Radar-Based Multi-Target Classification Using Deep Learning

Classification of Human Activities and Moving Targets using

Convolutional Neural Networks

Presented by:
Nyasha Ernest Mashanda

Prepared for:
Mr Neil Watson, Dr Yunus Abdul Gaffar and Mr Robert Berndt
Dept. of Statistical Sciences
University of Cape Town

Submitted to the Department of Statistical Sciences at the University of Cape Town in partial fulfilment of the academic requirements for a Master of Science degree in Data Science.

31 May 2022

Acknowledgements

I would like to express my gratitude to the following people whose contribution has been invaluable over the course of this project and my degree:

My family, friends and loved ones for the love and support you showed me. You helped me to fight through the storm. Special thanks to my father (**Lameck Mashanda**) and mother (**Monica Samusodza Mashanda**) for encouraging me to continue working hard. You are my best role models.

My Supervisors, Mr Neil Watson, Dr Yunus Abdul Gaffar and Mr Robert Berndt for their support and guidance throughout this dissertation. Through listening to your wise instructions, I learned more about Machine Learning, radar and above all, working in a team.

The Mandela Rhodes Foundation for providing me with a scholarship to pursue my studies. Beyond the scholarship, you allowed me to grow as a young leader who wants to make the world a better place.

Council for Scientific and Industrial Research (CSIR) and the Department of Science and Innovation (DSI) for providing the measured data used for this work.

Abstract

Real-time, radar-based human activity and target recognition has several applications in various fields. Examples include hand gesture recognition, border and home surveillance, pedestrian recognition for automotive safety and fall detection for assisted living. This dissertation sought to improve the speed and accuracy of a previously developed model classifying human activity and targets using radar data for outdoor surveillance purposes. An improvement in accuracy and speed of classification helps surveillance systems to provide reliable results on time. For example, the results can be used to intercept trespassers, poachers or smugglers.

To achieve these objectives, radar data was collected using a C-band pulse-Doppler radar and converted to spectrograms using the Short-time Fourier transform (STFT) algorithm. Spectrograms of the following classes were utilised in classification: one human walking, two humans walking, one human running, moving vehicles, a swinging sphere and clutter/noise.

A seven-layer residual network was proposed, which utilised batch normalisation (BN), global average pooling (GAP), and residual connections to achieve a classification accuracy of 92.90% and 87.72% on the validation and test data, respectively. Compared to the previously proposed model, this represented a 10% improvement in accuracy on the validation data and a 3% improvement on the test data.

Applying model quantisation provided up to 3.8 times speedup in inference, with a less than 0.4% accuracy drop on both the validation and test data. The quantised model could support a range of up to 89.91 kilometres in real-time, allowing it to be used in radars that operate within this range.

Nomenclature

Adam Adaptive Moment Estimation

BN Batch Normalisation

CAE Convolutional Auto-Encoder

CNN Convolutional Neural Network

CP Coherent Processing

DL Deep Learning

GAP Global Average Pooling

Lidar Light Detection and Ranging

ML Machine Learning

Nadam Nesterov-accelerated Adaptive Moment Estimation

PSNR Peak Signal-to-Noise Ratio

PTQ Post-Training Quantisation

Radar Radio Detection and Ranging

ReLU Rectified Linear Unit

ResNet Residual Networks

SNR Signal-to-Noise Ratio

STFT Short-time Fourier Transform

List of Figures

List of Tables

Contents

1 Introduction

1.1 Background

Human activity and target classification has been gaining interest in recent years due to its applications in pedestrian recognition for automotive safety [1], fall detection for assisted living [2] and surveillance systems for border control and security [3, 4]. Surveillance applications are relevant to South Africa and other nations worldwide experiencing increasing poaching activities. For example, in December 2021, 24 rhinos were killed in just two weeks due to poaching [5]. To prevent the extinction of endangered species such as elephants and rhinos, it is crucial to have reliable area surveillance systems that can recognise human activity or vehicles used for poaching. These systems can also be used for home security, border control and protection of farmlands.

Different sensor systems have been used for human activity and target classification, including cameras [6], Lidar [7] and radar [8]. Unlike cameras or Lidar, the performance of a radar sensor is less sensitive to varying weather conditions and different levels of light [9]. Furthermore, radar offers a more extended detection range than optical sensors [8] and can detect targets behind opaque objects [10]. These advantages make radar more attractive for outdoor surveillance systems.

Various commercial radar systems have been developed for outdoor surveillance. These include Hensoldt's Spexer radars [11] which were designed for border security systems and the protection of critical infrastructure, and InnoSent radars [12] for home security and perimeter surveillance. The demand for radar surveillance systems is expected to surge due to the increasing demand for border security and military applications. Consequently, the Surveillance radar market is projected to grow from US$ 8.0 billion to US$ 11.5 billion by 2025 [13].

1.2 Problem Statement

Radar can collect the range or Doppler information of moving targets. In addressing the challenge of classifying human activities and targets using radar data, researchers have utilised spectrograms [14, 15], time-range maps [16, 17] and time-range-Doppler maps [18, 19]. However, spectrograms are commonly used because they contain sufficient information for classifying different activities or targets [20].

Fig. 1.1 shows spectrograms of a walking human, a dog and a horse. Note that spectrograms show how the Doppler frequency of targets changes with time and are sometimes called time-Doppler maps. The spectrograms in Fig. 1.1 show Doppler returns from the torso, hands and legs. A torso has a lower Doppler frequency since it moves slower than hands or legs. Hands and legs show repetitive Doppler patterns as they move back and forth periodically.

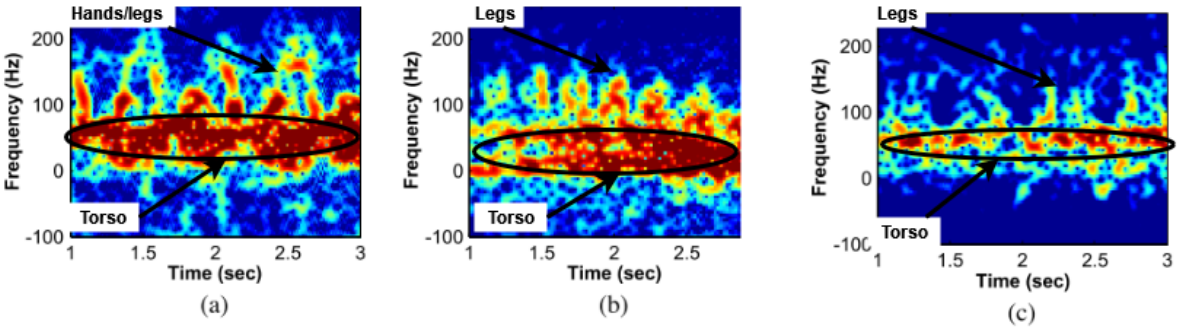

Figure 1.1: Spectrograms of (a) human, (b) a dog and (c) a horse [21].

The classification of spectrograms typically begins with extracting a pre-defined set of features from the spectrograms. Examples of such pre-defined features include those related to physical characteristics of the target [22], discrete cosine transform coefficients [23] and linear predictive coding coefficients [24]. A subset of the extracted features may then be chosen using feature selection [25] or dimension reduction [26]. The subset of features is then fed to a classifier of choice, such as support vector machines (SVMs) [27] or random forests (RFs) [28].

This approach requires human expertise and significant domain knowledge to define the features that should be extracted. Deep learning (DL) offers an alternative approach, where features are learned from spectrograms automatically without human supervision. In one of the first works on DL for micro-Doppler classification, [21] used a convolutional neural network (CNN) to classify six classes (running, walking, walking while holding a stick, crawling, boxing while moving forward, boxing while standing in place, and sitting still) at a correct classification rate of 90.9%, similar to that previously achieved with SVMs [29]. Later, [30] used a 2-layer Auto-Encoder structure with a total of 120 data samples to classify four classes of activities (falling, sitting, bending, and walking) and achieved an accuracy of 87%.

In classifying human activity and targets, the inference speed of CNNs must be considered. Inference refers to classifying new data after model training. High inference speeds allow real-time support in radar systems, assisting border authorities, nature reserve or security personnel to react promptly. Researchers are currently exploring methods of improving the inference speed of DL algorithms. Different methods of speeding up inference have been proposed, for example, quantisation [31] and pruning [32]. However, while these methods may improve a model's inference throughput or speed, they may degrade the model's performance [33].

This work focuses on the accurate and timely classification of human activity or targets that may be encountered in outdoor environments. Six classes were considered: one human

walking, one human running, two humans walking, moving vehicle/s, clutter or noise (which represents the absence of a human or vehicle) and a swinging sphere (which was added to diversify the data). The results of this study can be used to improve the accuracy and speed of detection in outdoor surveillance systems so that unwanted presence or activity can be detected correctly in a timely manner.

1.3 Objectives of this Study

The work in this study sought to achieve the following objectives:

1. Build a CNN model that would improve the accuracy results from previously done research [34] without over-fitting the model on the data. A higher accuracy gives confidence that the model will provide reliable results for surveillance purposes.
2. Improve the inference throughput speed of the proposed model while maintaining accuracy so that the model can be deployed in real-time surveillance systems.

1.4 Project Scope

Firstly, to achieve the objectives mentioned above, this study used data supplied by the Council for Scientific and Industrial Research (CSIR), collected using a C-band pulse-Doppler radar. The CSIR data comprised only six classes, namely: one human walking, one human running, two humans walking, moving vehicle/s, clutter and a swinging sphere.

Secondly, only spectrograms were considered for classification. Previous research on human activity and target classification has shown that micro-Doppler signatures found in spectrograms contain sufficiently unique and diverse information that can be used to distinguish different human activities or targets [35].

Thirdly, only CNNs were investigated for classifying spectrograms. Amongst other DL algorithms such as long short-term memory (LSTM) and recurrent neural networks (RNNs), CNNs have been the most effective in visual recognition tasks such as image classification [36]. Although spectrograms are not natural images, results from previous research on human activity and target classification have shown that CNNs have adapted well to the problem [37].

Lastly, only post training quantisation (PTQ) [38] was investigated for inference throughput improvement. Due to limited time, other methods proposed in research, such as quantisation aware training (QAT) [38] and Pruning, were not explored.

1.5 Research Contribution

There are generally two approaches to building machine learning models: from the ground up or using a transfer learning process [14]. This work develops a residual network model from the ground up for human activity and target classification applications. At the time of

writing this thesis, only one paper [39] had investigated the application of residual networks to address the problem, albeit in a transfer learning fashion.

Although there is much research on improving model throughput in the DL community, its application to radar classification has not been addressed in the literature. This research investigates PTQ to improve the proposed model's throughput without degrading the model's performance.

1.6 Report Outline

Chapter 2 reviews the basic operational principles of a pulse-Doppler radar. A description of the radar data processing to create spectrograms using collected data follows this. After that, CNNs are discussed, focusing on the basic principles, followed by a critical review of the literature on classifying human activity and targets using CNNs. Lastly, principles of PTQ are presented, followed by a review of the literature which applies the method.

Chapter 3 provides an overview of the research methods to achieve the objectives listed in Section 1.3. Data collection and pre-processing methods are also discussed. This is followed by a summary of the processes applied in building CNN models and performing PTQ. Finally, the details regarding the hardware and software used to perform experiments are presented.

Chapter 4 provides more details on the design decisions used to generate spectrograms from radar data. The data splitting into training, validation and test set is also discussed.

Chapter 5 focuses on implementing CNN models and their accuracy results on the validation and test datasets.

Chapter 6 provides details on the implementation and results of PTQ on the best model. Furthermore, the effect of PTQ on the model's speed and accuracy was analysed in this chapter. This was followed by investigating the maximum range that the model could support in real-time.

Chapter 7 discusses the research conclusions and recommendations for future work.

The Appendix provides details on additional experiments that were carried out after achieving the objectives.

2 Literature Review

This section covers the review of the relevant literature to the project. Section 2.1 examines the theory behind pulse-Doppler Phased Array radar used for data collection. This is followed by a summary of the theory of computing a spectrogram in Section 2.2. Section 2.3 focuses on the methods of extracting training, validation and testing data from spectrograms. Section 2.4 explores the theory behind CNNs and their applications in human activity and target classification. This is followed by a review of techniques for improving a model's inference speed, focusing on PTQ in Section 2.5. Section 2.6 suggests an approach to achieving this study's objectives based on the literature.

2.1 Radar Data Measurement

Continuous-wave (CW) [40, 41], pulse-Doppler [42, 43] and frequency-modulated continuous wave (FMCW) [44, 45] radars have been used to collect data for human activity and target classification. Data obtained using these radars can be used to extract the Doppler information of targets which is essential for distinguishing moving targets and corresponding activities. In this study, a C-band pulse-Doppler Phased Array radar was used to take measurements of different targets and human activities. Note that a C-band radar operates within the 4 GHz to 8 GHz frequency range.

Pulse-Doppler radars transmit radio pulses and then receive pulses reflected by illuminated objects. The pulses are generated by a radio frequency (RF) generator before passing through a Power Divider module which feeds the signal to both the transmitting antenna and mixer, as shown in Fig. 2.1.

Targets in the radar's path reflect the transmitted signal. The receiving antenna then collects the reflected signal. Note that the power of the received signal is given by the radar equation:

$$P_r = \frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 (R)^4 L} \tag{1}$$

where

- P_t = transmit power
- G_t = transmit antenna gain
- G_r = receive antenna gain
- λ = operating wavelength
- σ = target radar cross section
- R = target range

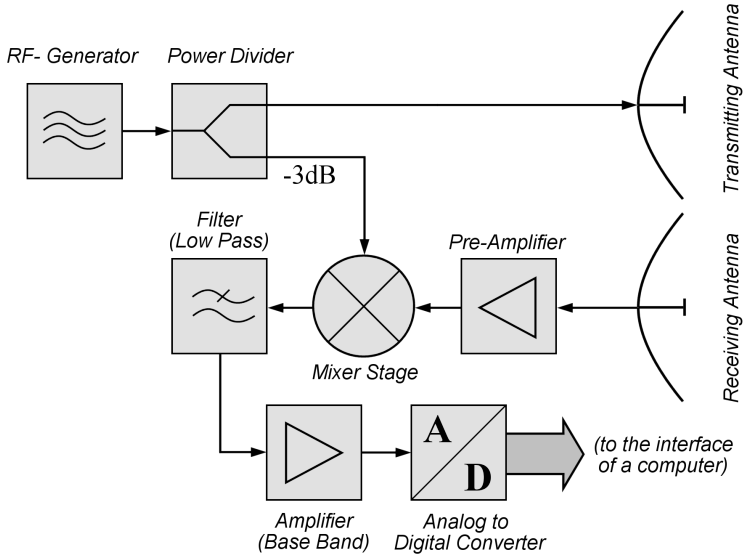

Figure 2.1: Block diagram of a simple pulse-Doppler radar module [46].

- L = loss, which includes atmospheric losses during propagation of the electromagnetic waves to and from the target, fluctuation losses during the reflection and internal attenuation factors of the radar set on the transmitting and receiving paths.

Thus, there is an inverse relationship between the received power and the range of the targets. Consequently, targets at longer ranges are more difficult to detect than targets closer to the radar.

As shown in Fig. 2.1, the received signal is amplified and fed to a mixer. The mixing process provides a way of extracting the frequency difference between the carrier frequency of the transmitted signal and the received signals. The output of the mixer passes through a low-pass filter which filters out unwanted signals. The resulting baseband signal is amplified and then digitised afterwards using an analog to digital converter (ADC).

In a pulse-Doppler Phased Array system, the antenna elements are fed a phase-shifted signal to create a beam pattern that is electronically steered to allow transmission in different directions. The electronic steering is more flexible and requires less maintenance than the mechanical steering to measure targets in different directions. The received signals are processed as previously described to obtain digitised baseband samples. Additional processing includes pulse-compression [47] to improve the signal-to-noise ratio (SNR) and beamforming [48] to attenuate signals that did not come from the direction in which there is a target of interest.

The processed baseband samples of the received echoes are ordered by their arrival time. Since time is proportional to the target's distance away from the radar, each sample is typically

referred to as a range bin. Samples belonging to the same range bin from different pulses are separated by the pulse repetition interval (PRI), the duration between consecutive pulses.

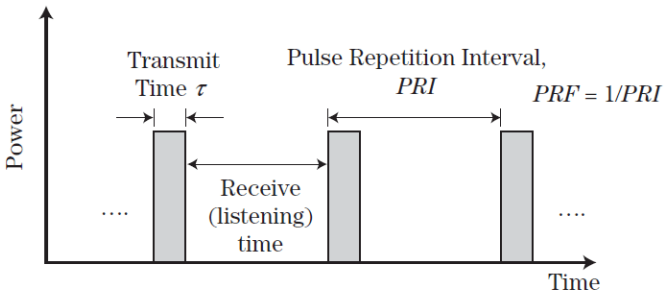

Figure 2.2: Waveform of a pulse radar [49].

Fig. 2.2 shows transmitted pulses separated by the PRI. The pulse width (τ) is the duration of the pulse. The range resolution ρ of an unmodulated pulse is given by:

$$\rho \geq \frac{c\tau}{2} \approx \frac{c}{2B} \tag{2}$$

where τ is the duration of the pulse and $B = \frac{1}{\tau}$ is the bandwidth of the signal.

Spectrograms are then generated by performing a Short-time Fourier transform (STFT) of the processed baseband samples.

2.2 Radar Data Pre-processing

2.2.1 Short-time Fourier Transform

The STFT is a valuable tool for analysing the frequency content of a non-stationary signal [50] as it gives a time-frequency domain representation of the signal. To compute the STFT, a signal is broken down into windows, which usually overlap. The discrete Fourier transform (DFT) is then applied to each window. Thus, the STFT of a signal $x[k]$ is defined as:

$$X_{STFT}[m,n] = \sum_{k=0}^{N-1} x[k]g[k-m]e^{-j2\pi nk/N} \tag{3}$$

where $g[k]$ denotes an N-point window function, m is the time index, and n is the frequency index. A spectrogram is created by applying the modulus of the STFT result:

$$S[m,n] = |X_{STFT}[m,n]|^2. \tag{4}$$

A spectrogram is thus a two-dimensional representation of the energy spectral density of the signal as it changes over time. In this study, spectrograms were used as input to CNNs, which extract relevant features in the data for classification purposes.

2.2.2 Short-time Fourier Transform Parameters

There are three main parameters to consider when computing the STFT: window type, window length, and window overlap. The window type affects the quality of Doppler information conveyed by the spectrogram. Rectangular windows are less commonly used because they give rise to high side-lobe levels in the frequency domain, as shown in Fig. 2.3. This results in the masking of weak frequency components close to strong components in the spectrum. Tapering is applied to windows in the time domain to reduce the side-lobe levels in the frequency domain. However, tapering the windows results in an increase in the width of the main lobe, which degrades the frequency resolution. Tapered windows such as Hamming and Hanning are commonly used because they offer a balance between side-lobe levels and an increase in main-lobe width [51].

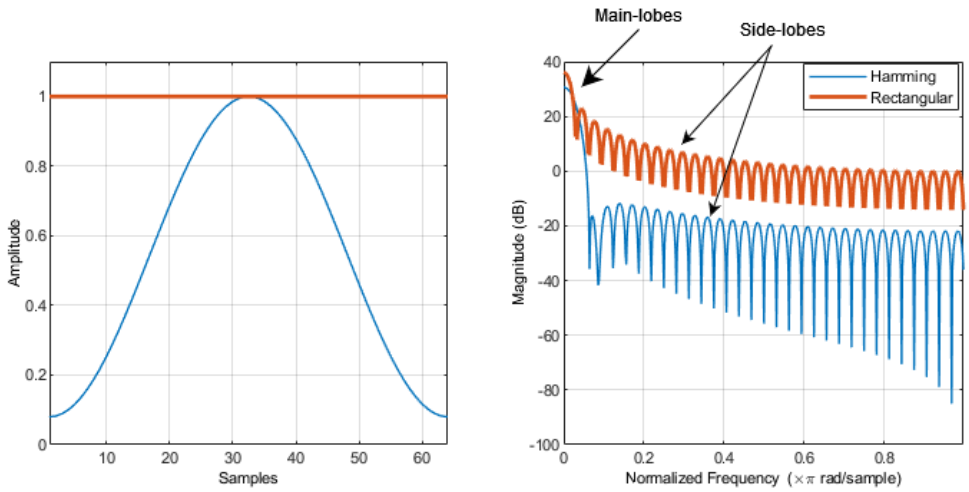

Figure 2.3: Rectangular and Hamming windows in the time domain (left) and frequency domain (right).

The tapered windows are applied in an overlapping fashion, as shown in Fig. 2.4, to retain the data in the window's tapered regions, which is potentially lost. An overlap factor of 50% is recommended for both the Hanning and Hamming windows as it allows all data samples to be weighted equally with minimal computational effort [52].

