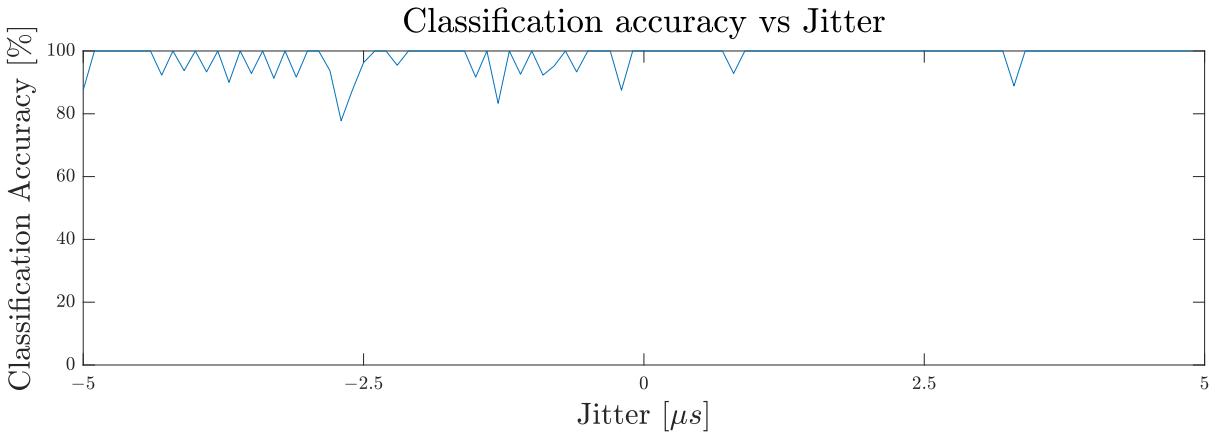


Figure 5.12: *Accuracy vs pulse detection jitter for the TFR-based classifier*

5.3 Instantaneous Measurement CNN Classifier

5.3.1 Dataset Information

This classifier was trained on the same dataset as the TFR classifier, albeit with different preprocessing to transform the data into the set of instantaneous measurements.

5.3.2 Classification Accuracy

The classifier achieved an overall classification accuracy of 84.07% across the 28 classes.

The training process took 39.37 minutes, an average of 6.74 seconds per epoch. The feedforward (prediction) time of the model was measured in batches of 64 pulses, and the model was found to be able to process approximately 11000 pulses per second.

The learning curves of the classifier are shown in Figure 5.13. The validation accuracy appears to converge at approximately 150 epochs, while the training accuracy improves across all 350 epochs. The seemingly unbounded improvement of training accuracy and convergence of validation accuracy shows that the classifier is beginning to overfit to the training data, but not to the point of being unable to generalise to new data. This indicates that the dropout in the fully connected layers is performing its intended function.

The confusion matrix for the classifier is shown in Figure 5.14, and is also tabulated in Table B.4 in Appendix B.

The effects of SNR, CFO, and pulse detection jitter variation can be observed through Figures 5.15, 5.16, and 5.17 respectively. Unlike the TFR-based classifier results in Section 5.2.2, these figures clearly show that the classification accuracy of the model depends strongly on SNR. The model began to exhibit poor performance (sub-80%) below 0 dB SNR. The amount of jitter and CFO did not have a significant impact on classification accuracy.

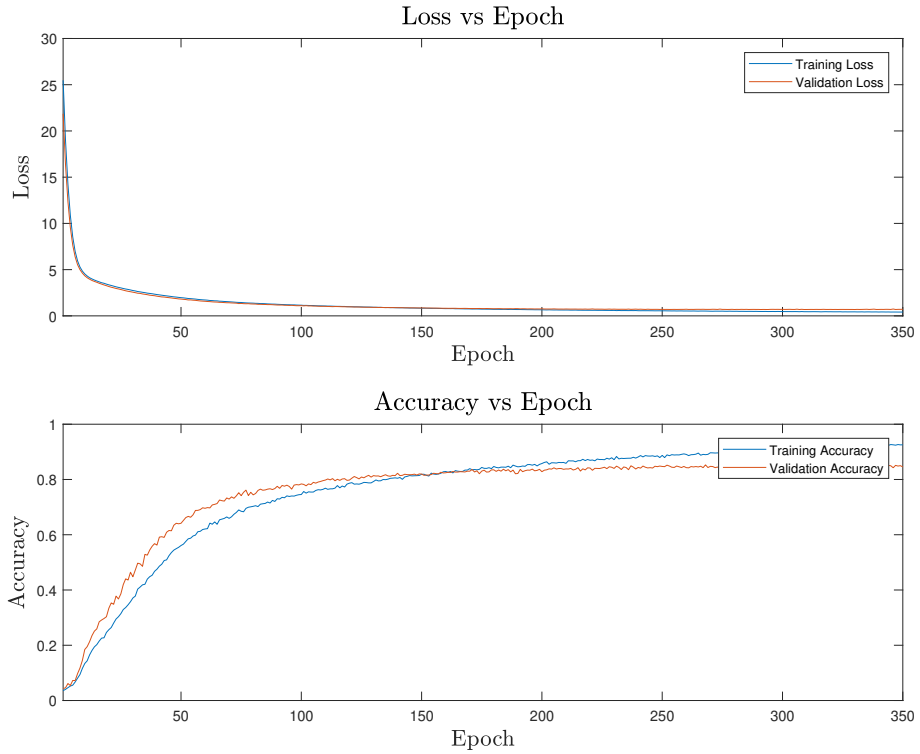


Figure 5.13: *Loss and accuracy learning curves for the instantaneous measurement classifier*

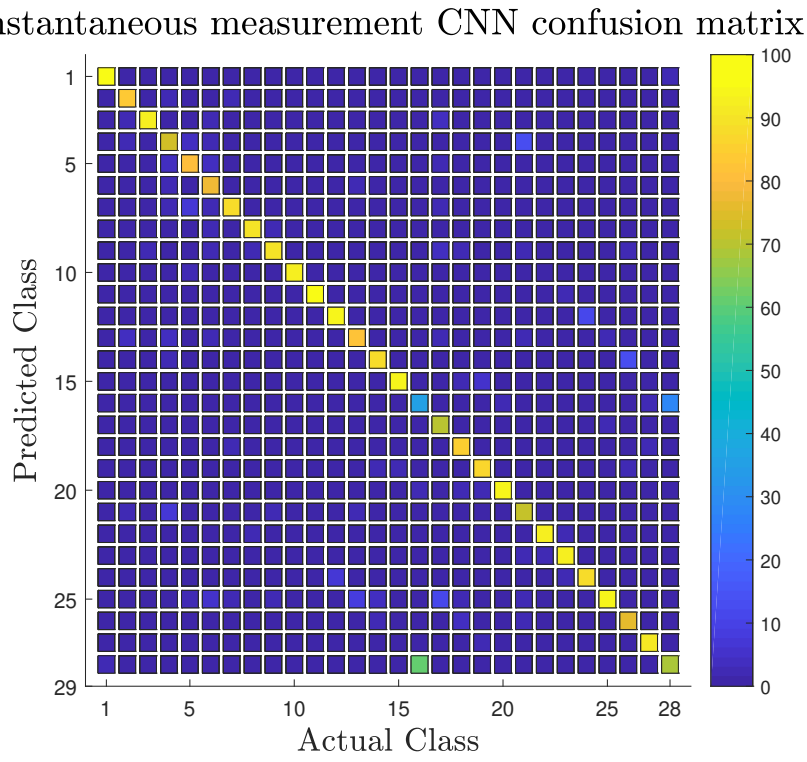


Figure 5.14: *Confusion matrix for the instantaneous measurement CNN classifier*

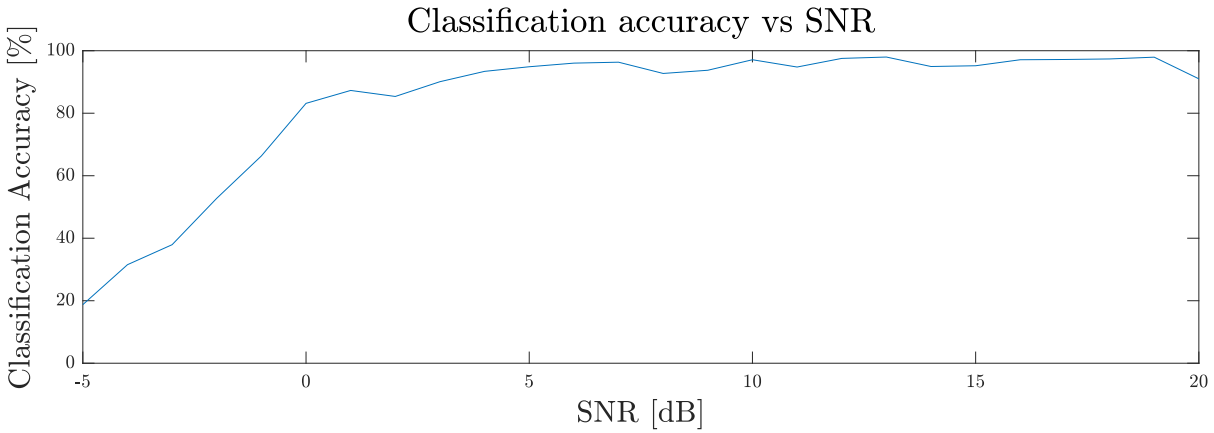


Figure 5.15: *Accuracy vs SNR for the instantaneous measurement classifier*

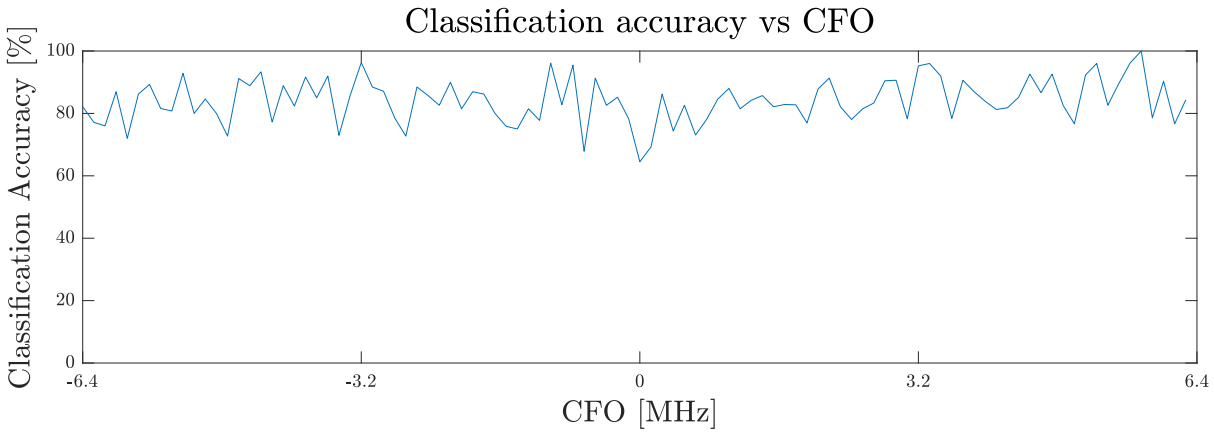


Figure 5.16: *Accuracy vs CFO for the instantaneous measurement classifier*

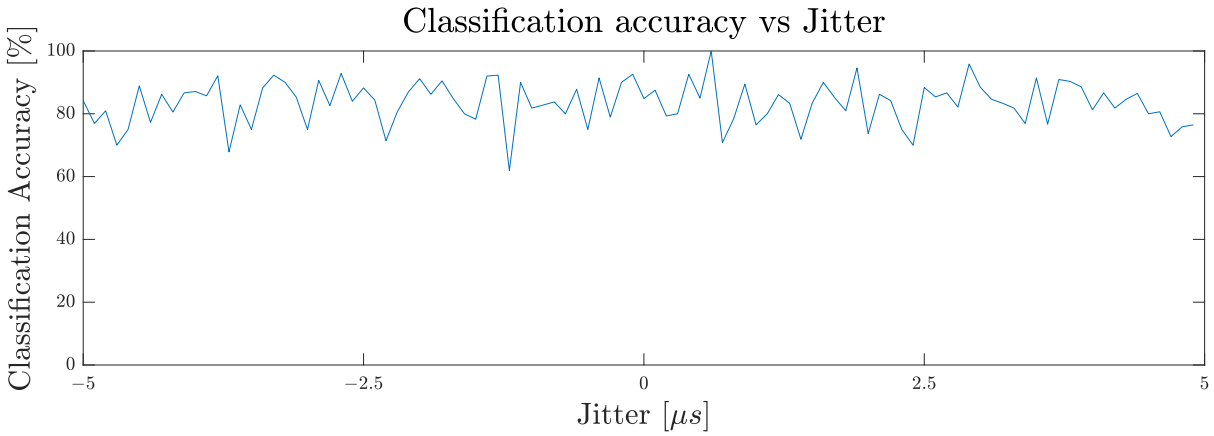


Figure 5.17: *Accuracy vs pulse detection jitter for the instantaneous measurement classifier*

5.4 Raw Complex Time Data CNN Classifier

5.4.1 Dataset Information

The raw complex time data classifier was trained, validated and tested on the same datasets as the other convolutional neural network classifiers mentioned in this section.

5.4.2 Classification Accuracy

The classifier achieved an overall classification accuracy of 86.57% across the 28 classes.

The training process took 42 minutes, an average of 5.04 seconds per epoch. The feedforward (prediction) time of the model was measured in batches of 64 pulses, and the model was found to be able to process approximately 18000 pulses per second.

The learning curves of the classifier are shown in Figure 5.18. The accuracy curves show the same phenomenon as observed in the instantaneous measurement classifier in Section 5.3.2 whereby the validation accuracy converges at approximately 250 epochs while the training accuracy continues to increase. This observation, coupled with the convergence of the loss curves, is an indication that the number of epochs could have been halved from 500 to 250.

The confusion matrix for the classifier is shown in Figure 5.19, and is also tabulated in Table B.5 in Appendix B.

The effects of SNR, CFO, and pulse detection jitter variation can be observed through Figures 5.20, 5.21, and 5.22 respectively. As shown for the instantaneous measurement classifier in Section 5.3.2, the classification accuracy of this model depends strongly on SNR, with a clear accuracy roll-off present below 0 dB SNR. There was once again no clear correlation between CFO, jitter, and classification accuracy.

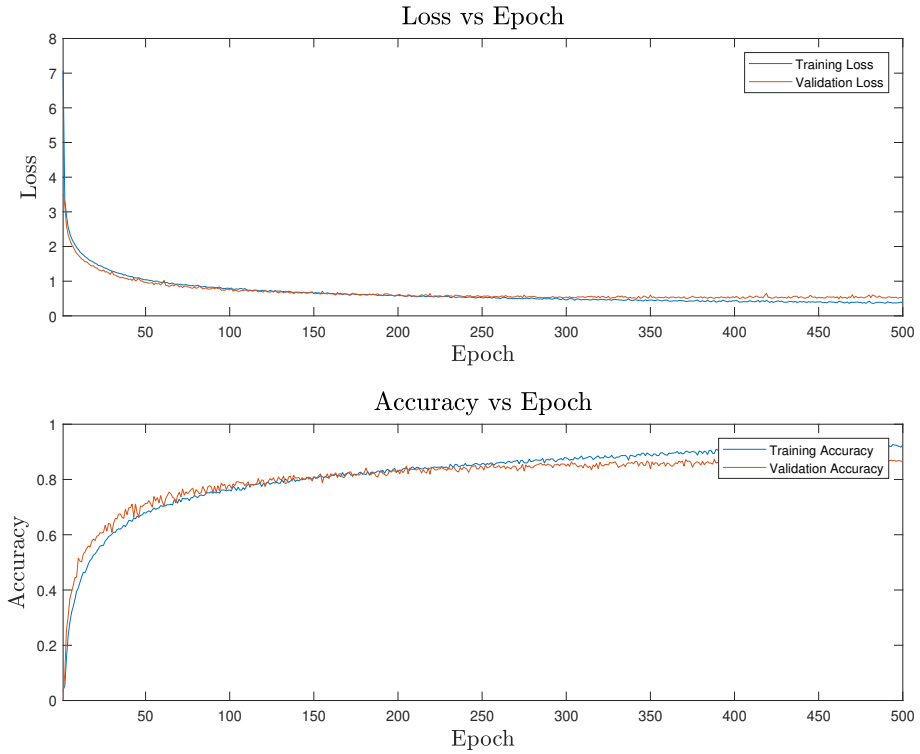


Figure 5.18: *Loss and accuracy learning curves for the raw complex time data classifier*

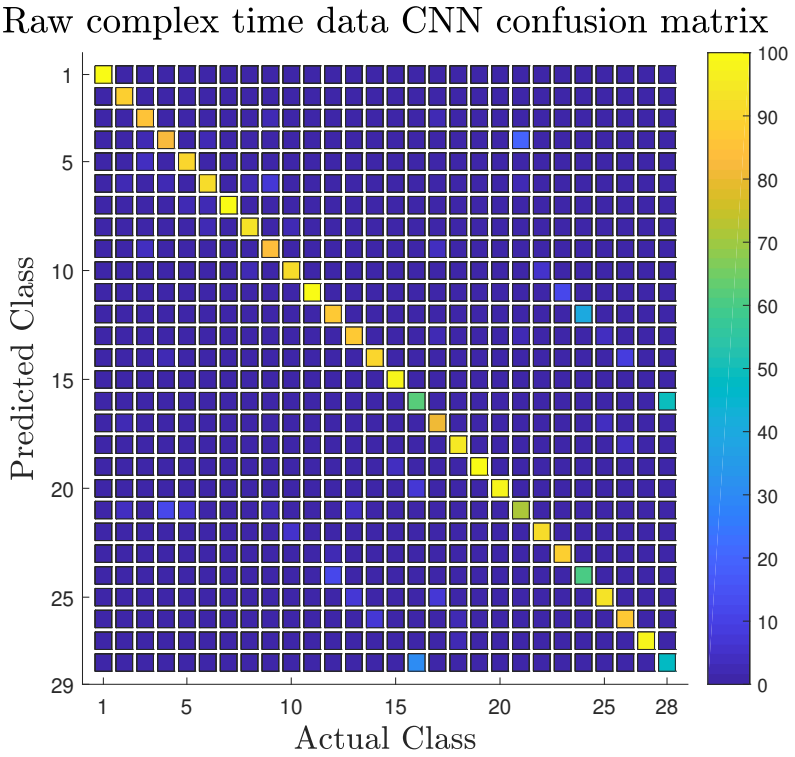


Figure 5.19: *Confusion matrix for the raw complex time data CNN classifier*

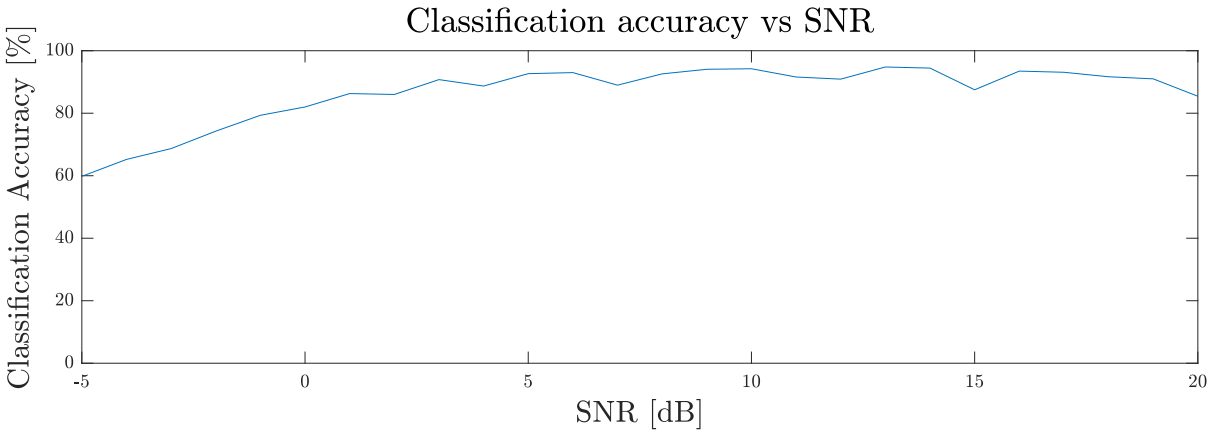


Figure 5.20: *Accuracy vs SNR for the raw complex time data classifier*

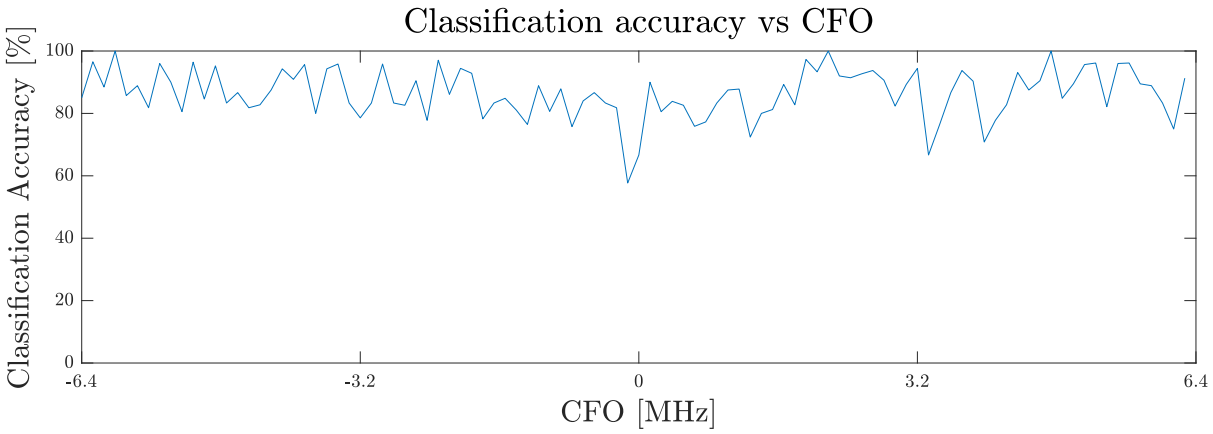


Figure 5.21: *Accuracy vs CFO for the raw complex time data classifier*

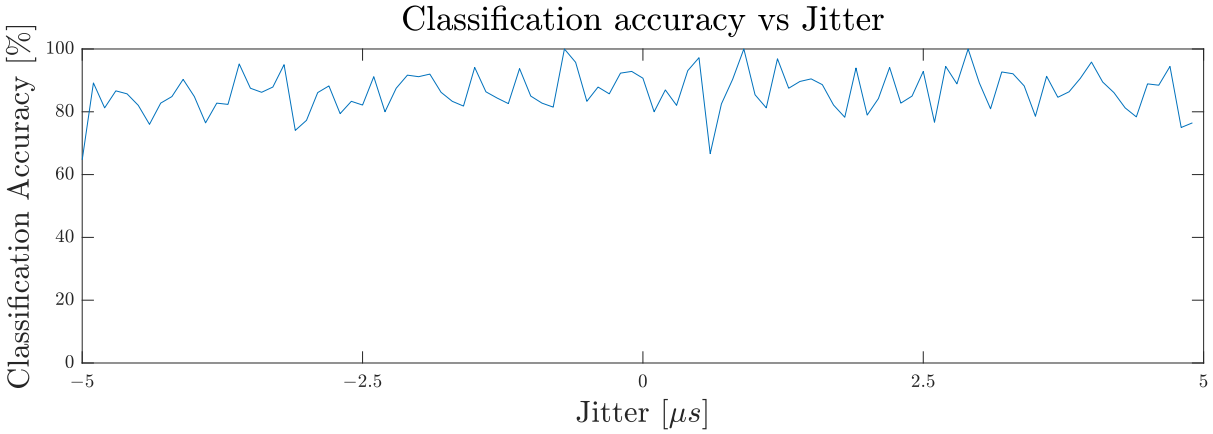


Figure 5.22: *Accuracy vs pulse detection jitter for the raw complex time data classifier*

5.5 Split I and Q Complex Time Data Classifiers

5.5.1 Separate Analysis

The classification accuracy and training times for the individual I and Q classifiers are shown in Table 5.2.

Table 5.2: *Split complex classifier results*

Classifier	Accuracy (%)	Training Time (min)
In-Phase (I)	84.71	41.02
Quadrature (Q)	80.71	39.86

The learning curves for the two classifiers were overlaid and are shown in Figure 5.23. As expected, these curves are extremely similar to the learning curves of the raw complex time data classifier in Section 5.4.2. Both classifiers showed clear convergence of validation accuracy and loss at approximately 250 epochs. The loss and accuracy for the Q classifier did not converge to exactly the same degree as the I classifier, implying that the gradient descent algorithm may have gotten stuck in a local minimum towards the end of the training process. It is expected that the two networks would train slightly differently due to the random initialisation of weights.

