University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Adapting the WaveNet Deep Learning Model for RADAR Classification

Andrew G. Mackay

March 27, 2019

Abstract

Being able to classify targets based on radar data has many applications such as assisted living, advanced driver assistance systems (ADAS), gesture recognition and security. Current leading methods use domain knowledge to process the data into micro-Doppler spectrograms. Image classification is then applied through a convolutional neural network (CNN). This processing not only reduces the resolution of the data but also assumes this to be the optimal representation. This project develops a model to challenge that assumption by attempting to classify directly from the range profiles. These profiles have a much higher temporal resolution, somewhat similar to audio. Due to this similarity, the model adapts the revolutionary work conducted by Oord et al. (2016a) in the development of the 'WaveNet' model. The WaveNet model is a dilated causal convolutional neural network for raw audio generation. The results show that the range model performed better than the CNN/micro-Doppler approach when evaluated on new data from subjects it had already been trained on however when evaluated on its ability to generalise to new subjects it performed significantly worse. This is hypothesised to be a result of the relatively small dataset available for training, only consisting of six subjects.

Acknowledgements

I would like to thank my supervisor, Professor Roderick Murray-Smith, for providing the novel concept behind this project and for his support and guidance throughout. I would also like to thank Aleksandar Angelov for conducting the valuable initial research that this project is built upon.

Education Use Consent

I hereby grant my permission for this project to be stored, distributed and shown to other University of Glasgow students and staff for educational purposes. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Signature: Andrew Gordon Mackay Date: 23 January 2019

Contents

1 Introduction

1.1 Overview

This project aims to improve the current performance for the classification of targets from RADAR data using recent advances in the field of deep learning known as dilated convolutions. To do this, the project will explore the use of a less processed version of the data than current leading methods. This data is likely to contain more information than the processed counterpart and therefore contribute to better classification performance. By operating on this lower level data such a model will remove the need for a manually designed processing pipeline that makes assumptions about how the data should be represented. By allowing the model to learn what it considers to be an optimum representation it is expected to improve the classification performance.

The current leading methods compute the micro-Doppler signature from the data (section 2.1.4) and apply image classification techniques to these spectrograms. The data that this project aims to utilise are known as range profiles (section 2.1.2) and have a much higher temporal resolution than the micro-Doppler signatures. To handle this data, the project aims to take advantage of a newly developed architecture that has proven to be revolutionary for text to speech and has shown promise for audio classification tasks. Both the range profiles and audio are of high temporal resolution and so it is hypothesised that this new technique will be transferable. To allow effective comparison between these techniques both the micro-Doppler signature and range profile based classifiers will be explored and compared using the same underlying data.

1.2 Applications of RADAR Classification

Traditionally RADAR (**ra**dio **d**etection **a**nd **r**anging) systems have been seen as a method for detecting a target and determining its range and speed. The first practical radar systems developed were utilised during World War II for the detection of enemy aircraft. However, from radar, the classification of targets is also possible. This section outlines why research in this area is important by highlighting some of the key applications that radar classification systems can be applied to.

1.2.1 Assisted Living

There often comes a time when it is no longer safe for an elderly person to live by themselves. There are numerous reasons why this may occur such as the increased risk of falling combined with more fragile bones or the onset of dementia preventing the carrying out of routine necessities such as eating. The United Nations, Department of Economic and Social Affairs, Population Division (2017, pg. 1) report states that 'virtually every country in the world is experiencing growth in the number and proportion of older persons in the population.'. Due to the rise in the elderly population, the demand for assisted living facilities has increased.

To provide this assisted living, monitoring systems need to be developed. For this section, we will consider a fall detection system. Critically such a system must achieve two main objectives.

It must never produce a false negative as this may result in death and it must be continuously functioning as an incident may happen at any time of the day. Additionally, if a system could be made unobtrusive so as not to interfere with day to day life and also maintain the privacy of its residents then it may have the effect of encouraging the voluntary transition into assisted living. False warnings should also be limited but not at the cost of a rise in false negatives.

This task may be well suited to a radar system. The radar data provides privacy as no image of the occupant is ever recorded or stored, unlike a camera. Furthermore, the system does not need to be attached to the occupant, unlike wearable fall detection systems. Lastly, the sensor is not affected by varying lighting conditions as it relies on the reflecting of radio waves and so would be operational during the night as well. Currently, however, the performance of such a system has only been evaluated in a research setting with limited field tests. Such tests did prove effective with Liu et al. (2016) detecting six out of six natural falls over one week in a real living environment. Despite achieving 100% detection the paper states that the false alarm rate was unacceptably high.

1.2.2 Advanced Driver Assistance Systems (ADAS)

The primary motivation for development in ADAS is to make roads safer. Jackson and Cracknell (2018) report that over 1,792 people were killed on the roads in Great Britain in 2016. The Department for Transport (2011) annual report found that 'Over 60 per cent of fatalities in reported road accidents had driver or rider error or reaction'. It is clear then that the need for effective ADAS is critical.

ADAS need to work in all conditions. The consistent performance of radar sensors independent of the weather conditions such as rain, light intensity and fog make it an appealing solution.