The window length (N) used to compute the STFT for each window is equivalent to the coherent processing (CP) length. The CP length determines the SNR of a signal in the frequency domain, hence the ease of resolving frequency components in the data. The SNR value is computed as the ratio of the signal's power to the noise's power [49]. If $x[n]$ is a distorted signal, formulated by:

$$x[n] = s[n] + v[n], \qquad 0 \leq n \leq N-1 \tag{5}$$

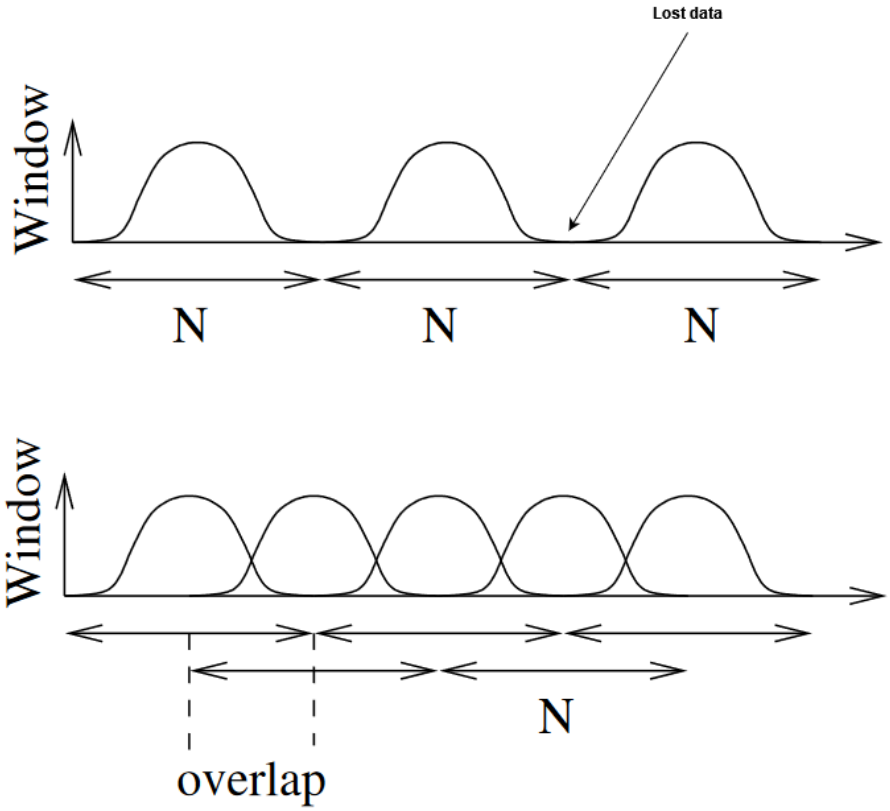

Figure 2.4: Segmented data stream with non-overlapping (top) and overlapping (bottom) windows [52].

where $s[n]$ is the pure signal and $v[n]$ is additive white noise with mean zero and variance σ^2 and N is the number of samples taken from the signal, then the SNR is defined as:

$$\text{SNR} = \frac{\sum_{n=0}^{N-1}[x[n]]^2}{\sigma^2}. \tag{6}$$

If the samples within the CP length are in phase, increasing the CP length increases the signal's power by N^2, which increases the SNR. Once the CP length includes samples that are out phase, the SNR gain is reduced [53]. Moreover, the CP length will determine the number of rows in the spectrogram. For example, when researchers use a CP length in the order of thousands for calculating the STFT, the spectrogram will also have thousands of rows.

Spectrograms with a very high number of rows are computationally expensive to process in CNNs. Therefore, researchers have to down-sample or resize the output [14]. This approach adds processing overhead, which increases the computational cost of classifying spectrograms. An optimal window length offers a reasonable compromise between the associated computational cost and the achieved SNR.

Using appropriately selected parameter values, the STFT can be computed to generate spectrograms for human activity and target classification.

2.3 Data Preparation

CNN require data of fixed dimensions (width, height, or depth). Thus, the number of rows and columns in the spectrogram data used for training, validation, and testing should be pre-defined. The CP length determines the number of rows, and the number of columns is determined by the specified time duration of observing an activity. Researchers have used durations that typically range from 2 to 4 seconds [14, 39, 54] for human activity and target classification. While the specific reasons for choosing these durations are not clear, these durations are sufficient to capture one period of repetitive motions such as walking. Additional research was carried out in this study to determine the spectrogram duration that would improve classification accuracy in Appendix A.

After selecting the spectrogram duration, segments of the spectrograms are then extracted using a sliding window [55] or via random sampling [56] along the time axis. The literature does not discuss which approach provides better results among these two methods. Therefore, additional research was also carried out to investigate the method that would yield better classification accuracy in Appendix A.

2.4 Convolutional Neural Networks for Spectrogram Classification

CNNs are a popular class of networks used in image classification. They can also be used to classify spectrograms [14, 15]. CNNs also form the backbone of most networks that have dominated the ImageNet [36] competition, for example, ResNets [57], EfficientNets [58] and most recently, CoAtNets [59].

2.4.1 Convolutional Neural Networks Background

CNNs can consist of a combination of the following layers:

- A convolution layer that is made up of kernels (matrices of weights) used to learn feature representations of the inputs. The feature representations are stored as feature maps. Feature maps are obtained by convolving the layer's input with a learned kernel and then applying an element-wise nonlinear activation function on the convolved results. The activation function introduces nonlinearities to CNN, which are desirable for multi-layer networks to detect nonlinear features [60]. Typical activation functions are Sigmoid, Tanh [61] and ReLU [62]. However, ReLU is commonly used because it helps to avoid the vanishing gradient problem and is computationally cheaper than Sigmoid and Tanh functions [63].
- A max-pooling layer that typically follows a convolutional layer. It is used to achieve shift or spatial invariance by reducing the resolution of the feature map [64]. Shift invariance allows CNNs to locate features/objects despite their location in the input data.
- A fully connected layer that follows the convolution and max-pooling layers (also known as feature extractors). It consists of fully connected neurons with weights used to learn the mapping between the feature maps and input labels.

Concatenating convolutional and max-pooling layers, as shown in Fig. 2.5, can allow CNNs to learn low to high-level features [65] that can be used for classification. For example, if the input data is an image, the kernels in the first convolutional layers learn low-level features such as edges, curves and shapes. Kernels in later layers learn more complex features such as textures and patterns in the data. Kernels in the last convolutional layers learn high-level features such as objects or parts of objects. Finally, the fully connected layers then learn to classify the data based on the extracted high-level features [65].

The training process of CNNs involves making predictions based on the current weight values and updating the weights based on the error of the predictions. The back-propagation algorithm [66] uses the chain rule to update the network weights based on rules defined by a given optimisation method.

There are various optimisation methods, including: Stochastic Gradient Descent [67], Adam

[68], AdaDelta [69], Adamax [68] and Nadam [70]. Adam is a popular optimiser because it achieves fast convergence [71] and is computationally efficient [68].

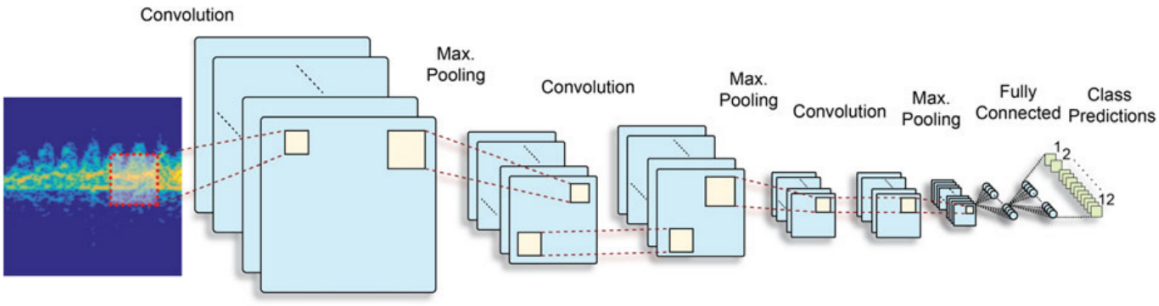

Figure 2.5: Three-layer CNN with two fully connected layers [40].

2.4.2 Convolutional Neural Networks in Radar-Based Classification

Various types of CNNs have been used in classifying human activity and targets, which include vanilla [21,41,55], convolutional auto-encoder (CAE) [37,40], residual networks (ResNets) [39] and hybrids of CNNs and LSTMs [72].

A vanilla CNN architecture comprises a combination of convolutional and max-pooling layers, using a fully connected network as a classifier. CAE (example shown in Fig. 2.6) utilise unsupervised pre-training to initialise the weights in the subsequent convolutional layers through a decoder-encoder structure. After unsupervised pre-training, the decoder is removed, and fully connected layers are added as classifiers at the end of the encoder. A hybrid network can include a combination of CNN and LSTM for spatial and temporal feature extraction [72], before applying a fully connected network for classification.

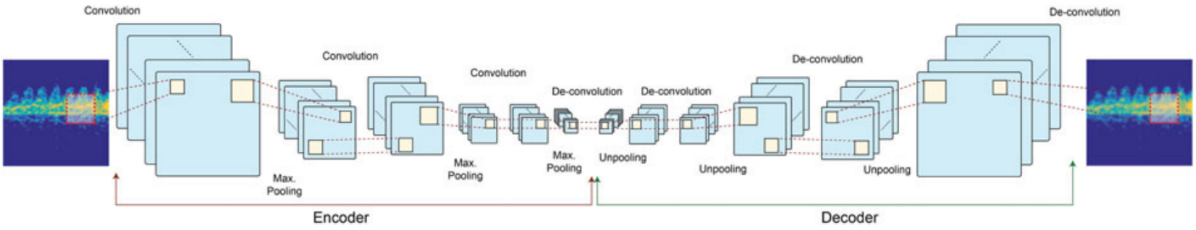

Figure 2.6: CAE with a decoder-encoder structure [40].

ResNets, introduced by Microsoft in 2015, utilise residual connections between convolutional layers. Adding residual connections improves model accuracy [57]. As discussed in [73], residual connections, shown in Fig. 2.7, help ameliorate the vanishing gradient problem which causes accuracy degradation as more layers are added to a network. ResNets can utilise GAP, as shown in Fig. 2.7. GAP allows better generalisation by downsampling the

feature extractor's output. This also minimises dependency on the fully connected layer that is prone to over-fitting [74].

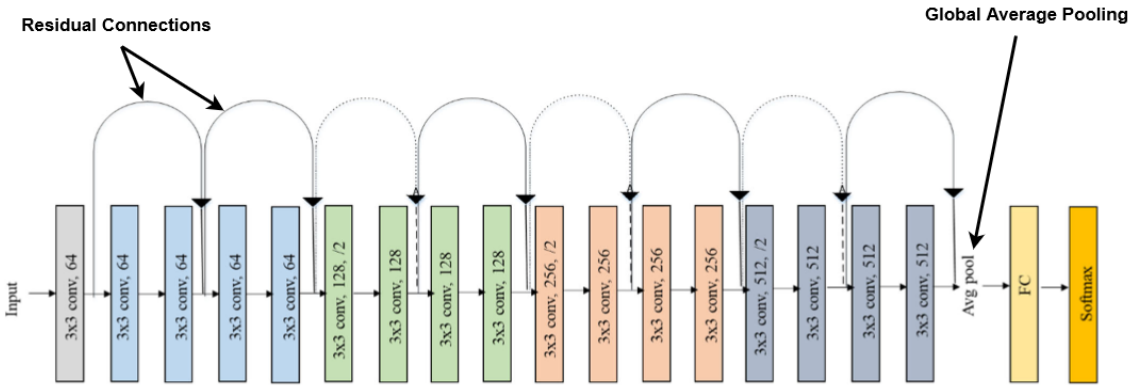

Figure 2.7: Resnet18 architecture with residual connections and GAP after the final convolutional layer [75].

Despite achieving state-of-the-art results in image recognition competitions such as ImageNet [76], ResNets application to human activity and target recognition has been limited. Only [39] applied ResNet18 in a transfer learning fashion to classify human activity and other targets. ResNet18 was trained on datasets such as the ImageNet, which have 1 000 classes. This justifies ResNet18's 11 million trainable parameters as it has to learn more features to distinguish classes. However, human activity and target classification comprise less than 15 classes, as shown in Table I. By applying ResNet18 to such problems, there is an increased risk of over-parameterisation, which leads to over-fitting [77]. Moreover, an over-parameterised network increases the computational cost of processing data, which negatively affects the inference speed.

The test accuracies (Table I) obtained from the experiments of classifying human activities and targets using CNNs cannot be directly compared because different datasets were used. However, they provide insight into the achievements made in human activities and target classification using CNNs. Accuracies range from 86.90% to 98.34% using vanilla, CAEs, CNN-LSTM and ResNet CNN models.

2.5 Improving Inference Speed

Various methods have been proposed to speed-up inference, for example, Pruning and Quantisation. Pruning [32] removes redundant weights from the network, resulting in a smaller and faster network. Quantisation [31] reduces the precision of the weights and activations in a network to achieve fast computing. There are two common types of Quantisation: PTQ and QAT. PTQ involves determining quantisation parameters after model training. In QAT, quantisation parameters are determined during model training. QAT

Model Type	Literature	Classes in Data	Test Accuracy (%)
Vanilla CNN	[41]	Walking human absent, one-person walking, two people walking, three people walking, four people walking and five people walking	86.90
Vanilla CNN	[55]	Boxing, crawling, creeping, jumping, running, standing, and walking	98.34
Vanilla CNN	[21]	Human, dog, horse, car	97.60
Vanilla CNN	[21]	Human running, walking, walking while holding a stick, crawling, boxing while moving forward, boxing while standing in place, and sitting still	90.90
CAE	[40]	Walking, jogging, limping, walking with cane, walking with walker, walking with crutches, crawling, creeping, wheelchair, falling, sitting. and falling from chair	94.20
CNN-LSTM	[72]	Running, walking, walking while holding a stick, crawling, boxing while moving forward, boxing while standing in place, and sitting still	98.28
ResNet	[39]	Running, walking, boxing, crawling, jumping, and standing casually	97.92

Table I: Results from literature.

is only recommended when PTQ has significantly degraded classification performance [38] because it is more complex to implement. Therefore, this study investigated the effect of PTQ on the proposed model. The following subsection focuses on the theory behind quantisation and the results achieved by other researchers in applying PTQ to CNNs.

2.5.1 Quantisation Theory

Network quantisation reduces the precision of weights and activations in neural networks by lowering the number of bits to represent these quantities [31]. Models are trained in high precisions such as 32-bit floating-point (FP32) to take advantage of the wide dynamic range these precisions afford. However, inference in 32-bit precision is computationally expensive, making deployment onto edge devices with strict power and compute requirements challenging [31]. Hence, a network's precision is reduced to lower precisions such as 16-bit floating-point (FP16) or 8-bit integer (INT8) for efficient use of computational resources and consequent increase in the speed of inference.

Reduction in precision speeds up inference because the available processors provide higher throughput math pipelines for the low-bit formats, speeding up math-intensive operations, such as convolutions and matrix multiplications [78]. Examples of processors include central processing units (CPUs) and graphics processing units (GPUs). However, the increase in throughput may come at the cost of the model's accuracy because of the loss of information that comes with quantisation.

PTQ is applied to a pre-trained network. The following equation is used to map model parameter values to a lower bit representation during quantisation:

$$x_q = \text{clip}\left(\left\lfloor \frac{x_f}{\Delta} \right\rceil\right) \tag{7}$$

where x_f is a floating-point value, Δ is the step size, $\lfloor \cdot \rceil$ is a function that applies a rounding-policy to round rational numbers to representable values in each precision, for example, rounding to integers in INT8 quantisation, *clip* is a function that clips outliers that fall outside of the dynamic range of a given precision and x_q is the quantised value.

The following equation can find the step size (Δ):

$$\Delta = \frac{q_{\text{range}}}{N} \tag{8}$$

where q_{range} is the size of the quantisation range and is determined from the distribution of values to be mapped to lower precision. N is the number of representable values in each precision. For example, for INT8 precision N is 256. Values outside of the quantisation range are clipped to the thresholds.

Quantization range setting [38] refers to the method of determining the clipping thresholds. This is a crucial step in determining the step size used for quantisation. Any parameter values outside the quantisation range are clipped to the threshold, thereby incurring a clipping error. To reduce the clipping error, the quantization range can be increased. However, increasing the quantisation range leads to increased rounding error. These two errors should be minimised as they lead to a loss of information carried by parameter values during quantisation, which may degrade model accuracy. To determine the quantisation range, various methods can be used, which include:

- Max: Uses the maximum absolute value of floating-point values observed [79].
- Cross-Entropy: Uses Kullback–Leibler (KL) divergence to minimize information loss between the original floating-point values and values that could be represented by the quantised format [80].
- Percentile: Sets the range to a percentile of the distribution of absolute values observed on different inputs [81]. For example, 99% calibration would clip 1% of the largest magnitude values.

The Max method leads to no clipping error. However, this approach is sensitive to outliers as strong outliers will cause excessive rounding errors. Cross-Entropy and Percentile methods clip outlier values to increase the resolution of inlier values [81] reducing the rounding error. The two methods aim to minimise both the clipping and rounding error and hence the loss of information in the model parameters.

Finding the optimal range for the weights of a model is easier because the distribution of weight values is constant. However, determining the optimal range for activation values is more challenging since activation values vary depending on the input data. Thus, setting the range pre-maturely may result in excessive clipping and rounding errors. Hence, a calibration dataset is fed to the network to find the optimal range for activation values. The distribution of the activation values is recorded as the network processes the data. The calibration dataset should represent the input data so that the recorded activation values resemble values that will be observed when new data is introduced to the model. The recorded distribution is then used to set the range.

2.5.2 The Effect of Quantisation on Inference Speed

Quantisation has different effects on a model inference speed depending on the network architecture, dataset and hardware used for inference. Paupamah, James and Klein [82] investigated the effect of INT8 quantisation on MobileNet, ShuffleNet and AlexNet using a combination of Nvidia GTX 1060 Ti and 1080 Ti GPUs. It was found that MobileNet achieved a 7.3× and 5.7× speed-up on the CIFAR10 [83] and FashionMNIST [84] datasets,

respectively. However, ShuffleNet's inference speed on the CIFAR10 dataset was halved due to quantisation. This was attributed to the complex architecture and small size of ShuffleNet.

Krishnamoorthi [31] investigated how INT8 quantisation affected the inference speed of MobileNet, NasNet, Inception and ResNet models. It was found that INT8 inference could provide 2×-3× speed-up on a CPU and close to 10× speed-up on specialized processors optimised for low precision wide vector arithmetic, such as the Qualcomm Digital Signal Processor (DSP) with Hexagon Vector Extensions (HVX).

2.5.3 The Effect of Quantisation on Accuracy

In [82], INT8 quantisation on MobileNet resulted in less than 1% loss in accuracy on both datasets. However, ShuffleNet's accuracy drop was more significant (12%) on the CIFAR10 dataset. The significant decline was attributed to the network's small size. Krishnamoorthi [31] also found that networks with fewer parameters suffered a more substantial loss in accuracy after quantising MobileNet, NasNet, Inception and ResNet models.

2.6 Suggested Approach

Although ResNets have performed better than vanilla CNNs [57], only one paper [39] has investigated their application in human activity and target classification. This study adopted a ResNet model that improved the performance of the model proposed in [34]. Parker [34] proposed a five-layer, vanilla CNN model with two fully connected layers which was used as a baseline. The model is discussed in more detail in Section 5.1.1.

The literature does not discuss how CNNs can be integrated into a radar system to support real-time classification. For real-time support, the inference speed of the model must be sufficiently high such that the radar system is not overwhelmed by incoming data. Therefore, this study also focused on improving inference speed using PTQ to support real-time applications without significantly degrading the model's accuracy. In addition, experiments were carried out to find the maximum range that the proposed CNN could support in real-time.

3 Methodology

This chapter provides an overview of the methods applied in collecting and pre-processing radar data for CNN model training, validation and testing. This is followed by a summary of the implementation of CNNs in classifying spectrograms. Afterwards, the implementation of PTQ in investigating the effect of quantisation on model accuracy and inference speed is discussed. Finally, the software and hardware used for data processing and experiments are presented.

3.1 Data Collection and Labelling

The CSIR was responsible for data collection and labelling. As discussed in Section 2.1, a C-band pulse-Doppler radar was used for data collection. The radar operated at a centre frequency of 5.45 GHz, pulse repetition frequency (PRF) of 10 kHz and pulse bandwidth of 25 MHz.

In collecting data, reflected signals received at the radar's antenna were mixed down to an intermediate frequency, digitised using an A/D converter and then mixed down to complex baseband samples. As illustrated in Fig. 3.1, the baseband samples were initially pulse-compressed to improve the SNR of the received signal. Beamforming was then applied to attenuate signals that did not come from the direction in which there was a target of interest. The beamformed signals were then decimated to an effective PRF of 714 Hz. Decimation was performed because the Doppler bandwidth provided by 10 kHz was much wider than needed. The decimation allowed for a much lower data rate while maintaining the region of the Doppler bandwidth that contained relevant micro-Doppler information for classification. Therefore, using Equation 2, the range resolution of the radar was found to be 5.994 m.

Data samples were collected from the range bins in which there was a target of interest according to the time the target was present. The samples were then labelled according to the target that was present in the range bin.

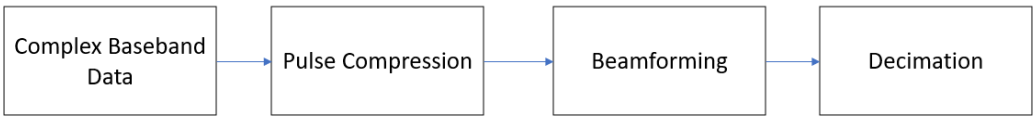

Figure 3.1: Pre-processing steps on complex baseband data.