5.5.2 Combined Analysis

After combining the I and Q complex time data classifiers using a confidence-based approach, a final accuracy metric of 88.143% was obtained. Note that this accuracy is higher than the classification accuracy realised through the separate analyses of the I and Q classifiers, and is higher than the combined raw complex time data classifier.

The feedforward speed of this classifier was not measured due to the fact that the classifier is comprised of two individual neural networks and an architecture that allowed for simultaneous feedforward of both networks was not explored.

Figure 5.24 shows the confusion matrix of the combination of the I and Q complex time data classifiers. The full confusion matrix is also tabulated in Table B.6 in Appendix B. The effects of SNR, CFO, and pulse detection jitter variation were analysed and plotted in Figures 5.25, 5.26, and 5.27. These figures clearly show that the model is, like other models discussed, sensitive to SNR. Interestingly, Figure 5.26 shows that the model also tended to perform better on signals with non-zero CFO.

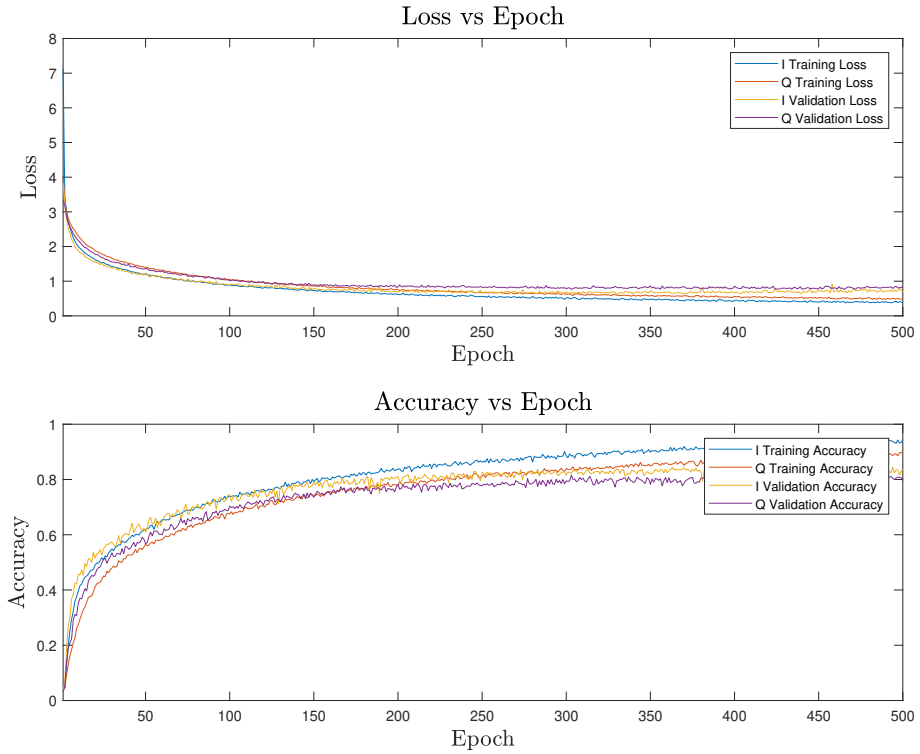


Figure 5.23: *Loss and accuracy learning curves for the I and Q split complex time data classifiers*

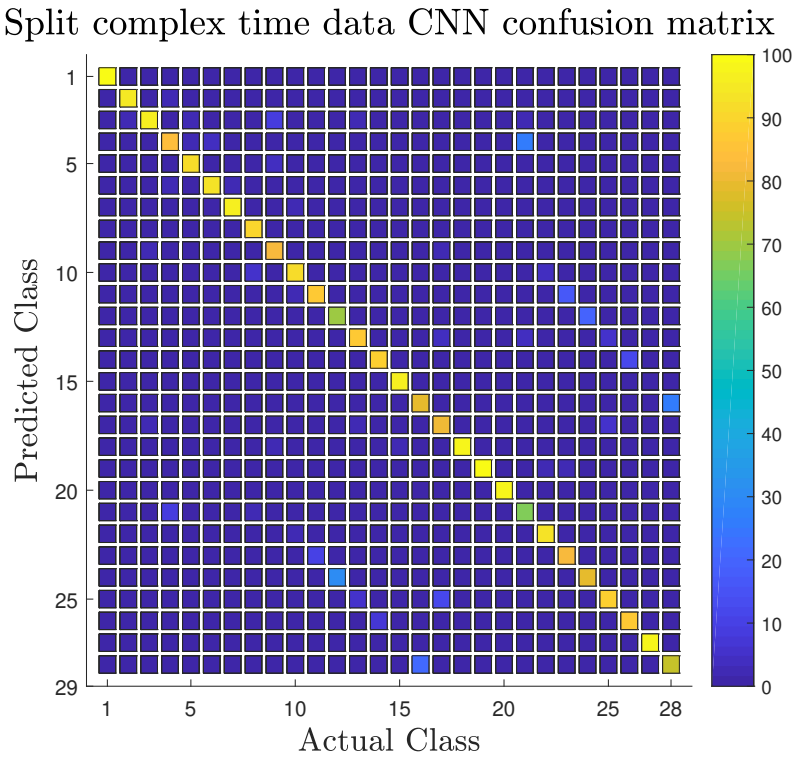


Figure 5.24: *Confusion matrix for the split complex time data CNN classifier*

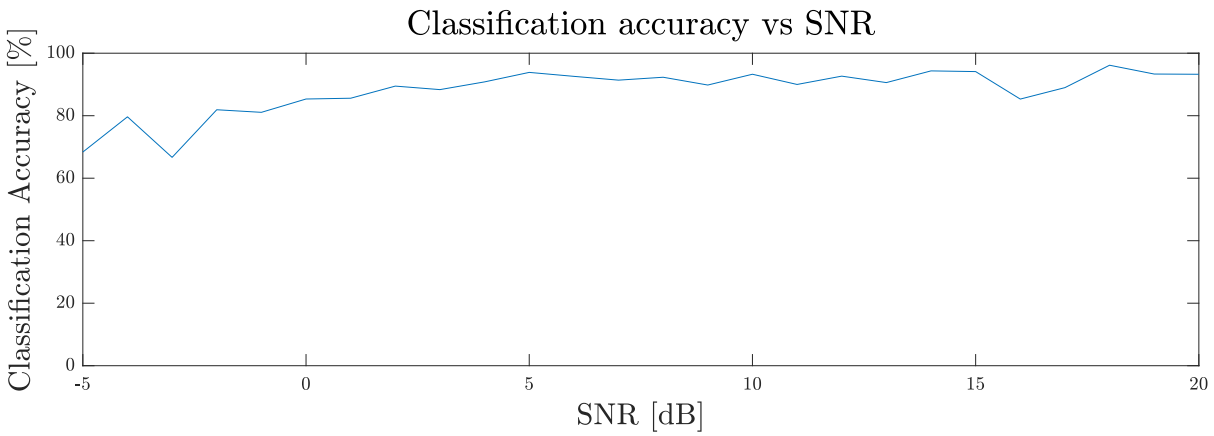


Figure 5.25: *Accuracy vs SNR for the split complex time data classifier*

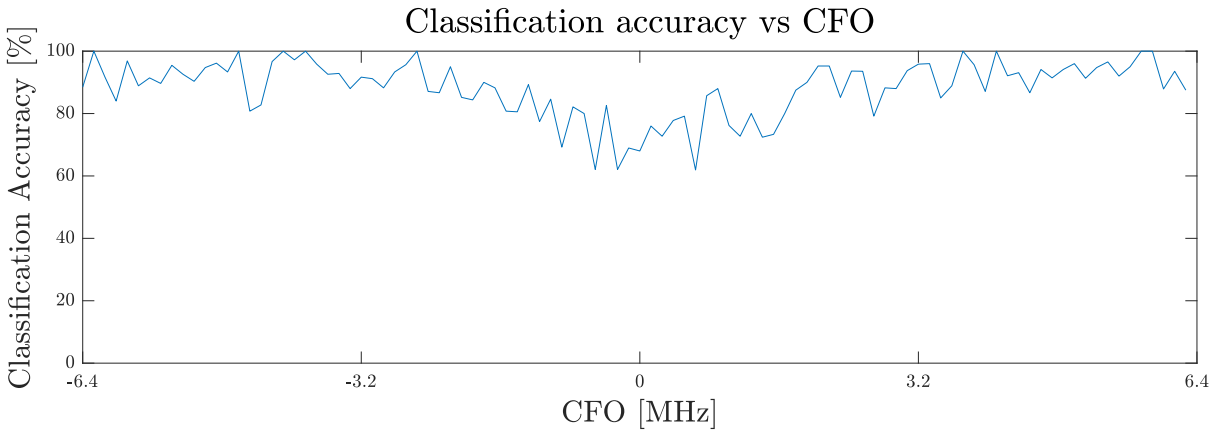


Figure 5.26: *Accuracy vs CFO for the split complex time data classifier*

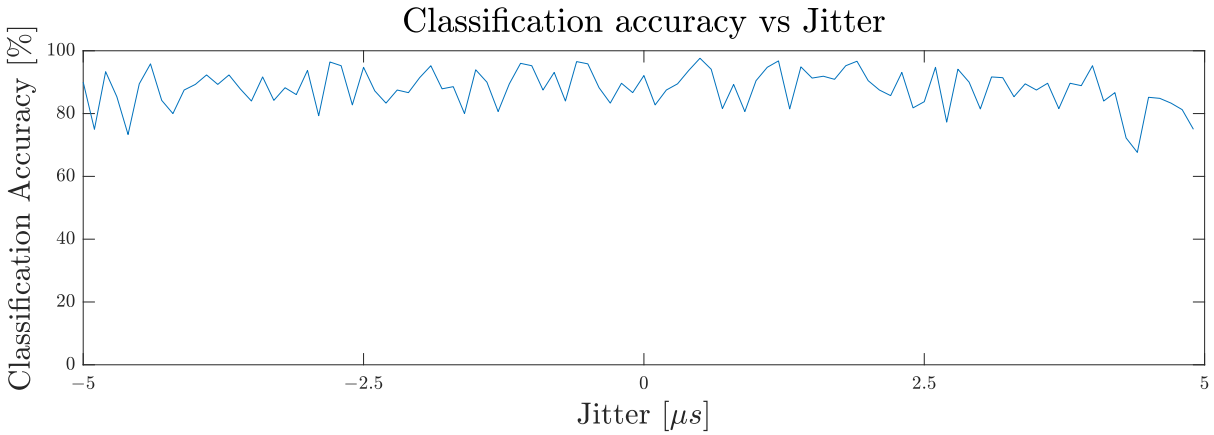


Figure 5.27: *Accuracy vs pulse detection jitter for the split complex time data classifier*

5.6 Low-SNR Analysis

With the exception of the TFR-based CNN classifier, all tested models show a strong dependence on SNR for accuracy. The models were tested with an extended dataset with SNR in the range $[-20, 20]$ dB in order to fully quantify the sensitivity of each model to noise. Figure 5.28 shows the results of these tests on the same axes for comparison.

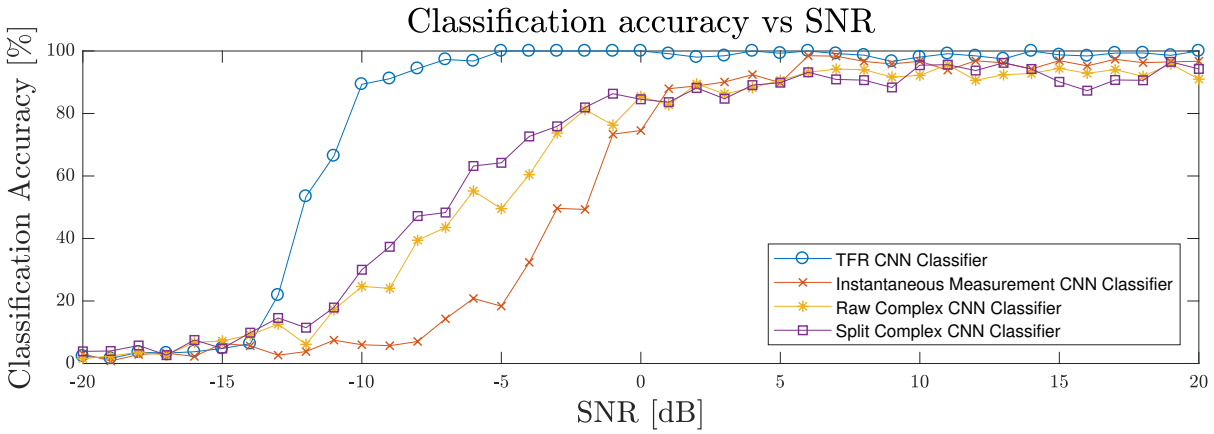


Figure 5.28: *Accuracy vs SNR for all classifiers over extended SNR range*

Figure 5.28 clearly shows that all of the tested models break down as SNR is decreased. In order to better understand these results, the performance of each classifier was tabulated in Table 5.3 using the following metrics:

- Overall average accuracy (Full SNR range)
- Average negative SNR accuracy (SNR < 0 dB)
- Average positive SNR accuracy (SNR ≥ 0 dB)

Table 5.3: *Low-SNR Analysis Results*

Classifier Data Type	Overall (%)	Negative SNR (%)	Positive SNR (%)
TFR Data	78.38	56.82	98.90
Instantaneous Measurement Data	55.74	15.76	93.81
Raw Complex Data	61.39	29.86	91.41
Split Complex Data	63.35	34.47	90.85

The TFR-based CNN classifier clearly exhibited the most resilience to low SNR values, while the other classifiers exhibited rapid performance drop-off as SNR was decreased. Of all the tested classifiers, the instantaneous measurement classifier was the least resilient to noise.

All classifiers achieved accuracies of over 90% for the positive SNR range, but the TFR-based CNN classifier was the best performing model for this range with an accuracy of 98.90%.

The reasoning behind the TFR-based classifier being the most robust can be inferred by visually analysing a piece of example data at various SNRs. A Barker 13 pulse was chosen for this task, but any modulation scheme could have been used. As a preface and a detection benchmark, Figures 5.29, 5.30, and 5.31 show the matched filter responses to a Barker 13 pulse with -20, -10, and 0 dB SNR respectively. These responses show that by using a matched filter, the detection of a Barker 13 pulse in signal with as little as -20 dB SNR is possible, but will likely result in high false alarm rates compared to higher SNR signals due to the low threshold that would be required.

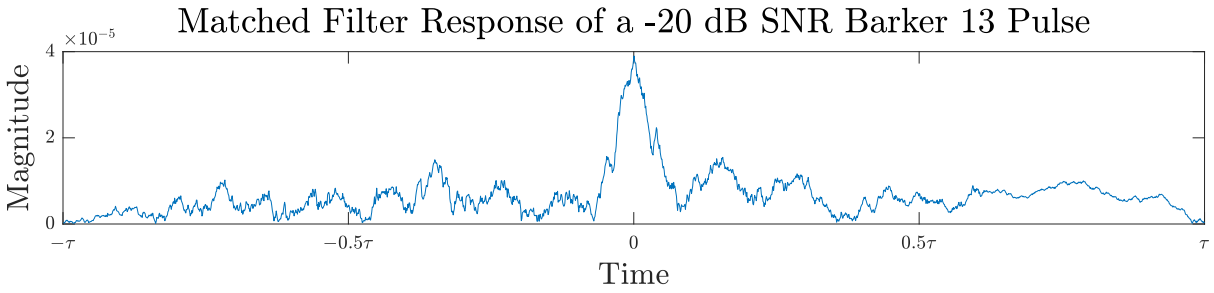


Figure 5.29: *Matched filter response of a Barker 13 pulse with -20 dB SNR*

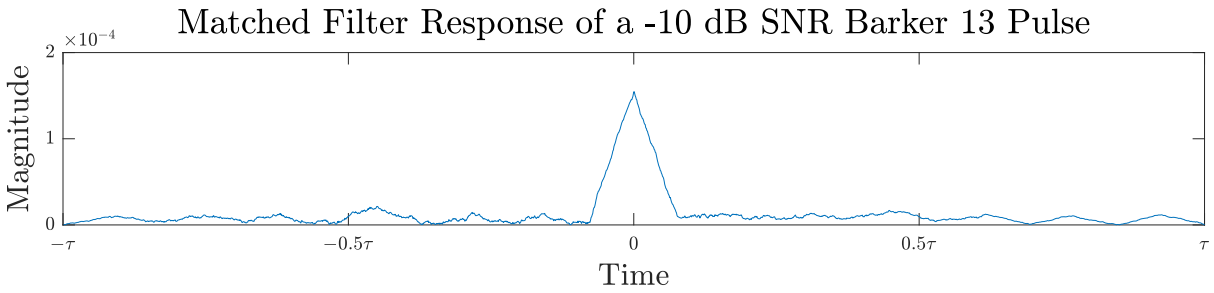


Figure 5.30: *Matched filter response of a Barker 13 pulse with -10 dB SNR*

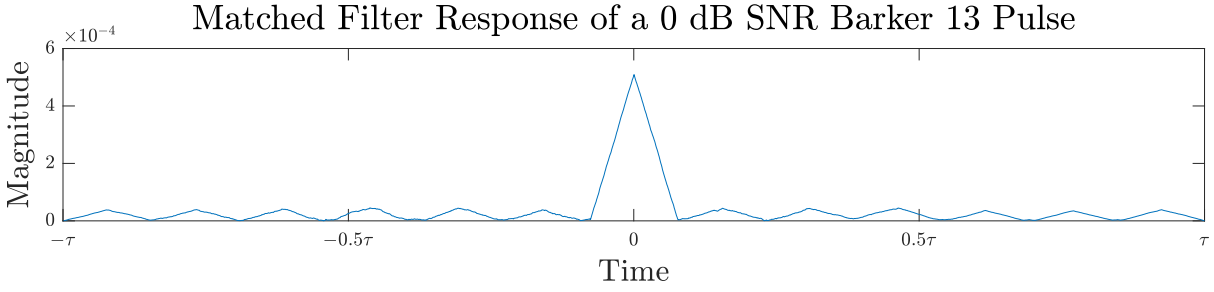
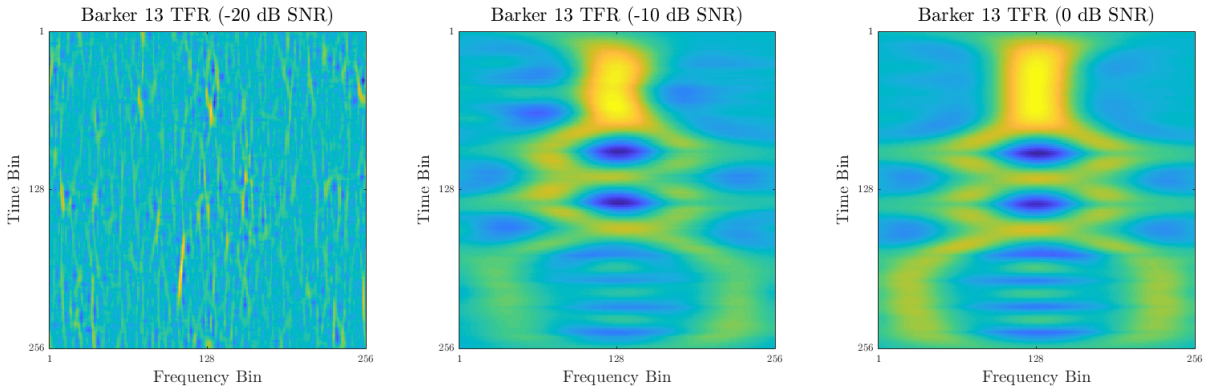


Figure 5.31: *Matched filter response of a Barker 13 pulse with 0 dB SNR*

Figures 5.32a, 5.32b, and 5.32c show the ZAM-GTFRs of a Barker 13 signal with -20, -10, and 0 dB SNR respectively.



(a) *-20 dB SNR* (b) *-10 dB SNR* (c) *0 dB SNR*

Figure 5.32: *Time-frequency representations of a Barker 13 pulse with varying SNR*

The TFR at -20 dB SNR in Figure 5.32a clearly contains no recognisable signal content and will likely be impossible for the TFR-based CNN classifier to process. Conversely, the signal content is clearly visible in the -10 dB SNR case shown in Figure 5.32b. While this signal exhibits some distortion compared to the 0 dB SNR TFR shown in Figure 5.32c, is is visually identifiable as a Barker 13 pulse. These results imply that the TFR-based CNN classifier should experience classification breakdown somewhere in the -20 dB to -10 dB SNR range, which matches the results obtained during low SNR testing.

The instantaneous frequency plots of the Barker 13 signal at -10 dB and 0 dB SNR are shown in Figures 5.33 and 5.34 respectively, and the instantaneous phase plots are shown in Figures 5.35 and 5.36.

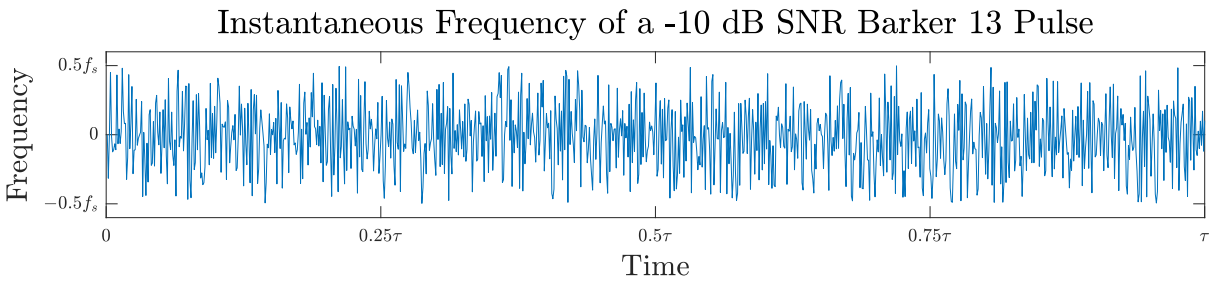


Figure 5.33: *Instantaneous frequency of a Barker 13 pulse with -10 dB SNR*

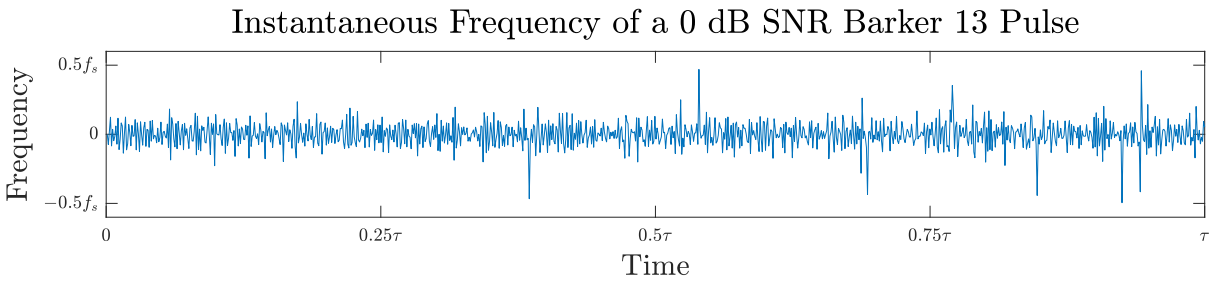


Figure 5.34: *Instantaneous frequency of a Barker 13 pulse with -0 dB SNR*

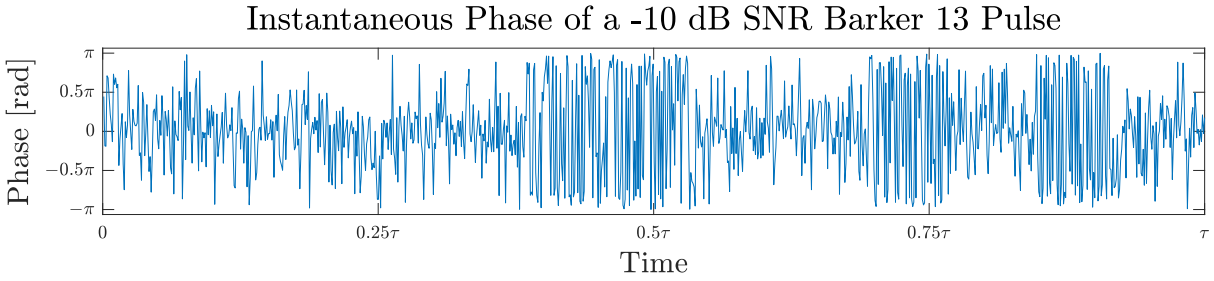


Figure 5.35: *Instantaneous phase of a Barker 13 pulse with -10 dB SNR*

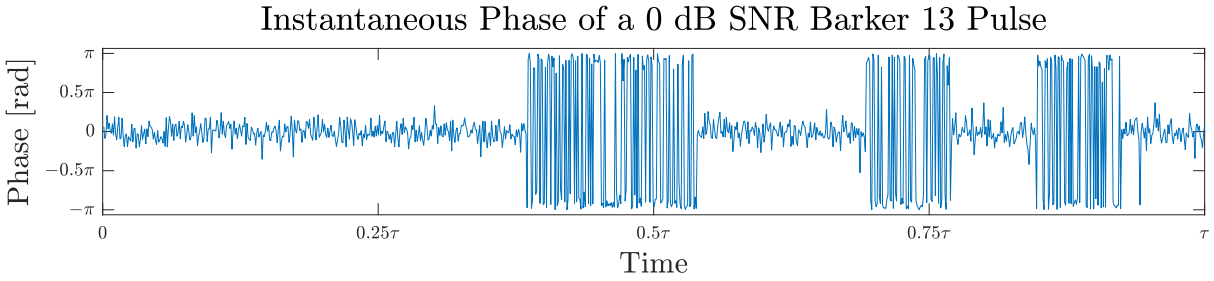


Figure 5.36: *Instantaneous phase of a Barker 13 pulse with -0 dB SNR*

These results show that the instantaneous frequency peaks characteristic to a Barker 13 signal are undetectable at -10 dB SNR signal, and are difficult to detect even at 0 dB SNR. The phase transitions of the Barker 13 pulse are clearly visible in the instantaneous phase for the 0 dB SNR case shown in Figure 5.36, but begin to become highly masked by noise for signals below 0 dB SNR. These results imply that degradation in classification accuracy for the instantaneous measurement CNN classifier can be expected from 0 dB SNR. Once again, this matches the behaviour observed during low SNR testing.