The radar sensor currently is not as effective as other methods for classifying objects and as a result is often combined with LiDAR and camera sensors. Cameras are affected by both weather conditions and varying light intensity whilst current LiDAR systems are expensive, bulky and weigh a significant amount. Despite the reduced performance in classification, The Tesla Team (2016) stated that Tesla believes their autopilot system would be able to use the radar as the primary sensor without relying on the camera sensors however they have not implemented this change.

1.2.3 Gesture Recognition

Recently radar sensors have also become a candidate for revolutionising the way we interact with devices. Google Advanced Technology and Projects group (Google ATAP 2019) has been working on project Soli (Project Soli 2019). Project Soli aims to use radar for gesture control of devices. The high positional accuracy for precise motion detection and the ability to sense through materials are key advantages of the sensor and has been demonstrated by Yeo et al. (2018). The development of such a system would result in a completely different way of interacting with devices. The Soli team highlight the benefits of using the system in a smartwatch where screen size is limited. When using touch controls there is an issue with the finger obscuring the display while the user performs an action. Using gestures would allow full screen visibility.

1.2.4 Security/Surveillance

The sensing and tracking of people is essential in security. Radar is a popular choice for this task due to its consistent performance in varying weather conditions. Fioranelli et al. (2015)

demonstrated the feasibility for radar systems to detect armed personnel based on the micro-Doppler signatures produced by the targets gait. In the paper, to simulate a rifle a metallic pole was carried. Carrying a rifle-like object is expected to result in a significant change in a human's gait however if this could also be applied to sidearms then it may have serious applications in the security industry.

1.3 Project Aim

The common disadvantage of radar classification systems for the previously mentioned applications is that they can be limited in their ability to classify objects when compared to a camera based sensor. To overcome this, a range of classification techniques has been developed.

One successful approach involves processing the radar data to form micro-Doppler spectrograms. Handcrafted features designed by domain experts can then be extracted and conventional classification techniques such as support vector machines can be applied. This technique is discussed in section 2.2.1.

A more recent approach has been inspired by the success of deep learning in the image classification field. Researchers have applied these techniques to automatically learn the key features from the micro-Doppler spectrograms directly. This helps to remove the need for expert domain knowledge whilst maintaining similar accuracy as feature extraction methods. Section 2.2.2 examines the work done in this area.

Although reducing the necessary domain knowledge, this technique still relies on pre-processing the data to produce the micro-Doppler signatures. With this pre-processing step, it is argued that there must be some amount of information lost from the original data returned by the radar sensor.

This project aims to investigate the application of deep learning to the rich time series data that is available before processing into micro-Doppler spectrograms. This is hoped to improve classification accuracy due to the increased fine-grained data available. Furthermore, such a system would remove the need for the Doppler processing stage, allowing the model to be trained and applied to a wide range of tasks directly without the need for pre-processing knowledge. This would be effectively creating a general-purpose tool to solve radar classification.

1.4 Chapter Summary

This chapter has provided the motivation behind the project.

Chapter 2 covers the necessary background theory behind the radar processing needed to understand the project as well as the research behind the development of the model.

Chapter 3 provides an analysis of the dataset used for the project and outlines the project requirements.

Chapter 4 covers the design of the experiments used to develop and evaluate the models.

Chapter 5 states how this work was practically implemented.

Chapter 6 first covers the results that shaped the design of the models then proceeds to discuss the results from the final evaluations of the developed models.

Chapter 7 concludes the project and suggests future work to be conducted.

2 Background

2.1 RADAR Theory and Processing Techniques

This section aims to introduce the main concepts of FMCW radar and the associated processing techniques needed to understand the project.

2.1.1 FMCW RADAR

FMCW stands for 'frequency modulated continuous wave' and is a type of radar system that has the ability to detect range, velocity and angle of arrival. The system works by continuously sending out chirps. A chirp is a sinusoidal wave. The wave's frequency increases linearly with time from the start frequency f_c up to $f_c + B$ where B is the bandwidth of the system. This wave is reflected off of a target and is picked up by the system's receiver. The received wave is mixed with the transmitted wave to obtain the intermediate frequency (IF) otherwise known as the beat-frequency. The instantaneous frequency of the IF is equal to the difference in instantaneous frequency of the transmitted and received signal. The phase of the IF is equal to the difference in phase of the transmitted and received signals (Ramasubramanian 2017, pg. 2-3). The IF signal is represented by using the In-phase/Quadrature (I/Q) representation which is needed to represent both the amplitude and phase (Wolff 2018). To store this data, complex numbers are utilised with the real part representing the I values and the imaginary part used to represent the Q values. Figure 2.1a shows the raw IF data for a person standing up and sitting down for 60 seconds. As the data is represented as complex numbers, the absolute value has been taken to provide the visualisation.

2.1.2 Range Profiles

The frequency of the IF signal is related to the distance of the object that the wave reflected off to the radar. Therefore, a range profile can be obtained by applying a fast Fourier transform (FFT) over each chirp in the signal (Ramasubramanian 2017, pg. 3). The FFT is simply a fast algorithm for computing the discrete Fourier transform and so objects and their distances can be determined by looking at the peaks in the FFT processed signal. Figure 2.1b shows the range profiles obtained from applying an FFT to the IF data found in figure 2.1a.