3.2 Additional Data Pre-processing

This subsection provides an overview of the pre-processing of labelled samples to produce spectrograms. More details on this topic are presented in Section 4.2.

The initial work in this study focused on pre-processing the data that the CSIR prepared. The pre-processing stage involved filtering out the clutter [85] in the complex data, followed by computing the STFT to generate spectrograms. Clutter comes from stationary (e.g., the ground and buildings) and slow-moving objects (e.g., wind affected grass and trees). Filtering was done to remove the clutter since it does not contain information relevant to distinguishing targets or activities of interest, such as moving vehicles or running humans.

Training, validation and testing data was extracted from the generated spectrograms in a sliding window fashion as described in Section 2.3.

3.3 Model Implementation

In implementing CNNs, the model proposed by [34] was used as the baseline to identify parameters that would improve this study's proposed model's performance. The proposed model applied concepts covered in literature such as residual connections and GAP. Full details regarding the design decisions and the experiment results are presented in Section 5.

3.4 Inference

FP32, FP16 and INT8 precisions were considered in investigating the effect of quantisation on the proposed model's accuracy and inference speed. Additional experiments were carried out to measure the inference speed of the whole data processing pipeline, from computing the STFT to classifying the obtained spectrograms. This was done to determine the maximum range that the inference pipeline could support in real-time. A detailed discussion of the experimental setup and results are presented in Chapter 6.

3.5 Software and Hardware

Machine learning model development was carried out in Python 3.6 using the PyTorch library. Python was selected because it has established support for machine learning libraries such as TensorFlow and PyTorch. PyTorch was selected for model building because it has a Pythonic programming style that makes debugging easy, and it also supports hardware accelerators such as GPUs [86].

Data pre-processing was carried out in MATLAB R2020b. However, the full inference pipeline was implemented in Python, from computing the STFT to model inference.

All development, testing and analysis was carried out on an ASUS TUF Gaming laptop with the following specifications:

- Operating System: Ubuntu 18.04.6.
- Processor: Intel Core i5-9300H (2.40 GHz).

- Graphics Processor: Nvidia GeForce GTX 1650.
- Memory: 8GB DDR4-2666 SO-DIMM.

Nvidia TensorRT [80] was used for model optimisation and quantisation. In TensorRT, model optimisation was achieved through layer and tensor fusion, kernel auto-tuning and parallel stream execution [80]. In addition to model optimisation, TensorRT provided libraries that supported FP16 and INT8 quantisation.

4 Data Collection and Pre-processing

Data collection and pre-processing affect the quality of input data to machine learning models and hence their performance. This chapter provides more detail on radar data collection and pre-processing.

4.1 Data Collection

The C-band pulse-Doppler radar used to record different activities and targets, was mounted on top of a building at the Pretoria CSIR campus, as shown in Fig. 4.1. The radar's line of sight was in the direction shown in Fig. 4.2, facing trees, roads, and buildings. All measurements of the six classes were taken at different times between 09h00 and 18h00 on the 8th of September, 2021.

Figure 4.1: Radar on top of the roof of building 44 at CSIR's main campus in Pretoria [34].

Note that the measured data was pulse compressed, beamformed and decimated as discussed in Section 3.1. Afterwards, data samples belonging to range bins in which there was a target of interest were collected. The size of the samples depended on the time in which the target was present in the range bin. The samples were labelled according to the target present.

4.1.1 One Human Walking

A person was recorded walking on the field shown in Fig. 4.3, approximately 175 meters away from the radar. The person walked in a straight line, in different directions, as illustrated

Figure 4.2: The line of sight of the radar showing the field in which a human (marked by a purple dot) was recorded walking [34].

in Fig. 4.3, at a speed of approximately 1.5 meters per second. Additional recordings were taken later in the afternoon with a person walking in a road approximately 1 200 metres away from the radar, seven times the initial distance. The person walked up and down the road marked by a purple oval shape in Fig. 4.4.

4.1.2 Two Humans Walking

Two humans were recorded walking in the same field used to record one human walking, (Fig. 4.3). A distance of two to three metres between the two people was maintained throughout the measurements. This was done to ensure that the distance between the micro-Doppler returns of the two people was always less than the range resolution of the radar (5.994 meters). In this way, the micro-Doppler returns would be captured within a single range bin such that they would appear in the same spectrogram after computing the STFT.

4.1.3 One Human Running

A person was recorded running in the same field and road used for recording the walking activities, shown in (Fig. 4.4). The directions of motion were like those in the walking activities.

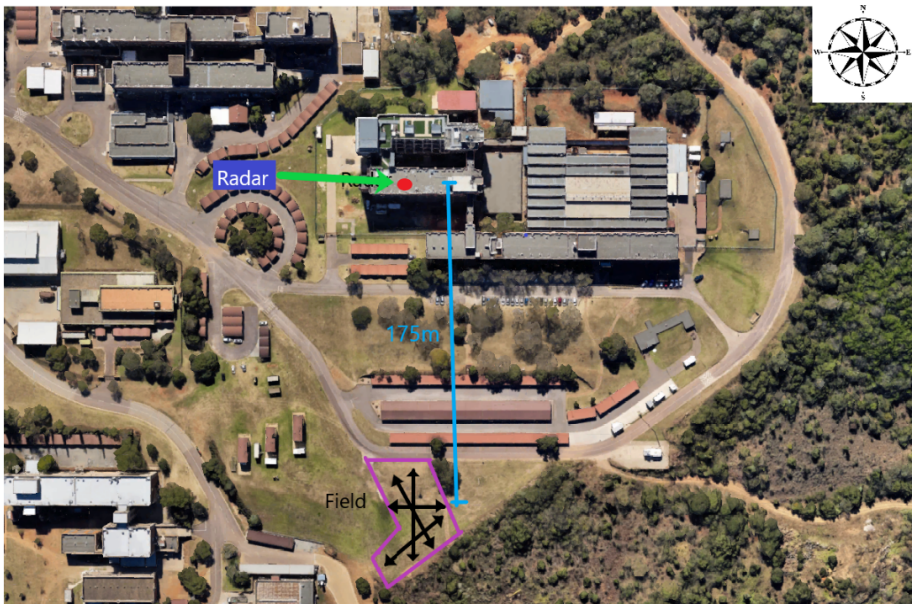

Figure 4.3: Field (marked in purple) from which human and sphere swing activities were measured [34].

4.1.4 Vehicles

Moving vehicles were recorded on multiple roads, between 600 meters and 1 400 meters away from the radar, travelling in opposite directions at locations marked in Fig. 4.5.

4.1.5 Clutter or Noise

The clutter or noise class consisted of all data collected in which there was no target of interest. This class will be referred to as the clutter class.

4.1.6 Sphere Swing

A person swinging a sphere that was attached to a rope was recorded in the middle of the field in Fig. 4.3. This class was added to increase the diversity of the dataset. The sphere was swung at different aspect and tilt angles and at varying velocities, in a circular motion, figure-of-8 patterns and clockwise and anti-clockwise directions.

4.2 Data Pre-processing

The pre-processing stage involved computing the STFT from collected and labelled baseband samples to generate spectrograms. This was followed by the process of extracting segments of the spectrograms to be used as input data for the CNN model training, validation and testing.

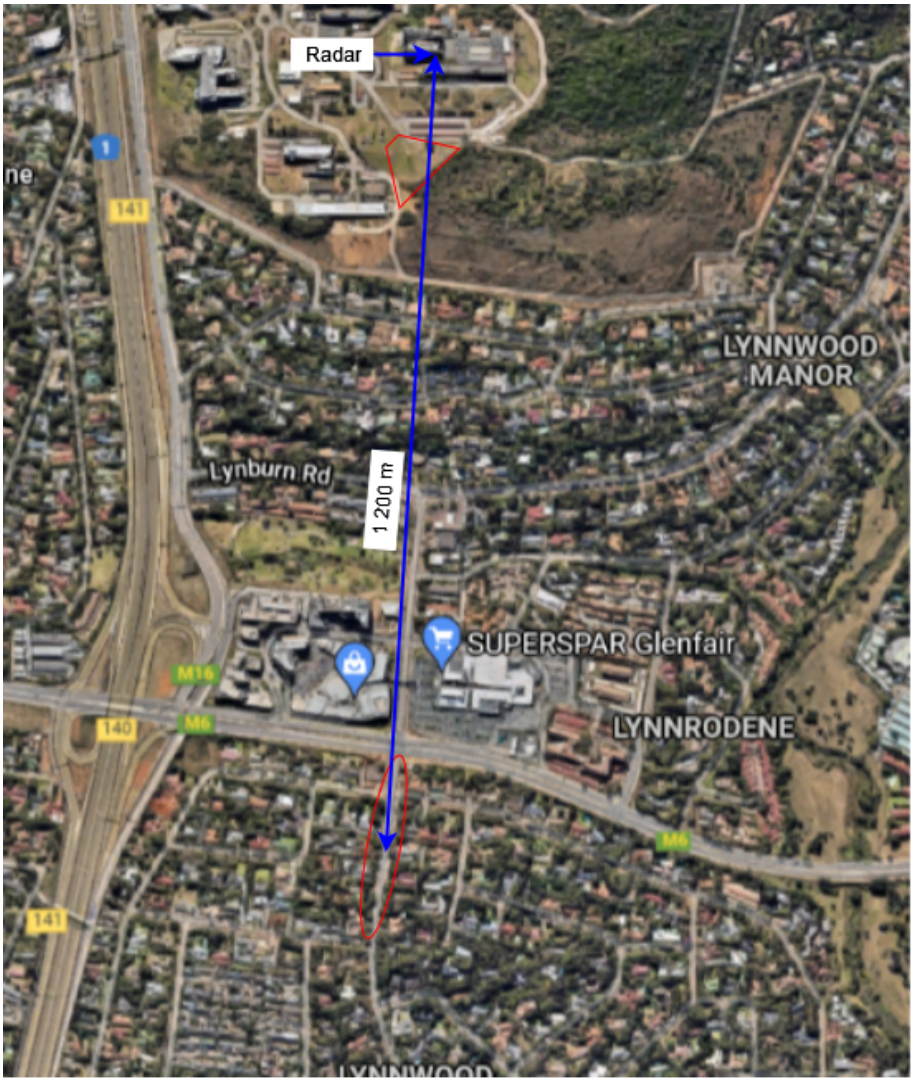

Figure 4.4: Field and road (marked in red) from which human activity measurements were taken.

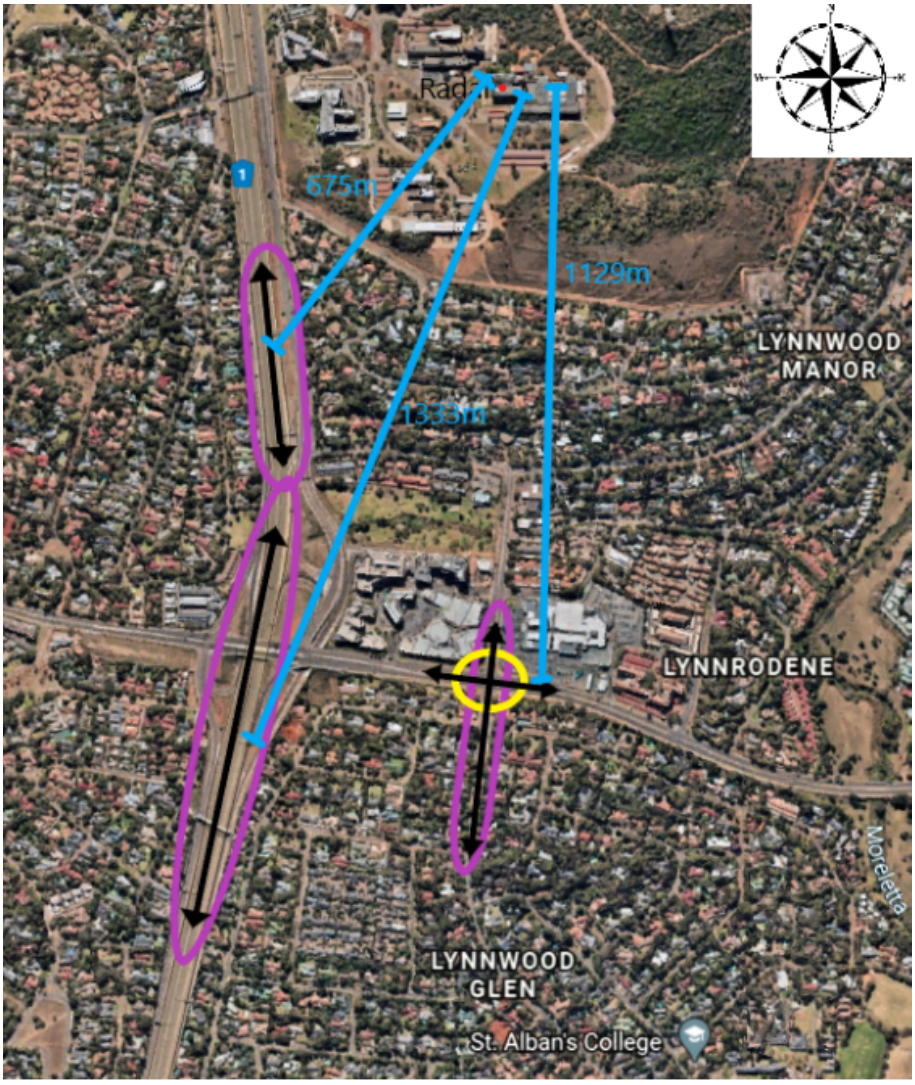

Figure 4.5: Roads (marked in purple) from which moving vehicles were measured [34].

4.2.1 Short-Time Fourier Transform

The baseband samples were contaminated with clutter due to returns from stationary and slow-moving objects. A sixth-order notch filter removed the clutter before computing the STFT. The notch filter had a 3 dB bandwidth of 20 Hz, centre frequency of 0 Hz and maximum pass-band ripple of 0.2 to suppress the clutter. A 3dB bandwidth of 20 Hz was used because the clutter was observed to fall under a radial velocity of 0.5 m/s (20 Hz Doppler frequency) in spectrograms that were generated from clutter class data. The filter's magnitude response in Fig. 4.6 shows that frequencies at 20 Hz were attenuated by 2.97 dB, while frequencies at 15 Hz were attenuated by 27 dB.

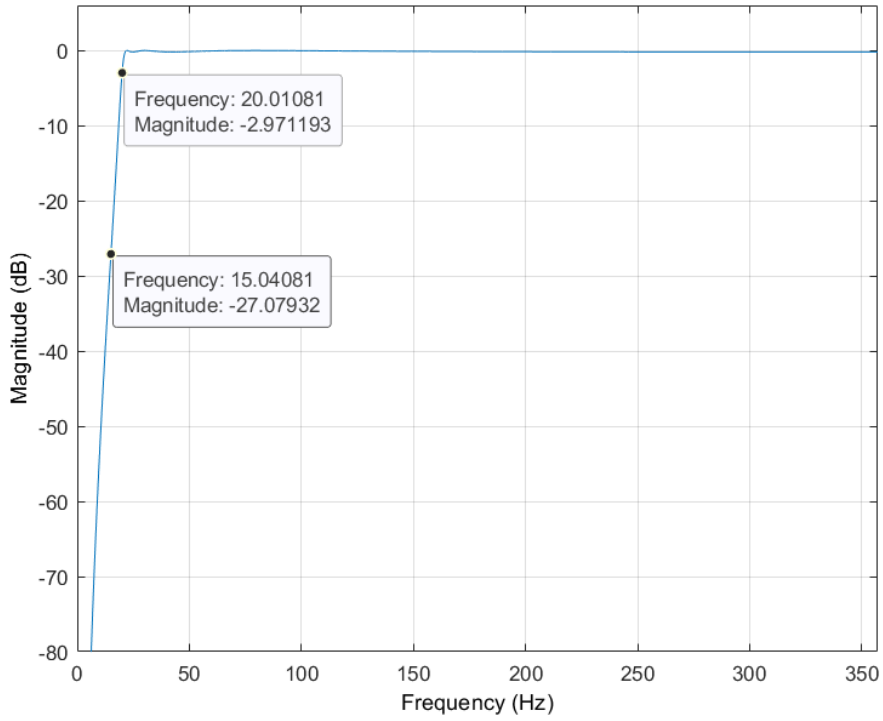

Figure 4.6: Magnitude response of the notch filter used to remove clutter.

After filtering, the STFT was applied to obtain a time-frequency domain representation of the signals in the form of spectrograms. In computing the STFT, a Hamming window with an overlap of 50% was applied. The Hamming window was chosen because it provides a balance between side-lobe levels and an increase in main-lobe width. An overlap of 50% was chosen since it allowed all data samples to be weighted equally, with minimal computational effort [52]. Both of these reasons were discussed in Section 2.2.2.

To find the optimal CP length, the concept of peak SNR (PSNR) was applied. Note that the SNR in the time-frequency domain varied with time due to the targets' movements and

their changing distance from the radar. Consequently, the PSNR was used as a measure of the quality of information conveyed by a spectrogram. The PSNR was defined as the ratio of peak power of the measured signal to the power of the noise.

The peak signal power was estimated using the peak value in the time-frequency domain, after clutter removal. The power of the noise was estimated using filtered complex data samples from the clutter class (Section 4.1.5). It was observed that the complex noise samples followed a Gaussian distribution with zero mean, as shown in Fig 4.7. Hence, the noise power was estimated using the variance of the samples [87].

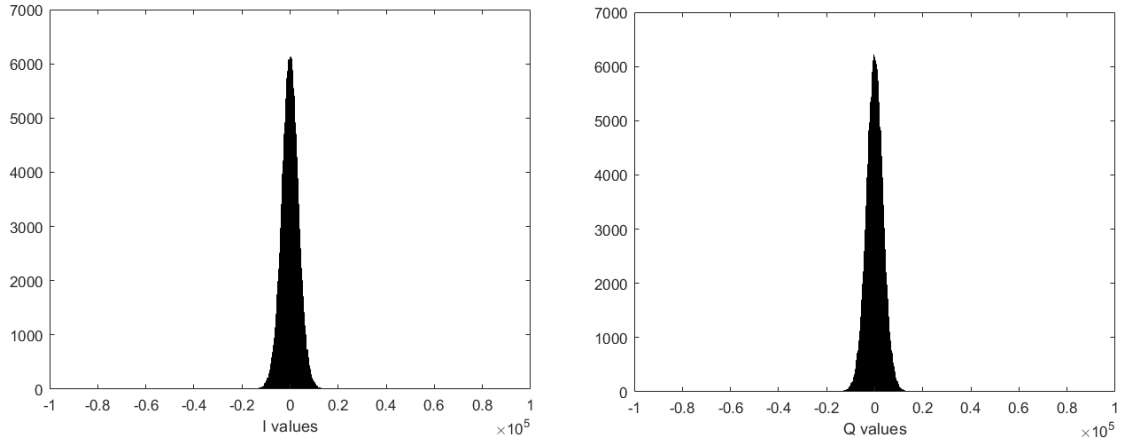

Figure 4.7: Distributions of complex noise samples taken from filtered clutter class data.

Five data samples were randomly selected in each class to investigate the relationship between PSNR and CP length. The data samples were filtered to remove clutter, and the STFT was applied to convert the data into the time-frequency domain. The PSNR was then calculated from the time-frequency domain data and the average PSNRs of each class were determined. The CP length was varied between 32, 64, 128 and 256 (0.045, 0.09, 0.18 and 0.36 seconds in time) in computing the STFT.

The results in Fig. 4.8 show that the average PSNRs of human activities were higher than those of moving vehicles. One plausible reason for this is the target's distance away from the radar. Most human activities were measured in a field closer to the radar (175 meters). Moving vehicles were measured at larger distances (600 meters - 1400 meters). From equation 1, the power of the received signal is more sensitive to a target's range than the target's radar cross-section (RCS). The difference in range was potentially more significant than the difference in RCS between the two targets. As a result, lower power was received for vehicles, which resulted in lower SNR.

Human activities also had a higher PSNR than a swinging sphere, although the activities were recorded at similar distances from the radar. One plausible reason for this was that the

power received from the swinging sphere was lower (compared to human activities) due to the sphere's smaller cross-sectional area. The smaller cross-sectional area reduced the RCS, which is directly proportional to the received power. Thus, a lower power resulted in a lower SNR compared to human activities.