The robustness of the raw and split complex time data classifiers is more difficult to quantify as all preprocessing is performed by the convolutional filters of the models. It is reasonable to assume that, if feature extraction is being performed effectively, the performance degradation due the SNR of the raw and split complex time data classifiers should lie somewhere between that of the TFR-based classifier and the instantaneous measurement classifier.

5.7 Filter Activation Analysis

In order to analyse the convolutional filters generated by the training process and the effects they had on the input complex time data, the outputs of the filters on the first convolutional layer of the raw complex time data classifier were extracted and visualised.

This process was performed on a Barker 13 pulse and a P4 64 pulse generated specifically for this experiment. A Barker 13 pulse was selected for analysis as it contains easily extractable salient features in the form of binary phase transitions, while the P4 64 pulse was selected on account of its polyphase nature (not limited to 180° phase shifts) and the large number of phase transitions in the code.

Carrier frequency offset was deliberately added into the pulses in order to introduce artefacts into the instantaneous phase of the signals. The instantaneous frequency was visualised and compared to the outputs of the filters as it is the the clearest way to visualise the locations and magnitudes of the phase transitions in these pulses.

5.7.1 Barker 13 Analysis

The parameters of the Barker 13 pulse under test are listed below:

- SNR: 10 dB
- CFO: 3.2 MHz ($\frac{f_s}{8}$)
- Jitter: 0 μs

Figure 5.37 shows the instantaneous frequency plot of the Barker 13 pulse, clearly showing the peaks corresponding to 180° phase transitions.

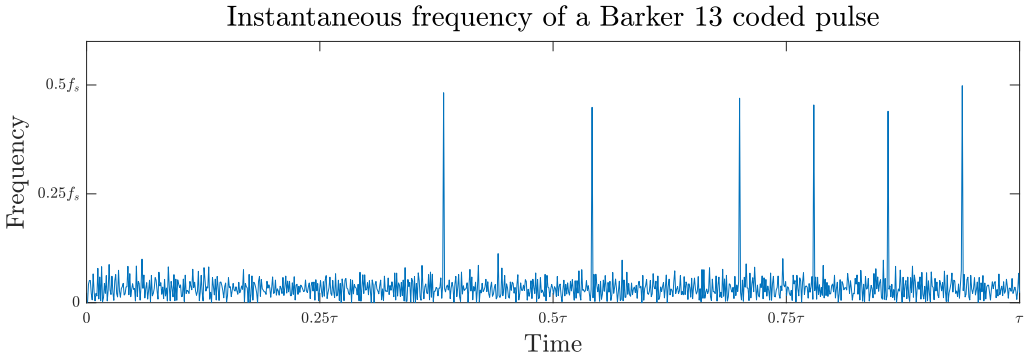


Figure 5.37: *Instantaneous frequency of the Barker 13 pulse under analysis*

Figures 5.38 to 5.41 show the output of convolutional filters 1 through 4 on the first convolutional layer when applied to the above pulse. Note the presence of spikes in all of the filter outputs at the same times as the phase transitions in the original pulse. It appears that filter 2, shown in Figure 5.39, shows the clearest correlation to the instantaneous frequency of the pulse.

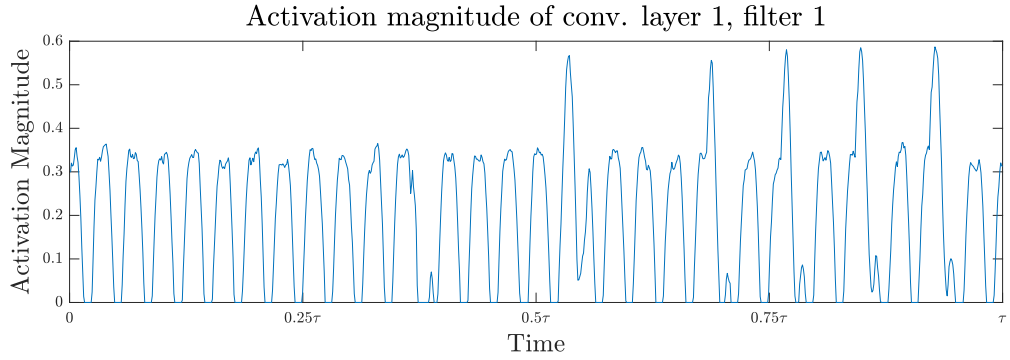


Figure 5.38: *Convolution output of layer 1, filter 1 in the presence of a Barker 13 pulse*

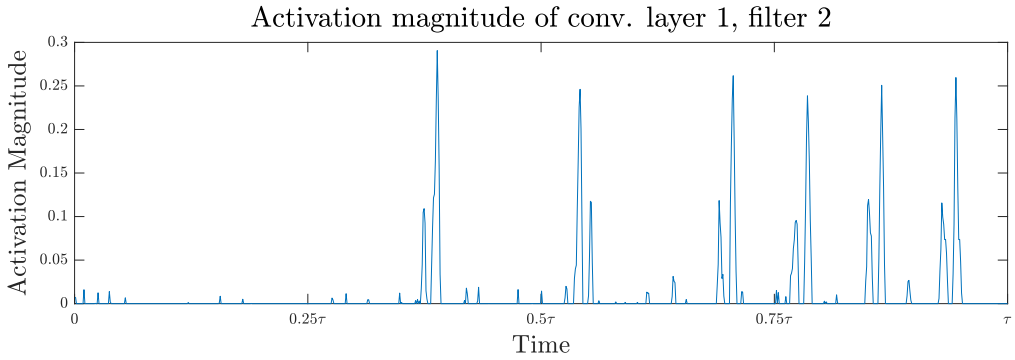


Figure 5.39: *Convolution output of layer 1, filter 2 in the presence of a Barker 13 pulse*

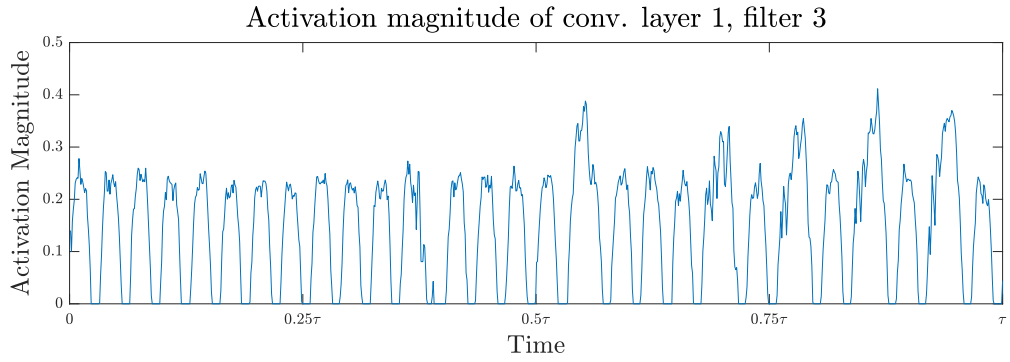


Figure 5.40: *Convolution output of layer 1, filter 3 in the presence of a Barker 13 pulse*

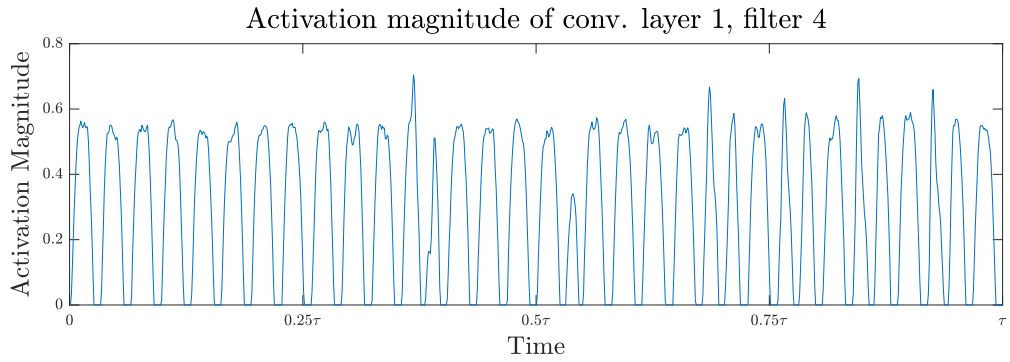


Figure 5.41: *Convolution output of layer 1, filter 4 in the presence of a Barker 13 pulse*

5.7.2 P4 64 Analysis

The parameters of the P4 64 pulse under test are listed below:

- SNR: 10 dB
- CFO: 3.2 MHz ($\frac{f_s}{8}$)
- Jitter: 0 μs

Figure 5.42 shows the instantaneous frequency plot of the P4 64 pulse. There are clear peaks at the locations of the phase transitions, although they are not uniform in size due to the fact that the phase transitions in the code vary in magnitude.

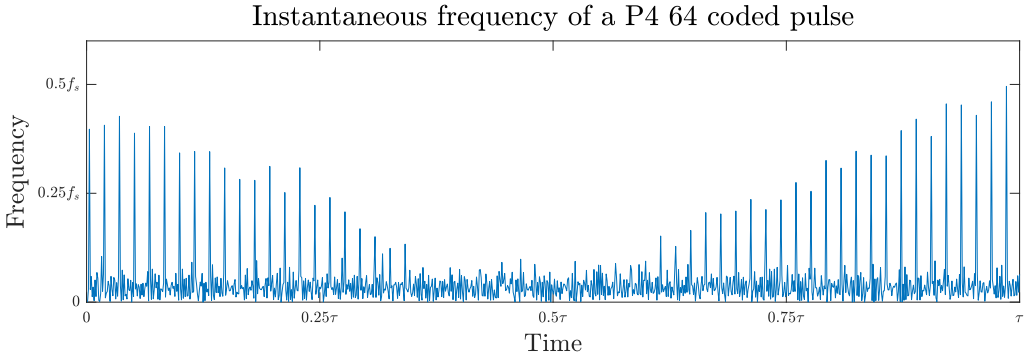


Figure 5.42: *Instantaneous frequency of the P4 64 pulse under analysis*

Figures 5.43 to 5.46 show the output of convolutional filters 1 through 4 on the first convolutional layer when applied to the P4 64 pulse. As observed previously, there are clear peaks appearing throughout the filter activations, corresponding in time to the locations of the phase transitions. This effect is noticeably more masked than for the Barker 13 case due to the sheer number of phase transitions. Once again, it appears that the second filter (Figure 5.44) shows the clearest correlation to the instantaneous frequency of the pulse.

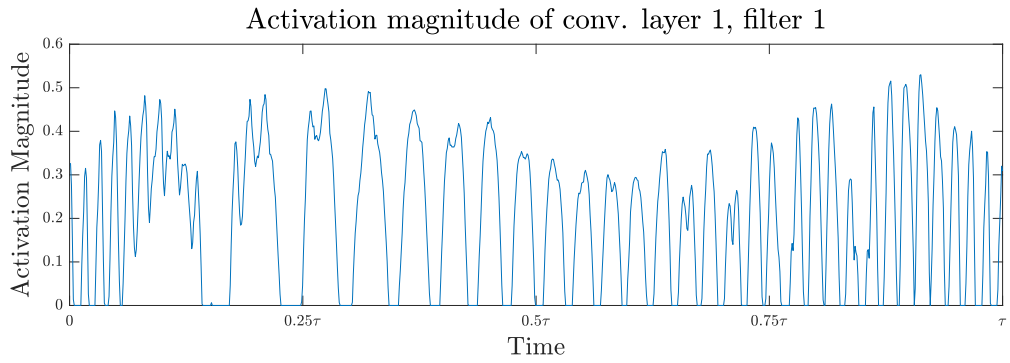


Figure 5.43: *Convolution output of layer 1, filter 1 in the presence of a P4 64 pulse*

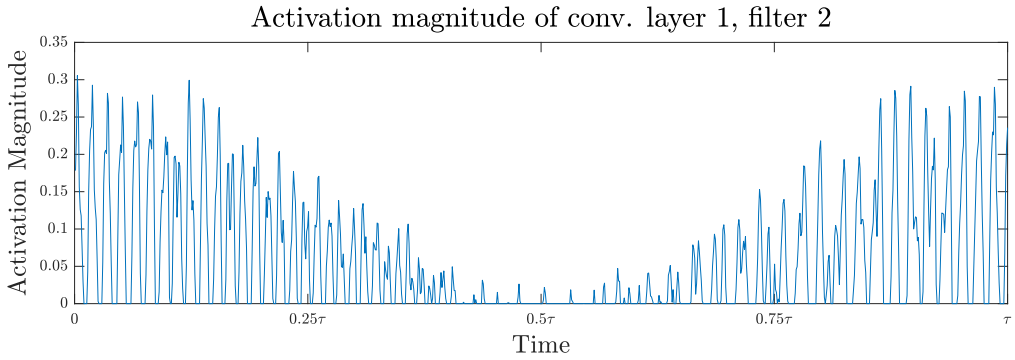


Figure 5.44: *Convolution output of layer 1, filter 2 in the presence of a P4 64 pulse*

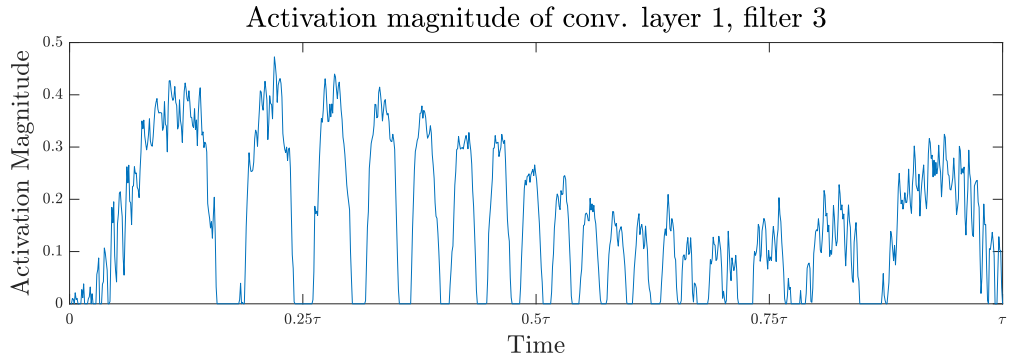


Figure 5.45: *Convolution output of layer 1, filter 3 in the presence of a P4 64 pulse*

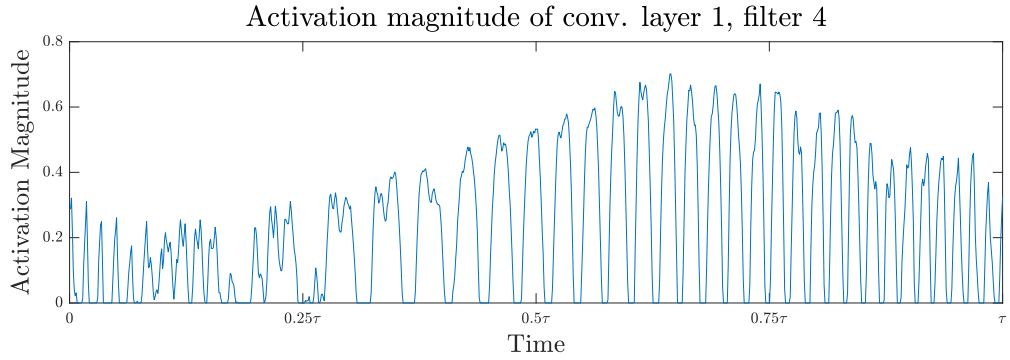


Figure 5.46: *Convolution output of layer 1, filter 4 in the presence of a P4 64 pulse*

5.8 Complex Time Data Reproduction

The gradient ascent optimisation method described in Section 4.9.1 was applied to the raw complex time data classifier. For the sake of brevity and consistency with prior experiments, this experiment was only performed on two classes - Barker 13 and P4 order 64.

5.8.1 Barker 13 Analysis

5.8.1.1 Instantaneous Measurements

Figures 5.47, 5.48, and 5.49 show the comparisons between the instantaneous magnitude, frequency, and phase of the reference and "optimal" Barker 13 signals. Note that "optimal" in this sense refers to the fact that the signal was generated through the process of gradient ascent optimisation, and has no relation to the performance or feasibility of the signal as a radar waveform.

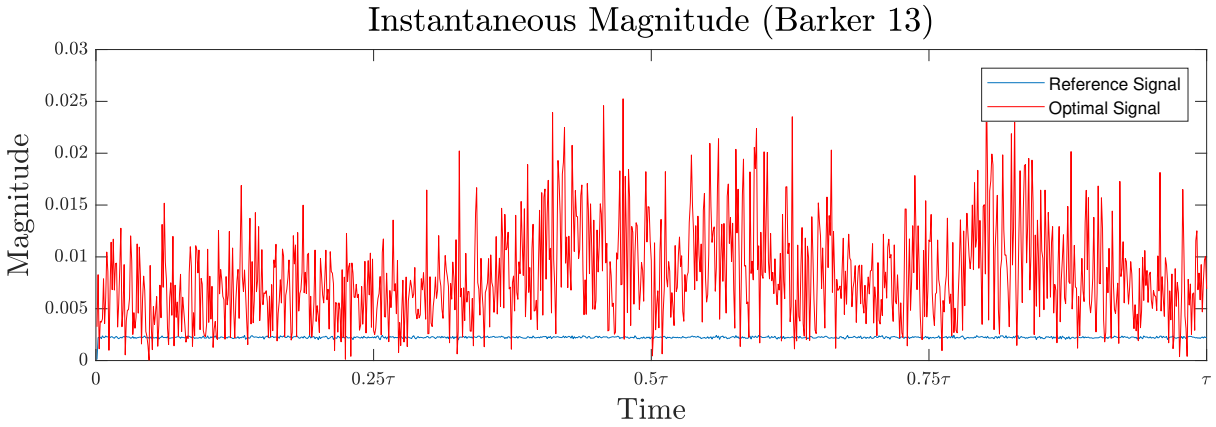


Figure 5.47: *Reference vs. optimal instantaneous magnitude comparison (Barker 13)*

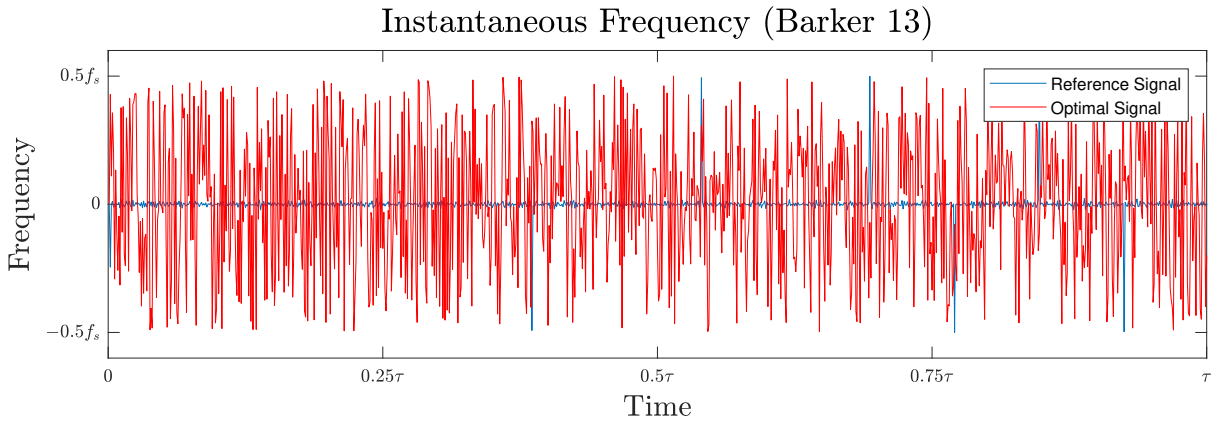


Figure 5.48: *Reference vs. optimal instantaneous frequency comparison (Barker 13)*

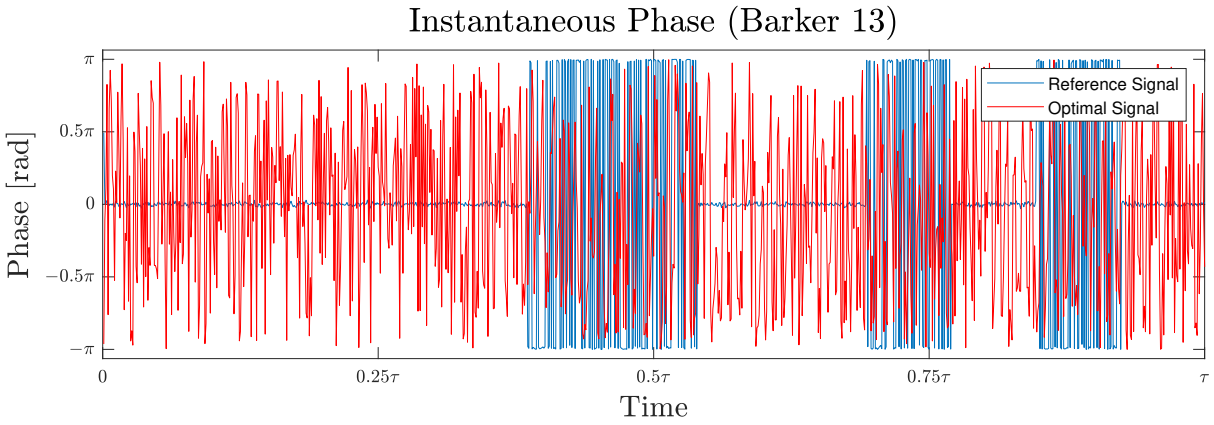
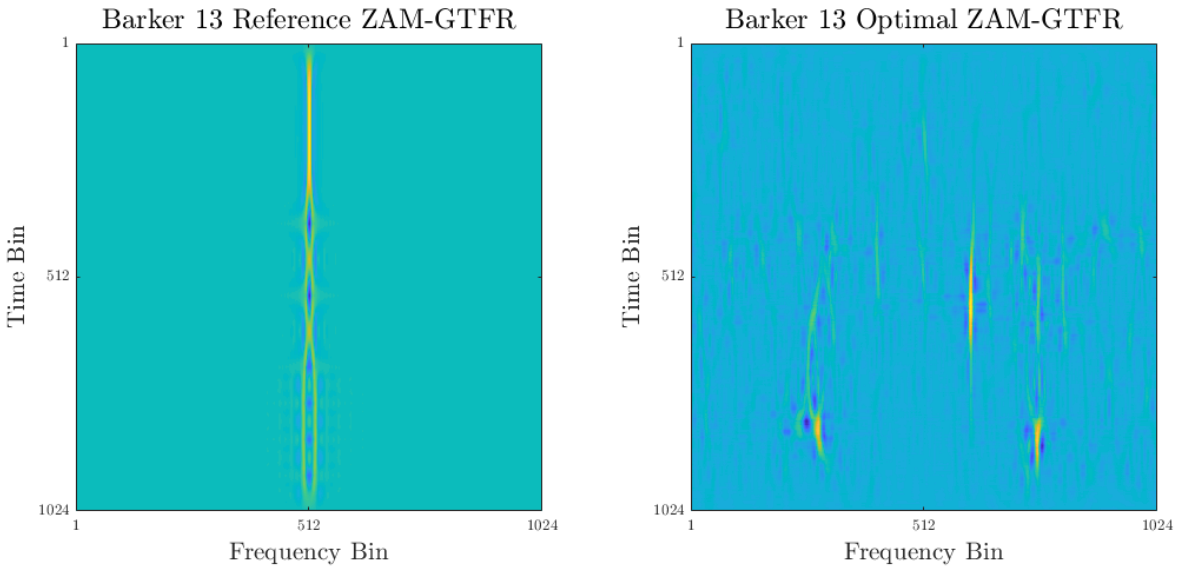


Figure 5.49: *Reference vs. optimal instantaneous phase comparison (Barker 13)*

Comparing the signals in this manner shows that there is no visual correlation between the reference and optimal instantaneous magnitude, frequency and phase plots. The optimal signal appears to be almost entirely noise-like, with a small amount of variance in signal magnitude across the duration of the pulse. Even though the new waveform bears no resemblance to the original signal, feeding the optimal signal into the input of the classifier results in the class being correctly classified with near 100% confidence.