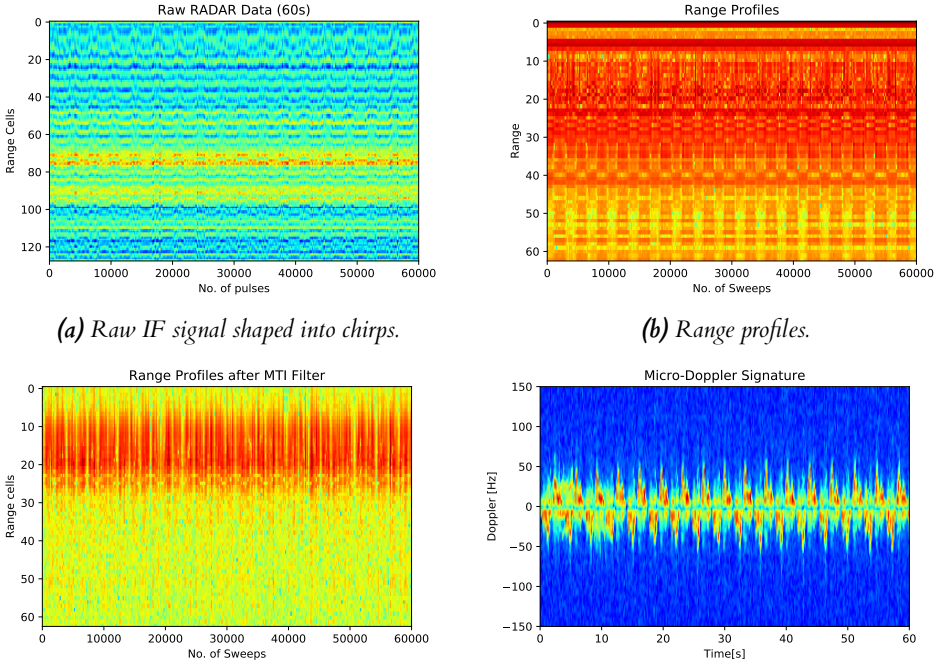
2.1.3 MTI Filter

As figure 2.1b shows, the produced range profiles can be very noisy. A Moving Target Indicator (MTI) filter can be used to help remove clutter from the range profiles. The filter removes stationary objects by calculating the Doppler frequencies. If an object is close to stationary then it will have low Doppler frequencies (Sebastian 2017, pg. 31). Figure 2.1c shows the result of applying an MTI filter to the range profiles displayed in figure 2.1b.

2.1.4 Micro-Doppler Signatures

By applying an FFT across the chirps in the range profiles the relative velocity of moving objects can be detected (Ramasubramanian 2017, pg. 4). Figure 2.1d shows the result of applying the FFT to the MTI filtered range profiles in figure 2.1c. As the data is for a subject standing up and sitting down, alternating peaks can be seen in the spectrogram (figure 2.1d) representing the target moving towards (standing up) and moving away from (sitting down) the radar. Some of the noise can be identified and removed from the spectrogram by comparing the histograms of a signature with a target and one without. Alternatively, the minimum value of the spectrogram can be increased using visual inspection to determine the cut-off.

From observation of the spectrogram (figure 2.1d), it is clear that there is more information present than simply the subject moving towards and away from the radar. This is due to the micro-Doppler effect. Any small motions additional to the bulk movement of the target can generate additional changes in the reflected frequency of the transmitted chirp. This additional information is present in the spectrogram and can be used to help in the classification of targets. For example, different engines found in cars produce vibrations which induce these micro-Doppler effects and can be used to classify the type of car (Chen et al. 2006, pg. 2).



(a) *Raw IF signal shaped into chirps.*

(b) *Range profiles.*

(c) *Range profiles after applying the MTI filter.*

(d) *Micro-Doppler signature.*

Figure 2.1: *The stages of FMCW RADAR processing demonstrated on data of a human sitting down and standing up repeatedly for 60s. Due to the complex nature of the data in all images, the absolute value has been taken for visualisation purposes. To allow clearer visualisation, for figures (b), (c) and (d),* $20 \cdot \log_{10}$ *is applied to the data. (a) shows the raw radar data shaped into chirps. (b) shows the result of applying an FFT to the raw data. (c) shows the result of applying the MTI filter to the range profiles. Lastly, (d) shows the final Micro-Doppler signature obtained by applying an FFT across the chirps of the MTI filtered range profiles.*

2.2 Current Methods Developed for RADAR Classification

As mentioned in section 2.1.4, the micro-Doppler signature can be used to help with classification. Section 2.2.1 describes work that has been done to classify radar data based on extracting features from the micro-Doppler signature and feeding these features into various classifiers. Section 2.2.2 discusses deep learning approaches that work by feeding in the raw micro-Doppler signatures directly into the classifier and allowing the classifier to 'learn' the relevant features.

2.2.1 Methods Based on Feature Extraction

One of the first demonstrations into the feasibility of identifying targets based on micro-Doppler signatures and machine learning techniques can be found in Lei and Lu (2005). The paper used simulated micro-Doppler signatures of four micro motions (vibration, rotation, coning and tumbling) to compare the performance of three classifiers. Features were extracted from the signatures by the use of Gabor filters followed by principal component analysis (PCA). The paper compared the performance of a Bayes linear classifier, a k-nearest neighbours (k-NN) classifier and a support vector machine (SVM) classifier. The results showed the SVM to be superior achieving a recognition rate of 92% when using 2600 features. The Bayes linear classifier and K-NN classifier achieved 78% and 83% respectively. The paper also compared the use of Gabor filters against applying PCA directly to the micro-Doppler signatures and found the Gabor filters to be more effective by over 10%.