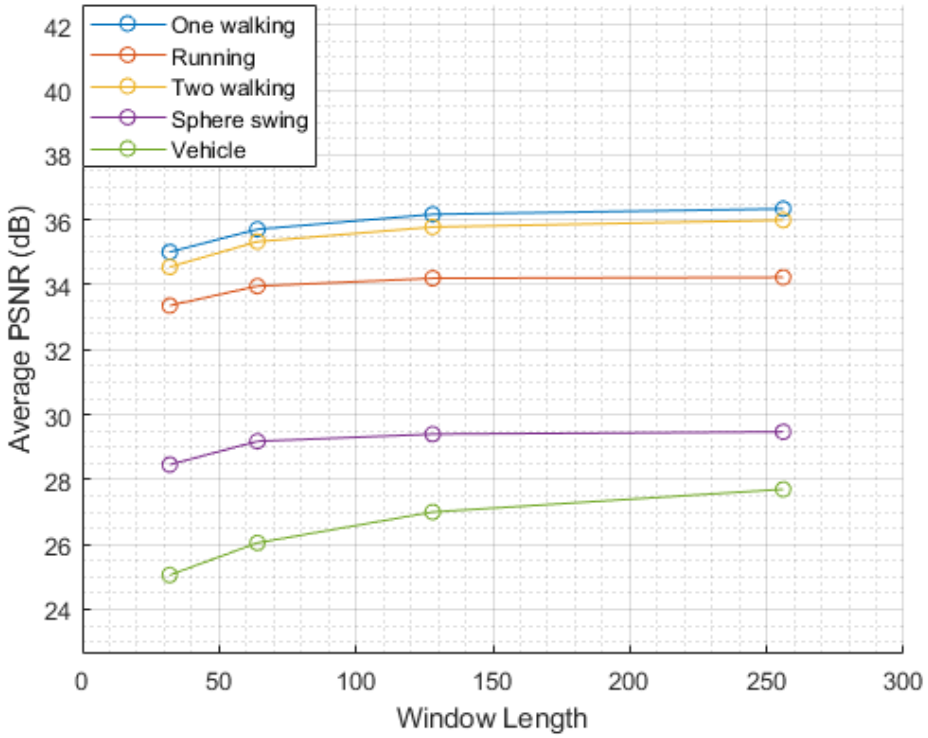

Figure 4.8: Curves of average PSNR of each class for various window lengths.

Fig. 4.8 also shows that the average PSNR of all classes plateaued after a window length of 128. As a result, doubling the window length beyond 128 would not significantly improve the average PSNR but would create spectrograms with a higher computational cost for processing in CNNs. Therefore, a window length of 128 was chosen to generate spectrograms using the STFT.

The following STFT parameter values were chosen:

- Window type - Hamming.
- Window length - 128.
- Window overlap length - 64 (50%).

4.2.2 Spectrogram Segmentation

The data that was used as input to the CNN models were extracted from spectrograms in a sliding window fashion following the practice in [55]. Given that there was little information on how researchers selected the time duration of examples, a spectrogram duration of 4 seconds was initially chosen as done in [14]. A duration of 4 seconds was sufficient to capture the full gait of the slowest class (walking human), which is approximately 1 second for men [88]. However, after model building, additional research was carried out to determine the relationship between spectrogram duration and model accuracy in Appendix A.

The sliding windows had an overlap of 25%, which is equivalent to 1 second for a duration of 4 seconds. An overlap of 25% was chosen to ensure that there was a significant variance between spectrograms in consecutive windows. Note that this was an overlap used for extracting input data for CNNs while the previously mentioned overlap of 50% was used to compute the STFT.

Fig. 4.9 shows how four spectrograms of 4 seconds duration were extracted from a spectrogram of one person walking away from the radar. There is a constant overlap of 1 second between consecutive windows. The number of rows in the spectrograms was 128, while the number of columns was 45. Therefore, the CNN models were fed input data of size 128×45.

Fig. 4.10 shows extracted spectrograms of the six classes in the data. The vehicle spectrogram shows the radial velocity of the vehicle decreasing over time because of slowing down. The human activity spectrograms show periodic patterns resulting from the motion of the torso, hands, and legs. The sphere swing example shows the radial velocity of the sphere changing rapidly as the sphere follows a circular path spatially. There were no significant micro-Doppler returns in the clutter spectrogram due to the absence of a moving target.

Fig. 4.10 also shows that the spectrograms of different targets each have a visually unique signature. Thus, they contain a rich source of information for target classification using DL techniques, especially CNNs that have excelled in image recognition tasks.

4.3 Splitting the data

A total of 17 939 spectrogram segments were extracted from the generated spectrograms. Fig. 4.11 shows the distribution of the extracted spectrograms across various classes. The vehicle class had the most spectrograms (4 443) while the sphere swing class was the least represented, with only 2 049 examples.

The distribution of spectrograms across the various classes was uneven. Therefore, the concept of class imbalance was explored. Class imbalance is present when there are significantly fewer training examples in one or more classes compared to other classes [89]. The imbalance is such that a model can achieve high accuracy by simply biasing itself to the majority class. Therefore, choosing accuracy as the performing criterion in class imbalance problems may

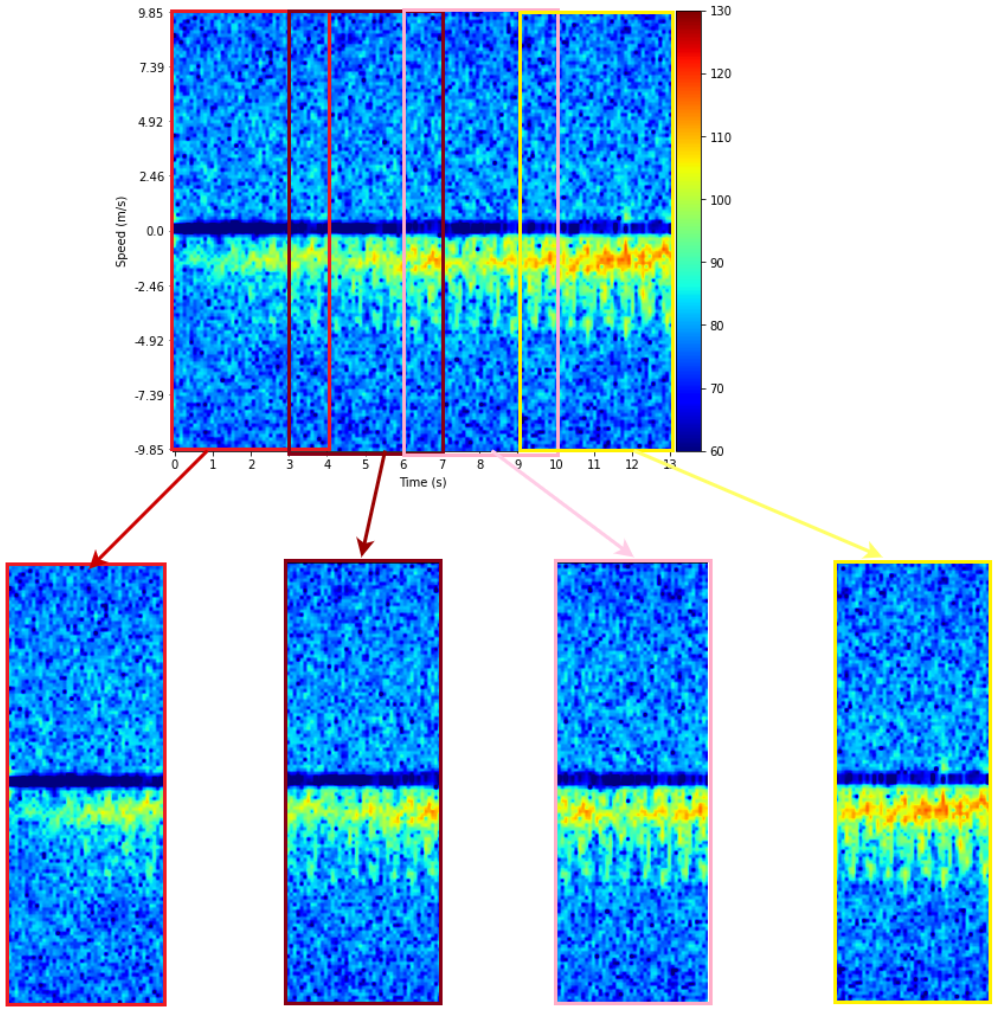

Figure 4.9: Four spectrograms extracted from a spectrogram of one person walking away from the radar.

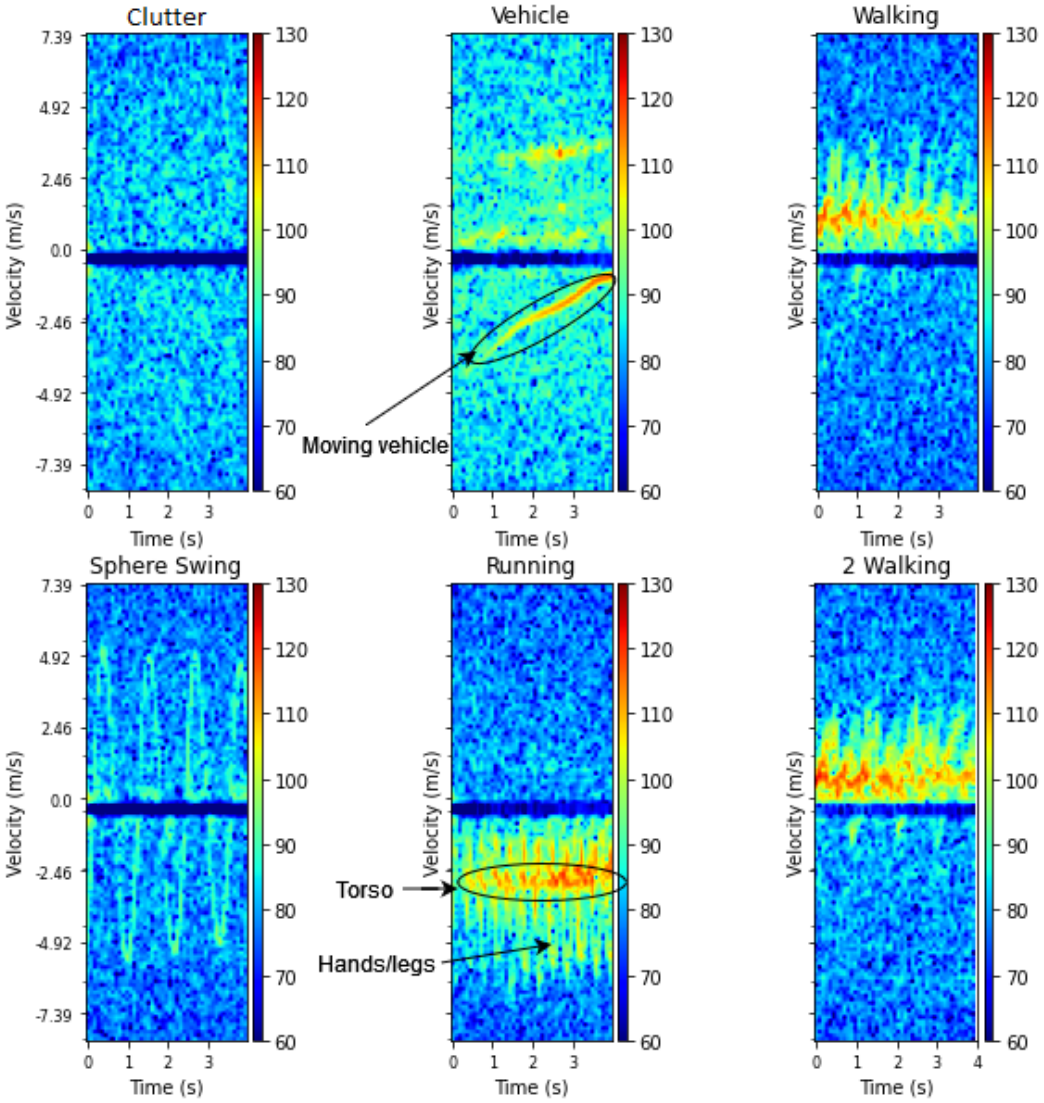

Figure 4.10: Spectrogram segments of six classes extracted from spectrograms.

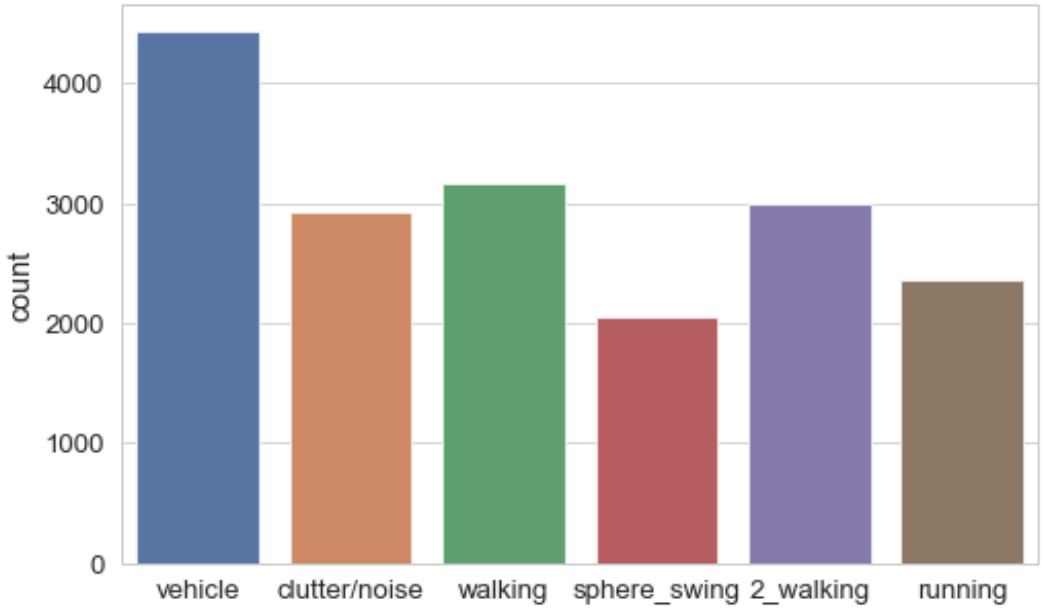

Figure 4.11: Distribution of extracted spectrograms across classes.

give inaccurate and misleading information about a classifier performance [90–93]. Although the classes in the data in this study were not evenly distributed, it was not considered as a severely imbalanced dataset for the following reasons:

- The majority class did not vastly outnumber the other classes, e.g., by 10 times or 100 times [93, 94].
- The vehicles class made up 24 % of the data, and if the classification is biased towards this class, the model's performance would be poor as it would get only 24 % (4 443/17 939) of the predictions correct.

Given that severe class imbalance did not apply to this data, accuracy was used as a performance metric for the model and confusion matrices were used to evaluate the model's performance in each class. The accuracy was calculated using the following formula:

$$Accuracy(\%) = \frac{T}{T+F} \cdot 100\% \tag{9}$$

where T is the number of correctly classified spectrograms, and F is the number of incorrectly classified spectrograms.

The spectrograms were split 70%/15%/15% into the train, validation and test datasets on a per-class basis using their recording time. As shown in Fig 4.12, spectrogram segments from earlier recordings were assigned to the training set, and spectrogram segments from the latest recordings were assigned to the test set.

Spectrogram segments were divided on the time of recording because some spectrograms recorded the same activity from the perspective of different range bins. These spectrograms were highly correlated. Therefore, dividing the data on the time of recording instead of randomly, ensured that correlated spectrograms used for training did not leak into the validation and testing datasets. Furthermore, splitting data based on recording time allowed a better assessment of how the model would perform in the future, having been trained on data captured earlier [95].

Among the 3 169 spectrograms of one human walking, 238 spectrograms belonged to measurements taken from the road, 1 200 metres away. The rest of the spectrograms belonged to measurements taken from the field 175 metres away. Similarly, amongst the 2 354 spectrograms of one human running, 73 spectrograms belonged to measurements taken from the road while the rest were taken from the field. The 238 and 73 spectrograms were divided 70%/15%/15% between the training, validation and testing datasets on a per-class basis. This ensured that the models were also trained on data from further distances before validation and testing.

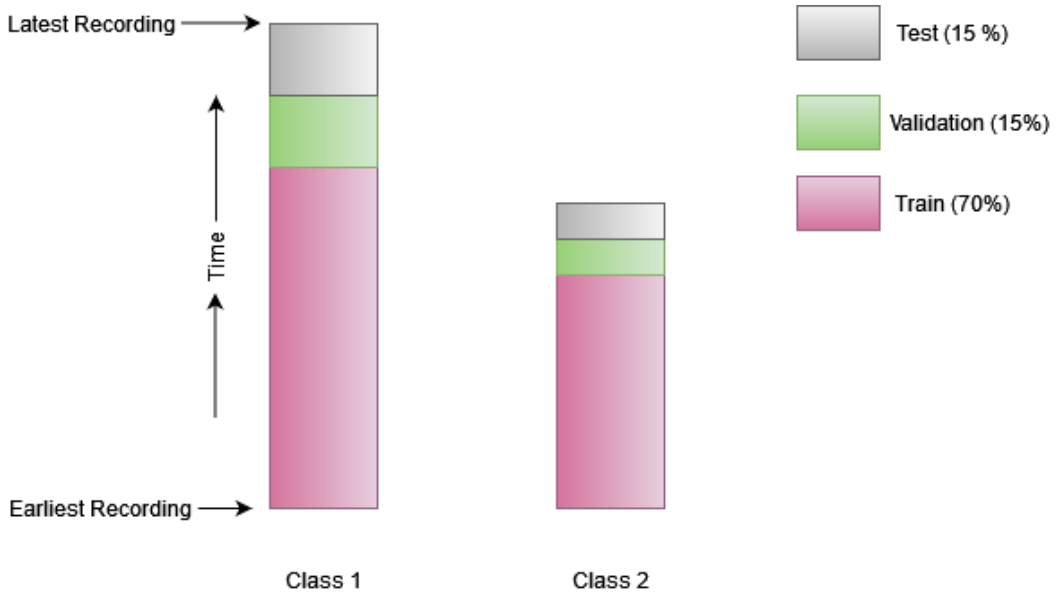

Figure 4.12: Splitting data from different classes into training, validation and test datasets.

The following chapter focuses on the implementation, training and testing of CNN models using the data generated from the pre-processing steps.

5 Model Implementation, Training and Results

This chapter gives an overview of the experimental process and the results that were achieved in the model building stage. The first objective of this study was to improve the accuracy of the model proposed by [34]. Therefore, four CNN models were considered. The first model was the proposed model from [34], which provided a performance baseline. The second model was a three-layer vanilla CNN, with BN [96]. The third model was a three-layer vanilla CNN, with BN and GAP. The final model was a CNN that adopted BN, GAP, and residual connections in its architecture following the practice in [57].

5.1 Model Implementation and Training

5.1.1 Baseline Convolutional Neural Network (Model 1)

Model 1, shown in Fig. 5.1, was used as the baseline model for performance comparison. Model 1 had five convolutional and two fully connected layers resulting in 442 886 trainable parameters. The first layer had 128 units with 5×5 filters, followed by two layers of 128 units with 3×3 filters and then two layers of 64 units with 3×3 filters. BN and dropout [97] were not used in this model because [34] observed that the two techniques did not improve the model's performance.

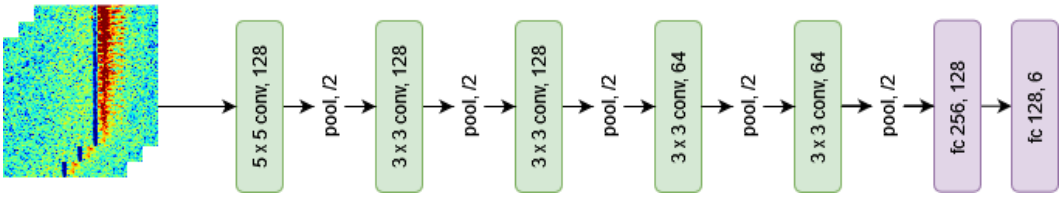

Figure 5.1: Architecture of the baseline model with five convolutional layers and two fully connected layers.

Parker [34] found that the optimal training parameters for model 1 were a batch size of 100 and a learning rate of 0.0005 using Adam as the optimiser. Therefore, the same parameters were used for training Model 1 in this study.

5.1.2 Convolutional Neural Network without Global Average Pooling (Model 2)

Model 2, shown in Fig. 5.2, was inspired by VGG16 CNN created by the members of the Visual Geometry Group at the University of Oxford [98]. VGG16 showed that deep networks utilising 3×3 filters performed better than shallow networks with larger filters. Model 2 had three convolutional layers of 32 units with 3×3 filters, and three fully connected layers. There were 199 814 trainable parameters in this model. Model 2 also used BN between each convolutional and activation unit layer as recommended in [96], since it standardises the data passing through a model to allow faster training.

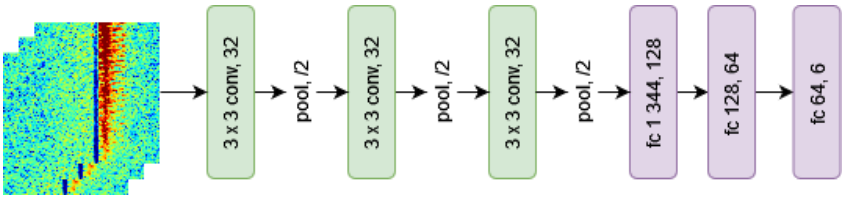

Figure 5.2: Architecture of Model 2 with three convolutional layers and three fully connected layers.

5.1.3 Convolutional Neural Network with Global Average Pooling (Model 3)

Lin et al. [74] showed that GAP acts as a regulariser to improve the classification performance of a CNN. Model 3, shown in Fig. 5.3, introduced GAP after the feature extractor. Due to GAP, Model 3 only needed one fully connected layer, which reduced the number of trainable parameters by a factor of 10, from 199 814 to 19 206. Reducing the number of parameters helps to prevent over-fitting and to achieve fast inference. Model 3 also used BN between each convolutional and activation unit.