5.8.1.2 Time-Frequency Representation

Figure 5.50 presents a side-by-side view of the ZAM-GTFRs of the reference and optimal Barker 13 signals. No cropping or scaling has been performed on these images.



(a) *Barker 13 reference TFR*

(b) *Barker 13 optimal TFR*

Figure 5.50: *Time-frequency representations of reference and optimal Barker 13 signals*

There are no visual similarities between the reference and optimal TFRs. While the signal in the reference TFR has a clear structure, the signal in the optimal TFR appears to be noise-like with three main nodes of "activity".

5.8.2 P4 Order 64 Analysis

5.8.2.1 Instantaneous Measurements

Figures 5.51, 5.52, and 5.53 show the instantaneous magnitude, frequency, and phase plots of the reference and "optimal" P4 order 64 signals overlaid.

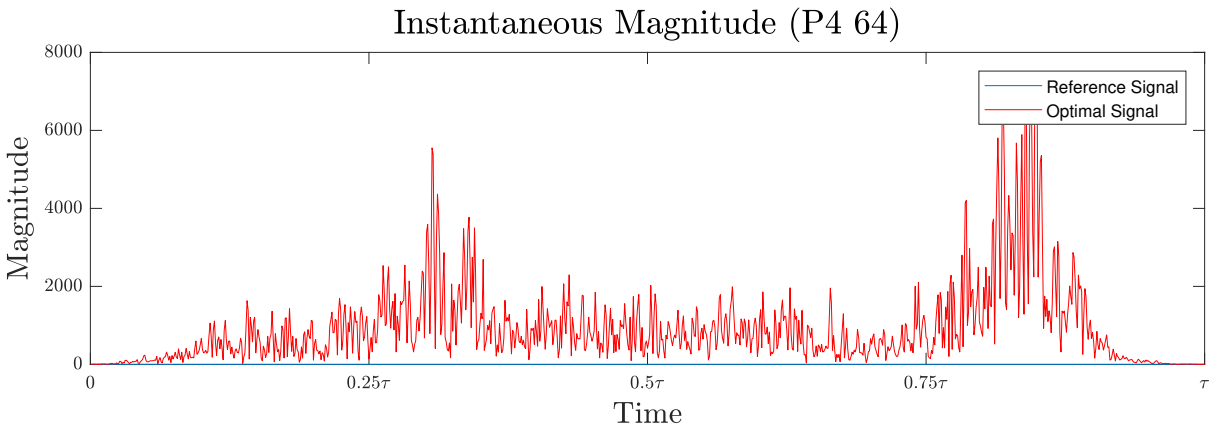


Figure 5.51: *Reference vs. optimal instantaneous magnitude comparison (P4 64)*

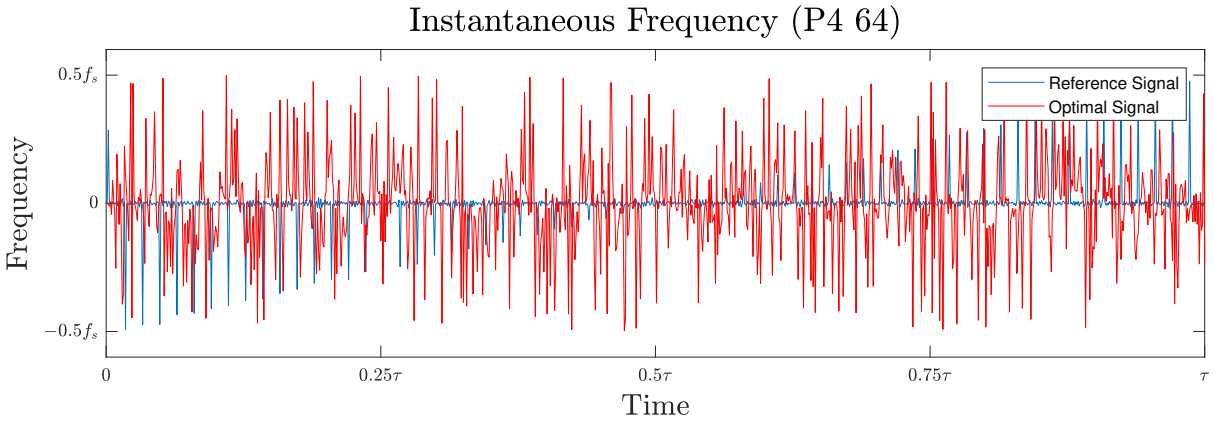


Figure 5.52: *Reference vs. optimal instantaneous frequency comparison (P4 64)*

As was the case for the Barker 13 signal, there is no clear correlation between the reference and optimal signals in any of these plots. The optimal signal appears to be noise-like apart from the variation in magnitude, but results in correct classification with extremely high confidence when fed back to the input of the classifier.

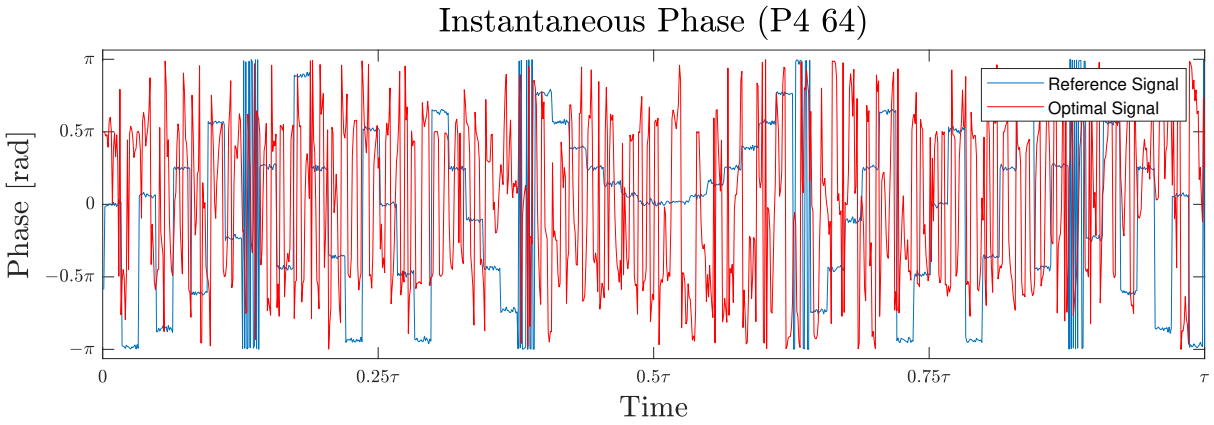
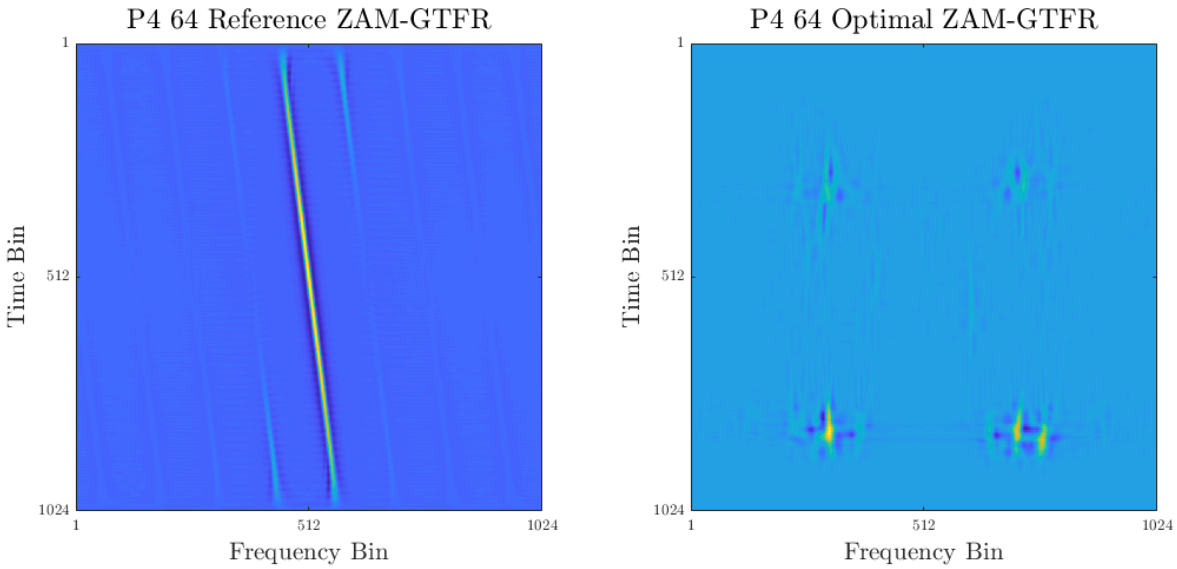


Figure 5.53: *Reference vs. optimal instantaneous phase comparison (P4 64)*

5.8.2.2 Time-Frequency Representation

Figure 5.54 presents a side-by-side view of the ZAM-GTFRs of the reference and optimal P4 order 64 signals. Once again, these images are uncropped and unscaled.



(a) *P4 64 reference TFR*

(b) *P4 64 optimal TFR*

Figure 5.54: *Time-frequency representations of reference and optimal P4 64 signals*

The TFR of the optimal signal is slightly more structured than that of the Barker 13 pulse, but still bears no resemblance to the TFR of the reference signal.

The approach described in this section is extremely similar to some of the approaches to generating adversarial examples as discussed Section 2.5.6, and has shown the same results - a set of examples that bear no visual similarity to the real class, but that are classified with extremely high confidence by the model.

5.9 Antenna to Antenna Experiment

As described in Section 4.10, an experiment was performed whereby pulses from each class were transmitted over a real radio link and sampled before being classified. The aim of this experiment was to observe the robustness of the various classifiers and data representations to real-world signal imperfections.

5.9.1 Data Visualisation and Comparison

The time-frequency representations and instantaneous measurements of the received pulses were visualised and compared to their simulated counterparts in order to visually quantify the effects of the real-world radio link. Note that all time-frequency representations compared in this section have already undergone the cropping and resizing operations described in Section 4.3.

For the sake of brevity, only three classes are visualised in this section - one frequency modulated class (LFM), one binary phase modulated class (Barker 13) and one polyphase modulated class (P4 order 64).

5.9.1.1 Linear Frequency Modulation

Figure 5.55 shows the TFRs of the simulated and received LFM pulses. The TFR of the received signal clearly exhibits little to no degradation as a result of the radio link, and is visually identical to the TFR of the simulated signal.

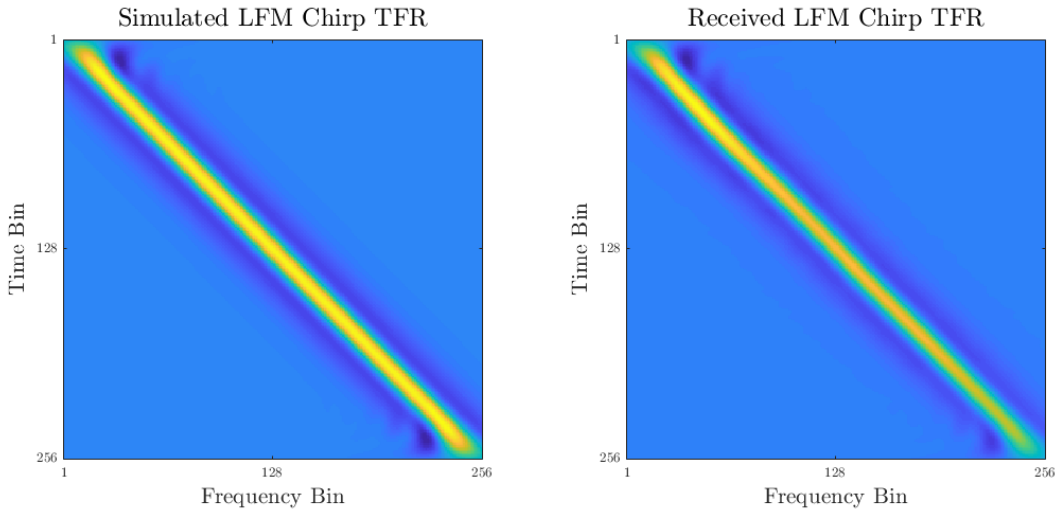


Figure 5.55: *Time-frequency representations of simulated and received LFM signals*

Figures 5.56 and 5.57 show the instantaneous magnitudes of the simulated and received LFM pulses. The level difference between the two signals is expected as the level of the received signal is a product of the transmitter power, antenna gains, and antenna separation, while the simulated level was chosen arbitrarily for visualisation. The received signal exhibits a degree of magnitude instability over the pulse duration.

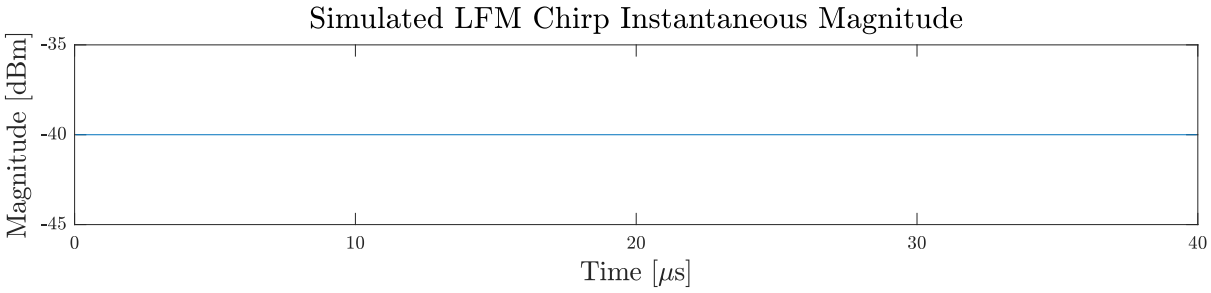


Figure 5.56: *Instantaneous magnitude of simulated LFM signal*

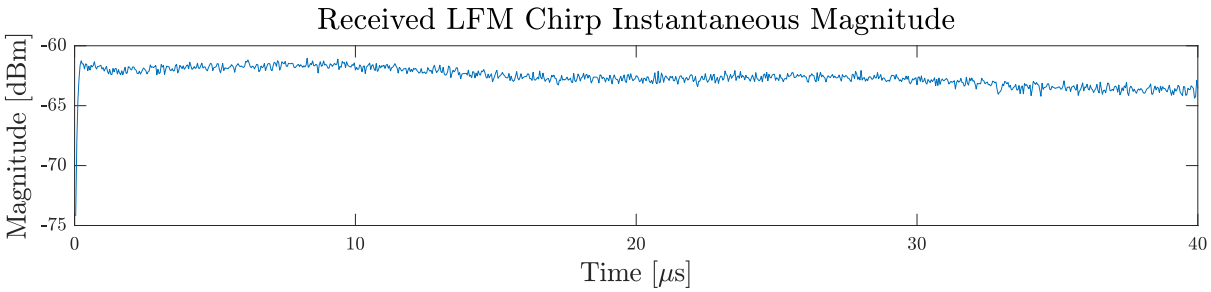


Figure 5.57: *Instantaneous magnitude of received LFM signal*

The instantaneous frequency plots in Figures 5.58 and 5.59 show that both pulses sweep over the same frequency range, but the sweep is noticeably noisy in the received signal.

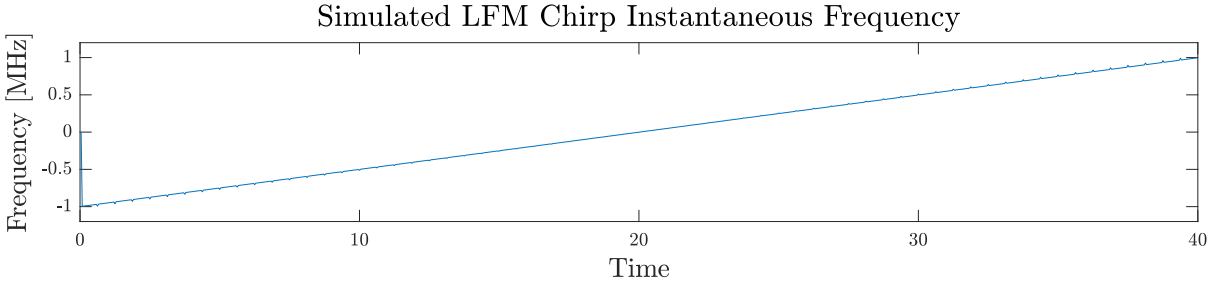


Figure 5.58: *Instantaneous frequency of simulated LFM signal*

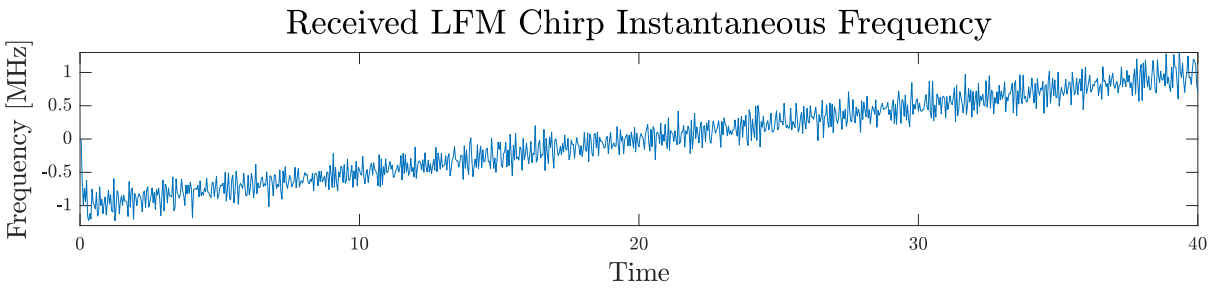


Figure 5.59: *Instantaneous frequency of received LFM signal*

The instantaneous phase plots of the simulated and received signals are shown in Figures 5.60 and 5.61. These are visually identical apart from a different starting phase in the received signal. This starting phase is expected to be be random in real-world environments.

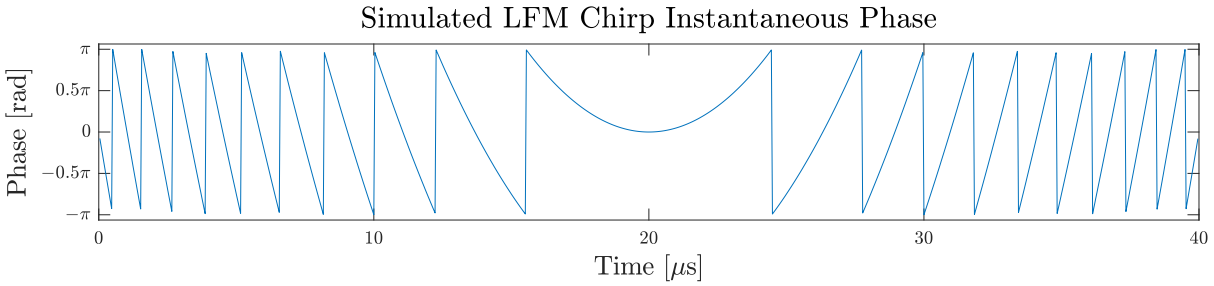


Figure 5.60: *Instantaneous phase of simulated LFM signal*

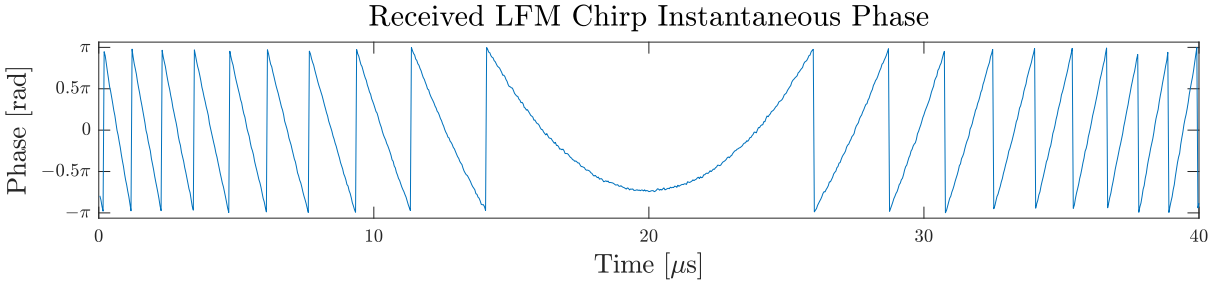


Figure 5.61: *Instantaneous phase of received LFM signal*

5.9.1.2 Barker 13

Figure 5.62 shows that, as was the case for the LFM signals, the TFR of the received Barker 13 signal is visually identical to that of the simulated signal.

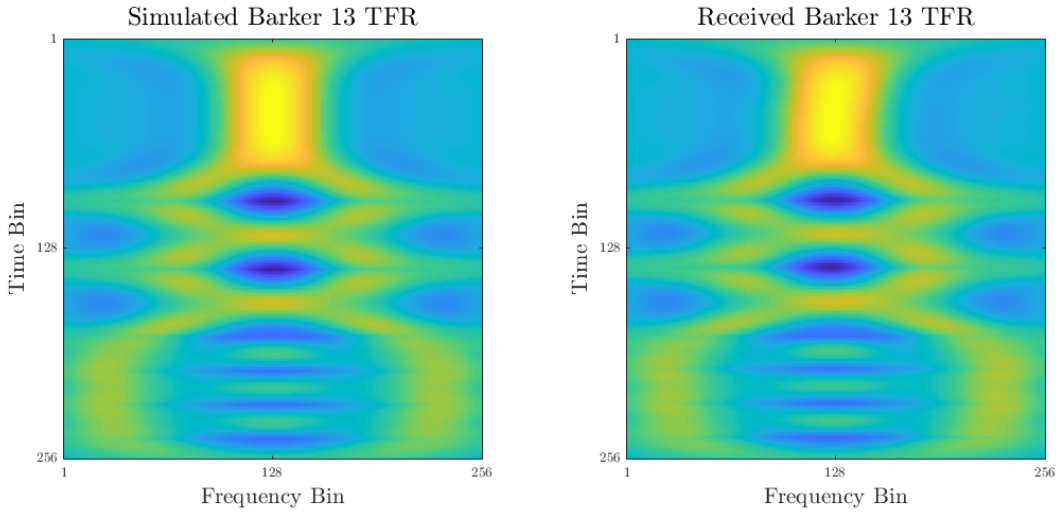


Figure 5.62: *Time-frequency representations of simulated and received Barker 13 signals*

Figures 5.63 and 5.64 show clear artefacts in the instantaneous magnitude of the received Barker 13 pulse in the form of negative peaks at the locations of the phase transitions. These artefacts are not present in the simulated signal, and are most likely a by-product of the amplifier in the transmitter.