Three years later Kim and Ling (2008) applied an artificial neural network (ANN) on handcrafted features extracted from micro-Doppler signatures for classifying human activities. A brief description of ANNs can be found in section 2.2.2 referred to as a basic neural network. Unlike Lei and Lu (2005), this was not a simulated dataset and contained seven classes as opposed to four making the task significantly harder. The dataset was made up of twelve humans each performing seven activities including walking, crawling and boxing whilst standing in one place. Six features were extracted from the spectrogram and fed into an ANN. The model was trained using eight of the human subjects then evaluated on the remaining four. A classification accuracy of 82.7% was achieved which demonstrated clearly that micro-Doppler signatures were indeed effective for classification.

Kim and Ling (2009) later showed that an SVM based classifier may be more effective than the ANN for micro-Doppler signature classification. The SVM classifier was applied to the same dataset with the same extracted features as described in Kim and Ling (2008). As an SVM is a binary classifier the paper combined the SVM with a decision-tree classifier to handle the seven classes. The experiment differed slightly from the original as nine of the human subjects were used for training the model. This left only three subjects for testing. The model achieved a classification accuracy of 91.9%. Although this is higher than the ANN approach it is not conclusive that the SVM is better than the ANN due to being exposed to more training data (one extra subject) and therefore evaluated on a different test set (one less subject).

From this work, it is clear that there is significant potential for classification based on micro-Doppler signatures. The techniques, however, rely on having the expert domain knowledge to perform efficient pre-processing and extraction of relevant features. This results in a lack of generalisation of the models. For example, in order to apply these proven techniques on a different research problem, effort must be put in to identify and extract the key features from the micro-Doppler signatures. Furthermore, by extracting these features, there is a loss of information from the original signature as demonstrated by the increased performance based on the number of extracted features shown in Lei and Lu (2005). These issues can be approached by the introduction of deep learning in the form of convolutional neural networks (CNNs).

2.2.2 Convolutional Neural Networks

Deep learning models have become feasible due to the significant increase in computational power that has been developed over recent years, specifically, the rise in dedicated graphics processing units (GPUs). GPUs are processing units that have been designed to handle large scale matrix computations. This increase in computational power can be used to let the model 'learn' what features are useful for classification instead of manually deciding this. At the same time, the model also learns how to classify the features. For these models to be successful, large amounts of labelled datasets are needed for training. In the field of image classification, convolutional neural networks (CNNs) have shown to be extremely effective with Krizhevsky et al. (2012) developing a CNN model which won the 2012 ImageNet Large-Scale Visual Recognition Challenge with a top 5 error rate of 15.3% which was over 10.8% better than the next runner up.

Figure 2.2 shows the structure of a basic neural network. A neural network works by having an input layer followed by one or more 'hidden' layers which are followed by an output layer. A layer is composed of 'nodes'. Traditionally each layer is fully connected to the following layer. That is to say, each node in one layer is connected to every node in the following layer. The strengths of these connections are defined by weights. Each node has an activation function which is non-linear. The weighted sum of the inputs to the node is passed into the activation function and the output becomes the output value from the node. To learn, labelled data is passed through the network and based on how far the output is from the true label (determined by a loss function) the weights of the network are updated based on gradient calculations to better fit the data. This is known as the backpropagation algorithm (Rumelhart et al. 1995). The network can be classified as deep based on the number of hidden layers. Goodfellow et al. (2016, chapter 6) is recommended for further reading on neural networks and deep learning.

For this section, two-dimensional CNNs are discussed although the same concepts can be applied to three dimensions to handle coloured images as input. A convolutional neural network is similar to a neural network however the fully connected layers are replaced with convolutional layers. The network will take as input a 2D matrix. A convolutional layer has many filters (a filter is just a matrix) which are used to detect structures in the input to the layer. This is done by convolving the filter over the input. By using many filters, different features of the data can be detected. During training, the filters are adjusted by the backpropagation algorithm. To allow detection of more complex structures the convolutional layers can be stacked. Filters in the lower layers tend to detect more basic features in the matrix such as edges and corners whilst filters higher up detect more complex shapes by combining these basic feature inputs. To help the backpropagation algorithm, regularization layers are added in between which apply an activation function to each value in the layer output. Pooling layers are often introduced in between these convolutional/regularization pairs to reduce the size of the matrices to save space. For classification purposes, a final fully connected layer with softmax activation is often used. Figure 4.1 shows the CNN architecture used in this project. For further information on CNNs, Goodfellow et al. (2016, chapter 9) can be used.