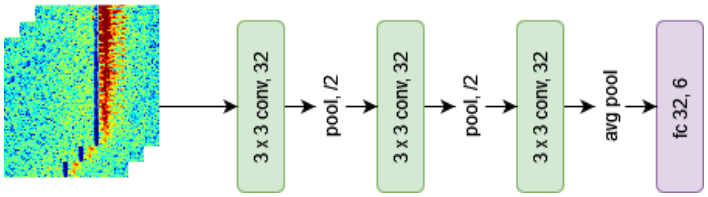

Figure 5.3: Architecture of Model 3 with three convolutional layers using GAP.

5.1.4 Convolutional Network with Residual Connections and Global Average Pooling (Model 4)

Model 4, shown in Fig. 5.4, was inspired by Microsoft's ResNet18 architecture [57], which used residual connection after every two convolutional layers with GAP after the feature extractor. He et al. [57] showed that networks utilising residual connections achieved better classification results than their vanilla counterparts. This is because residual connections allow networks to avoid the vanishing gradient problem, which degrades performance as network depth increases. The model consisted of seven convolutional layers with 32 units with 3×3 filters in each layer and a single fully connected layer, resulting in 56 454 parameters. Model 4 also used BN between each convolutional and activation unit.

5.2 Results

This subsection covers the results from training models discussed in Section 5.1. The training and validation results of the four models are presented. The results were used to determine the best model.

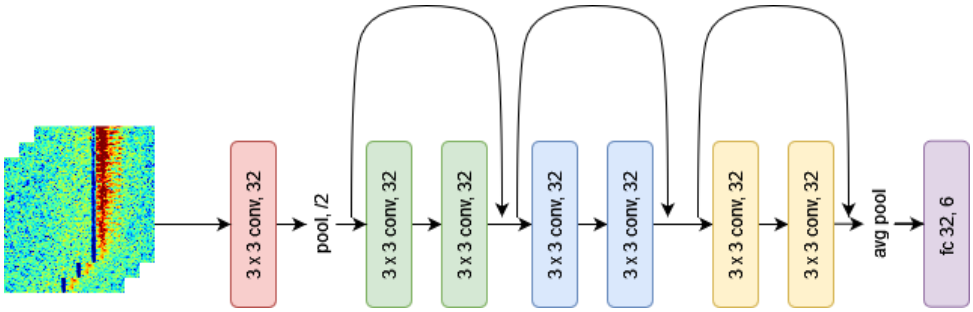

Figure 5.4: Architecture of Model 4 with three residual blocks using GAP.

ReLU was used as an activation function for reasons discussed in Section 2.4.1. Also, BN with momentum 0.1 and epsilon of 10^{-5} was applied on Models 2 to 4 to achieve fast training.

In training Models 2 to 4, SGD and Adam were considered as optimisers using different learning rates from 10^{-1} to 10^{-5} in orders of 10. Adam led to better accuracy results using a learning rate of 10^{-4} for the proposed models. Furthermore, decreasing Adam's learning rate by 50% after every ten epochs improved the accuracy. Decreasing the learning rate in this manner helped the models converge and not overshoot the minimum.

Early stopping [99] was also applied to stop the models from over-fitting after observing no improvement in validation loss for 20 epochs. This was done to prevent the models from over-fitting. Various batch sizes were also considered in training Models 2 to 4, including 16, 32, 64, 128 and 256. It was found that a batch size of 32 resulted in better accuracy. One plausible reason for this is that smaller batch sizes add noise to the training process, and this has a regularisation effect on the network [100].

5.2.1 Model 1 Results

The learning curves of Model 1 are shown in Fig. 5.5. The model reached a minimum validation loss after 27 epochs. Training stopped after 47 epochs because of early stopping. The baseline model was evaluated at the minimum validation loss, and it achieved a training accuracy of 94.16% and a validation accuracy of 82.99%. This was comparable to [34]'s validation accuracy of 83.18% using the same model. The small difference was potentially due to the randomness in DL algorithms introduced by weight initialisation and regularisation processes. This randomness makes it difficult to achieve the same results. Furthermore, [34] used different STFT parameters to generate spectrograms. Thus, the model was not trained on the same data. This was also a potential source of the difference in accuracy.

Fig. 5.6 shows the confusion matrix from evaluating Model 1 on the validation data. The diagonal elements represent the accuracy of each class. The model had difficulty distinguishing human activities (walking, two people walking and running) and achieved a minimum accuracy

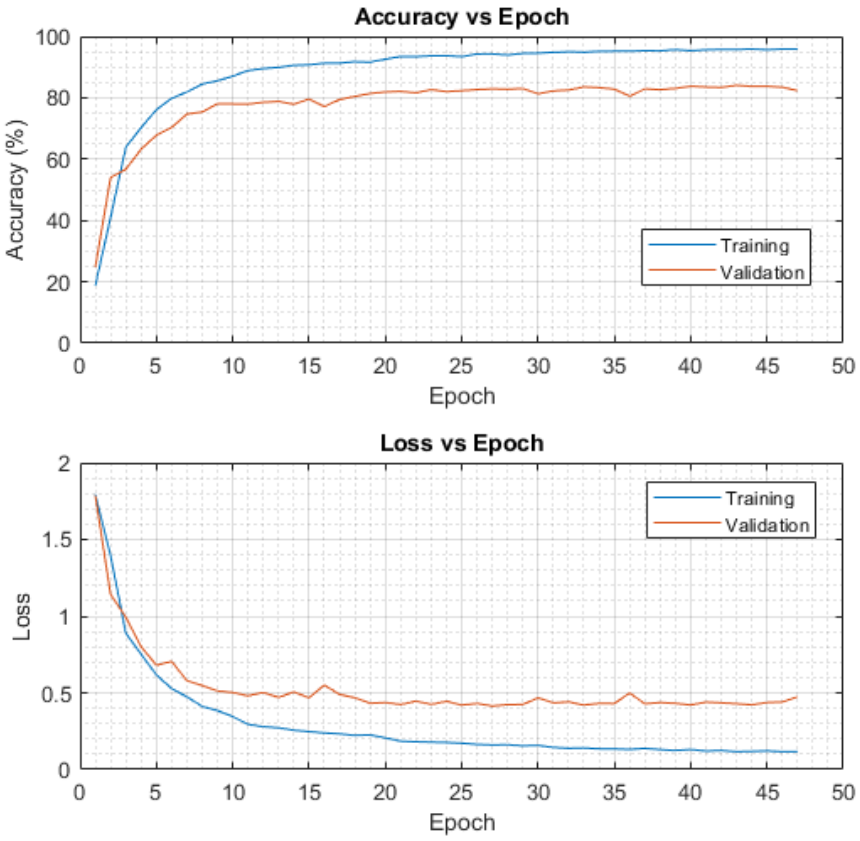

Figure 5.5: Training and validation curves of Model 1.

of 51% for two people walking. Despite achieving an accuracy above 96% in classifying clutter, sphere swing and moving vehicles, the overall accuracy was significantly limited by the model's inability to distinguish human activities correctly.

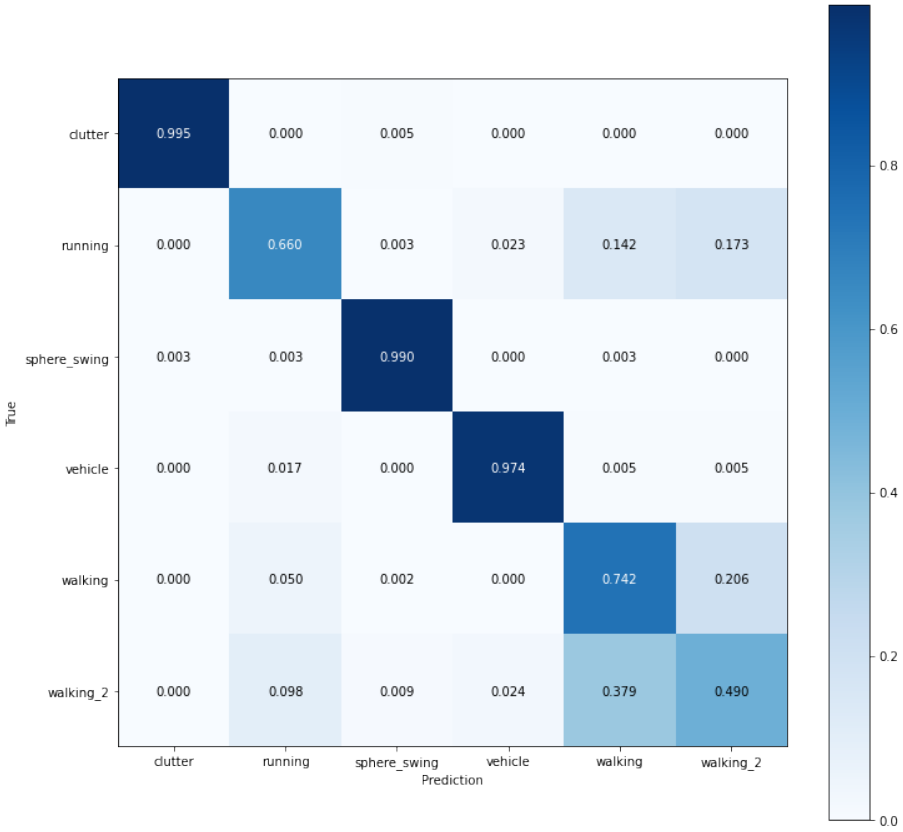

Figure 5.6: The confusion matrix of Model 1 fitted to validation data.

5.2.2 Model 2 Results

In optimising Model 2, it was found that a depth beyond three convolutional layers and applying dropout did not improve the model.

The learning curves of this model are shown in Fig. 5.7. Model 2 reached a minimum validation loss after nine epochs. Beyond nine epochs, the model started over-fitting, which can be seen by the decreasing training loss and increasing validation loss. Training stopped after 29 epochs because of early stopping. The model was evaluated at the minimum validation loss, and it reached a training accuracy of 97.02% and a validation accuracy of 83.83%.

The confusion matrix on the validation data (Fig. 5.8) shows that Model 2 was also limited in distinguishing the human activity classes. The minimum accuracy on the human activity classes was 64.6% for two humans walking. The total validation accuracy was comparable (less than 1% difference) to Model 1, but Model 2 had half the number of parameters. This

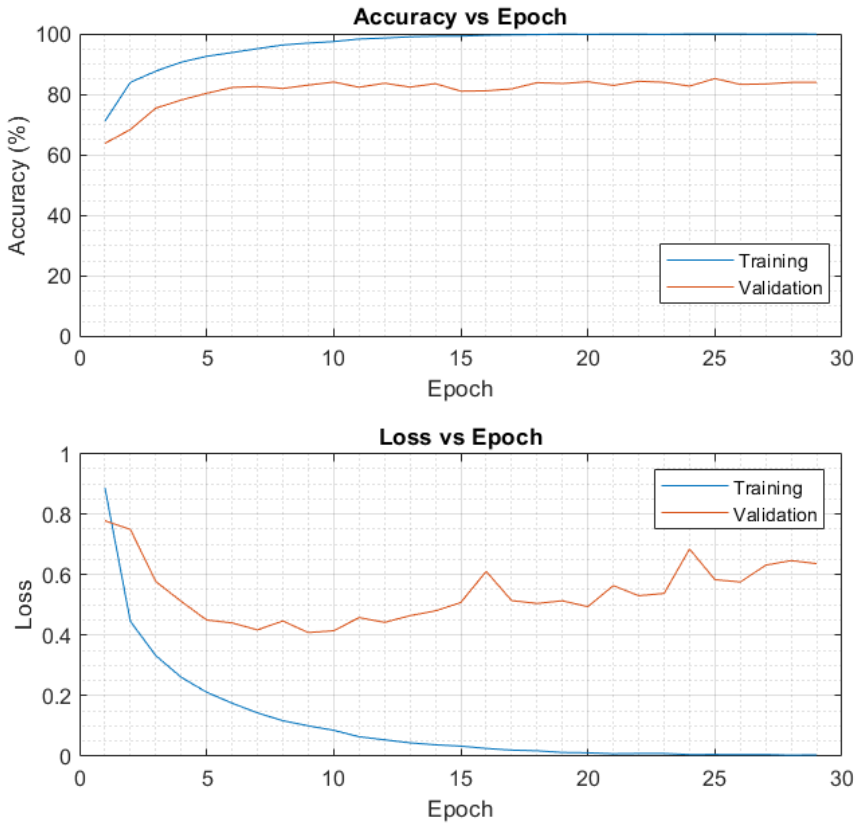

Figure 5.7: Training and validation curves of Model 2.

suggests that Model 1 had more trainable parameters than necessary.

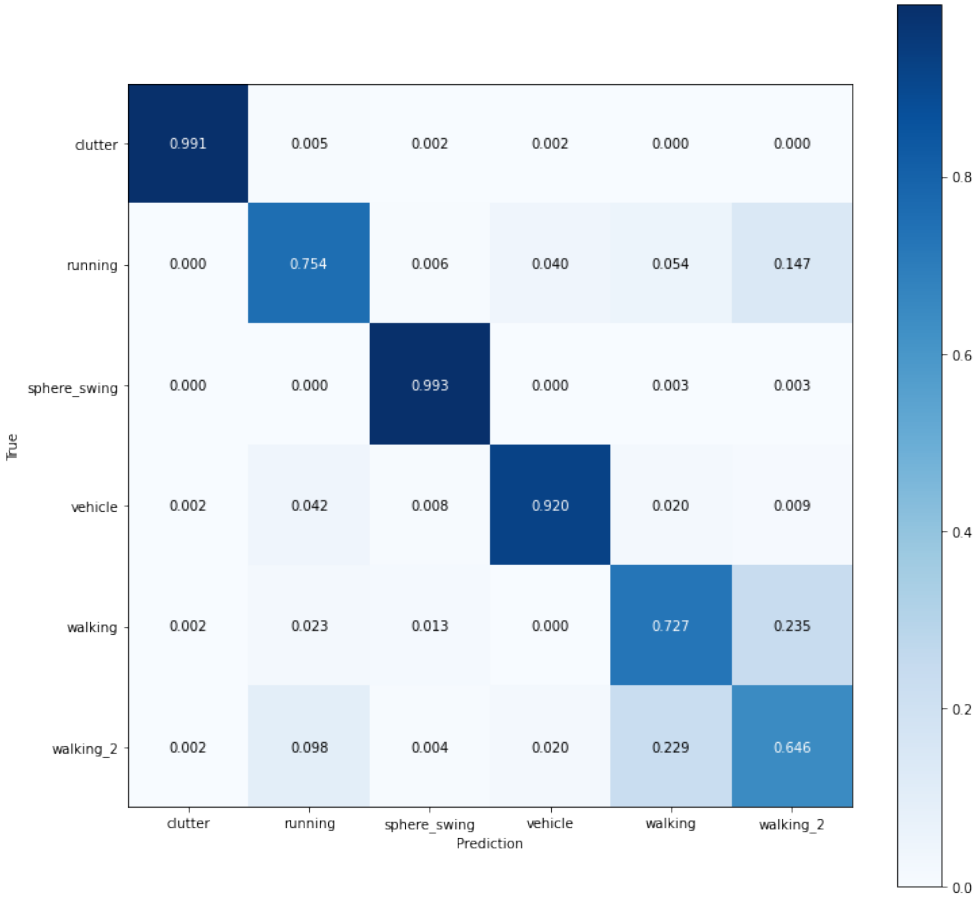

Figure 5.8: The confusion matrix of Model 2 fitted to validation data.

5.2.3 Model 3 Results

The learning curves of the model are shown in Fig. 5.9. The model reached minimum validation loss after 147 epochs. Training stopped after 167 epochs because of early stopping. It might not be possible to know with certainty why Model 3 trained longer than models 1 and 2. However, one explanation is that GAP helps the neural network optimization by smoothing, flattening, and stabilizing the loss landscape [101]. Consequently, the combination of a smoother loss landscape and a decaying learning rate (50% every 10 epochs), resulted in a model that took longer to reach the minimum.

The model was evaluated at the minimum validation loss, and it reached a training accuracy of 96.14% and a validation accuracy of 86.84%, an improvement over both Models 1 and 2.

The confusion matrix of model 3 (Fig. 5.10) shows that it had higher accuracy in distinguishing human activities than Model 2. The minimum accuracy in the human activity

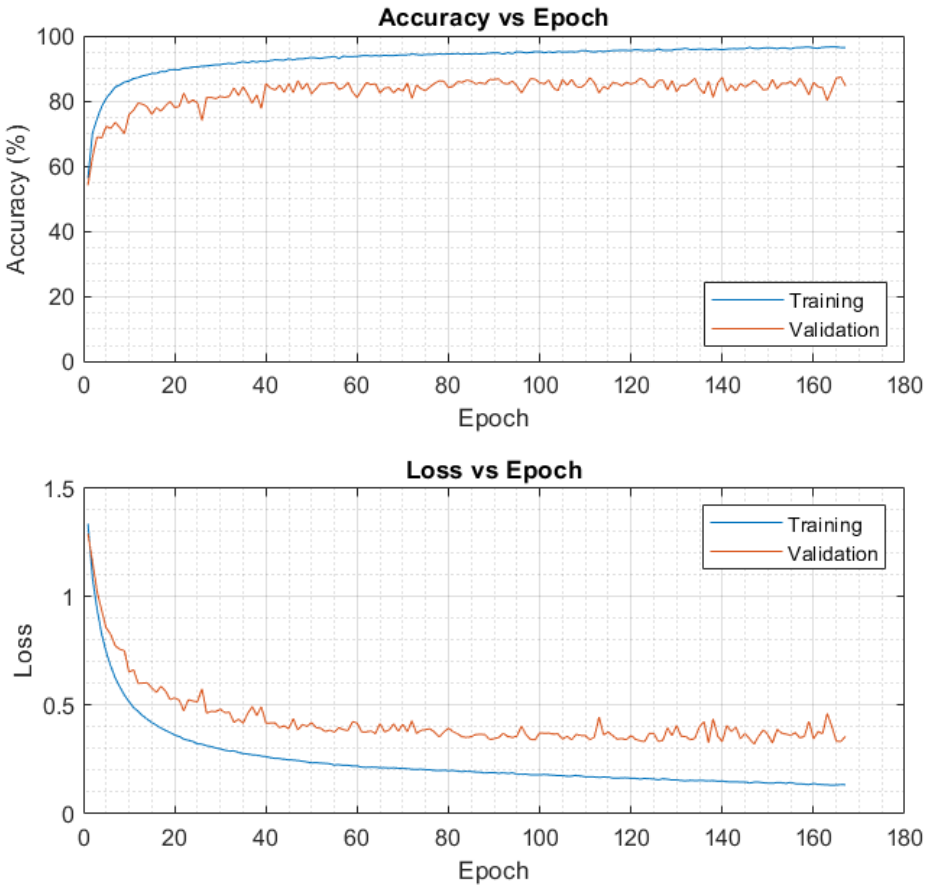

Figure 5.9: Training and validation curves of Model 3.

classes was 73.7% for one person walking. The improvement in validation accuracy compared to Model 2 is plausibly due to GAP, which improves model generalisation [74].

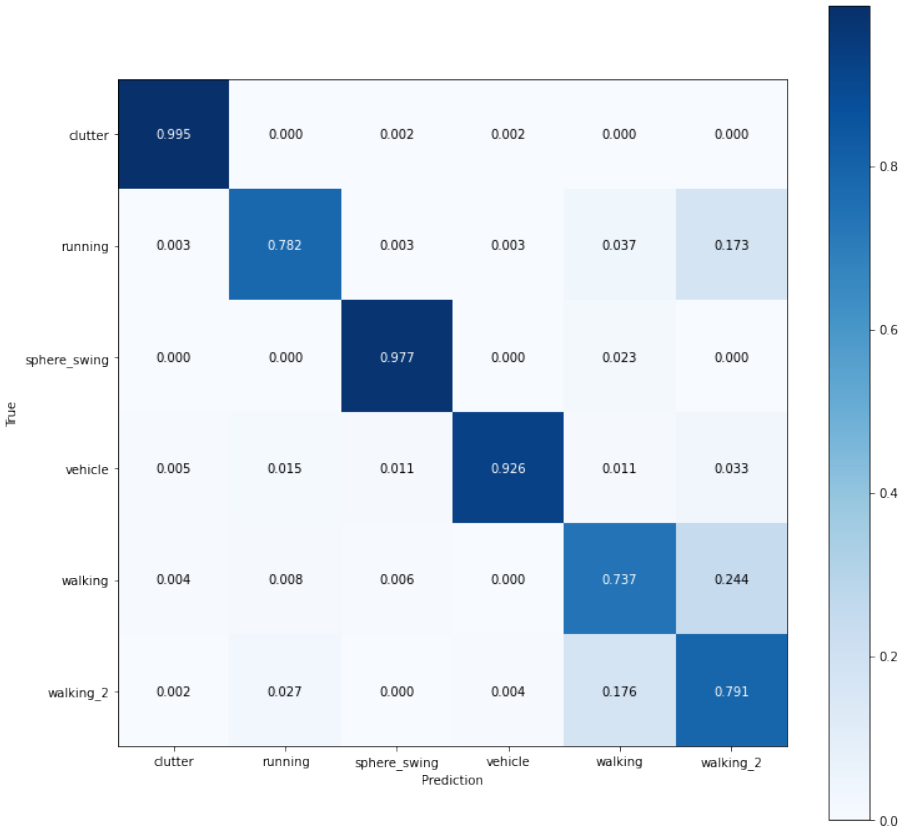

Figure 5.10: The confusion matrix of Model 3 fitted to validation data.