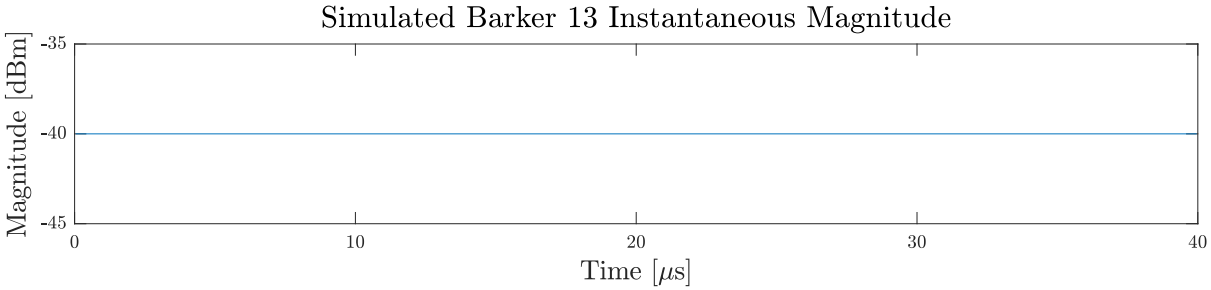


Figure 5.63: *Instantaneous magnitude of simulated Barker 13 signal*

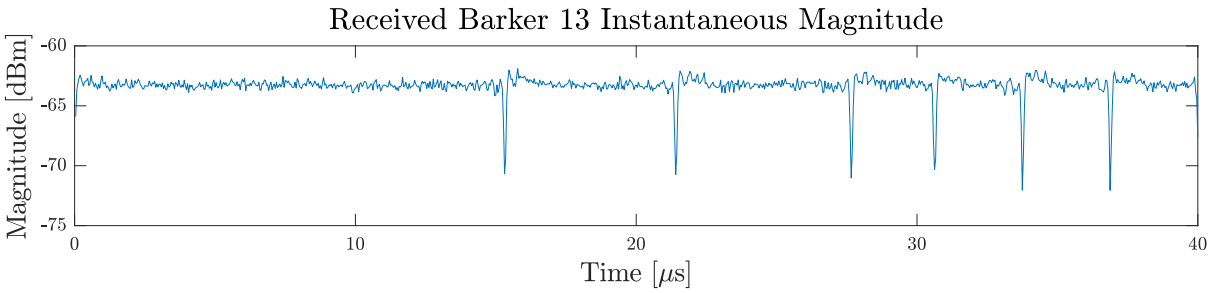


Figure 5.64: *Instantaneous magnitude of received Barker 13 signal*

Figures 5.65 and 5.66 show the instantaneous frequency plots of the simulated and received Barker 13 signals, while Figures 5.67 and 5.68 show their instantaneous phase.

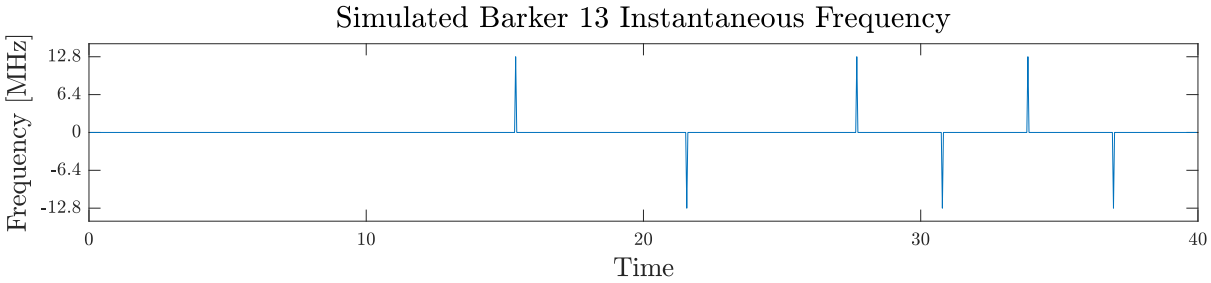


Figure 5.65: *Instantaneous frequency of simulated Barker 13 signal*

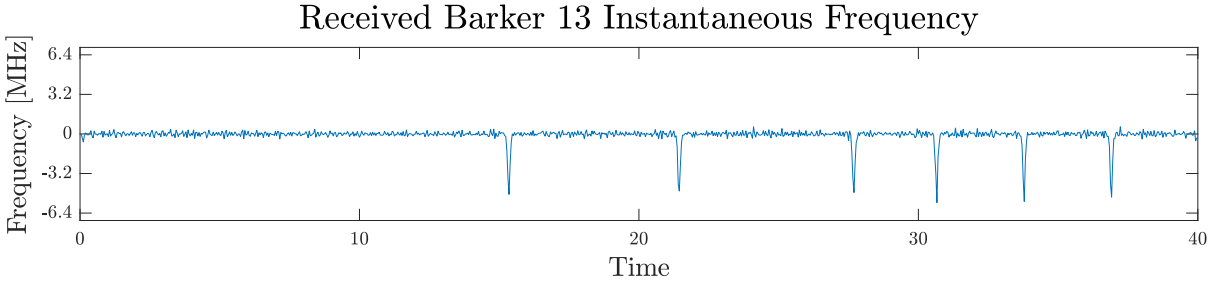


Figure 5.66: *Instantaneous frequency of received Barker 13 signal*

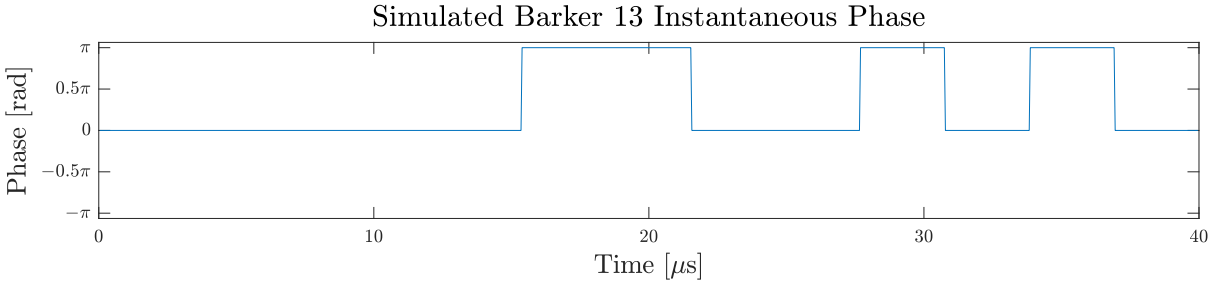


Figure 5.67: *Instantaneous phase of simulated Barker 13 signal*

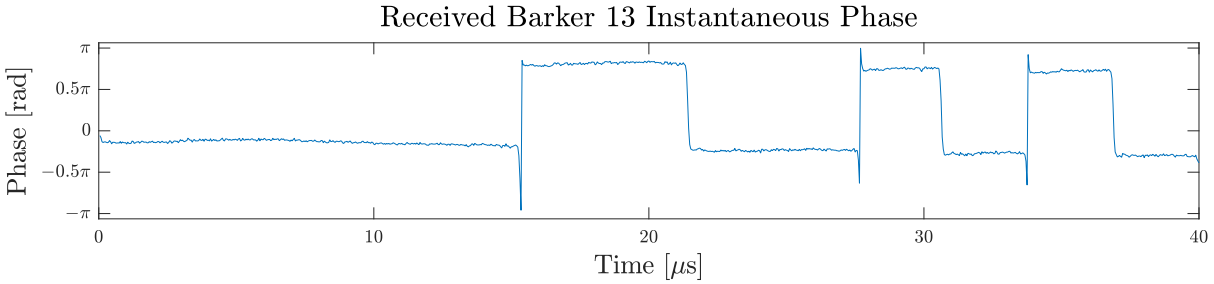


Figure 5.68: *Instantaneous phase of received Barker 13 signal*

The difference in the magnitude of instantaneous frequency peaks between the simulated and received signals can be attributed to the fact that the simulated phase transitions were instantaneous (infinite slew rate), while in reality the finite slew rate of the amplifier causes the phase transitions to be spread across more than one sample. The fact that the phase shifts shown Figure 5.68 are slightly less than the nominal π radians will also cause a reduction in the magnitude of these peaks. A π radian phase shift corresponds to an instantaneous frequency peak of $\frac{f_s}{2}$, while the received instantaneous frequency peaks appear to have magnitudes of approximately 5.5 MHz - slightly less than $\frac{f_s}{4}$ where f_s is 25.6 MHz. This value aligns with the hypothesis that the phase transitions are slightly less than the ideal π degrees and are spread over two samples.

Figure 5.68 shows clear transients on the leading edge of each phase transition. These leading-edge transients are responsible for the difference in direction between the instantaneous frequency peaks of the simulated and received signals.

5.9.1.3 P4 Order 64

Figure 5.69 shows the TFRs of the simulated and received P4 order 64 pulses. It appears that the TFR of the received pulse is slightly misaligned in frequency, resulting in the lower frequency component at the end of the signal (bottom-left of the TFR) being cut off. This is likely due to the cropping and resizing operations described in Section 4.3.

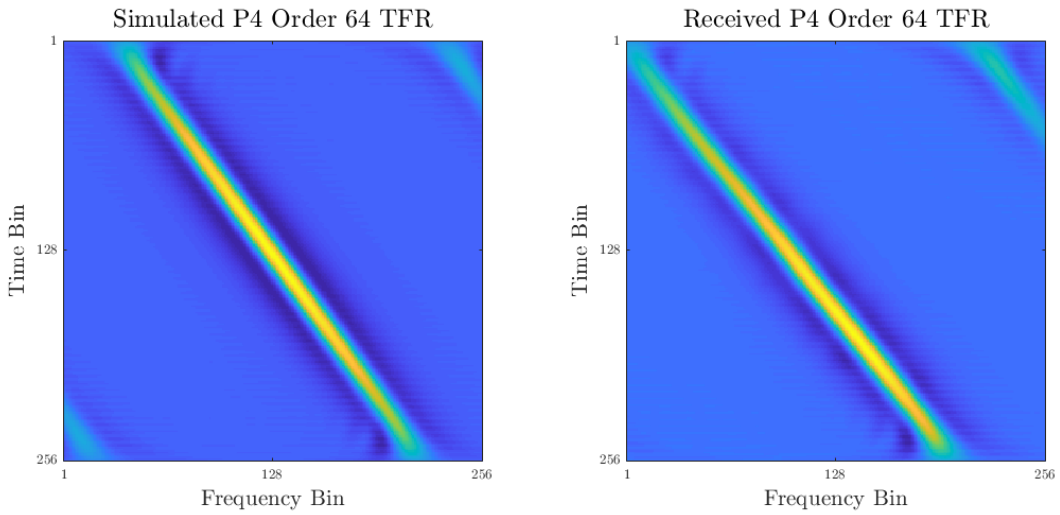


Figure 5.69: *Time-frequency representations of simulated and received P4 64 signals*

As observed in the Barker 13 pulse in Section 5.9.1.2, the instantaneous magnitude plots in Figures 5.70 and 5.71 show the presence of artefacts in the received signal at the locations of phase transitions.

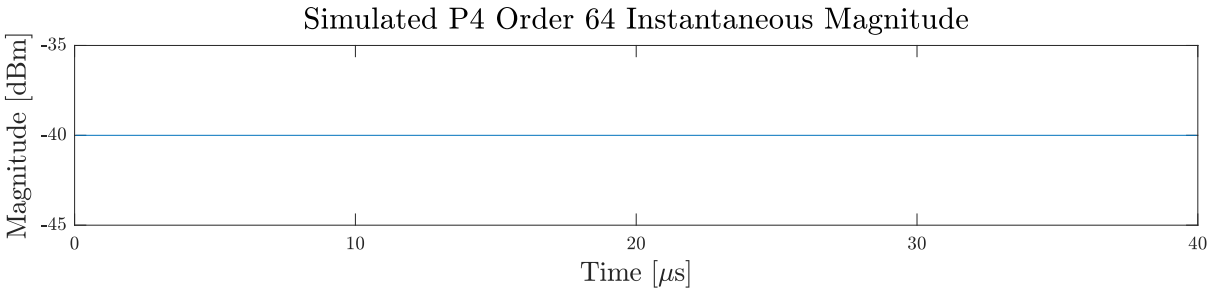


Figure 5.70: *Instantaneous magnitude of simulated P4 64 signal*

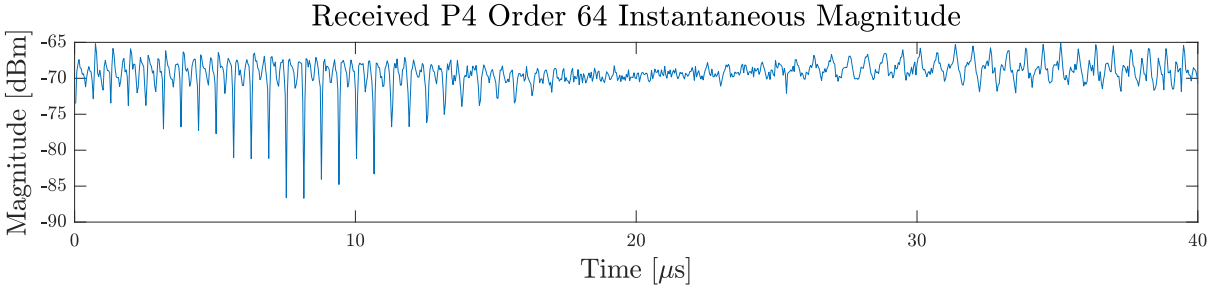


Figure 5.71: *Instantaneous magnitude of received P4 64 signal*

Figures 5.72 and 5.73 show that the received signal once again has smaller instantaneous frequency peaks than the simulated signal. The received signal does not have monotonically increasing instantaneous frequency peaks, and contains a number of outliers that appear to correspond to instantaneous phase transients in Figure 5.75.

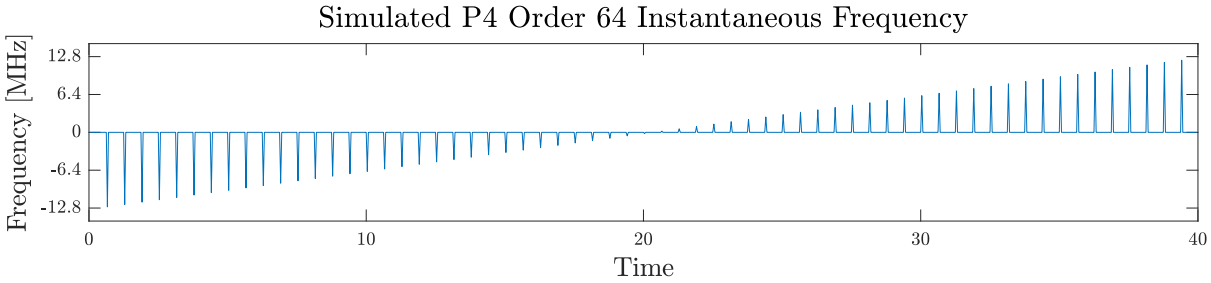


Figure 5.72: *Instantaneous frequency of simulated P4 64 signal*

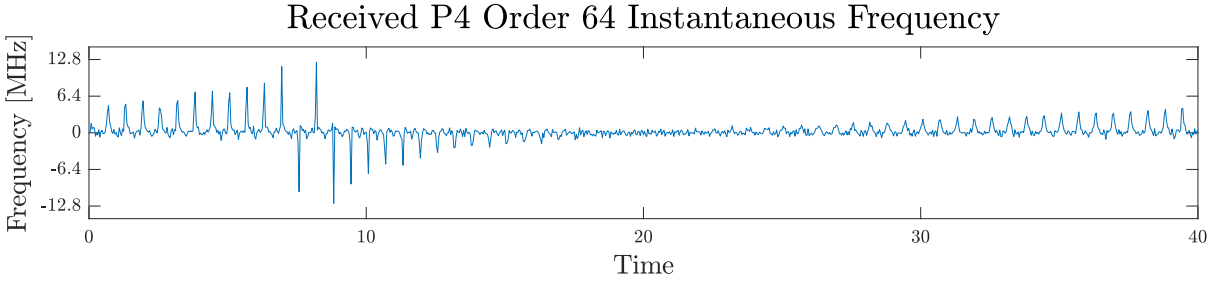


Figure 5.73: *Instantaneous frequency of received P4 64 signal*

As mentioned, the phase of the received signal in Figure 5.75 appears to contain large transients on the leading edges of certain phase transitions. The phase of the received signal in the periods between transitions also appears to have a degree of instability.

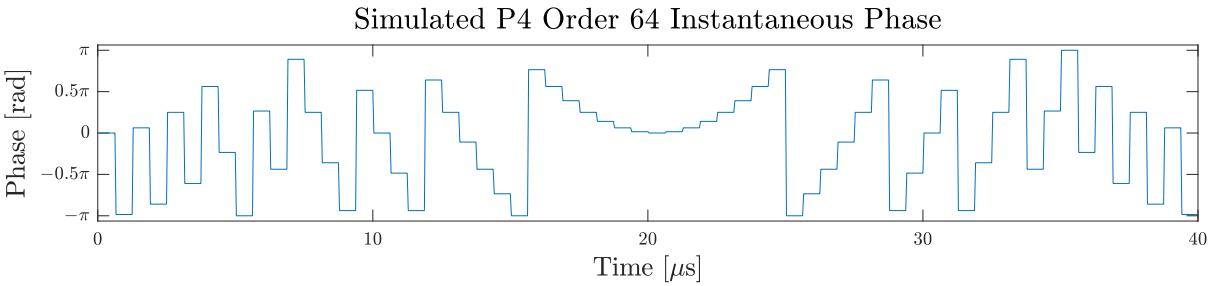


Figure 5.74: *Instantaneous phase of simulated P4 64 signal*

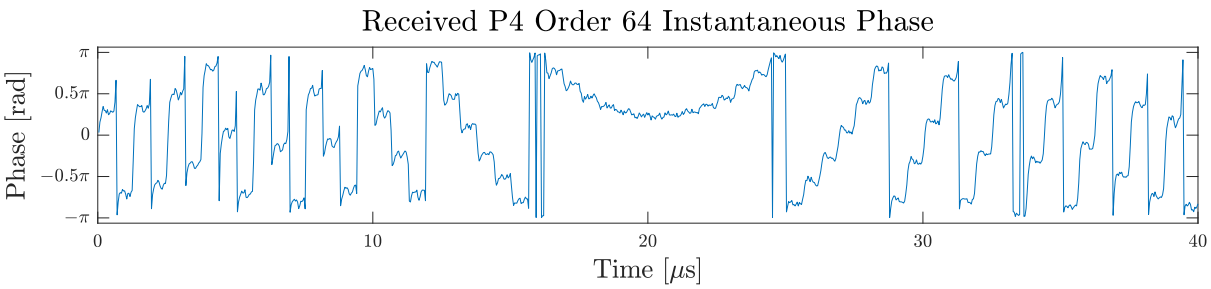


Figure 5.75: *Instantaneous phase of received P4 64 signal*

5.9.2 Classification Performance

The performance of each classifier on the received data is summarised in Table 5.4.

Table 5.4: *Antenna to Antenna Experiment Classification Performance*

Classifier Data Type	Accuracy (%)	Count
TFR Data	92.86	26/28
Instantaneous Measurement Data	82.14	23/28
Raw Complex Data	42.86	12/28
Split Complex Data	53.57	15/28

The following subsections will analyse the performance of each classifier and attempt to relate the results to the data type under test and how it was affected by the radio link.

5.9.2.1 TFR-Based CNN Classifier

The TFR-based classifier was the best performing classifier in the antenna to antenna experiment, with an accuracy of 92.86%. The confusion matrix for this classifier is shown in Figure 5.76.

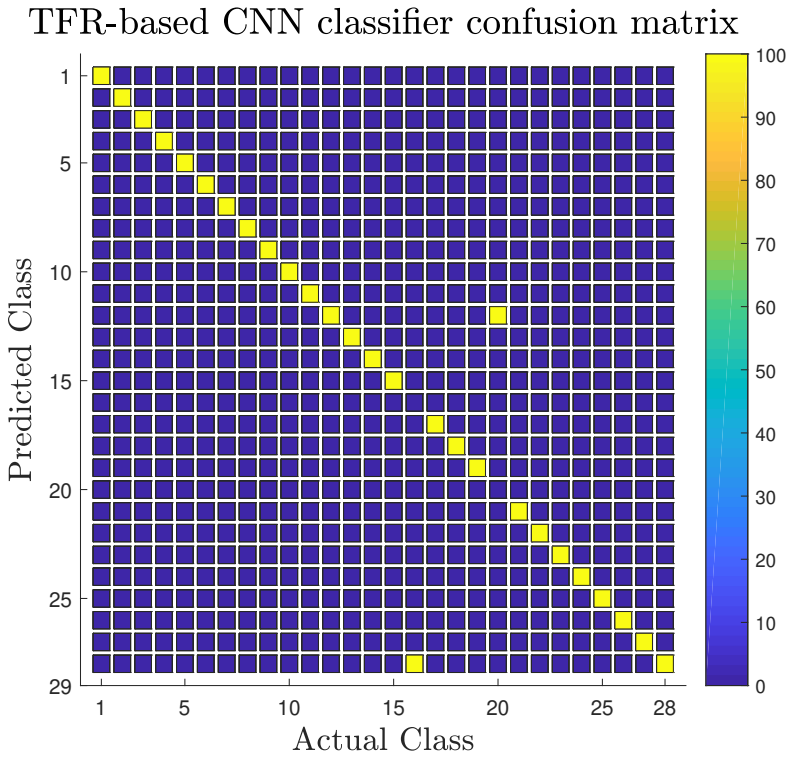


Figure 5.76: *Confusion matrix for the TFR-based CNN classifier in the antenna to antenna test*

The confusion matrix shows that classes 16 and 20 were misclassified as classes 28 and 12 respectively. These classes are P1 order 8, which was misclassified as P4 order 64, and P2 order 8, which was misclassified as a Frank code of order 8. Both of the misclassified classes are high order polyphase codes, and were misclassified as other high order polyphase codes. It is interesting to note that the two misclassifications were uni-directional. That is, while the P1 order 8 pulse was misclassified as a P4 order 64 pulse, the P4 order 64 pulse was classified correctly. The same applies to the P2 order 8 and the Frank code order 8 pulses.

Figure 5.77 shows the TFRs of the P1 order 8 and P4 order 64 pulses that were fed into the classifier. There are some clear similarities between the images - both TFRs show the primary signal component as being a positively swept signal over time, and both TFRs have minor signal components in the corners of the image. The TFRs differ in that the P1 order 8 TFR possesses periodic nulls and periods of amplitude variation, while the P4 order 64 TFR smoothly ramps up to a maximum amplitude before ramping down again.

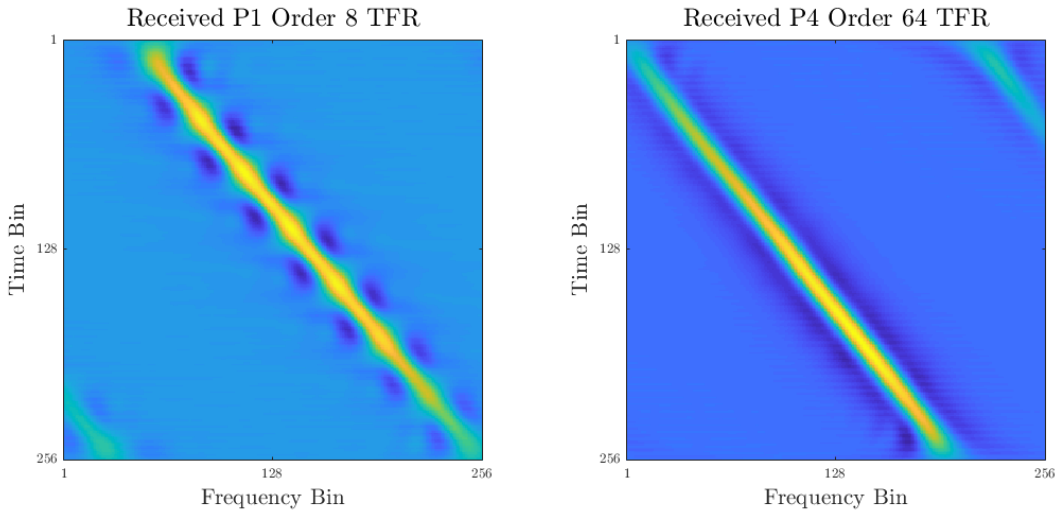


Figure 5.77: *Time-frequency representations the received P1 order 8 and P4 order 64 pulses*

Figure 5.78 shows the TFRs of the P2 order 8 and Frank code order 8 pulses that were fed to the classifier. The primary signal component of the P2 order 8 signal appears to disperse towards the end of the pulse, while the Frank code of order 8 appears to be comprised of two primary components, both of which maintain a chirp-like structure throughout pulse.

The confusion in classification may stem from the upper-right hand region (positive frequency, first half of the pulses) of both TFRs, where the two signals are visually extremely similar. In this region, both signals exhibit amplitude variations and periodic nulls, similar to what was seen in the P1 order 8 signal.