Kim and Moon (2016a) first proposed the idea of applying deep learning to remove the need for handcrafted features in radar classification. The paper applied a CNN to the raw Doppler spectrogram signatures. CNNs had already proven to be very successful in the field of image classification and therefore by treating the Doppler spectrograms as images, it was possible to benefit from the extensive research and experimentation that had been already conducted. The model was applied to two different tasks. The first task was the classification of a target from a dataset. The dataset contained a human, a dog, a car and a bike. For this task, the model achieved 97.6%. The second task looked at applying the model to the same dataset as used in Kim and Ling (2009), described in section 2.2.1. For this task, the model achieved an average accuracy of 90.9%. Although the accuracy is slightly less than the SVM technique used in Kim and Ling (2009), the paper serves as a strong proof of concept for the potential of the CNN technique.

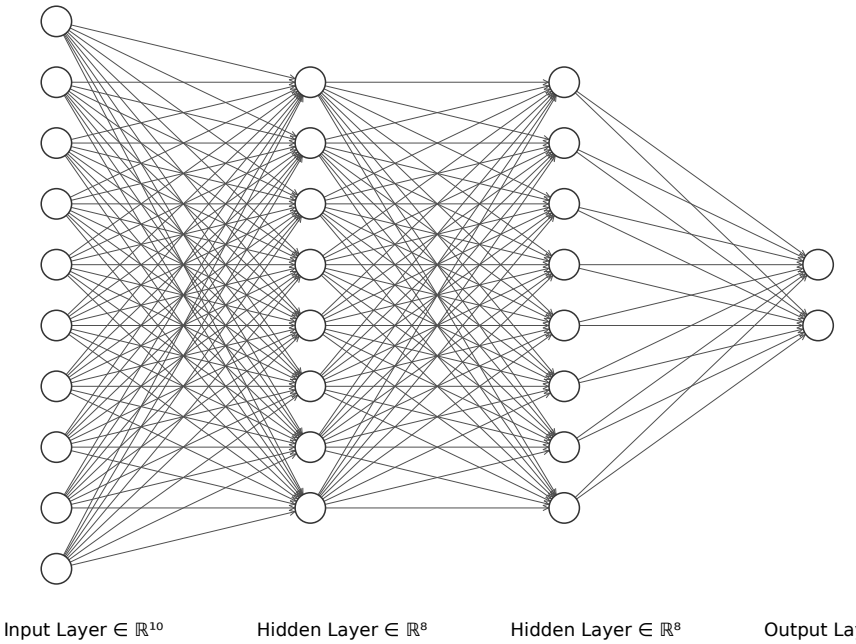


Figure 2.2: *Basic neural network architecture. The network depicted consists of an input layer with ten input nodes followed by two hidden layers, both of eight nodes, followed by an output layer consisting of two nodes. Each layer is fully connected to the following layer.*

Not only did this technique achieve similar performance compared with the handcrafted feature method but the paper also suggested that the use of deep learning could remove the dependence on expert domain knowledge for applying classification. From this proof-of-concept, several papers have been released applying the method for other research problems regarding radar classification.

In Kim and Toomajian (2016) the method is applied to the classification of ten hand gestures, achieving an accuracy of 85.6%. Furthermore, in Kim and Moon (2016b) the technique is applied to the difficult domain of classification of human activities on water. Due to the movement of waves and the increased variation in movement it is expected that handcrafted feature extraction would be less effective in this domain. It should be noted however that the experiment was conducted in a swimming pool so the effect of waves would be significantly reduced. The activities included a variation of swimming strokes and a person pulling and a person rowing an inflatable boat. The model achieved an average accuracy of 87.8% after 5-fold cross-validation.

Since this initial work, more complex deep learning models have been investigated. Angelov et al. (2018) investigated various model architectures for classifying moving targets. The dataset was composed of four types of target which were all automotive related (one person, car, bike and two people). The targets were all moving. The three architectures compared were a scaled down version of VGG16 (Simonyan and Zisserman 2014), a convolutional residual network known as Res-NET50 (He et al. 2015) and lastly a model which combined a CNN with long short-term memory (LSTM)(Hochreiter and Schmidhuber 1997). An LSTM architecture is a modification made to the standard recurrent neural network (RNN) architecture to address the issue of vanishing/exploding gradients during training. Information on RNNs can be found in Goodfellow et al. (2016, chapter 10). The modification allows long term dependencies to be modelled. The paper describes how this CNN-LSTM combination differs from the conventional approach of treating Doppler signatures as individual images as it instead considers the data as 'temporal data sequences'. When evaluated on a test set composed of target types: car, bicycle

and person, the CNN-LSTM significantly outperformed the CNN obtaining 93% accuracy compared with 79%. When evaluated on a test set composed of types: car, person and two people, the accuracy was very similar however the CNN-LSTM performed slightly worse by 1%, achieving 80% accuracy. Due to the small dataset, the results are not conclusive but the paper serves as a proof-of-concept. For future work, the paper suggests the possibility of exploring classification based on sequences of range profiles using purely recurrent network architectures.

Although the use of Doppler signatures is an effective method for removing clutter as it removes the presence of static objects, this processing causes some information to be lost from the original range profiles. By developing a model that can act directly on the rich time series data that is the range profiles it is hoped to improve classification accuracy. Furthermore, this method would also reduce the need for domain knowledge further as it removes the need for applying the Doppler processing.

In order to handle this increased amount of information present, new techniques must be considered.