5.2.4 Model 4 Results

In optimising Model 4, it was found that beyond seven convolutional layers, the classification accuracy improved by less than 0.05% for a significant increase in the number of trainable parameters. Therefore, Model 4 used seven convolutional layers.

Model 4 learning curves are shown in Fig. 5.11. The model reached minimum validation loss after 84 epochs. Training stopped after 104 epochs because of early stopping. The model was evaluated at the minimum validation loss and achieved a training accuracy of 96.69% and a validation accuracy of 92.90%, a significant improvement over Model 3.

The confusion matrix of Model 4 (Fig. 5.12) shows that the model improved in distinguishing human activity. The minimum accuracy in the human activity classes was 83.8% in the one human walking class. This was significantly higher than the 51%, 64.6% and 73.7% achieved by Models 1, 2 and 3. The overall 6% improvement in validation accuracy, compared to model 3, was plausibly due to residual connections, which facilitated the addition of more layers to

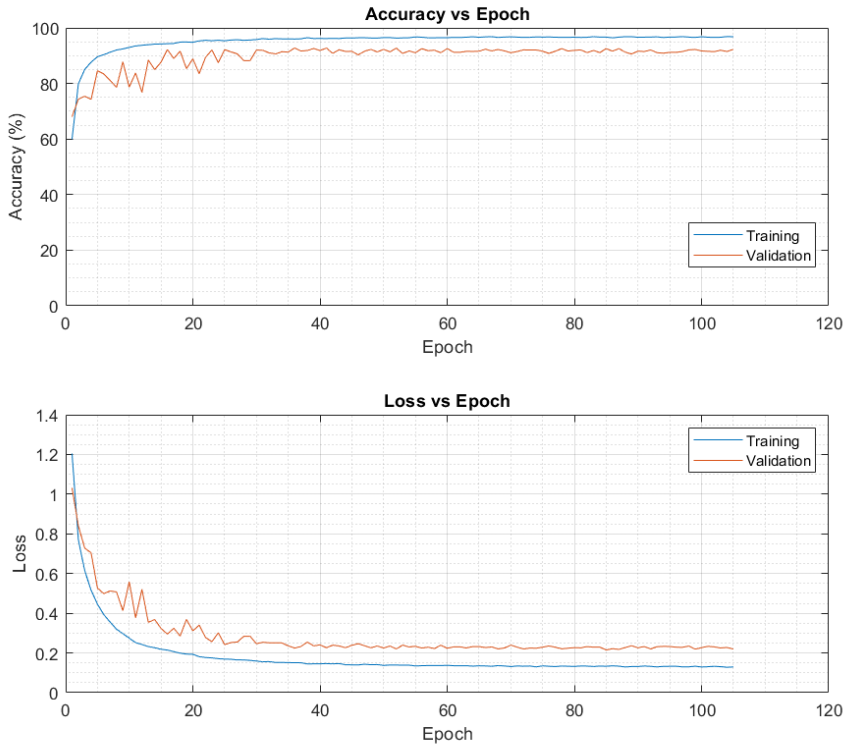

Figure 5.11: Training and validation curves of Model 4.

the network to learn more abstract features without degrading its performance [57].

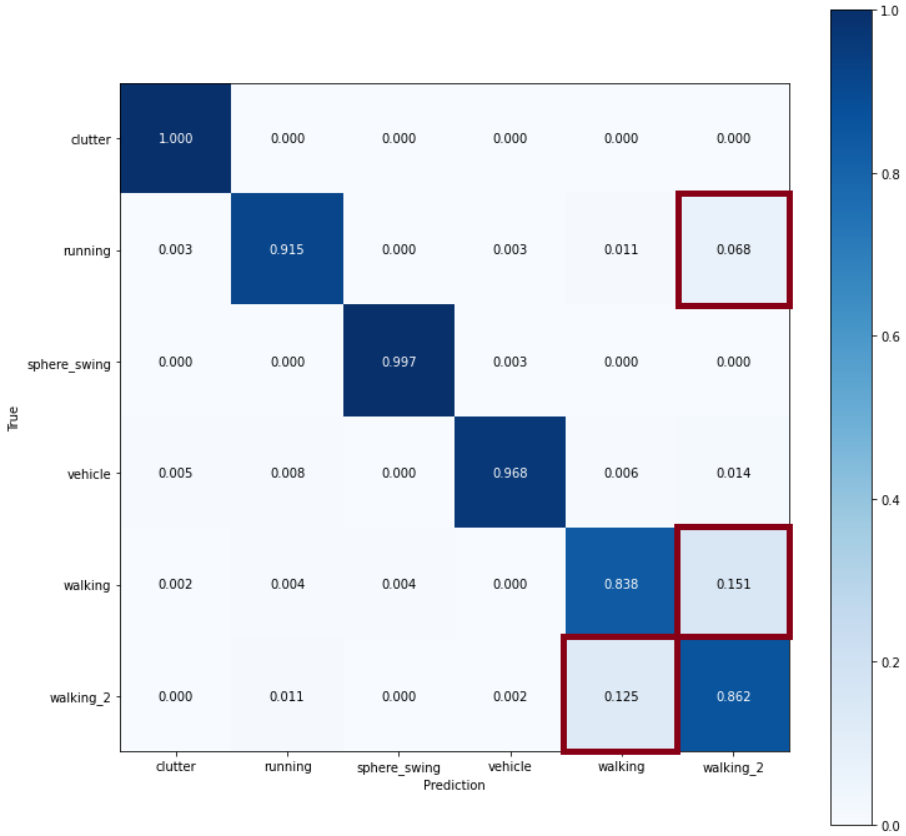

Figure 5.12: The confusion matrix of Model 4 fitted to validation data highlighting (in red) three pairs of most confused classes.

The results from all the models are summarised in Table II. Model 1 had 442 886 trainable parameters. It achieved a significantly lower accuracy (82.99%) than models 3 (86.84%) and 4 (92.90%). Model 3 and 4 applied GAP after the feature extractor. GAP averaged the results from output feature maps of the feature extractor. Consequently, fewer parameters were needed in the fully connected layer, reducing the number of total parameters in the model, which helps prevent over-fitting [74]. Plausibly, the reduction in the number of parameters resulted in better model generalisation, which improved the classification results.

Model 4 outperformed all models by using residual connections. Residual connections allowed the depth of the model to be increased without suffering a degradation in performance. As networks grow deep, they learn increasingly powerful features that are essential for discriminating between different classes [102]. This is a plausible reason for Model 4 achieving higher accuracy than Models 1, 2 and 3.

Model	Number of Parameters	Training Accuracy (%)	Validation Accuracy (%)
Model 1	442 886	94.16	82.99
Model 2	199 814	97.02	83.83
Model 3	19 206	96.14	86.84
Model 4	56 454	96.69	92.90

Table II: Accuracy results of Model 1, 2, 3 and 4.

5.3 Cross-Validation

Ten-fold cross-validation [103] was used to assess the accuracy of the top two models (Models 3 and 4) on unseen data. The training and validation data were combined, and the test data was excluded from the process. The combined data was divided into ten folds. Examples from each class were divided equally across the folds based on time, as shown in Fig. 6.1. The model was trained on nine folds in each training session, using the left-over fold for validation. Ten training sessions were held to ensure that each fold was used for validation exactly once.

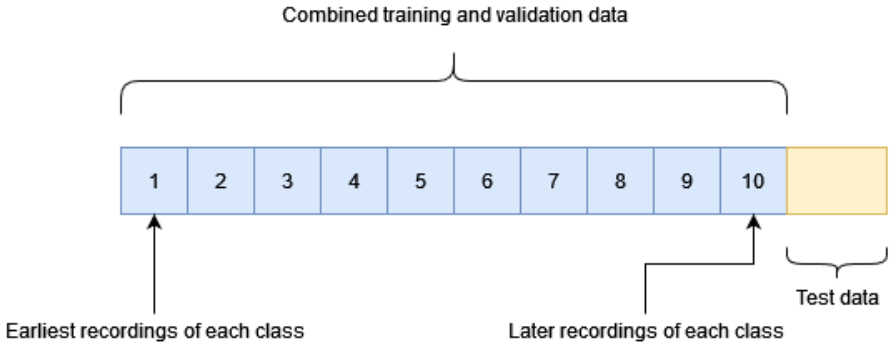

Figure 5.13: Data preparation for cross-validation.

Model 3 achieved an average cross-validation accuracy of 91.32%, with a minimum accuracy of 90.42% on the last fold and a maximum of 96.54% on the second fold. Model 4 reached an average cross-validation accuracy of 93.02%, with a minimum accuracy of 91.15% on the first fold and a maximum of 96.21% on the second fold. The results indicated that the data in the second fold was the easiest to classify for both models.

Models 3 and 4 achieved 91.32% and 93.02% average cross-validation accuracy, respectively. Therefore, Model 4 was chosen as the best model as it showed better predictive performance than Model 3.

5.4 Analysis of Best Model

Model 4 reached a validation accuracy of 92.90% and a cross-validation accuracy of 93.02%. An analysis was done on the top three classes of examples that the model found most difficult to classify, highlighted in red in Fig. 5.12.

Most of misclassified spectrograms had weak micro-Doppler returns. These weak returns result from measurements taken from longer distances. From Equation 1, the power of the received signal is inversely proportional to the fourth power of the target's range. Therefore, longer target ranges result in weaker micro-Doppler returns. Such spectrograms are shown in Fig. 5.14 (b, d, h, i and j), Fig. 5.15 (c, d, e, h and i), and Fig. 5.16 (c and g).

Other misclassified spectrograms did not have visible micro-Doppler returns. These spectrograms include Fig. 5.14 (c and f) and Fig. 5.16 (a). One plausible reason for this was that the baseband samples used to generate the spectrograms were taken from range bins in which a target of interest had left. Given that the model classified these spectrograms under human activity instead of the clutter class suggests that there might be very weak micro-Doppler returns in the spectrograms that the CNN model can detect.

A small proportion of the misclassified spectrograms appeared to be mislabelled. Examples of these include Fig. 5.14 (j), Fig. 5.15 (b) and Fig. 5.16 (e), which show micro-Doppler returns of a swinging sphere that were classified as human activities. The spectrograms were potentially misclassified as human activities because of the stronger micro-Doppler returns between the -2.46 m/s and +2.46 m/s region of the spectrum, as highlighted in the figures. Strong returns within this region usually come from torso micro-Doppler returns in human activity data.

5.4.1 Test Data Performance

Model 4 was evaluated on the test data and achieved a test accuracy of 87.72%. Note that [34]'s model achieved a test accuracy of 84.41%. Thus, the proposed model achieved an accuracy that was 3% higher than the baseline model.

The highest accuracy was 100% on the clutter class, as shown in Fig. 5.17. The model's accuracy in each class was higher in the test set than the validation set except for the one human walking class, which achieved an accuracy of 49.4%. This was significantly low compared to the 83.8% accuracy achieved in the validation dataset on the same class.

An analysis of the misclassified spectrograms of the one human walking class revealed that most of the misclassified spectrograms had weak micro-Doppler signatures, as shown in Fig 5.18. This was possibly due to a longer target range than other recorded measurements. The weak micro-Doppler returns make it more difficult to distinguish features between the two classes. To prevent this in the future, it may be necessary to have multiple radars deployed

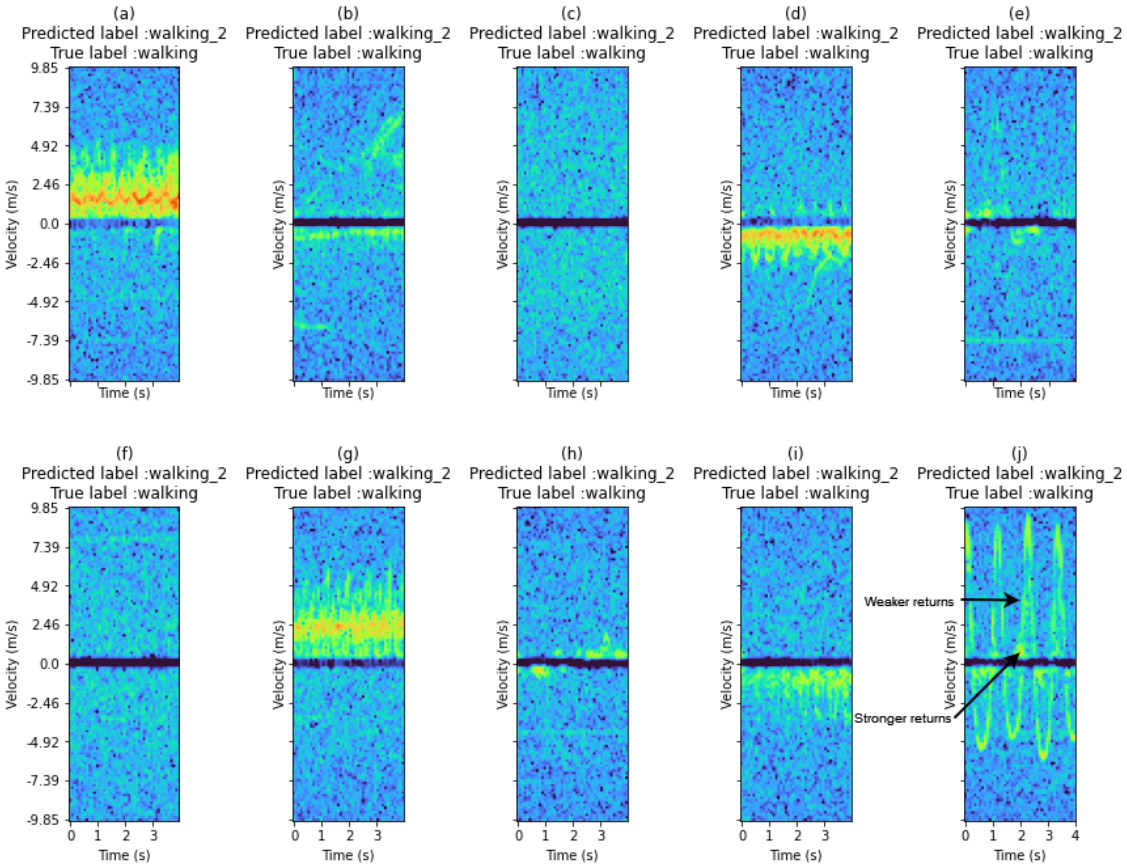

Figure 5.14: Examples of misclassified one-person walking spectrograms.

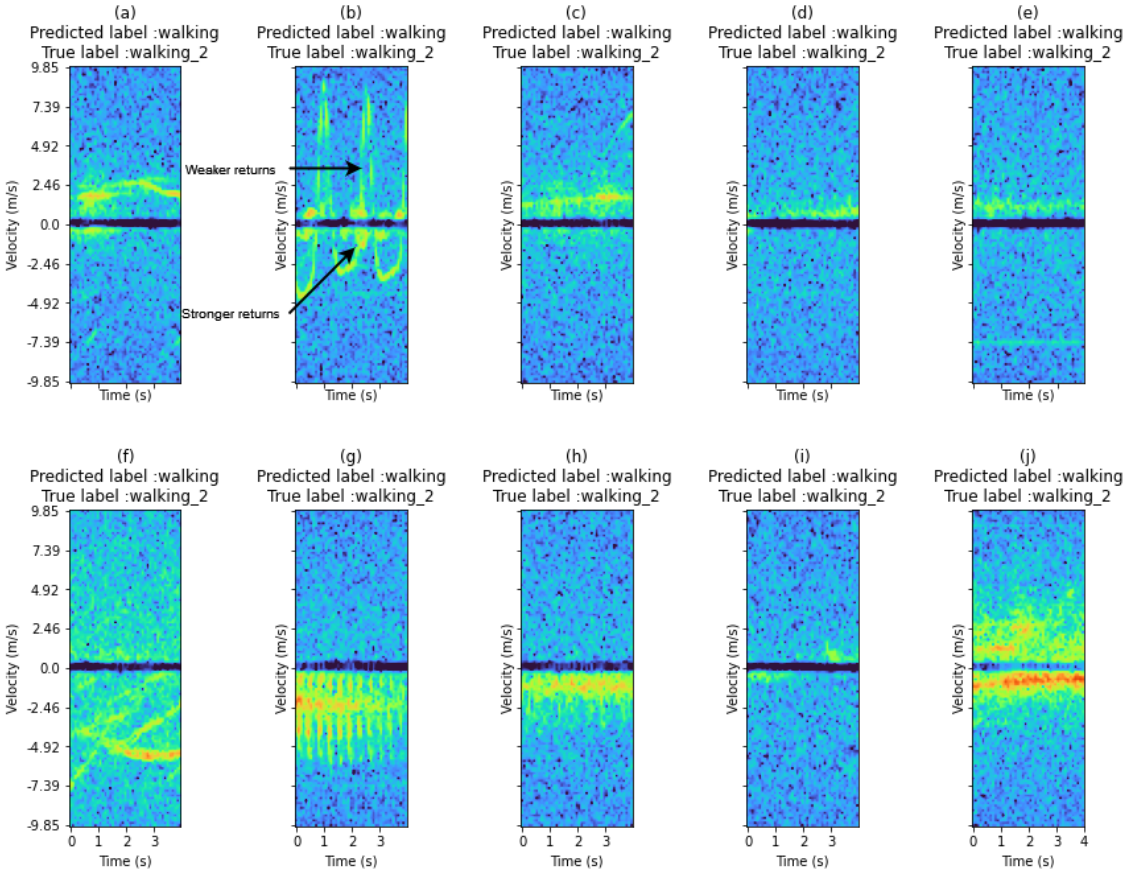

Figure 5.15: Examples of misclassified two-people walking spectrograms.

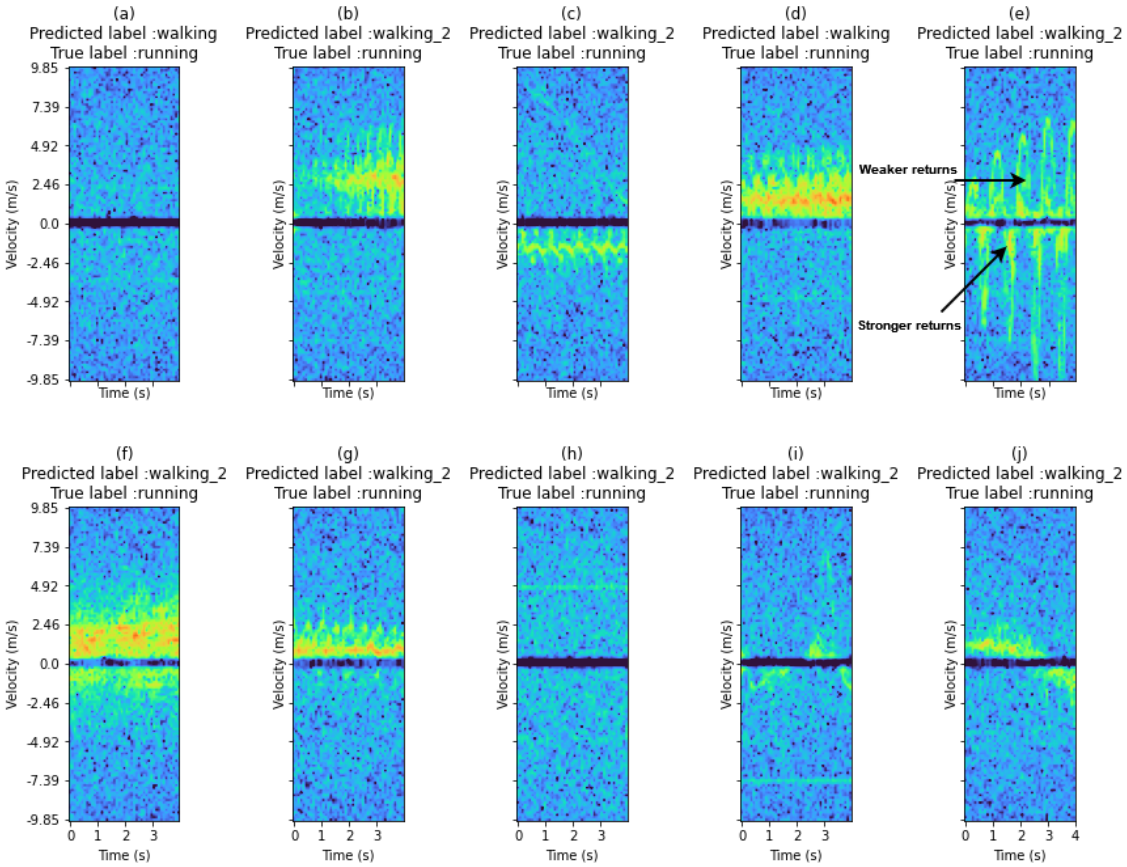

Figure 5.16: Examples of misclassified one-person running spectrograms.