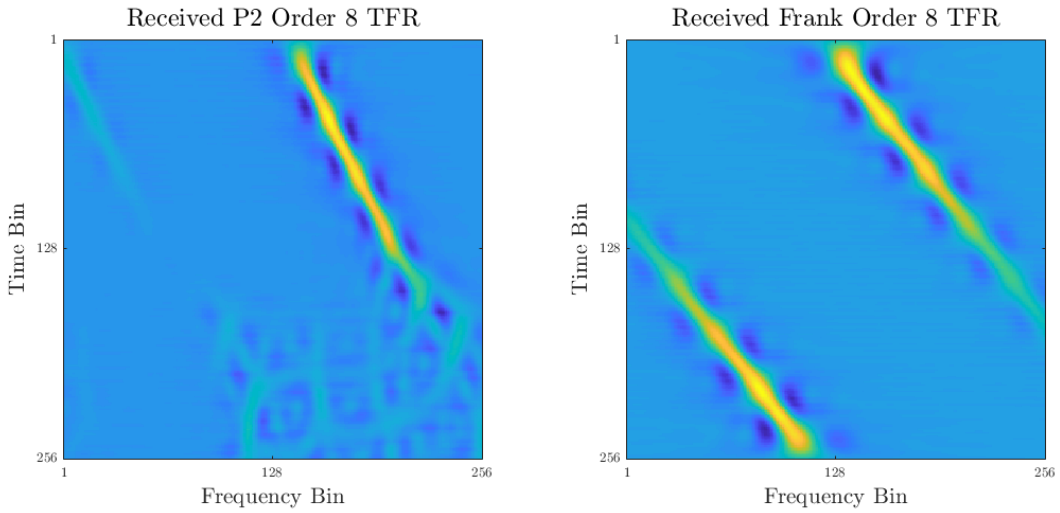


Figure 5.78: *Time-frequency representations of the received P2 order 8 and Frank order 8 pulses*

5.9.2.2 Instantaneous Measurement CNN Classifier

The confusion matrix showing the performance of the instantaneous measurement CNN classifier in the antenna to antenna test is shown in Figure 5.80.

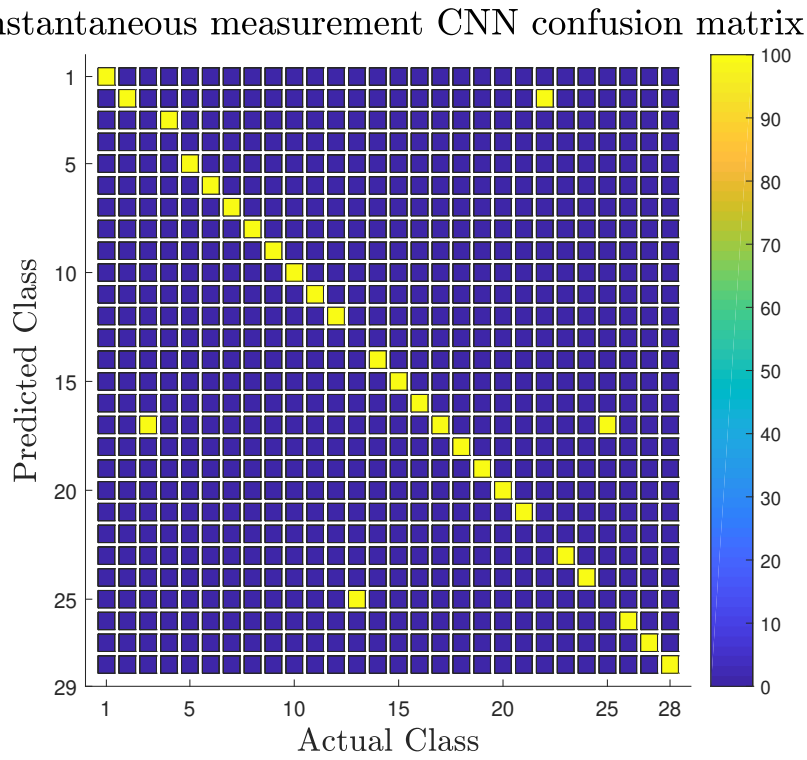


Figure 5.79: *Confusion matrix for the instantaneous measurement CNN classifier in the antenna to antenna test*

While this classifier did not perform as well as the TFR-based classifier in this experiment, it still achieved a reasonable classification accuracy of 82.14%, with only 5/28 classes being misclassified.

The classes that were misclassified and the classes that they were incorrectly classified as are shown in Table 5.5.

Table 5.5: *Misclassified Classes in Instantaneous Measurement Classifier*

Actual Class	Predicted Class
Barker 3	P2 2
Barker 4	Barker 3
P1 2	P4 4
P3 16	Barker 2
P4 4	P2 2

While the TFR classifier operates on a single image, the instantaneous measurement classifier effectively operates on a set of three time-series. This makes the sort of analysis performed on the misclassifications in the TFR-based classifier impractical, as visual similarities and differences would have to be extracted from each measurement component of each misclassified pair.

The contents of Table 5.5 once again show that certain polyphase codes tended to be misclassified as other polyphase codes with the same number of phase transitions. This is especially evident in the misclassification of P1 order 2 for P4 order 4, where the phase transitions occur at the exact same point in the code.

It is highly likely that the effects observed in the visual comparison of the simulated and received signals, particularly the difference in instantaneous frequency peak magnitudes and the artefacts in the instantaneous magnitudes, are responsible for a degree of these misclassifications. If the classifier was highly weighting the magnitude of the peaks in the instantaneous frequency series for classification, the degradation in peak magnitude may have been enough to cause the confusion between classes that is being observed in these results.

5.9.2.3 Raw Complex Time Data CNN Classifier

The confusion matrix showing the performance of the raw complex time data classifier in the antenna to antenna test is shown in Figure 5.80. This classifier was the worst performing classifier in this experiment, with a classification accuracy of 42.86%.

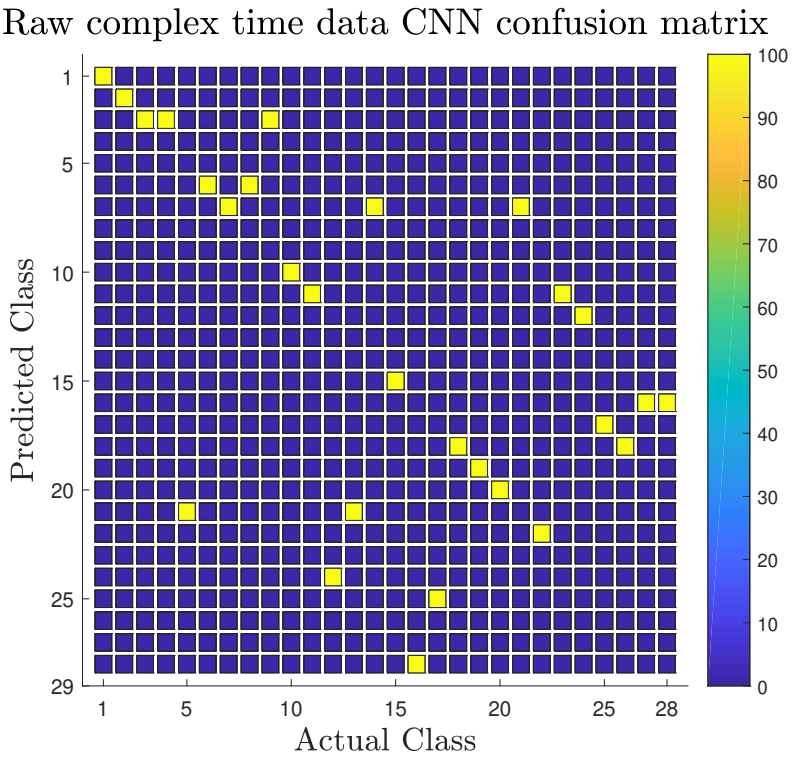


Figure 5.80: *Confusion matrix for the raw complex time data CNN classifier in the antenna to antenna test*

The mean and range normalisation for this classifier had to be performed dynamically instead of using the stored values computed during the training due to the massive differences in signal level between the simulated and received data. It is hypothesised that these level differences were the reason for the majority of the poor performance in this classifier, even after normalisation was applied.

There are a number of other factors that could have contributed to the poor performance of this classifier. The artefacts observed in the instantaneous magnitude plots suggest that the real and imaginary components were impacted unpredictably, which may have affected performance if the classifier relied strongly on perfect alignment between the two components during phase transitions. The instability of the signal level over the pulse duration may also have affected performance as it introduces differences in magnitude between different time segments of the signal. It is also possible that the differences between the simulated noise (Additive white Gaussian noise (AWGN)) and the real noise in the environment had more of an impact on this classifier than on the instantaneous measurement or TFR-based classifiers.

The hypothesis that the classifier performance was strongly impacted by the level differences between the simulated and received data is reinforced by the performance of the instantaneous measurement classifier. Two out of the three instantaneous measurements are naturally normalised in that they are confined to a fixed range for all

possible signals - the instantaneous frequency is computed as normalised frequency in the range $[-1, 1]$, while the instantaneous phase is wrapped between $[-\pi, \pi]$. This implies that if the instantaneous measurement classifier puts a strong weighting on the instantaneous frequency and phase vectors, which is likely, level differences in the instantaneous amplitude vector are unlikely to have of an effect on performance. In comparison, the raw complex time data classifier does not have any naturally normalised components, and relies solely on the magnitudes of the real and imaginary signal components.

5.9.2.4 Split I and Q Complex Time Data Classifiers

The confusion matrix showing the performance of the combined I and Q classifiers in the antenna to antenna test is shown in Figure 5.81. This classifier was second to worst performing classifier in this experiment, with a classification accuracy of 53.57%.

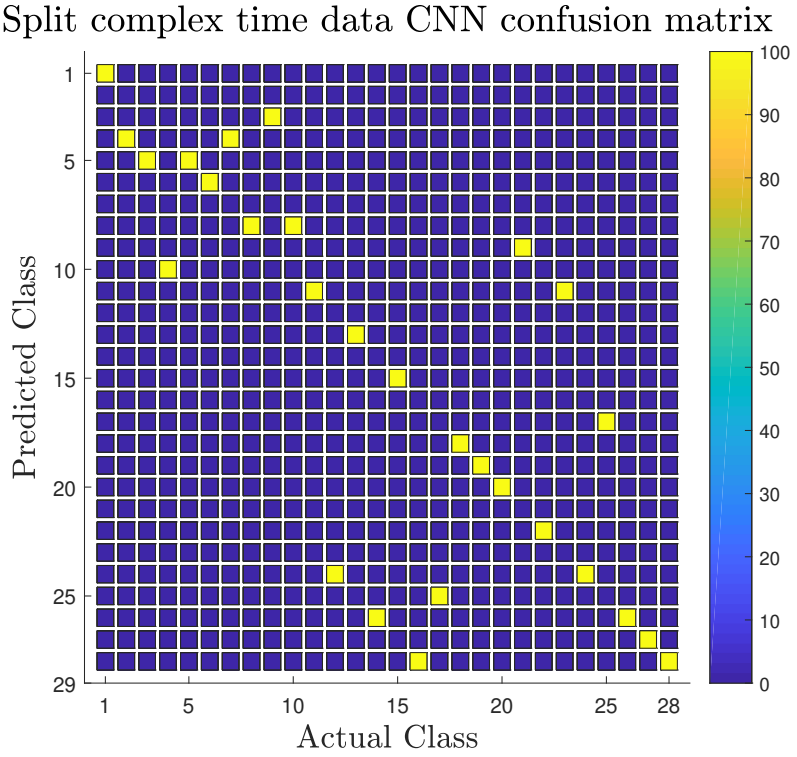


Figure 5.81: *Confusion matrix for the split complex time data CNN classifier in the antenna to antenna test*

It is interesting to note that this classifier was able to achieve approximately 10% higher classification accuracy than the raw complex time data classifier in this experiment.

The performance of the individual I and Q classifiers on the experimental data is shown in Table 5.6.

Table 5.6: *Antenna to Antenna Experiment Classification Performance*

Classifier	Accuracy (%)	Count
Real (I)	50.00	14/28
Imaginary (Q)	42.86	12/28
Combined	53.57	15/28

It is clear that the same effects that plagued the raw complex time data classifier impacted the performance of these classifiers, but that by combining two independent predictions based on their confidence the combined classifier was able to offset these negative effects and boost its classification accuracy.

It is also interesting to note that the individual I and Q classifiers both matched or exceeded the performance of the raw complex time data classifier. This strongly implies that the raw complex time data classifier was unable to effectively extract additional information from the relationship between the I and Q time-series in this experiment, and that the simplistic confidence-based combination approach taken in this classifier was more effective in this regard.

Chapter 6

Discussion

6.1 Dataset Validity

As stated earlier in this report, the primary threat to the validity and integrity of this study is the fact that the dataset used for training, classification and analysis was comprised entirely of simulated examples.

Incorporating real-world imperfections such as negative signal-to-noise ratios, carrier frequency offsets and pulse detection jitter aided in making the dataset more realistic, even though in reality there are far more imperfections and complications that a classification system would need to deal with. In reality, imperfections are not perfectly decoupled from one another. A low SNR signal may require a lower threshold in the level detector, which may cause pulse detection jitter if the pulse is captured incorrectly. It is also possible for SNR and jitter to cause errors in CFO estimation for ramp-like signals (LFM and high order P4 for example), as the apparent midpoint of the ramp will differ depending on how much of it is captured.

The results of the low SNR and antenna to antenna experiments described in Sections 5.6 and 5.9 served to validate the datasets in use. The antenna to antenna experiment showed that the models trained on simulated data could be used to classify data that has passed through a real-world radio link. In turn, the low SNR experiments pushed all classifiers to the point of complete breakdown and, where possible, showed manual (visual) feature extraction was still possible wherever the models were producing accurate results, and that classification breakdown was related to salient features being masked by noise.

As it stands, due to the scope of the project being limited to the classification of pulses with certain fixed parameters (pulse width and LFM bandwidth), it is the belief of the author that the dataset used was sufficiently realistic for the results to be reliable and for the architectures designed to be treated as a base for future real-world modulation classification problems. The limitations on the pulse width and LFM bandwidth were added as the purpose of the models in this study is to classify, and not to measure. In future work the limitation on pulse width could be removed with appropriate preprocessing (resampling), and the limitation on LFM bandwidth could be removed by varying the bandwidth of the signal in the training set and adding an extra post-classification bandwidth measurement step.

6.2 Classifier Performance

The classification accuracy and throughput of all models in this study is shown in Table 6.1. Note that the throughput of the MSE-Based ridge classifier and the split complex time data classifier was not measured.

Table 6.1: *Complete summary of classifier performance*

Classifier	Accuracy (%)	Throughput (pulses/sec)
MSE-Based ridge classifier	70.57	N/A
TFR-Based CNN classifier	98.82	725
Instantaneous measurement CNN classifier	84.07	11000
Raw Complex CNN classifier	86.57	18000
Split Complex CNN classifier (I)	84.71	N/A
Split Complex CNN classifier (Q)	80.71	N/A
Split Complex CNN classifier (Combined)	88.14	N/A

Section 5.1 showed that the MSE-based ridge classifier was able to achieve a classification accuracy of 70.57%, while Sections 5.2 to 5.5 showed that classifiers making use of machine learning techniques were able to achieve an average classification accuracy of 87.17%. These results immediately show that the classifiers making use of convolutional neural networks were able to outperform the tested non-machine learning counterpart by a significant margin.

These classifiers differed in the sense that the MSE-based ridge classifier took a more "manual" approach to classification, where heavy preprocessing was performed to extract features that could be compared to a reference set for classification. In contrast to this, the machine learning classifiers made use of convolutional neural networks where the only preprocessing necessary was to transform the raw data to the desired input representation and to apply normalisation and scaling operations. All feature extraction was then performed automatically.

The advantage of a more manual approach to feature extraction is that it lends itself to a deeper understanding of the modulation types in question. The entire classification process is transparent and purpose-built, which could allow for known, undesirable effects to be mitigated if developed correctly.

This being said, the MSE-based ridge classifier showed that it did not generalise well to all pulse types, and the feature set used was not always diverse enough to allow similar classes to be separated. The lack of clear "steps" in certain pulse ridges made them unsuitable for classification using this system, indicating that it may be necessary for a secondary classification stage to be implemented that searches for and extracts

different features from pulses with "LFM-like" characteristics. The lack of algorithmic flexibility is the primary disadvantage of manual classification approaches, and the problems experienced during this study will simply be compounded as more classes are added and the classification scope is increased.

Conversely, the primary advantage noted during the testing of the convolutional neural network classifiers is that they were able to generalise equally well to all classes, and required very little manual signal processing to implement successfully.

The main disadvantage to the machine learning based classifiers is that they are highly opaque systems, and are generally regarded as a "black-box" approach where the internals are hidden from the user and are not validated or analysed in depth.

6.3 Preprocessing Requirements

6.3.1 Preprocessing Summary

The MSE-based ridge classifier required the raw complex time data to be extensively preprocessed before it could be used. These steps are listed below:

- TFR computation
- TFR thresholding and cropping
- Ridge extraction
- Ridge normalisation and preprocessing
- Step counting and step parameter extraction

Computing the time-frequency representation of each signal to be classified is a computationally expensive operation, and will likely place more of a limit on the throughput of the system than the classification process itself. This being said, it is feasible that this classifier could be used in a real-time system if the preprocessing steps are accelerated by a graphics processing unit (GPU) or field-programmable gate array (FPGA).

The classifiers making use of CNNs required far fewer preprocessing steps than the MSE-Based ridge classifier. The preprocessing steps for each CNN classifier are as follows. Note that the split complex CNN classifier is not listed as it is identical to the raw complex CNN classifier in terms of preprocessing.

TFR-Based CNN Classifier

- TFR computation
- TFR thresholding and cropping
- TFR zero-centring (subtraction by constant)

Instantaneous Measurement CNN Classifier

- Instantaneous amplitude, frequency and phase computation
- Mean and range normalisation

Raw Complex CNN Classifier

- Mean and range normalisation

Even though the TFR-Based CNN classifier was able to classify data far more accurately than the MSE-Based ridge classifier, being forced to compute a TFR for every single pulse is still a substantial amount of preprocessing. Once again, this operation may limit the throughput of the classifier more than the feedforward step through the model itself.

Computing the instantaneous amplitude, frequency and phase required to operate the instantaneous measurement CNN classifier is a computationally simple operation and can be run significantly faster than the TFR computation required for the TFR-Based CNN classifier. However, Sections 5.4 and 5.5 showed that a convolutional neural network was able to learn from raw complex time data with no preprocessing besides mean and range normalisation while still maintaining excellent classification accuracy on the simulated signals.

Raw complex time data is undoubtedly the most computationally viable data representation for intra-pulse modulation classification. The use of this data representation resulted in significantly higher throughput than any other tested system, both due to the lack of requisite preprocessing, and due to the relatively small size of the classifier model. It is crucial to note that while the raw complex time data classifier may have been the most efficient of the tested classifiers, it was not the best performing or most robust. The TFR-based CNN classifier exhibited significantly higher accuracy, resilience to noise, and resistance to the effects of a real-world radio link.

6.3.2 Hardware Acceleration

If left in its current state, the preprocessing required in the MSE-Based ridge classifier will likely place more of a limit on throughput than the classification process itself. It is possible that a number of these preprocessing steps can be accelerated using a GPU or a FPGA. However, the actual classification process itself is highly sequential with little room for parallelisation apart from the simultaneous classification of multiple pulses (batching).

In contrast to the MSE-Based ridge classifier, the classifiers designed using convolutional neural networks are well suited to massively parallel processing and can have their processing offloaded to a GPU or FPGA with relative ease using open source software.

After observing that the raw complex time data CNN classifier was capable of classifying approximately 18000 pulses per second under GPU acceleration, the test was repeated without GPU acceleration, which showed that the system was only capable of classifying approximately 1000 pulses per second. This is a speed-up of almost 20 times, and shows that without hardware acceleration it may be challenging to run these processes in real-time with real-world pulse densities.

The speed of the classifier without GPU acceleration is solely based on the speed and parallelism of the CPU in the system (3.3 GHz, 4 cores, no hyperthreading for the example above), and the ability of the software to exploit that parallelism. Since these tests were carried out using the TensorFlow back-end [49], the model is potentially able to scale to a large number of CPUs, each with many cores. However, given the low cost of consumer-grade GPUs, it is entirely likely that a GPU driven approach will always be cheaper and more efficient than a CPU driven approach that strives for similar throughput.

6.4 Effects of Pulse Imperfections

6.4.1 Parameter Variations

While all tested classifiers exhibited a degree of performance degradation for signals with low SNR, the TFR-based CNN classifier was by far the most robust in this regard. Figure 5.10 shows that this classifier was capable of classifying signals with extremely high accuracy up to the lower SNR bound of the initial test - long after the other tested classifiers began exhibiting negative effects. The MSE-based ridge classifier was more sensitive to jitter than SNR, but removing jitter showed that there is a clear

roll-off in classification accuracy below approximately 10 dB SNR. All other CNN-based classifiers (instantaneous measurement, raw complex, and split complex) began to show performance degradation for signals below approximately 0 dB SNR.

The low SNR analysis experiment was designed to push the various CNN-based classifiers to the point of complete classification breakdown by testing them over a much wider range of SNRs than the original tests. For this purpose, the SNR range was extended from the original $[-5, 20]$ dB to $[-20, 20]$ dB. The results of this experiment in Section 5.6, particularly Figure 5.28, show that the TFR-based CNN classifier is capable of producing accurate results at SNRs up to 10 dB lower than other classifiers. Analysis of the various data representations at various SNRs provided some insight into this result. In general, it appears that it is far easier to visually extract salient features in time-frequency representations than instantaneous measurements at low (≤ 10 dB) SNR.

The only classifier that proved to be sensitive to jitter was the MSE-based ridge classifier. This classifier showed a direct relationship between classification error and the amount of jitter in the signal. This is due to the fact that pulse detection jitter results in a linear shift of the locations of steps in the ridge of the signal. The start, end and peak locations of each step are one of the primary features used for classification, so it is understandable that any deviation from their nominal values will result in increased classification uncertainty.

6.4.2 Radio Link Effects

Section 5.9 presents the results of the antenna to antenna experiment designed to test the feasibility of using simulation-trained CNN classifiers on data that has passed through a real-world radio link. Upon analysis of the received signals, it was immediately clear that the combination of the transmitter, channel, and receiver resulted in significant deviations from the original simulated signals. While these deviations are discussed in full in Section 5.9, the most significant observations are noted below.

- Ripple in signal magnitude over the duration of the received pulse
- Noise in measured instantaneous frequency
- Randomised starting phase of received signals
- Artefacts in received instantaneous magnitude at the locations of phase transitions
- Reduction in size of instantaneous frequency peaks in received signals
- Transients on leading edges of phase transitions in received signals

The received signals containing these channel and hardware-related imperfections were fed through the CNN-based classifiers developed in this study, with varying degrees of success.

It was found that the TFR-based CNN classifier was the most resilient to the effects of the radio link, classifying 26 out of the 28 classes correctly. The classes that were misclassified were both high-order polyphase codes, and clear visual similarities were observed between the classes causing confusion. That being said, there were also enough visual differences between the misclassified classes that one can expect the model to classify them correctly if the training set is supplemented with new examples containing these new imperfections.

The CNN classifier operating on raw complex data showed the worst resilience to the effects of the radio link, with only 12 of the 28 classes being correctly identified. It is hypothesised that the poor performance is due in part to the classifier being unable to generalise to the level differences between the simulated training signals and the sampled test signals. It is likely that the performance of this classifier could be substantially improved by training on more signals with varying levels and by improving the normalisation techniques in use. Adding examples of sampled signals that have undergone the effects of the radio link to the training set would likely also substantially improve the ability of the classifier to generalise to this data.

The manner in which misclassifications manifested themselves during the antenna to antenna experiment suggests that a two-stage approach to modulation classification may be a worthwhile topic for future research. The first stage would be responsible for identifying the modulation scheme of the signal using a single classifier, and the second stage could make use of a bank of modulation-specific classifiers that are purely responsible for identifying the modulation order. This approach may counteract the confusion between different high order polyphase codes that was experienced during this experiment.

6.5 Data Representations

This study evaluated three different representations of radar pulses in order to analyse the impact of various degrees of preprocessing and abstraction on the effectiveness of machine learning models. The data representations that were analysed were, in order of decreasing amounts of preprocessing, a time-frequency representation (ZAM-GTFR), a set of instantaneous measurements (amplitude, frequency and phase), and raw complex time data.