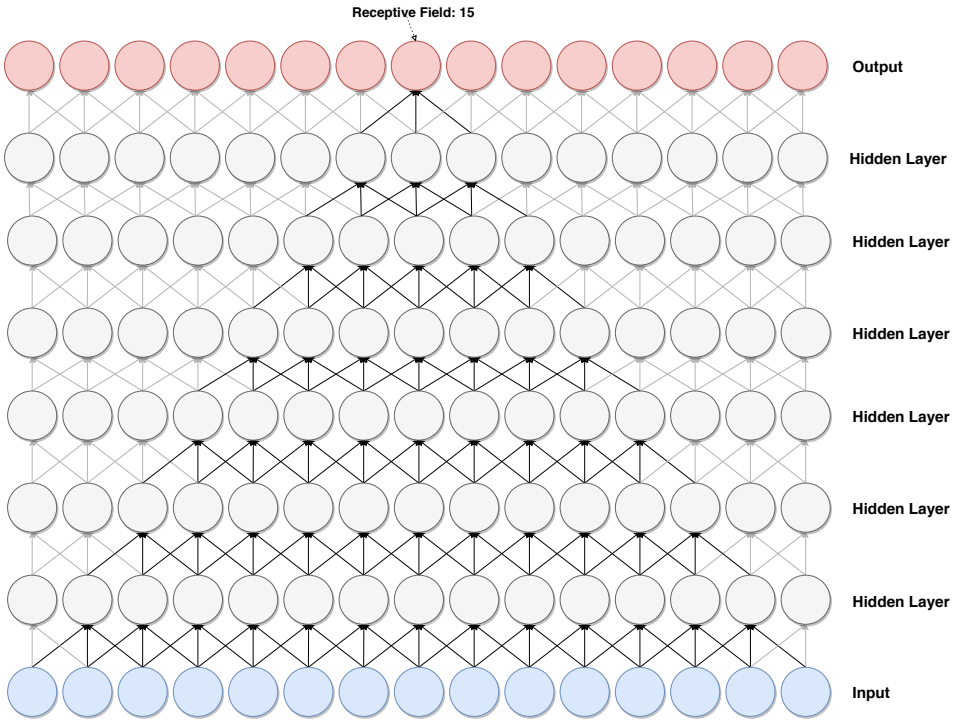
2.3 Dilated Convolutional Neural Networks

As discussed in section 2.2.2, Angelov et al. (2018) suggested the possibility of utilising a purely recurrent network architecture for the task of target recognition from the range profiles for future work. Standard RNNs have been found to suffer from the vanishing/exploding gradient problem when the number of steps gets large. As the range profile data used has a high temporal resolution of 1000 samples per second, different model architectures need to be considered.

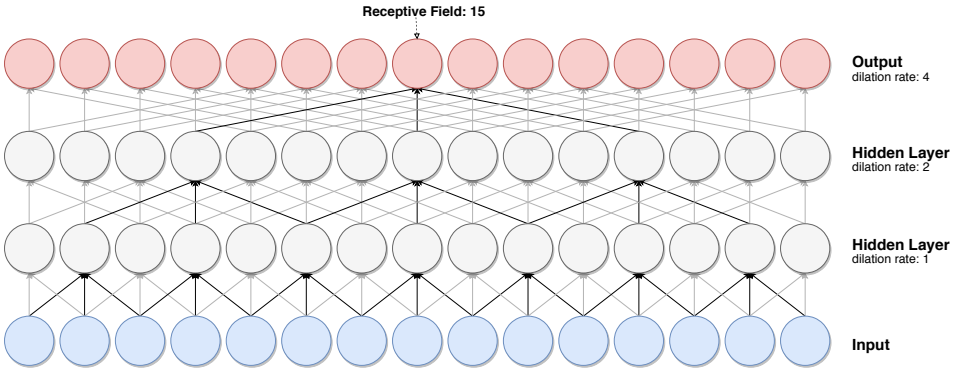
LSTM adaption to RNN networks has proven to be an effective solution to tackle the exploding/vanishing gradient problem and introduce the ability to model long term dependencies. LSTM models, however, can be slow to train as they cannot fully utilise the parallel processing advantage of the GPU. Instead, this project has been inspired by recent advances in the use of dilated convolutions for modelling long sequences.

Oord et al. (2016a) revolutionised the task of text to speech with the introduction of the 'WaveNet' model. The paper presented a generative model for raw audio waveforms which performed significantly better when compared to other models on a text to speech task. Importantly for this project, the paper also described how the model could be adapted for speech recognition tasks. With this modification, the model achieved 18.8 phone error rate (PER) on the TIMIT (Garofolo et al. 1993) test set. This is proposed as the 'best score obtained from a model trained directly on raw audio'. However, based on the results produced by Michalek and Vanek (2018) this does not make it the best performing model with other architectures reaching as low as 15.58% (lower is better).

The WaveNet model is based on stacking layers of dilated causal convolutions (Yu and Koltun 2015) combined with gated activation units. A dilated convolution is similar to a normal convolution layer however it skips over values depending on the specified dilation rate. By doubling the dilation rate for each layer, it is found that the receptive field expands exponentially with the number of layers added as opposed to a standard CNN which would expand linearly. As a result, much fewer layers are required to obtain the same receptive field as a standard CNN which means the model is easier to train and will be less affected by the exploding/vanishing gradient problem. Figure 2.3 shows a comparison of how the receptive field grows for a standard CNN (figure 2.3a) compared with a dilated CNN (figure 2.3b). In the final layer, both networks have a receptive field of 15, however, the standard CNN has three times the number of hidden layers to achieve this. In both figures, the kernel size is 3. The dilation rate for the dilated CNN is doubled for each layer. The advantage of a large receptive field is that long term dependencies can be modelled.