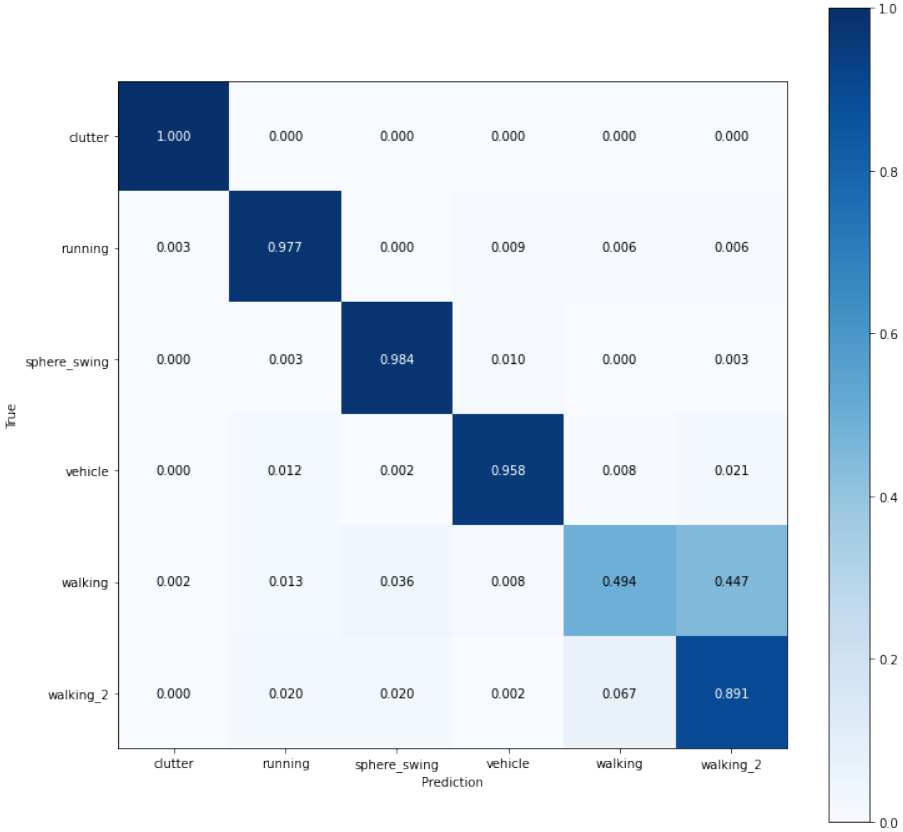

Figure 5.17: The confusion matrix of Model 4 on test data.

in the surveillance area so that the distance between the radar and target is minimised, to receive stronger micro-Doppler returns.

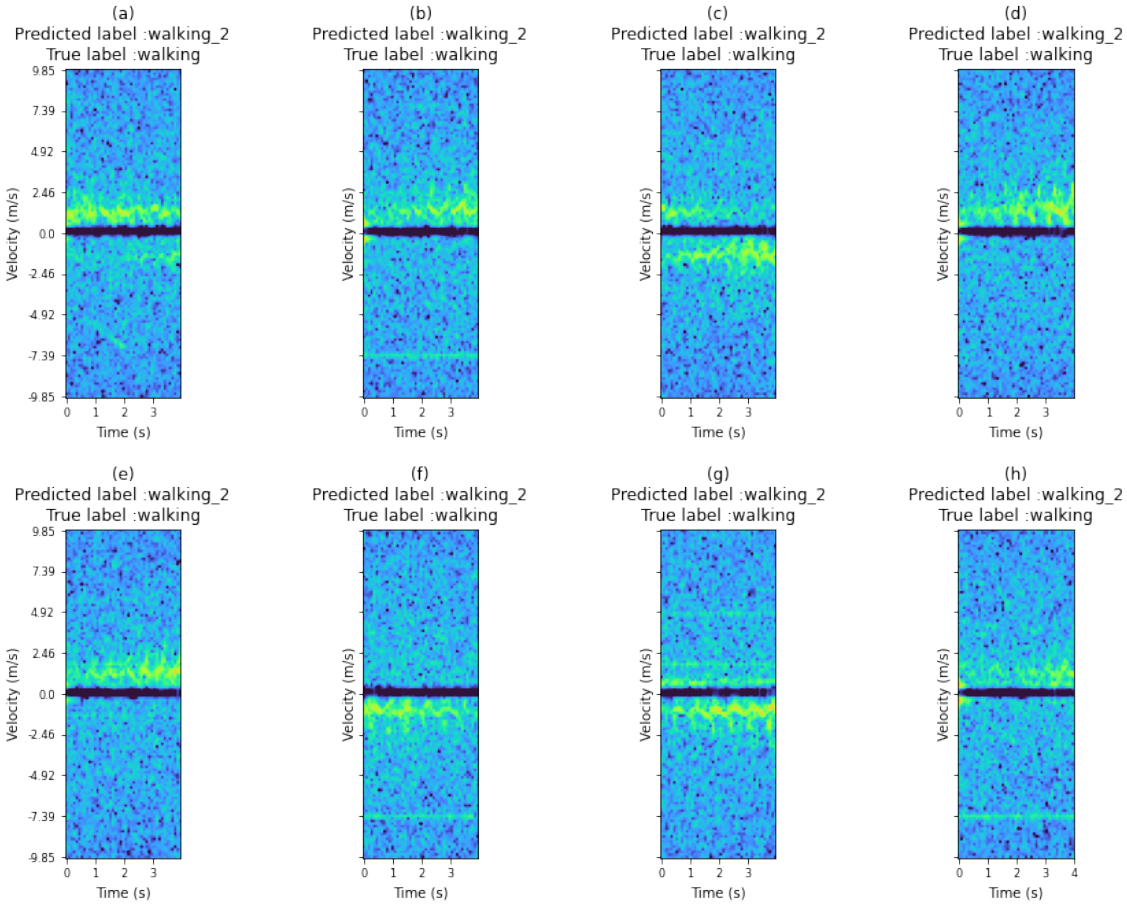

Figure 5.18: Misclassified spectrograms in the test dataset.

6 Inference

Inference is the process of classifying data after model training. The inference speed (throughput) of a model must be sufficiently high so that the model can support real-time applications. As discussed in Section 2.5, one way of improving a model's throughput is by PTQ. PTQ reduces the precision of a pre-trained model's parameters (weights and activations) and input data to run computations in low-bit precision, which is faster. However, the speed gain may come at the cost of accuracy.

This section investigated the effect of PTQ on the throughput and accuracy of model 4. The results from the investigation were used to determine if the throughput of the model could be improved without a significant degradation in performance. In addition, an inference pipeline was simulated to determine the maximum range supported by Model 4 in real-time applications.

6.1 Throughput and Accuracy

6.1.1 Implementation

PTQ was implemented in TensorRT [104] using the PyTorch FP32 (PFP32) Model 4 from Section 5.1.4. The PFP32 model was optimised by TensorRT and quantised to FP16 and INT8, resulting in the following models:

- TFP32 - The TensorRT optimised FP32 model.
- TFP16 - TensorRT optimised FP16 model.
- TINT8 - TensorRT optimised INT8 model.

Quantisation was performed using a calibration dataset of 84 randomly selected examples from each of the six classes to make a total of 504 examples as recommended in [38]. Cross-entropy was used for quantisation range setting as it was found to minimise accuracy loss compared to the Percentile approach discussed in Section 2.5.1.

To measure throughput, there was no need to use real measured data because the focus was on the speed at which data was processed, regardless of its class. Therefore, the throughput of the four models was calculated using randomly generated spectrograms in 1 000 batches. Batch sizes of 16, 32, 64, 128 and 256 were considered. The formula used to calculate throughput was as follows:

$$Throughput = \frac{N}{T} \qquad (spectrograms/s) \tag{10}$$

where N is the total number of classified spectrograms and T is the total time taken to complete inference.

After calculating throughput, the accuracy of the four models was compared to the validation and test set to assess how quantisation affected the model's performance.

6.1.2 Throughput Results

The throughput results are shown in Fig. 6.1. The model's curves show an increase in throughput as the batch size increases. This is because bigger batch sizes allow more spectrograms to be processed in parallel, leading to higher throughput. However, the throughput plateaus after a batch size of 128 due to limitations on the number of parallel processes that can be run using the available computational resources.

Fig. 6.1 also shows that the PFP32 model had the lowest throughput among the four models. The PFP32's highest throughput was 5 000 spectrograms/s using a batch size of 256. TensorRT optimisation doubled the throughput of the model to 10 000 spectrograms/s. Quantising the TensorRT optimised model to INT8 precision resulted in a 3.8 times speed gain from 10 000 spectrograms/s to 38 000 spectrograms/s.

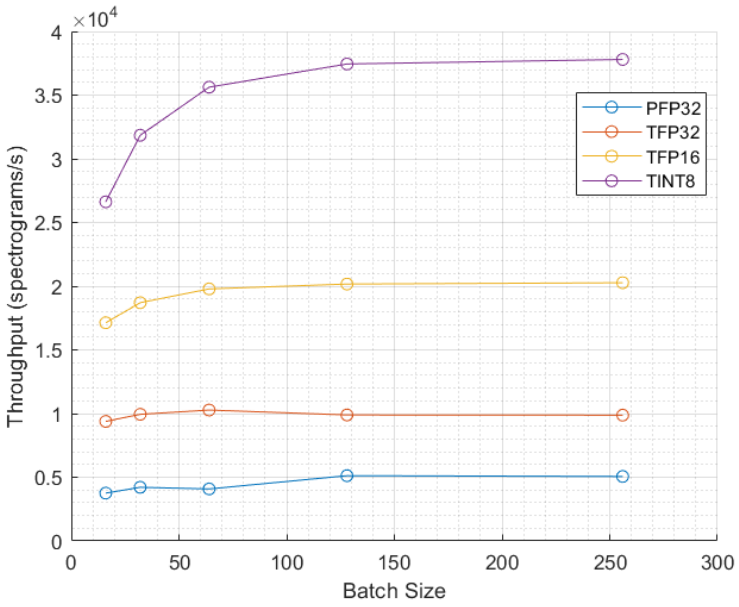

Figure 6.1: Curves of data throughput varying with batch size and quantisation precision.

6.1.3 Accuracy Results

The predictive performance of the four models was assessed on the same validation and test data that was used to evaluate Model 4's performance in the previous chapter. The accuracy results are summarised in Table III.

Quantisation led to a slight decrease in accuracy in both the validation and test data. It was observed that INT8 quantisation had a higher accuracy drop compared to FP16 in both

datasets. The maximum percentage drop was 0.33% on the validation data and 0.38% on the test data due to INT8 quantisation. The percentage drop in accuracy was higher for the INT8 model than the FP16 model, potentially because of an increase in information loss when moving from 32-bit to 8-bit precision compared to 16-bit precision. However, quantisation may help regularise the model in some instances, which may improve accuracy.

These accuracy results cannot be directly compared to the results from literature because results from other researchers show that the effect of quantisation depends on the model and dataset type, as discussed in Section 2.5.3.

The accuracy and throughput experiments showed that the loss in accuracy due to quantisation was less than 0.4%, and the gain in speed reached 3.8 times using TensorRT optimised models. This showed that the proposed model could be quantised to increase throughput while effectively maintaining high accuracy.

6.2 Inference Pipeline Experiments

6.2.1 Implementation

An inference pipeline was created to simulate the processes followed in classifying activities or targets using radar data. The pipeline setup assumed that data was already available in memory, and other processes such as pulse-compression, beamforming, and decimation had already been performed.

Fig. 6.2 shows the pre-processing and inference stages of the pipeline. The pre-processing stage involved filtering out clutter and computing the STFT using complex random samples in range bins. The samples were stored as complex floats of size 64-bits, with each component of the complex number being assigned 32-bits. The sampling frequency of the samples was the effective PRF of 714 Hz, which is the same as that of the real data used in this study.

The pipeline would wait for 2 856 (PRF$\times$4 seconds) samples to accumulate in a range bin when processing data. Note that 2 856 samples are needed to generate a spectrogram of 4 seconds duration with size 128 $\times$ 45 using the STFT parameters from Section 4.2.1. The spectrograms would then be collected and fed to the TINT8 model, which would process the

Model	Validation Accuracy (%)	Test Accuracy
PFP32	92.90	87.72
TFP32	92.90	87.72
TFP16	92.75	87.70
TINT8	92.57	87.34

Table III: Comparison results of Model 4 with different precisions.

data in batches of size 256.

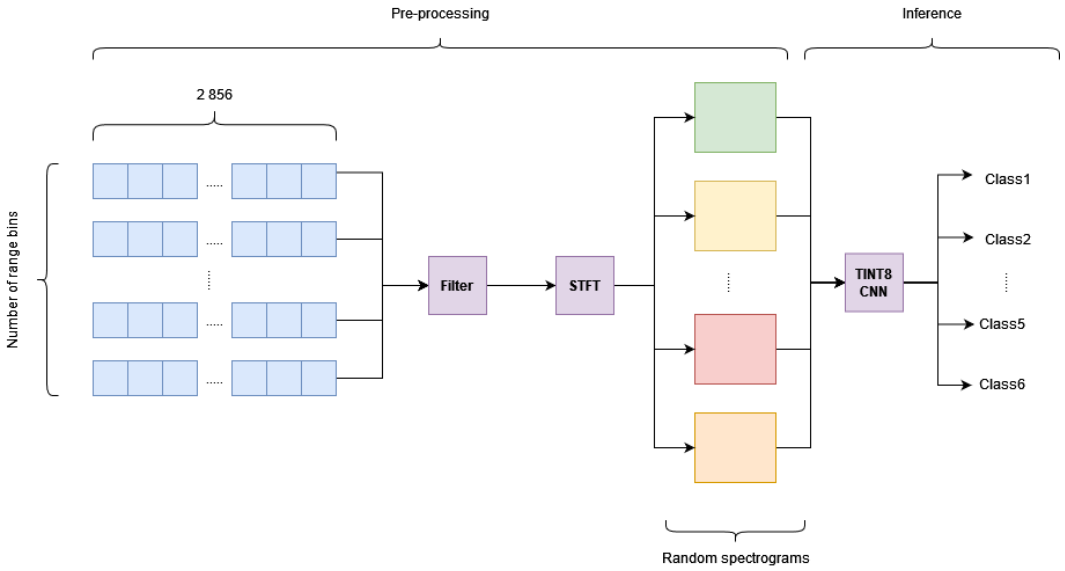

Figure 6.2: Inference pipeline combining the pre-processing and inference stages.

The total time taken for data to pass through the pipeline was measured for various range bins, starting from 3 000 to 18 000 in steps of 3 000. The total processing time was measured 100 times for each number of range bins.

6.3 Results

The results from the experiments (Fig. 6.3) showed that the inference pipeline could support a maximum of 15 000 range bins in real-time. In 15 000 range bins, all the processing times are within four seconds, which is the time taken for 2 856 samples to accumulate in a range bin. Beyond 15 000 range bins, the pipeline will not be able to process the data in under four seconds. Consequently, there will be no real-time support since the rate at which data will be entering the pipeline (PRF) would exceed the rate it can be processed. The pipeline could support a range of 89.91 km (range resolution $\times$ 15 000) in real-time, assuming the data was coming from a single receive beam. If there are two receive beams, then the data will be coming in at twice the rate, and the pipeline will only be able to support half of the range. Thus, the maximum range will scale linearly in this manner depending on the number of receive beams available.

Note that in a radar system, various processes occur before computing the STFT, which were not considered in this pipeline. These processes include pulse-compression, beamforming, and additional steps like decimation. Therefore, the throughput bottleneck might come from these preceding processes, which would reduce the maximum range that the entire system can support.

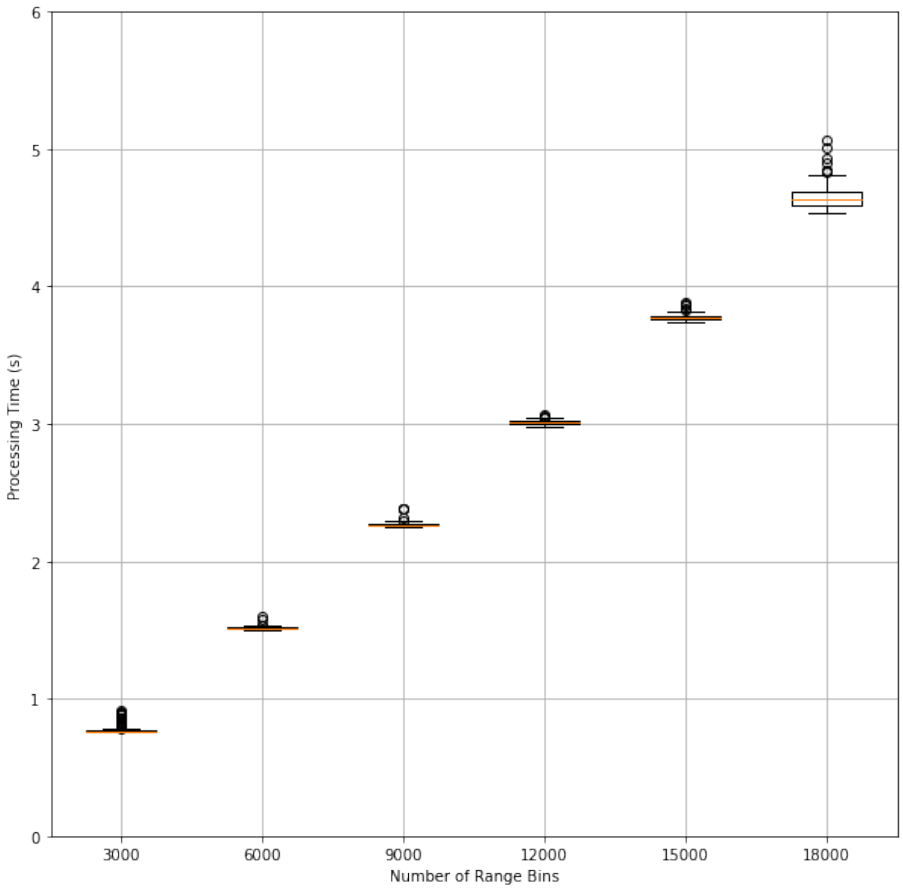

Figure 6.3: Box and whisker plots of the time taken to pre-process and infer data using the inference pipeline in Figure 6.2.

7 Conclusion & Recommendations

7.1 Conclusion

This study's first objective was to improve the results achieved by [34] in classifying human activities and targets using radar data collected at the CSIR campus in Pretoria. Parker's [34] work was the first to use the CSIR dataset to perform classification using a CNN. For this reason, [34]'s model was used as a baseline model, and work was done to improve the performance. The second objective was to investigate if PTQ can be used to improve the inference speed of the model while maintaining its performance.

7.1.1 Objective I

To achieve the first objective, complex baseband samples collected and labelled by the CSIR were converted to spectrograms using the STFT algorithm. The parameter values of the algorithm were selected using available literature and the PSNR of the data. Examples of 4 seconds duration were extracted from the generated spectrograms in a sliding window fashion and were fed to CNNs for training. It was found that residual connections, BN and GAP, improved model performance. This resulted in a validation accuracy of 92.90% and a test accuracy of 87.72%. The validation accuracy was 10% more than [34]'s model, and the test accuracy was 3% more than [34]'s. Despite the proposed model having eight times fewer parameters than [34]'s model, higher accuracy was achieved. One feasible reason for this is that [34]'s model was over-parameterised, which led to its poor generalisation. The proposed model had a cross-validation accuracy of 93.02%, which showed its ability to maintain performance on encountering new data.

Despite improving the results from [34], the proposed model showed that it had difficulty in distinguishing one person walking from two people walking, which limited its overall performance on both the validation and test data. Weak micro-Doppler returns were present in one person walking spectrograms that the model struggled to classify. One reason for weak Doppler returns is a longer range between the target and the radar, as deduced from the radar Equation 1. To solve this issue, radars can be placed in multiple places in the surveillance area so that at any point in time, the distance between the target and the radar is minimised. This will ensure that the signal received by the radar has a high SNR, which makes it easier to identify micro-Doppler patterns in spectrograms.

7.1.2 Objective II

The second objective was achieved by utilising TensorRT to optimise and quantise the proposed model. It was found that PTQ resulted in a 3.8 times speedup of the model with less than 0.4% accuracy loss using 8-bit integer precision. The results from the inference pipeline showed that it could support up to 89.91 km of range in real-time. This can enable the model

to monitor vast amounts of land provided other preceding processes are not bottlenecks to data throughput. It is important to note that model inference throughput is also dependent on the hardware, as discussed in Section 2.5.2. If the model is a bottleneck, the throughput can be improved by using more powerful GPUs such as Nvidia TITAN V or Tesla V100.

7.2 Recommendations for Further Research

The recommendations below are made from this study's data collection, model development, and inference processes.

Firstly, a more diverse dataset would facilitate the development of a model with better generalisation capabilities. The diversity of the dataset can be improved by collecting data in different weather conditions or using more subjects for human activity measurements as done in [54]. This may help the model maintain its performance in varying weather conditions or when there are humans with different walking patterns from those of the individuals used to record the data. In addition, it is vital to know the range threshold at which a model can distinguish activities or targets accurately. Future research should consider collecting various datasets grouped by ranges. For example, one dataset may have activities and targets recorded from 100 to 500 meters away from the radar, while the other may be from 2000 to 3000 meters away. Comparing model accuracy between the datasets will help determine the maximum range that would still yield reliable results.