These levels of complexity also correspond to the efficiency of visual analysis by a human. Waterfall plots and time-frequency representations are a tried and tested method of displaying data to users and operators, so it is natural that they were considered as an input to classifiers. These time-frequency representations are also the closest to regular images out of the three selected representations, which made them a logical choice for classification by traditional image-based CNNs.

Instantaneous measurements are a commonly used data representation in signal analysis, and offer different information to a time-frequency representation. While a set of instantaneous measurements can be extremely useful to a trained operator, it is a substantially less intuitive data representation than a time-frequency representation, and may be less efficient to analyse manually.

Raw complex time data is the simplest possible data representation and is extremely inefficient for human-driven analysis without preprocessing. This input representation tested the ability of the classifiers to adapt to and learn from data representations that are not limited to the scope of human visual recognition.

It was found that all three of the aforementioned data representations performed extremely well when paired with their respective convolutional neural network models. While human-driven classification with a time-frequency representation may be possible at low signal-to-noise ratios, the machine learning models were able to perform accurate classification long after visually observable features in instantaneous measurements and raw complex time data would have become masked by noise.

As stated in Section 6.3, more complex preprocessing also leads to increased hardware demands in order to potentially achieve real-time operation. The results of the three benchmarked CNN classifiers indicate that feedforward speed is inversely proportional to the size of the classifier model, which is, to a large extent, driven by the complexity of the data representation in use. Having a low feedforward speed and being required to perform complex preprocessing makes data representations such as time-frequency representations more computationally demanding for potential real-time operation, and points to simpler representations, such as raw complex time data, as being the optimal choice for high-throughput classification.

While raw complex time data may be ideal for applications where speed and computational efficiency is critical, time-frequency representations such as the ZAM-GTFR were found to be far more resilient to noise than other representations. The imperfections due to channel effects were clearly visible in the complex time data and instantaneous measurements of signals, they did not have any major visual impact on the ZAM-GTFR of the same signals. The TFR-based CNN classifier consistently performed

better than other tested models, which suggests that time-frequency representations are the optimal choice for applications where performance and robustness are more critical than throughput and low amounts of preprocessing.

The models developed for the instantaneous measurement classifier and the raw complex time data classifier required significant divergences from traditional image-based CNN models to achieve high classification accuracy on time-series data. These changes took the form of restrictions in the dimensionality reduction operations and expansion of the convolutional filters to allow for a greater range of temporal relationships to be observed. These changes and the observations that led to them may be of use in future work where CNN-based classification of time-series data is required.

6.6 Split I and Q Complex Time Data Classifiers

The results of the separate analysis of the I and Q complex time data classifiers showed that it was possible for a convolutional neural network to learn to classify modulation types extremely accurately from only a single component of the signal. In fact, the method employed to combine the results of the split I and Q classifiers resulted in better performance than the raw complex time data classifier in all tests.

This is an interesting result, as splitting the I and Q components in this manner means that the two independent classifiers are unable to make use of any information contained in the relationship between the I and Q time series. This is not to say that the two components do not carry important phase information independently. The real (I) component of the analytic signal is, by definition, the real signal itself, and the imaginary (Q) component is the Hilbert transform of that signal. As expected, combining the two classifiers through their confidence metrics had the effect of compensating for the uncertainty of incorrect predictions in the individual classifiers, resulting in a higher overall classification accuracy.

It is worth noting that the method used to fuse the results of the two classifiers does not make up for the fact that any information contained in the relationship between the two time series was effectively lost. The confidence-based data-fusion method used was relatively trivial and could potentially be improved upon by making use of other performance metrics or architectures.

6.7 Filter Activation Analysis

Section 5.7 presented the results of the convolutional filter analysis that was performed on the raw complex time data classifier for Barker 13 and P4 64 pulses.

By comparing the convolutional filter activations and the instantaneous frequency plots of the input pulses, it was possible for some observations on the features extracted by the CNN to be made. In both cases, the results showed a strong correlation between the locations of the phase transitions in the signal (spikes in the instantaneous frequency plots) and the regions of maximum activation in the convolutional filters.

These results suggested that the first convolutional layer of the complex time data CNN operates in a similar manner to the process of manual classification, where a coarse separation of pulse types is performed using the locations of phase transitions or the number of instantaneous frequency peaks as features.

An interesting and important topic for future investigation would be to delve into the convolutional filters deeper into the network in an attempt to relate the other features extracted by the network to humanly understandable concepts. This may assist in optimising the existing CNN architectures or providing guidelines for the implementation of new architectures that are more robust to the signal imperfections and channel effects observed in the antenna to antenna experiment.

6.8 Complex Time Data Reconstruction

Thanks to the concept of gradient ascent, it is possible to reconstruct an input image that corresponds to the maximisation of the confidence of a specific class on a trained neural network. This process is extremely similar to some of the approaches used in the generation of adversarial examples or models as described in Section 2.5.6. The images produced through this process are not always intuitively related to the images used to train the model for the class in question, and often seem to resemble textures or patterns rather than distinct objects [53].

Section 5.8 applied this technique to the certain classes in the trained model for the raw complex time data classifier, and went on to analyse the resultant data to see if it bears any similarity to the actual class under test.

The results of the various forms of analysis in Figures 5.48 to 5.54 showed that there were no obvious visual similarities between the “optimised” data resulting from the gradient ascent method and the actual data. The TFR results in Figures 5.50 and 5.54 show that

there is some structure to the generated signals, but that they share no traits with the original data.

While the complex time data reconstruction experiment did not reveal any major findings, it did draw further attention to an interesting topic for future research. Even though there were no apparent visual similarities between the original and the reconstructed data, these newly generated examples were classified by the model as their corresponding classes with extremely high confidence. The signals generated through this method are very likely unique to the model in question and could form the basis for a line of research into electronic countermeasures against trained CNN-based pulse or modulation classifiers. The fact that the "optimised" data had very little resemblance to the original data implies that it may be possible to introduce additional modulation to radar pulses in an attempt to fool or mislead CNN-based classifiers, similar to how adversarial examples for image-based classifiers have been shown to mislead popular CNN models.

Chapter 7

Conclusions

This chapter serves to draw meaning from the results presented in previous chapters and their subsequent discussion in order to answer the research questions proposed at the beginning of this study.

7.1 Research Questions

Can radar intra-pulse modulation schemes be reliably classified without machine learning techniques?

The results of the MSE-based ridge classifier showed that a classification accuracy of approximately 70% could be reached without the use of machine learning techniques. The developed classifier exhibited very little robustness against pulse detection jitter, and a strong dependence on high SNR when jitter was removed from the equation. The classification accuracy was shown to degrade rapidly for signal-to-noise ratios below 10 dB, whereas other classifiers maintained reasonable performance past 0 dB SNR. In real-world scenarios it is likely that one would have to deal with radar signals that have been sampled with extremely low SNR. This classifier also exhibited tendencies to repeatedly confuse certain classes, which showed that the feature set being used for classification was not diverse enough to extract unique characteristics for each class.

Extensive preprocessing in the form of TFR and ridge computation is required for this classifier to operate, which reduces the potential for real-time processing. This is not to say that real-time processing is not possible using this classification approach, but hardware acceleration techniques for parallel processing would need to be investigated.

The points discussed above are unique to the non-machine learning classifier developed specifically for this study. As discussed in Section 2.5, prior literature has shown that a number of researchers have had much more successful forays into the field of deterministic pulse classification. In conclusion, intra-pulse modulation classification is viable without machine learning techniques, but results of this study indicate that CNN-based approaches may be more computationally efficient, more accurate, and more robust than certain simplistic approaches.

Could the application of machine learning improve on non-machine learning based classification techniques?

The results of all of the classifiers using machine learning showed substantial improvements over the tested non-machine learning classifier, with an average classification accuracy of 87.17%. Prior literature already confirmed that machine learning can improve on other modulation classification techniques, but none of the studies that were researched considered as broad a range of modulation schemes as this one.

The drawbacks observed in the tested non-machine learning classifier (poor resilience to pulse detection jitter and low SNR, confusion between classes) were not observed in the results of any of the CNN-based classifiers. Solutions to the problem of extensive preprocessing were also successfully explored through the use of low-level data representations such as raw complex time data. The actual classification approach used in CNN-based classifiers also lends itself more easily to hardware acceleration, with GPU-based implementations being easily accessible through open-source software.

In summary, yes - machine learning can offer significant improvements over other classification approaches, but has the disadvantage of being a more opaque process to designers and operators.

Which data representations are best suited to classification using a convolutional neural network?

It was found that using time-frequency representations such as the ZAM-GTFR resulted in the highest performing and most robust classifier model. The model developed using this data representation had the highest classification accuracy in all tests. It also showed significantly higher performance at low SNR ($\leq$ 0 dB) than other models, and was the least impacted by the effects of the transmit-receive chain and the radio link in the antenna to antenna experiment.

The only drawbacks of the TFR-based classifier are the requisite preprocessing and the relatively low throughput compared to other models. It is possible that, with some optimisation, the size of the classifier model can be reduced to increase feedforward speeds through the neural network. The TFRs in use were all of size 256 $\times$ 256, which is relatively large in comparison to the inputs of other time series based models. The 256 $\times$ 256 pixel image results in 65536 input neurons in the model, whereas the next largest model (the instantaneous measurement classifier) only needed 3072 input neurons - a factor of 21 times fewer. Future development of this classifier model should explore reducing the size of the input image until classification accuracy begins to degrade, as any reduction in input size will result in increased throughput.

What level of classification accuracy can be achieved using a fully trained convolutional neural network for each of the approaches explored in the previous question?

All tested CNN models exhibited extremely high classification accuracies across the entire range of pulse imperfections. The performance of the various models is discussed fully in Section 6.2, but the classification accuracies of the individual CNN models corresponding to the different data representations under test are given in Table 7.1.

Table 7.1: *Convolutional neural network classifier results*

Data Representation	Accuracy (%)
Time-frequency Representation	98.82
Instantaneous Measurements	84.07
Raw Complex Time Data	86.57
Split Raw Complex Time Data	88.14

The highest classification accuracy was achieved using the time-frequency representation based classifier, where ZAM-TFR of each incoming signal is computed, cropped, and scaled before classification. Conversely, while it was shown to insufficiently robust in certain regards, the raw complex time data model made use of the simplest architecture and proved to be capable of significantly higher throughput that other models.

What is the effect of low (negative) signal-to-noise ratio on the accuracy of the classification techniques?

The MSE-based ridge classifier showed the most severe performance degradation as the SNR of test signals was decreased. When all jitter was removed from the test examples, this classifier began showing accuracy roll off for signals below 10 dB SNR.

The machine learning based classifiers exhibited remarkable robustness to low signal-to-noise ratios, with all classifiers achieving over 90% accuracy for positive SNR. A separate experiment using a new dataset of signals with an extended SNR range was required to evaluate the performance of the various CNN-based classifiers for negative SNR.

The TFR-based CNN classifier showed the most robustness to negative SNR, maintaining extremely high classification accuracy until -10 dB SNR. Conversely, all other CNN-based classifiers experienced a relatively smooth roll-off of classification accuracy from approximately 0 dB SNR. The model making use of instantaneous measurement data was the most SNR dependent of all CNN-based classifiers, achieving only 15.76% accuracy for signals in the range of $[-20, 0)$ dB SNR. For comparison, the TFR-based CNN classifier achieved 56.82% accuracy for signals in the same range.

What is the effect of carrier frequency offset on the accuracy of the classification techniques?

Carrier frequency offset had minimal impact on the classification accuracy of any of the tested classifiers. The reason for this is probably due to the fact that a number of the data representations are not strongly affected by frequency offsets.

A frequency offset in a time-frequency representation simply shifts the location of the signal portion in the "image", which is accounted for during the thresholding and cropping preprocessing operations.

Frequency offsets in the instantaneous measurement representation take the form of a constant vertical shift in the instantaneous frequency data and the addition of a constant rate of change to the instantaneous phase data, resulting in phase wrapping. It is quite likely that due to the presence of phase wrapping the model made use of the instantaneous frequency data as the primary feature source (this is backed up by the convolutional filter analysis performed), in which case a constant vertical offset is easily accounted for.

What is the effect of pulse detection jitter on the accuracy of the classification techniques?

The MSE-based ridge classifier was the only technique that showed a relationship between pulse detection jitter and classification accuracy. This finding is related to the method in which the classifier was implemented, and the strong dependence on fixed time-based locations of features for accurate classification.

The performance of classifiers making use of machine learning techniques proved to be largely invariant in the face of pulse detection jitter. It is highly likely that the convolutional filters that were learned during training placed more of an emphasis on the locations of features relative to one another, instead of their absolute location with respect to the start or the end of the signal as a whole.

Can parallels be drawn between the features extracted by the convolutional neural networks and the features extracted using common visualisation methods?

The convolutional filter activation analysis experiment showed a strong correlation between the locations of phase transitions in the signals and the regions of maximum activation in the first-layer convolutional filters. This suggests that salient visual features such as the number and the locations of the phase transitions in the signal are still heavily weighted by convolutional neural networks. Analysis of the features extracted by deeper convolutional layers is an avenue for future research, and may provide valuable insights into how time-domain signals are perceived by machine learning systems.

Can classifiers trained on simulated data classify data that has passed through a real-world radio link?

The analysis of signals during the antenna to antenna experiment showed that the transmission of signals through a physical medium resulted in the addition of clear, and sometimes unpredictable, imperfections. It is to be expected that these added imperfections would degrade the performance of simulation-trained classification models.

The CNN-based classifiers that were tested on data that had undergone transmission, channel, and sampling effects performed with varying degrees of success. The TFR-based CNN classifier was by far the most robust of the tested classifiers, while the classifier operating on raw complex time data was most sensitive to the changes made to the signals.

The results of this experiment suggest that it is possible to train classifiers on simulated data and still achieve reasonable performance on data that has passed through a physical medium, but that certain data representations show the effects of the transmission process more clearly than others. It is also suggested that, while training on an entirely real-world dataset may not always be possible in practice, supplementing a simulated dataset with a number of real-world examples may significantly improve the robustness of the resulting models.

Can classification be carried out in real time if a realistic pulse density is assumed?

The convolutional neural network classifier that made use of raw complex time data as an input data representation was capable of classifying approximately 18000 pulses per second. This rate of classification would allow for simultaneous real-time classification of a number of medium PRF emitters or a single high PRF emitter. In comparison, the TFR-based CNN classifier was much more robust than the complex time data based classifier, but was only capable of classifying approximately 725 pulses per second. As previously stated, the throughput of the TFR-based classifier could likely be substantially improved by reducing the size of the input to the model.

The rate of classification for all models can be increased if pulses are allowed to be batched into groups before classification. The throughput metric used in this study was measured using batches of 64 pulses at a time, but larger groups could result in significantly higher throughput due to reduced computational overhead. The only disadvantage of this approach is that it introduces additional latency between pulse detections and pulse classifications. It is possible that the batch size parameter could be automatically adjusted to ensure that a fixed maximum latency requirement is met while maintaining optimal classification throughput.

Are the performance improvements obtained from convolutional neural networks worth the loss in transparency from non-machine learning approaches?

The answer to this question remains largely up to the designer of the classification system.

This study has shown that convolutional neural networks offer significant advantages over the tested decision tree based classification approach in terms of accuracy, robustness, speed, and requisite preprocessing. However, only a minor amount of research was undertaken in attempting to investigate the operation and features extracted by the various convolutional neural networks, and they remain highly opaque processes.

There remain many systems in which a deterministic classifier that is potentially more conservative and transparent than a neural network may be a better choice - especially those systems where a misclassification or an incorrect prediction may result in damage, loss of property, or even loss of life.

In general, it is the belief of the author that machine learning is a more viable option than deterministic or non-machine learning based techniques for intra-pulse modulation classification, but that the applicability of machine learning to other areas of radar and electronic defence should be evaluated on a case-by-case basis, both in terms of performance and morality.

Chapter 8

Recommendations

8.1 Further Evaluation on Real-World Data

While the antenna to antenna experiment evaluated the performance of the trained classifiers on data that had passed through a real-world transmission medium, the data analysed was all from a single emitter, and thus lacked diversity. In order to further validate the results of this study, a dataset consisting of real-world recordings from various emitters should be created and used for training and testing.

Such a dataset may be challenging to obtain due to the classified nature of many defence radar systems, but a collaborative effort between academic and commercial research groups may be able to produce a dataset large enough for meaningful analysis. It is likely that the most robust classifiers will be produced using a dataset with a mixture of simulated and recorded radar signals.

Making use of more real-world data implies that more advanced preprocessing would need to be implemented to remove the constraint of a fixed pulse width, and to improve the robustness of the time-series classifiers presented in this study.

8.2 Broaden Scope of Classification

The results of this study indicate that classifiers making use of convolutional neural networks performed extremely well on all tested modulation types and orders. The range of modulation orders under test could be broadened, and additional modulation types could be added in order to expand on the capabilities of the classifiers.

In addition to improving upon the abilities of the classifiers, these classifiers could be applied to larger systems for the tasks of emitter identification and deinterleaving. This task would fall hand in hand with the task of evaluating the system on real-world data and would serve as further validation of the results of this study.

8.3 Improve Throughput of TFR-Based Model

The TFR-based CNN classifier presented in this study makes use of 256 × 256 pixel images, which results in a model with a large number of trainable parameters. This large model proved to be significantly slower than other time series-based models. Further research on this classifier should include reducing the size of the input images, while evaluating the effects on performance, throughput, and robustness.

8.4 Investigate CNN Feature Extraction

The analysis on convolutional filter activations revealed some interesting relationships between the locations of phase transitions and the regions of maximum convolutional filter activation.

This analysis should be continued and extended in order to identify other parallels between traditional analysis techniques and the features extracted by convolutional neural networks. It may also be possible to glean information on previously unexplored signal features and characteristics of modulation techniques through this approach.

8.5 Research Adversarial Techniques

The complex time data reconstruction experiment revealed results that may be relevant to the field of electronic counter-countermeasures (ECCM). In future research, one could investigate whether introducing additional modulation on frequency modulated or phase coded pulses is a viable method for degrading or confusing a trained classifier model. Conversely, one could also explore what kind of preprocessing techniques could be developed to reduce the vulnerability of a classifier to such adversarial approaches.

8.6 Evaluate Two-Stage Modulation Classifier

The misclassified examples in the antenna to antenna experiment suggest that separating the classification of modulation type from the classification of modulation order may yield improved robustness against transmitter and channel effects. Further research might be well spent on training a generic and robust modulation classifier that is agnostic to modulation order. This classifier could then be paired with a bank of modulation scheme-specific classifiers that are responsible for the extraction of modulation order.

Bibliography

[1] M. A. Richards, J. A. Scheer, and W. A. Holm, Principles of modern radar. Scitech Publishing, 2010.

[2] G. Callanan and J. Swart, "A comparative analysis of phase modulation techniques," University of Cape Town, EEE5105Z.

[3] R. Frank, "Polyphase codes with good nonperiodic correlation properties," IEEE Transactions on Information Theory, vol. 9, no. 1, pp. 43–45, 1963.

[4] B. Lewis and F. Kretschmer, "A new class of polyphase pulse compression codes and techniques," IEEE Transactions on Aerospace and Electronic Systems, no. 3, pp. 364–372, 1981.

[5] B. L. Lewis and F. F. Kretschmer, "Linear frequency modulation derived polyphase pulse compression codes," IEEE Transactions on Aerospace and Electronic Systems, no. 5, pp. 637–641, 1982.

[6] N. J. Beaudry and R. Renner, "An intuitive proof of the data processing inequality," arXiv preprint arXiv:1107.0740, 2011.

[7] A. Wilkinson, "Notes on Radar/Sonar Signal Processing: Fundamentals," 2017.

[8] H.-I. Choi and W. J. Williams, "Improved time-frequency representation of multicomponent signals using exponential kernels," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, no. 6, pp. 862–871, 1989.

[9] X. Wu and T. Liu, "Spectral decomposition of seismic data with reassigned smoothed pseudo wigner–ville distribution," Journal of Applied Geophysics, vol. 68, no. 3, pp. 386–393, 2009.

[10] L. Cohen, "Time-frequency distributions-a review," Proceedings of the IEEE, vol. 77, no. 7, pp. 941–981, 1989.

[11] A. Papandreou and G. F. Boudreaux-Bartels, "Generalization of the choi-williams distribution and the butterworth distribution for time-frequency analysis," IEEE transactions on signal processing, vol. 41, no. 1, pp. 463–472, 1993.

[12] A. M. Zoubir, M. Viberg, R. Chellappa, and S. Theodoridis, Academic Press Library in Signal Processing: Array and Statistical Signal Processing. Academic Press, 2014.

[13] J. Lundén and V. Koivunen, "Automatic radar waveform recognition," IEEE Journal of Selected Topics in Signal Processing, vol. 1, no. 1, pp. 124–136, 2007.

[14] Y. Zhao, L. E. Atlas, and R. J. Marks, "The use of cone-shaped kernels for generalized time-frequency representations of nonstationary signals," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 38, no. 7, pp. 1084–1091, 1990.

[15] S. Oh and R. J. Marks, "Some properties of the generalized time frequency representation with cone-shaped kernel," IEEE Transactions on Signal Processing, vol. 40, no. 7, pp. 1735–1745, 1992.

[16] M. N. Murty and V. S. Devi, Pattern recognition: An algorithmic approach. Springer Science & Business Media, 2011.

[17] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol. 61, pp. 85 – 117, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608014002135

[18] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 315–323.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[20] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in 2012 IEEE Conference on Computer Vision and Pattern Recognition, June 2012, pp. 3642–3649.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[22] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," Journal of Machine Learning Research, vol. 12, no. Jul, pp. 2121–2159, 2011.

[23] T. Tieleman and G. Hinton, "Lecture 6.5 - rmsprop," COURSERA: Neural networks for machine learning, vol. 4, no. 2, pp. 26–31, 2012.

[24] The MathWorks, Inc., "Convolutional Neural Network - MATLAB & Simulink," https://www.mathworks.com/discovery/convolutional-neural-network.html, 2018, Accessed: 28/02/2018.

[25] W. Fenghua, H. Zhitao, Z. Yiyu, and J. Wenli, "A new approach for intra-pulse modulation recognition," in Radar, 2006. CIE'06. International Conference on. IEEE, 2006, pp. 1–5.

[26] M. Svarc, L. Brnak, and M. Richterova, "Perspective methods of radar pulse classification," in International Conference on Military Technologies (ICMT) 2015. IEEE, 2015, pp. 1–8.

[27] A. Singh and K. S. Rao, "Detection, identification and classification of intra pulse modulated lpi radar signal using digital receiver," International Journal of Emerging Technology and Advanced Engineering, vol. 2, no. 9, 2012.

[28] D. Zeng, X. Zeng, G. Lu, and B. Tang, "Automatic modulation classification of radar signals using the generalised time-frequency representation of zhao, atlas and marks," IET radar, sonar & navigation, vol. 5, no. 4, pp. 507–516, 2011.

[29] Y. Guo and X. Zhang, "Radar signal classification based on cascade of stft, pca and naïve bayes," in 2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS). IEEE, 2016, pp. 191–196.

[30] B. Barshan and B. Eravci, "Automatic radar antenna scan type recognition in electronic warfare," IEEE Transactions on Aerospace and Electronic Systems, vol. 48, no. 4, pp. 2908–2931, 2012.

[31] M. W. Aslam, Z. Zhu, and A. K. Nandi, "Automatic modulation classification using combination of genetic programming and knn," IEEE Transactions on wireless communications, vol. 11, no. 8, pp. 2742–2750, 2012.

[32] A. K. Mishra and B. Mulgrew, "Radar signal classification using pca-based features," in 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, vol. 3. IEEE, 2006, pp. III–III.

[33] M. D. DeVore and J. A. OSullivan, "A performance-complexity study of several approaches to automatic target recognition from synthetic aperture radar images," IEEE Transactions on Aerospace and Electronic Systems, 1999.

[34] G. P. Noone, "A neural approach to automatic pulse repetition interval modulation recognition," in Information, Decision and Control, 1999. IDC 99. Proceedings. 1999. IEEE, 1999, pp. 213–218.

[35] M. Conning and F. Potgieter, "Analysis of measured radar data for specific emitter identification," in 2010 IEEE Radar Conference. IEEE, 2010, pp. 35–38.

[36] D. Evans and A. J. Jones, "A proof of the gamma test," Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, vol. 458, no. 2027, pp. 2759–2799, 2002.

[37] T. J. OShea, 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.