(a) *Receptive field growth of a standard 1D CNN with a kernel size of 3. As shown by the bold arrows, each node in the final layer has a receptive field of 15.*



(b) *Receptive field growth of a dilated 1D CNN with a kernel size of 3 and dilation rate that is doubled for each layer. As shown by the bold arrows, each node in the final layer has a receptive field of 15.*

Figure 2.3: *Comparison of receptive field growth between a standard CNN (a) and a dilated CNN (b). Both architectures have a kernel size of 3 and a receptive field of 15 however the standard CNN architecture requires three times more hidden layers to achieve this. The receptive field is found to grow linearly with the number of hidden layers for a standard CNN whilst exponentially in the dilated CNN.*

As the network is generating raw audio it needs to be causal and so each node is only conditioned on data that has been generated before it. Figure 2.4 shows a representation of this causal structure. The network shown has a kernel size of 2 which is the same as the WaveNet model.

For the WaveNet model, each node in figure 2.4 is a gated activation unit. The structure of this unit is shown in figure 2.5. This gated unit was taken from Oord et al. (2016b) who justified its use over a conventional rectified linear unit activation function as it is expected to be able to 'model more complex interactions'. As shown, the unit applies the dilated convolution to the

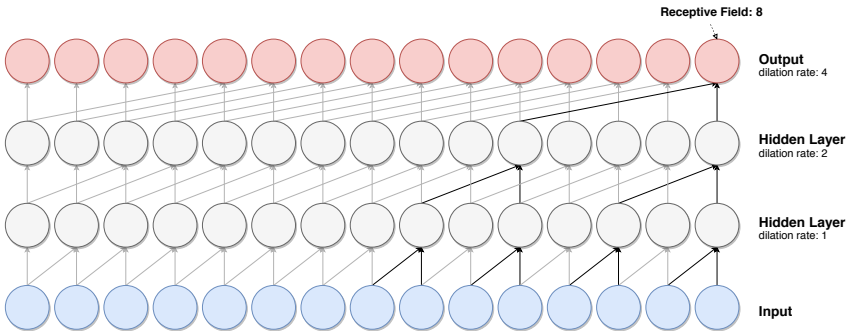


Figure 2.4: *Receptive field growth of a dilated causal CNN with a kernel size of 2 and dilation rate that is doubled for each layer. As shown by the bold arrows, each node in the final layer has a receptive field of 8.*

input then splits this output into both a sigmoid activation function and a hyperbolic tangent activation function. The outputs of these functions are then merged by multiplication to form one output. A 1×1 convolution is then applied. The output of this is then added to the original input which is known as a residual connection. From the 1×1 convolution, a skip connection is also taken. These skip connections are later added to the output of the final layer.

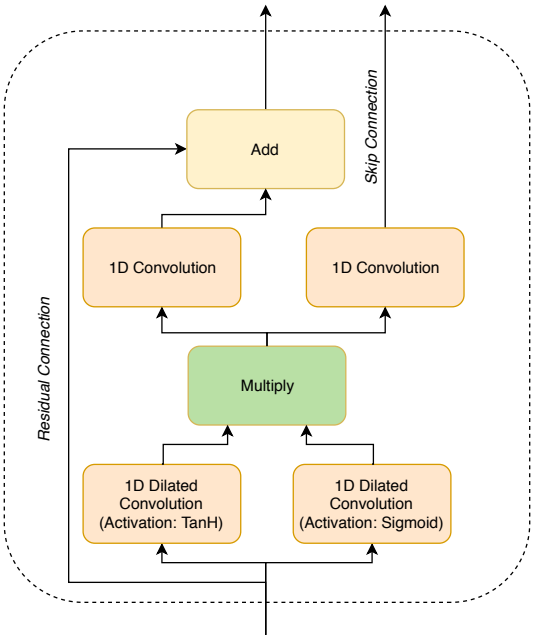


Figure 2.5: *Gated activation unit.*

To make the model suitable for classification, a mean pooling layer is added to the output of the final layer then several standard convolutional layers are added followed by a final softmax layer.

The WaveNet model uses a kernel size of two and doubles the dilation rate for each layer up to 512. The model also stacks these blocks of 512 on top of each other to further increase the receptive field. This architecture formed the core of the final model evaluated in this project. Figure 4.2 shows an annotated diagram of the model.

Strubell et al. (2017) later applied dilated convolutions for languages modelling tasks. The motivation for applying dilated convolutions instead of RNN based models is that the dilated

convolution models can take better advantage of the parallel nature of the GPU producing more efficient models which are needed to handle the large-scale data that is now available. The results showed that generally the dilated model performed similarly to the LSTM model however it was significantly faster, ranging from 8x to over 14x the speed of the LSTM model depending on the task.