Secondly, at the time of writing, vision Transformers (ViTs) [105] and Convolution and self-Attention Networks (CoAtNets) have been outperforming other models such as ResNets in the ImageNet competition. Future research should consider applying these models to the problem of human activity and target classification to improve the results achieved by the best-performing model in this study.

Finally, INT8 quantisation boosted the throughput by 3.8 times in the inference stage. However, this came at the cost of a 0.38% drop in accuracy. For further research, QAT [38] can be investigated to retain the model's accuracy after quantisation.

Appendix

A Results from Additional Research

As discussed in Section 2.3, CNN input data can be extracted from spectrograms using a sliding window or random sampling along the time axis. A sliding window was used to extract the input data in this study. However, additional experiments were conducted to determine if random sampling would improve accuracy. In addition, the relationship between the selected spectrogram duration and accuracy was investigated.

A.1 Sliding Window or Random Sampling

Model 4 was trained on data created using the sliding window approach and achieved a validation accuracy of 92.90% and a test accuracy of 87.72%. Note that a spectrogram duration of 4 seconds was used to extract input data for Model 4. The exact duration was applied in the random sampling approach. A total of 10 000 spectrograms were generated for training, 2 138 for the validation and 2 142 for testing. Model 4 was then re-trained on the new data using the same hyper-parameters that were used in Section 5.

It was found that using randomly sampled data resulted in a validation accuracy of 91.12% and a test accuracy of 88.25%. Although the random sampling approach achieved a slightly higher test accuracy (by 0.53%), it had a lower validation accuracy (by 1.78%) than that achieved using data generated in a sliding window fashion. Since the random sampling approach produced a model with a higher test accuracy, it was considered slightly better than using a sliding window. However, more research needs to be carried out since the 0.53% difference was considered too small to provide a sound conclusion.

A.2 Relationship between Spectrogram Duration and Accuracy

The second set of experiments investigated the relationship between spectrogram duration and accuracy. Different datasets were created using spectrogram durations of 1, 2, 3 and 5 seconds in a sliding window fashion. The spectrogram overlap was 25%, as done in Section 4.2.2. Note that data of 4 seconds duration was already available from the pre-processing done in Section 4. Five instances of Model 4 CNNs were trained on the five datasets using the same hyper-parameters used in Section 5. The trained models were then evaluated on their dataset's validation and test set.

The results in Fig. A.1 show that model accuracy increases with an increased spectrogram duration. The increase was expected since a longer spectrogram duration results in micro-Doppler patterns being observed for a more extended time. Consequently, the model is given more information for classification, which helps improve prediction capabilities. Kim

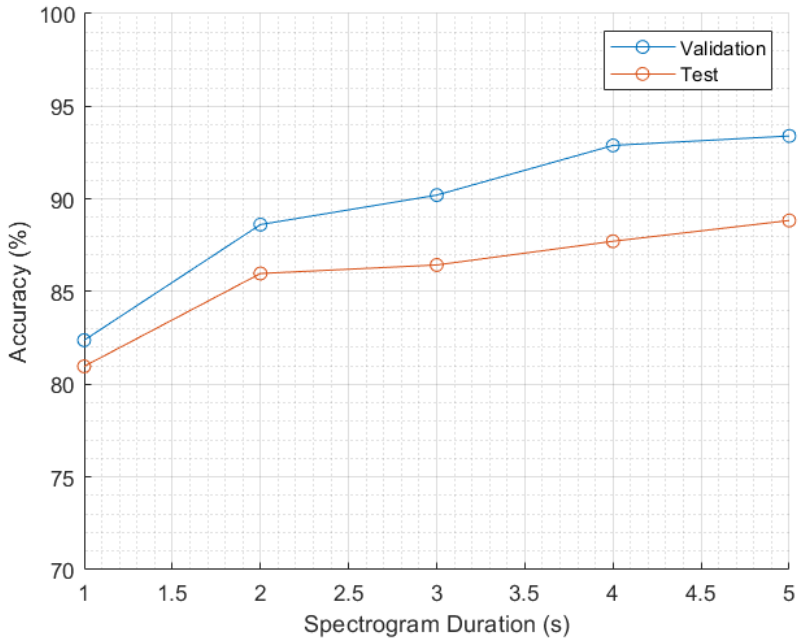

Figure A.1: Comparison results of Model 4 using different spectrogram durations.

and Moon [54] also found that an increase in spectrogram duration increased accuracy.

A.3 Conclusion

The first experiment compared randomly sampling spectrograms to using a sliding window to generate Model 4's input data. The results from the first experiment were inconclusive since both methods achieved test accuracies within 0.6% of each other.

The results from the second experiment showed that accuracy increases with an increase in spectrogram duration. However, if the spectrogram duration is increased, the waiting time also increases prior to inference. The waiting time increases to allow enough samples to accumulate in the range bins to generate spectrograms of the specified duration. Thus, there is a trade-off between accuracy and waiting time, which must be considered for inference purposes.

B Link to Code

The code for implementing data processing, model training and the inference can be found at `https://github.com/nemashy/Masters-Nyasha`

References

[1] P. Khomchuk, I. Stainvas, and I. Bilik, "Pedestrian motion direction estimation using simulated automotive MIMO radar," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 3, pp. 1132–1145, Jul. 2016.

[2] M. G. Amin, Y. D. Zhang, F. Ahmad, and K. D. Ho, "Radar signal processing for elderly fall detection: The future for in-home monitoring," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 71–80, Mar. 2016.

[3] S. Z. Gürbüz, Ü. Kaynak, B. Özkan, O. C. Kocaman, F. Kıyıcı, and B. Tekeli, "Design study of a short-range airborne UAV radar for human monitoring," in *2014 48th Asilomar Conference on Signals, Systems and Computers*, Nov. 2014, pp. 568–572.

[4] F. Fioranelli, M. Ritchie, and H. Griffiths, "Classification of unarmed/armed personnel using the NetRad multistatic radar for micro-Doppler and singular value decomposition features," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 9, pp. 1933–1937, Jun. 2015.

[5] P. Greenfield, "Poachers kill 24 rhinos in just two weeks in South Africa," *The Guardian*, Dec. 16, 2021. [Online]. Available: https://www.theguardian.com/environment/2021/dec/16/rhino-deaths-in-south-africa-from-poaching-reach-24-in-december-aoe

[6] X. Yang and Y. Tian, "Super normal vector for human activity recognition with depth cameras," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 5, pp. 1028–1039, May. 2016.

[7] F. Luo, S. Poslad, and E. Bodanese, "Temporal convolutional networks for multiperson activity recognition using a 2-D lidar," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7432–7442, Mar. 2020.

[8] D. Tahmoush, J. Silvious, and J. Clark, "An UGS radar with micro-Doppler capabilities for wide area persistent surveillance," in *Radar Sensor Technology XIV*, vol. 7669. International Society for Optics and Photonics, Apr. 2010, p. 766904.

[9] A. S. Mohammed, A. Amamou, F. K. Ayevide, S. Kelouwani, K. Agbossou, and N. Zioui, "The perception system of intelligent ground vehicles in all weather conditions: A systematic literature review," *Sensors*, vol. 20, no. 22, p. 6532, May. 2020.

[10] L. Frazier, "Surveillance through walls and other opaque materials," *IEEE Aerospace and Electronic Systems Magazine*, vol. 11, no. 10, pp. 6–9, 1996.

[11] Hensoldt, "Radar, Iff and Comms," Jan. 2022. Accessed on: Jan. 21, 2022. [Online]. Available: https://www.hensoldt.net/products/radar-iff-and-comms/#c12501

[12] InnoSent, "Security through radar technology," Jan. 2022. Accessed on: Jan. 25, 2022. [Online]. Available: https://www.innosent.de/en/sector/security/

[13] MarketsAndMarkets, "Surveillance radars market," Aug. 2020. Accessed on: Feb. 5, 2022. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/security-surveillance-radar-market-35330984.html

[14] M. S. Seyfioglu and S. Z. Gurbuz, "Deep neural network initialization methods for micro-Doppler classification with low training sample support," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 12, pp. 2462–2466, Nov. 2017.

[15] R. P. Trommel, R. I. A. Harmanny, L. Cifola, and J. N. Driessen, "Multi-target human gait classification using deep convolutional neural networks on micro-Doppler spectrograms," in *2016 European Radar Conference (EuRAD)*, Oct. 2016, pp. 81–84.

[16] Y. Shao, S. Guo, L. Sun, and W. Chen, "Human motion classification based on range information with deep convolutional neural network," in *2017 4th International Conference on Information Science and Control Engineering (ICISCE)*, Nov. 2017, pp. 1519–1523.

[17] Z. Zhang, Z. Tian, and M. Zhou, "Latern: Dynamic continuous hand gesture recognition using FMCW radar sensor," *IEEE Sensors Journal*, vol. 18, no. 8, pp. 3278–3289, Feb. 2018.

[18] S. Wang, J. Song, J. Lien, I. Poupyrev, and O. Hilliges, "Interacting with soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, Oct. 2016, pp. 851–860.

[19] J. Lien, N. Gillian, M. E. Karagozler, P. Amihood, C. Schwesig, E. Olson, H. Raja, and I. Poupyrev, "Soli: Ubiquitous gesture sensing with millimeter wave radar," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–19, Jul. 2016.

[20] X. Li, Y. He, and X. Jing, "A survey of deep learning-based human activity recognition in radar," *Remote Sensing*, vol. 11, no. 9, p. 1068, Apr. 2019.

[21] Y. Kim and T. Moon, "Human detection and activity classification based on micro-Doppler signatures using deep convolutional neural networks," *IEEE geoscience and remote sensing letters*, vol. 13, no. 1, pp. 8–12, Nov. 2015.

[22] Y. Kim, S. Ha, and J. Kwon, "Human detection using doppler radar based on physical characteristics of targets," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 2, pp. 289–293, Jul. 2014.

[23] P. Molchanov, J. Astola, K. Egiazarian, and A. Totsky, "Ground moving target classification by using DCT coefficients extracted from micro-Doppler radar signatures and artificial neuron network," in *2011 Microwaves, Radar and Remote Sensing Symposium*, Oct. 2011, pp. 173–176.

[24] R. J. Javier and Y. Kim, "Application of linear predictive coding for human activity classification based on micro-Doppler signatures," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 10, pp. 1831–1834, Apr. 2014.

[25] B. Tekeli, S. Z. Gurbuz, and M. Yuksel, "Information-theoretic feature selection for human micro-Doppler signature classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 5, pp. 2749–2762, Jan. 2016.

[26] J. Zabalza, C. Clemente, G. Di Caterina, J. Ren, J. J. Soraghan, and S. Marshall, "Robust PCA micro-Doppler classification using SVM on embedded systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 3, pp. 2304–2310, Dec. 2014.

[27] Y. Kim and H. Ling, "Human activity classification based on micro-Doppler signatures using a Support Vector Machine," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 5, pp. 1328–1337, Mar. 2009.

[28] M. Ritchie, M. Ash, Q. Chen, and K. Chetty, "Through wall radar classification of human micro-Doppler using singular value decomposition analysis," *Sensors*, vol. 16, no. 9, p. 1401, Aug. 2016.

[29] Y. Kim and H. Ling, "Human activity classification based on micro-Doppler signatures using a support vector machine," *IEEE transactions on geoscience and remote sensing*, vol. 47, no. 5, pp. 1328–1337, Mar. 2009.

[30] B. Jokanovic, M. Amin, and F. Ahmad, "Radar fall motion detection using deep learning," *2016 IEEE Radar Conference, RadarConf 2016*, Jun. 2016.

[31] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, Jun. 2018.

[32] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, Nov. 2016.

[33] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *arXiv preprint arXiv:2103.13630*, Jun. 2021.

[34] M. Z. Parker and E. Engineering, "Radar-based classification of moving targets," Nov. 2020.

[35] X. Li, Y. He, and X. Jing, "A survey of deep learning-based human activity recognition in radar," *Remote Sensing*, vol. 11, no. 9, Apr. 2019.

[36] L. Fei-Fei, J. Deng, and K. Li, "Imagenet: Constructing a large-scale image database," *Journal of vision*, vol. 9, no. 8, pp. 1037–1037, Jun. 2009.

[37] M. S. Seyfioglu, S. Z. Gurbuz, A. M. Ozbayoglu, and M. Yuksel, "Deep learning of micro-Doppler features for aided and unaided gait recognition," *2017 IEEE Radar Conference, RadarConf 2017*, pp. 1125–1130, Apr. 2017.

[38] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A White Paper on Neural Network Quantization," *arXiv preprint arXiv:2106.08295*, Feb. 2021.

[39] H. Du, Y. He, and T. Jin, "Transfer learning for human activities classification using micro-Doppler spectrograms," in *2018 IEEE International Conference on Computational Electromagnetics (ICCEM)*, Apr. 2018, pp. 1–3.

[40] M. S. Seyfioglu, A. M. Ozbayouglu, and S. Z. Gurbuz, "Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 4, pp. 1709–1723, 2018.

[41] R. Trommel, R. Harmanny, L. Cifola, and J. Driessen, "Multi-target human gait classification using deep convolutional neural networks on micro-Doppler spectrograms," in *2016 European Radar Conference (EuRAD)*, 2016, pp. 81–84.

[42] Y. Kim and B. Toomajian, "Application of Doppler radar for the recognition of hand gestures using optimized deep convolutional neural networks," in *2017 11th European Conference on Antennas and Propagation (EUCAP)*, 2017, pp. 1258–1260.

[43] ——, "Hand gesture recognition using micro-Doppler signatures with convolutional neural network," *IEEE Access*, vol. 4, pp. 7125–7130, 2016.

[44] Y. Lin, J. Le Kernec, S. Yang, F. Fioranelli, O. Romain, and Z. Zhao, "Human activity classification with radar: Optimization and noise robustness with iterative convolutional neural networks followed with random forests," *IEEE Sensors Journal*, vol. 18, no. 23, pp. 9669–9681, 2018.

[45] X. Qiao, T. Shan, and R. Tao, "Human identification based on radar micro-Doppler signatures separation," *Electronics Letters*, vol. 56, no. 4, pp. 195–196, 2020.

[46] C. Whisky, "Block diagram of a simple continuous wave radar module," Jan. 2012. Accessed on: Jan. 17, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Continuous-wave_radar#/media/File:Bsp2_CW-Radar.EN.png

[47] J. R. Klauder, A. Price, S. Darlington, and W. J. Albersheim, "The theory and design of chirp radars," *Bell System Technical Journal*, vol. 39, no. 4, pp. 745–808, 1960.

[48] B. D. Van Veen and K. M. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE assp magazine*, vol. 5, no. 2, pp. 4–24, 1988.

[49] W. L. Melvin and J. A. Scheer, *Principles of modern radar: Vol. III: Radar applications*, 2014.

[50] E. Sejdić, I. Djurović, and J. Jiang, "Time–frequency feature representation using energy concentration: An overview of recent advances," *Digital signal processing*, vol. 19, no. 1, pp. 153–183, Apr. 2009.

[51] "Window function," https://en.wikipedia.org/wiki/Window_function, accessed: 2021-10-24.

[52] G. Heinzel, A. Rüdiger, and R. Schilling, "Spectrum and spectral density estimation by the Discrete Fourier Transform (DFT), including a comprehensive list of window functions and some new at-top windows," 2002.

[53] R. Awadhiya and R. Vehmas, "Analyzing the effective coherent integration time for space surveillance radar processing," in *2021 IEEE Radar Conference (RadarConf21)*, Jun. 2021, pp. 1–6.

[54] Y. Kim and T. Moon, "Human detection and activity classification based on micro-Doppler signatures using deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 1, pp. 8–12, Apr. 2016.

[55] Y. He, Y. Yang, Y. Lang, D. Huang, X. Jing, and C. Hou, "Deep learning based Human activity classification in radar micro-Doppler image," *2018 15th European Radar Conference, EuRAD 2018*, pp. 230–233, Mar. 2018.

[56] J. Park, R. J. Javier, T. Moon, and Y. Kim, "Micro-Doppler based classification of human aquatic activities via transfer learning of convolutional neural networks," *Sensors*, vol. 16, no. 12, p. 1990, 2016.

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

[58] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[59] Z. Dai, H. Liu, Q. V. Le, and M. Tan, "Coatnet: Marrying convolution and attention for all data sizes," *arXiv preprint arXiv:2106.04803*, 2021.

[60] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[61] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[62] V. Nair and G. E. Hinton, "Rectified Linear Units improve restricted Boltzmann machines," in *ICML*, 2010.

[63] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323. [Online]. Available: https://proceedings.mlr.press/v15/glorot11a.html

[64] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International Conference on Artificial Neural Networks*. Springer, 2010, pp. 92–101.

[65] C. Molnar, *Interpretable Machine Learning*, Jan. 2019, https://christophm.github.io/interpretable-ml-book/. [Accessed on: Jan. 23, 2022].

[66] H. Leung and S. Haykin, "The complex backpropagation algorithm," *IEEE Transactions on Signal Processing*, vol. 39, no. 9, pp. 2101–2104, 1991.

[67] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

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

[69] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[70] T. Dozat, "Incorporating Nesterov momentum into Adam," 2016.

[71] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[72] J. Zhu, H. Chen, and W. Ye, "A hybrid CNN-LSTM network for the classification of human activities based on micro-Doppler radar," *IEEE Access*, vol. 8, pp. 24 713–24 720, Oct. 2020.

[73] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," *Advances in neural information processing systems*, vol. 29, pp. 550–558, 2016.

[74] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[75] F. Ramzan, M. U. G. Khan, A. Rehmat, S. Iqbal, T. Saba, A. Rehman, and Z. Mehmood, "A deep learning approach for automated diagnosis and multi-class classification of Alzheimer's disease stages using resting-state fMRI and residual neural networks," *Journal of medical systems*, vol. 44, no. 2, pp. 1–16, 2020.

[76] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[77] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018.

[78] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," *arXiv preprint arXiv:2004.09602*, 2020.

[79] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," 2011.

[80] S. Migacz, "8-bit inference with Tensorrt," in *GPU technology conference*, vol. 2, no. 4, 2017, p. 5.

[81] J. L. McKinstry, S. K. Esser, R. Appuswamy, D. Bablani, J. V. Arthur, I. B. Yildiz, and D. S. Modha, "Discovering low-precision networks close to full-precision networks for efficient inference," in *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, 2019, pp. 6–9.

[82] K. Paupamah, S. James, and R. Klein, "Quantisation and pruning for neural network compression and regularisation," in *2020 International SAUPEC/RobMech/PRASA Conference*, 2020, pp. 1–6.

[83] A. Krizhevsky, "The CIFAR-10 dataset," Apr. 2009. Accessed on: Feb. 7, 2022. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[84] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-Mnist: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[85] P. Wang, H. Li, and B. Himed, "Moving target detection using distributed MIMO radar in clutter with non-homogeneous power," *IEEE Transactions on Signal Processing*, vol. 59, no. 10, pp. 4809–4820, 2011.

[86] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[87] W. L. Melvin and J. A. Scheer, *Principles of modern radar: Vol. III: Radar applications*, 2014.

[88] M. P. Murray, A. B. Drought, and R. C. Kory, "Walking patterns of normal men," *JBJS*, vol. 46, no. 2, pp. 335–360, Apr. 1964.

[89] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, "Classification with class imbalance problem," *Int. J. Advance Soft Compu. Appl*, vol. 5, no. 3, 2013.

[90] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2011.

[91] N. V. Chawla, "Data mining for imbalanced datasets: An overview," *Data mining and knowledge discovery handbook*, pp. 875–886, 2009.

[92] G. E. Batista, R. C. Prati, and M. C. Monard, "Balancing strategies and class overlapping," in *International Symposium on Intelligent Data Analysis*. Springer, 2005, pp. 24–35.

[93] S. Visa and A. Ralescu, "Issues in mining imbalanced data sets - A review paper," in *Proceedings of the sixteen midwest artificial intelligence and cognitive science conference*, vol. 2005. sn, 2005, pp. 67–73.

[94] G. H. Nguyen, A. Bouzerdoum, and S. L. Phung, "Learning pattern classification tasks with imbalanced data sets," *Pattern recognition*, pp. 193–208, 2009.

[95] R. R. Picard and K. N. Berk, "Data splitting," *The American Statistician*, vol. 44, no. 2, pp. 140–147, Feb. 1990.

[96] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[97] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[98] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[99] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[100] D. R. Wilson and T. R. Martinez, "The general inefficiency of batch training for gradient descent learning," *Neural networks*, vol. 16, no. 10, pp. 1429–1451, 2003.

[101] N. Park and S. Kim, "Blurs make results clearer: Spatial smoothings to improve accuracy, uncertainty, and robustness," *arXiv preprint arXiv:2105.12639*, 2021.

[102] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[103] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation." *Encyclopedia of database systems*, vol. 5, pp. 532–538, 2009.

[104] N. T. Documentation, "Tensorrt Release 7.x.x," Jan. 2022. Accessed on: Jan. 22, 2022. [Online]. Available: https://docs.nvidia.com/deeplearning/tensorrt/release-notes/tensorrt-7.html#rel_7-2-3

[105] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Scaling vision transformers," *arXiv preprint arXiv:2106.04560*, 2021.