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

[39] J. R. van der Merwe, W. P. du Plessis, F. D. Maasdorp, and J. E. Cilliers, "Introduction of low probability of recognition to radar system classification," in 2016 IEEE Radar Conference (RadarConf). IEEE, 2016, pp. 1–5.

[40] C. Sturm and W. Wiesbeck, "Waveform design and signal processing aspects for fusion of wireless communications and radar sensing," Proceedings of the IEEE, vol. 99, no. 7, pp. 1236–1259, 2011.

[41] S. D. Blunt, P. Yatham, and J. Stiles, "Intrapulse radar-embedded communications," IEEE Transactions on Aerospace and Electronic Systems, vol. 46, no. 3, pp. 1185–1200, 2010.

[42] D. Ciuonzo, A. De Maio, G. Foglia, and M. Piezzo, "Intrapulse radar-embedded communications via multiobjective optimization," IEEE Transactions on Aerospace and Electronic Systems, vol. 51, no. 4, pp. 2960–2974, 2015.

[43] J. G. Metcalf, C. Sahin, S. D. Blunt, and M. Rangaswamy, "Analysis of symbol-design strategies for intrapulse radar-embedded communications," IEEE Transactions on Aerospace and Electronic Systems, vol. 51, no. 4, pp. 2914–2931, 2015.

[44] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," arXiv preprint arXiv:1605.07277, 2016.

[45] 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.

[46] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," IEEE Transactions on Evolutionary Computation, 2019.

[47] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," arXiv preprint arXiv:1706.06083, 2017.

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

[49] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[50] MATLAB, version 9.2.0.538062 (R2017a). Natick, Massachusetts: The MathWorks Inc., 2017.

[51] F. Auger, P. Flandrin, P. Gonçalvès, and O. Lemoine, "Time-frequency toolbox," CNRS France-Rice University, vol. 46, 1996.

[52] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[53] F. Chollet, "How convolutional neural networks see the world," https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html, 2016, Accessed: 11/07/2018.

[54] Ettus Research, "USRP B200 Software Defined Radio," https://www.ettus.com/all-products/ub200-kit, 2019, Accessed: 20/04/2019.

[55] GEW Technologies, "MRR8001 Wideband Receiver Family," http://www.gew.co.za/electronic-warfare/products/mrr8001-wideband-receiver-family/, 2018, Accessed: 20/04/2019.

[56] Independent Communications Authority of South Africa, "National Radio Frequency Plan 2018 (NRFP-18)," https://www.icasa.org.za/uploads/files/National-Radio-Frequency-Plan-2018-41650.pdf, 2018, Accessed: 05/05/2019.

Appendix A: Ridge-based Classifier Data

For the tables and pages to follow, let the following definitions apply:

- W: The average width (number of frequency bins) of the TFR
- L: The average length (number of time bins) of the TFR
- m: The (absolute) gradient of the ridge plot
- A_x: The amplitude of step x as a fraction of half the width of the TFR
- S_x: The starting point of step x as a fraction of the length of the TFR
- E_x: The ending point of step x as a fraction of the length of the TFR
- P_x: The location of the peak of step x as a fraction of the length of the TFR

Tables A.1, A.2, A.3, A.4, and A.5 show the ridge-based classifier baseline datasets grouped into modulation schemes with 1, 2, 3, 4, and 5 steps respectively.

Table A.1: *Ridge-based classifier baseline dataset ($N = 1$)*

Class	Order/BW	W	L	m	A_1	S_1	E_1	P_1
Barker	2	18.6607	901.574	0.0004	0.8711	0.4168	0.593	0.4874
Barker	3	18.6378	901.3036	0.0018	0.8713	0.6059	0.7835	0.6767
Frank	2	18.6329	900.5899	0.0029	0.8705	0.7035	0.8798	0.774
Frank	4	37.595	904.74	0.0151	0.8901	0.4932	0.988	0.6601
Frank	6	80.8706	913.6542	0.042	0.9716	0.4215	0.9325	0.614
Frank	8	141.8047	916.526	0.0791	1.0028	0.3953	0.9163	0.5828
P3	16	39.0102	907.743	0.0214	0.9238	0.342	0.8553	0.5283
P3	36	84.8738	914.4827	0.0478	0.9583	0.2892	0.8016	0.5137
P3	64	146.2801	916.0785	0.0847	0.986	0.2763	0.7882	0.5108
P4	4	15.5651	899.4922	0.0041	0.7585	0.143	0.3613	0.2008

Table A.2: *Ridge-based classifier baseline dataset ($N = 2$)*

Class	Order/BW	W	L	m	A_1	A_2	S_1	S_2	E_1	E_2	P_1	P_2
FM	2 MHz	143.0608	913.9671	0.1544	0.9489	0.9489	0.0011	0.7561	0.2601	1	0.0011	0.9952
Barker	4	19.5609	901.4924	0.003	0.8286	0.8316	0.4232	0.7075	0.5887	0.8711	0.488	0.7732
Barker	5	19.8212	900.6927	0.0035	0.8197	0.8359	0.5383	0.7662	0.7008	0.9229	0.6034	0.8322
P1	4	31.1202	883.1816	0.03	0.7107	0.8627	0.0122	0.7728	0.2735	1	0.0366	0.9157
P1	6	42.9415	889.3257	0.0295	0.8016	0.8111	0.0011	0.833	0.2113	1	0.0054	0.9674
P1	8	110.024	879.5612	0.1233	0.9386	0.9488	0.0011	0.7577	0.2627	1	0.002	0.9877
P2	2	12.6127	896.4338	0.0006	0.6148	0.5237	0.1505	0.732	0.3363	0.9033	0.1921	0.7606
P3	4	19.0941	903.2327	0.0041	0.6049	0.6339	0.4269	0.7071	0.6063	0.9085	0.4817	0.7651
P4	16	30.9743	873.1877	0.0303	0.8106	0.8069	0.0011	0.7825	0.2922	1	0.0078	0.9554
P4	36	63.2388	872.9851	0.0691	0.9185	0.9119	0.0011	0.7649	0.2698	1	0.0017	0.9833
P4	64	109.6633	878.2532	0.1233	0.9563	0.9513	0.0011	0.7583	0.2621	1	0.0012	0.9919

Table A.3: *Ridge-based classifier baseline dataset ($N = 3$)*

Class	Order/BW	W	L	m	A_1	A_2	A_3	S_1	S_2	S_3	E_1	E_2	E_3	P_1	P_2	P_3
Barker	7	20.9501	895.1571	0.0044	0.7749	0.7775	0.817	0.3476	0.6759	0.8486	0.5062	0.8134	0.9651	0.4081	0.7369	0.8949
Barker	11	26.5887	878.9974	0.007	0.6087	0.8047	0.8176	0.2117	0.5261	0.8425	0.3277	0.7508	0.9999	0.2435	0.5958	0.9203
Barker	13	31.4169	885.4629	0.0105	0.5167	0.5159	0.8623	0.3532	0.5251	0.6911	0.448	0.6266	1	0.3705	0.5476	0.8383
P1	2	16.4222	900.3802	0.0042	0.4611	0.4621	0.4916	0.1662	0.4516	0.7272	0.3279	0.6118	0.9124	0.1937	0.4796	0.7603
P2	8	107.6273	902.7702	0.1259	1.0031	0.6437	0.9127	0.0011	0.638	0.796	0.2525	0.7412	1	0.002	0.6399	0.7973

Table A.4: *Ridge-based classifier baseline dataset ($N = 4$)*

Class	Order/BW	W	L	m	A_1	A_2	A_3	A_4	S_1	S_2	S_3	S_4	E_1	E_2	E_3	E_4	P_1	P_2	P_3	P_4
P2	4	31.0634	901.4049	0.019	0.7179	0.7597	0.7254	0.6309	0.0046	0.5582	0.8063	0.935	0.253	0.7192	0.8432	0.9635	0.027	0.6243	0.8083	0.9414

Table A.5: *Ridge-based classifier baseline dataset ($N = 5$)*

Class	Order/BW	W	L	m	A_1	A_2	A_3	A_4	A_5	S_1	S_2	S_3	S_4	S_5	E_1	E_2	E_3	E_4	E_5	P_1	P_2	P_3	P_4	P_5
P2	6	51.5978	900.8804	0.0471	0.5756	0.5975	0.5898	0.6046	1.2107	0.039	0.1297	0.571	0.6389	0.722	0.0904	0.1835	0.5876	0.6894	1	0.0748	0.1317	0.577	0.6405	0.9587

Appendix B: Confusion Matrices

This appendix provides the full confusion matrices for all classifiers in this study. Each cell in a confusion matrix shows the percentage of how many times the actual class on the x-axis was predicted as the predicted class on the y-axis.

Table B.1 is an index relating the confusion matrices in this appendix to their respective classifiers and the section in which they are presented and discussed.

Table B.1: *Confusion Matrix Index*

Table Number	Classifier	Section Number
Table B.2	MSE-Based Ridge Classifier	Section 5.1
Table B.3	TFR-Based CNN Classifier	Section 5.2
Table B.4	Instantaneous Measurement CNN Classifier	Section 5.3
Table B.5	Raw Complex Time Data CNN Classifier	Section 5.4
Table B.6	Split Complex Time Data CNN Classifier	Section 5.5

B.1 MSE-Based Ridge Classifier

Table B.2: *MSE-based ridge classifier confusion matrix*

Predicted Class \ Actual Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	74.29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22.89	0	0	0	0	0	0	0	0	0	0	0	15.24
2	0	96.19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	80.37	0	0	0	0	0	8.82	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	74.53	2.94	0	0	0	0	0	0	0	1.1	0	0	0	0	0	0	0	48.94	0	0	0	0	0	0	0
5	0	0	0	8.49	62.75	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4.26	0	0	0	0	0	0	0
6	0	0	0	0	0	92.86	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	39.6	2	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	91.21	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0
9	0	0	18.69	0	0	0	0	0	91.18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	98	8.16	0	0	0	0	0	0	0	0	0	0	8.49	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	2	73.47	19.83	0	0	0	0	0	0	0	0	0	16.98	17.71	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	4.08	75.86	0	0	0	0	0	0	0	0	0	0.94	2.08	39.83	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	81.32	0	0	0	2.35	0	8	0	0	0	0	0	1.05	0	0	0
14	0	0	0	3.77	24.51	7.14	0	6.59	0	0	0	0	0	71.03	5.88	0	16.47	0	0	6.67	15.96	0	0	0	0	22.52	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	93.14	0	0	0	0	0	0	0	0	0	0	0	0	0
16	7.62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34.94	0	0	0	25.56	0	0	0	0	0	0	1.02	7.62
17	0	1.9	0	0	0	0	0	0	0	0	0	0	16.48	0	0	0	81.18	0	0	0	0	0	0	0	1.05	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	1.1	0	0	0	0	47.52	47	4.44	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7.92	26	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.98	0	0	0.99	4	38.89	0	0	0	0	0	0	0	0
21	0	0	0	13.21	9.8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30.85	0	0	0	0	0	0	0
22	0	0.95	0	0	0	0	0	0	0	0	14.29	0	0	0	0	0	0	0	0	0	0	45.28	25	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	1.72	0	0	0	0	0	0	0	0	0	26.42	11.46	1.69	1.05	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	2.59	0	0	0	0	0	0	0	0	0	1.89	43.75	58.47	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	93.68	0	0	0
26	0	0.95	0.93	0	0	0	0	0	0	0	0	0	0	28.97	0	4.82	0	0	0	0	0	0	0	0	3.16	73.87	2.04	1.9
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4.82	0	0	0	5.56	0	0	0	0	0	3.6	96.94	0
28	18.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32.53	0	0	0	18.89	0	0	0	0	0	0	0	75.24
29	0	0	0	0	0	0	0	2.2	0	0	0	0	0	0	0	0	0	3.96	6	0	0	0	0	0	0	0	0	0

B.2 TFR-Based CNN Classifier

Table B.3: *TFR-based CNN classifier confusion matrix*

Predicted Class \ Actual Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	72.84	0	0	0	0	0	0.93	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0.88	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	27.16	0	0	0	0	0	99.07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	98.92	0	0	0	0	0	0	0	0	0	0	0	1.18	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0.94	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	96	0	0	0	1.75	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	97.37	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	1.08	0	0	0	0	0	0	0	0	0	0	0	98.82	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	99.06	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	100

B.3 Instantaneous Measurement CNN Classifier

Table B.4: *Instantaneous measurement CNN classifier confusion matrix*

		Actual Class																											
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Predicted Class	1	95.74	0	0	0	0	0	0	0	0	0	0.98	0	0	0	0	0	0	0	0	0	0	0	0	1.12	0	0	0	1.92
	2	0	82.73	0	2.02	0	0	1.85	0	0	0	0	0	0.89	0	0	0	1.05	0	0	0	0	0	0	0	0	0	0	0
	3	0	2.73	92.63	1.01	0	1.87	0	0	0.98	0.97	0	0	0.89	0	1.1	0	4.21	0	0	0	0	1.01	0	0	0.97	0	0	0
	4	0	1.82	1.05	72.73	3.45	1.87	0	0	0	0.97	0	0	0	0	0	0	0	1.04	0	0	12.84	1.01	0	0	0	0.94	0	0
	5	0	0.91	1.05	3.03	80.46	3.74	0	1.16	0	0	0	0	0.89	0	0	0	2.11	1.04	0	0	0.92	1.01	0	0	0	0	0	0
	6	0	0	0	2.02	0	77.57	0	0	0	0	0	0	1.79	1.03	0	0	1.05	0	0	0	0.92	0	0	0	0	0.94	0	0
	7	0	0	1.05	2.02	6.9	1.87	87.96	0	0.98	0.97	0	0	0.89	1.03	0	0	2.11	0	0	0	0.92	0	0	0	0	0	0	0
	8	0	0.91	1.05	1.01	0	2.8	0.93	89.53	1.96	0.97	0	0	0	0	0	0	1.05	0	0	0	0	0	0	0	0	0	0	0
	9	0	0.91	2.11	0	1.15	1.87	0.93	2.33	89.22	0	0	0	0	0	0	0	4.21	2.08	0	0	1.83	0	0	0	0	1.89	0	0
	10	0	0	0	0	2.3	0	0	0	0	92.23	0.98	0	0	0	0	0	0	0	0	0	0	1.01	0	0	0	0	0	0
	11	0	0	0	1.01	0	0	0	1.16	0	0.97	96.08	0	0.89	0	0	0	0	0	0	0	0	1.01	1.98	0	0	0.94	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	93.97	0	0	0	0	0	0	0	0	0	0	0	11.24	0	0	0	0
	13	0	3.64	0	4.04	0	1.87	0.93	0	0	0	0	0	81.25	0	0	0	2.11	0	0	0	1.83	1.01	0	0	2.91	0.94	0	0
	14	0	0.91	0	2.02	0	0	0	0	0.98	0	0	0	0.89	87.63	0	0	0	0	1.14	0	0	0	0	0	0	12.26	1.03	0
	15	0	0.91	0	0	0	0	0.93	0	0	0.97	0	0	0.89	0	94.51	0	0	2.08	4.55	0	0.92	0	0	0	0	0.94	1.03	0
	16	1.06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35.64	0	0	0	2.91	0	0	0	0	0	0	2.06	27.88
	17	0	0	0	0	1.15	0	0	0	0.98	0	0	0	0	1.03	0	0	69.47	0	0	0	0	0	0	0	1.94	0.94	0	0
	18	0	0	0	1.01	1.15	0	1.85	1.16	0	0	0	0	0	1.03	1.1	0	0	85.42	0	0	0	0	0	0	0	0.94	2.06	0
	19	1.06	0.91	0	0	0	0	0	0	0	0.97	0	0	0	2.06	2.2	0.99	0	0	86.36	0	0	0	0	0	0.97	2.83	1.03	0
	20	0	0	0	0	0	0	0	0	0	0	0.98	0	0	0	0	1.98	0	2.08	2.27	94.17	0	0	0	0	0	0	1.03	0.96
	21	0	1.82	0	7.07	0	0	0.93	1.16	1.96	0	0	0	1.79	0	0	0	1.05	1.04	1.14	0	71.56	0	0	0	0	0	0	0
	22	0	0.91	0	0	0	0.93	0.93	2.33	0	0.97	0	0	0	0	0	0	0	0	1.14	0	0.92	91.92	2.97	0	0	0	0	0
	23	0	0	1.05	0	0	0	0	1.16	0	0	0.98	0	0	0	0	0	0	0	0	0	1.83	2.02	93.07	0	0	0	0	0
	24	0	0	0	0	0	0	0	0	0	0	0	6.03	0	0	0	0	0	0	0	0	0	0	1.98	87.64	0	0	0	0
	25	0	0.91	0	1.01	2.3	4.67	2.78	0	2.94	0	0	0	8.04	3.09	0	0	11.58	2.08	0	0	2.75	0	0	0	93.2	0.94	0	0
	26	0	0	0	0	0	0.93	0	0	0	0	0	0	0.89	2.06	1.1	0	0	3.13	1.14	0.97	2.75	0	0	0	0	75.47	0	0
	27	0	0	0	0	1.15	0	0	0	0	0	0	0	0	0	0	0.99	0	0	2.27	0.97	0	0	0	0	0	0	90.72	0.96
	28	2.13	0	0	0	0	0	0	0	0	0	0	0	0	1.03	0	60.4	0	0	0	0.97	0	0	0	0	0	0	1.03	68.27

B.4 Raw Complex Time Data CNN Classifier

Table B.5: *Raw complex time data CNN classifier confusion matrix*

		Actual Class																											
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Predicted Class	1	98.96	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.99
	2	0	88.5	0.97	0	1.04	0	0	0	0	0	0	0	0	0	0	0	0.93	0	0	0	1	0	0	0	0	0	0	0
	3	0	0.88	85.44	0	1.04	0	0	0	2.88	0	0	0	0	0	0	0	1.85	0	0	0	0	0	0	0	0	0	0	0
	4	0	0.88	1.94	81.44	0	1.77	0	1.06	0.96	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0
	5	0	0	3.88	1.03	89.58	0	0	0	0	0	0	0	0	0	0	0	0.93	0	0	0	1	0	0	0	0	0	0	0
	6	0	1.77	0.97	3.09	0	92.04	0	2.13	6.73	0	0	0	1.06	0	0	0	0.93	0	0	0	3	0	0	0	0	0	0	0
	7	0	0	0.97	0	2.08	1.77	100	0	0	0	0	0	0	1.18	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	1.77	0	93.62	0	2.8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	3.88	0	1.04	0.88	0	0	83.65	0	0	0	0	0	0	0	4.63	0	0	0	1	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	2.13	0	90.65	0	0	0	0	0	0	0	0	0	0	0	5.88	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	0.93	98.77	0	0	0	0	0	0	0	0	1.02	0	0.98	11.63	0	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	86.36	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0
	13	0	0.88	0	0	0	0	0	0	0.96	0	0	0	87.23	0	0	0	2.78	0	0	0	2	0	0	0	3.42	0	0	0
	14	0	0.88	0.97	2.06	0	0	0	0	0	0	0	0	1.06	89.41	0	0	0	0	0	0	0	0	0	0	0	8.49	0	0
	15	0	0	0	0	0	0.88	0	0	0	0	0	0	0	1.18	96.88	0	0	0	1.05	0	0	0.98	0	0	0	0	0	0
	16	1.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	62.37	0	0	0	1.02	0	0	0	0	0	0	0	49.5
	17	0	1.77	0.97	0	0	0	0	0	0.96	0	0	0	1.06	0	0	0	80.56	0	0	0	0	0	0	0	3.42	0	0	0
	18	0	0	0	0	0	0	0	1.06	0	0	0	0	0	0	0	0	0	94.85	0	0	0	0	0	0	0	3.77	0	0
	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3.13	0	0	1.03	98.95	0	0	0.98	0	0	0	0	0	0
	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6.45	0	0	0	97.96	0	0	0	0	0	0	1.2	0.99
	21	0	4.42	0	12.37	5.21	0.88	0	0	2.88	0	0	0	3.19	0	0	0	0	0	0	0	71	0	0	0	0	0	0	0
	22	0	0	0	0	0	0	0	0	0	5.61	0	0	0	0	0	0	0	0	0	0	0	91.18	0	0	0	0	0	0
	23	0	0	0	0	0	0	0	0	0	0	1.23	1.82	0	0	0	0	0	0	0	0	0	0	88.37	0	0	0	0	0
	24	0	0	0	0	0	0	0	0	0	0	0	11.82	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0
	25	0	0	0	0	0	0	0	0	0	0	0	0	6.38	1.18	0	0	7.41	0	0	0	1	0	0	0	93.16	0	0	0
	26	0	0	0	0	0	0	0	0	0.96	0	0	0	0	7.06	0	0	0	2.06	0	0	0	0	0	0	0	86.79	0	0
	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.06	0	0	0	0	0	0	0	0	0.94	97.59	0.99
	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31.18	0	0	0	0	0	0	0	0	0	0	1.2	47.52

B.5 Split I and Q Complex Time Data Classifiers

Table B.6: *Split complex time data CNN classifier confusion matrix*

		Actual Class																											
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Predicted Class	1	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	94.95	0	3.06	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0.98	0	0	0	0	0	0	0
	3	0	2.02	95.65	0	2.06	0	0.85	0	8	0	0	0	2.02	0	0	0	0	0	0	0	2.94	0	0	0	1.05	0	0	0
	4	0	0	0	83.67	2.06	3.19	0	0	1	0	0	0	1.01	0	0	0	0	0	0	0	25.49	0	0	0	0	0	0	0
	5	0	0	0	0	90.72	0	0.85	0.94	4	0	0	0	0	0	0	0	0	0	0	0	0.98	0	0	0	0	1.16	0	0
	6	0	0	0	3.06	1.03	93.62	1.71	0.94	0	0	0	0	1.01	0	0.86	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	1.06	95.73	0	0	2.02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	1.06	0	90.57	0	0	0	0	0	0	0	0	0	1.04	0	0	0	0	0	0	0	0	0	0
	9	0	0	2.17	1.02	1.03	0	0	0	82	0	0	0	1.01	0	0	0	2.56	0	0	0	0	0	0	0	0	0	0	0
	10	0	0	0	0	1.03	0	0	5.66	0	91.92	0	0	0	0	0	0	0	0	0	0	0	4.17	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	2.02	87.36	0	0	0	0	0	0	0	0	0	0	0	15.74	0	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	68.97	0	0	0	0	0	0	0	0.9	0	0	0	19.81	0	0	1.09	0
	13	0	1.01	0	0	1.06	0	0	0	0	0	0	0	86.87	0	0	0	4.27	0	0	0	3.92	1.04	0	0	5.26	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	0	88.54	0.86	0	0	0	0	0	0	0	0	0	0	11.63	0	0
	15	0	1.01	0	0	0	0	0	0	0	0	0	0	0	1.04	96.55	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	78.7	0	0	0	0.9	0	0	0	0	0	0	0	26
	17	0	0	2.17	0	0	0	0	0	1	0	0	0	2.02	0	0	0	81.2	0	0	0	0	0	0	0	5.26	0	0	0
	18	0	0	0	0	1.03	0	0	0	2	0	0	0	0	1.04	1.72	0	0	97.92	0.96	0	0	0	0	0	0	1.16	0	0
	19	0	0	0	0	0	0	0.85	0	0	1.01	0	0	0	1.04	0	0	0	0	99.04	0	0	2.08	1.85	0	0	0	0	0
	20	0	0	0	0	0	0	0	0	0	0	1.15	0	0	0	0	0.93	0	0	0	97.3	0	0	0	0	0	0	1.09	0
	21	0	1.01	0	9.18	1.03	0	0	1.89	0	0	0	0	1.01	0	0	0	0	0	0	0	65.69	0	0	0	0	0	0	0
	22	0	0	0	0	0	0	0	0	0	3.03	1.15	0	0	0	0	0	0	0	0	0	0	92.71	0	0	0	0	0	0
	23	0	0	0	0	0	0	0	0	0	0	10.34	1.15	0	0	0	0	0	0	0	0	0	0	82.41	0.94	0	0	0	0
	24	0	0	0	0	0	0	0	0	0	0	0	29.89	0	0	0	0	0	0	0	0	0	0	0	79.25	0	0	0	0
	25	0	0	0	0	0	0	0	0	0	0	0	0	5.05	1.04	0	0	11.97	0	0	0	0	0	0	0	88.42	0	0	0
	26	0	0	0	0	0	0	0	0	1	0	0	0	0	7.29	0	0	0	1.04	0	0	0	0	0	0	0	86.05	0	0
	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9	0	0	0	0	0	0	97.83	0
	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20.37	0	0	0	0	0	0	0	0	0	0	0	74

Appendix Removed Due to Visible and Unremovable Signature(s)