Gupta and Rush (2017) used dilated convolutions for the task of modelling long-distance genomic dependencies. The large receptive field provided by the dilated convolutions allowed for much larger DNA snippets to be used as opposed to conventionally short snippets. As the data is not time-related, a non-causal implementation was used. On a short sequence dataset, the dilated models were found to outperform standard CNN models but were slightly beaten by a bi-directional LSTM model. The paper then introduced their own dataset containing significantly longer DNA sequences. On this dataset, the dilated model outperformed both the standard CNN and LSTM models on two out of the three tasks but was marginally beaten by the standard CNN on one of the tasks.

In Oord et al. (2016a), Strubell et al. (2017) and Gupta and Rush (2017), dilated convolution models were shown to perform either similarly or better when compared with LSTM based models. These tasks all involved some degree of processing long sequences of data. This success has inspired this project to investigate this concept on the high temporal resolution radar range profile data.

Since the introduction of WaveNet, the architecture has been applied to several Kaggle (Kaggle 2019) competitions in different fields. Ferreira and Silva (2016) applied a WaveNet inspired model to raw EEG signals for the Melbourne University AES/MathWorks/NIH Seizure Prediction challenge (Seizure Prediction 2016). The model is claimed to have achieved a score of 0.63 (AUC) after 10 epochs of training. The winner of the challenge achieved a score of 0.81. Although significantly less, it is stated that the team did not have access to a GPU for training and so the number of epochs were limited. For the 'Quick, Draw! Doodle Recognition Challenge' (Quick, Draw! 2018) hosted by Google AI, Mader (2018) tried a WaveNet based solution. The competition was scored by Mean Average Precision at 3 and the model achieved a score of 0.63. The top solutions all achieved scores above 90 with the winner achieving a score of 0.95. This suggests that this architecture may not be very suitable for the task of doodle recognition however it is only one sample.

2.4 Project Base

This project builds off of previous work conducted by Aleksander Angelov for his Honours project. The project consisted of using a CNN model to classify various human actions based on the micro-Doppler signature of the action. As the work is unpublished, to reference this work the following citation will be used: (Angelov et al. 2017). For this project, access was provided to the dataset created by Angelov et al. (2017), the processing code written in MATLAB (MATLAB 2019) for creating the Doppler spectrograms, scripts used for manipulating these generated spectrograms into datasets and the code used for building, training and evaluating the CNN models proposed by Angelov et al. (2017).

The radar system used to record the dataset was a FMCW radar. The specifications of the radar are listed in table 2.1

Table 2.1: *The RADAR specifications used for recording the dataset*

FMCW RADAR Specifications	
Carrier Frequency (GHz)	5.8
Bandwidth of Linear Chirp (MHz)	400
Duration of Linear Chirp (ms)	1
Doppler Frequency Range (Hz)	+/- 500
Transmitted Power (dBm)	+19
Gain of Transmitter and Receiver Antennas (dBi)	17
Antenna Type	Yagi
Antenna Beam Width Elevation (°)	24
Antenna Beam Width Azimuth (°)	24
Radar System Height from Floor (m)	1.2
Separation Between Transmitter and Receiver (cm)	40

2.4.1 Dataset

The dataset was created by recording subjects for 60 seconds performing an action repeatedly in front of the radar system. Six subjects each performed seven actions twice. The actions recorded were:

- pushing
- pulling
- moving one arm in a circle
- clapping
- picking up an object from the floor
- standing and sitting repeatedly from a chair
- walking back and forth in front of the radar

Descriptions of actions are taken from Angelov et al. (2017).

Angelov et al. (2017) developed a MATLAB program to process the raw data recordings into micro-Doppler signatures. The theory behind this processing is described in section 2.1. The 60-second spectrograms were split into 3-second spectrograms. To create sufficient data to train the CNN this was done by taking the first 3 seconds then moving on by 0.1 seconds to take the next 3-second window until the full 60 seconds had been covered.

2.4.2 Model Choice and Results

This data was then used as input to a CNN model. The model was based on a scaled down version of VGG16 (Simonyan and Zisserman 2014). For training the model, five of the six subjects were used then the model was evaluated on the remaining subject. Angelov et al. (2017) experimented with varying the number of filters used and found that with an increased number of filters, the model became more stable. All models achieved over 99% accuracy. It is not stated however which subject this was tested on. The work also investigated using different subjects for testing and found that results varied from around 75% to above 99% depending on which subject was evaluated. This suggests that there may be significant variance between how subjects performed the actions.

3 Analysis/Requirements

For this project, it was decided to focus solely on maximising classification accuracy. This was in order to more thoroughly investigate what machine learning is capable of achieving classification wise without considering constraints such as processing speed or memory footprint.

3.1 Dataset Analysis

As mentioned in section 2.4, the dataset used for this project was created by Angelov et al. (2017). The dataset consisted of six subjects (labelled A-F) each performing seven actions for 60 seconds twice in front of the RADAR. A description of the actions is listed in section 2.4.1.

Subject E only performed the walking and pushing actions twice whilst performing the remaining five actions once. This means that there is a slight class bias in the dataset towards these two actions, however, as it was only one subject, it was decided that discarding the extra actions or using sub-sampling would not outweigh the advantage of having the extra data due to the limited size of the dataset.

Large datasets are needed to train deep learning models. To increase the size of this dataset a sliding window, sliding at a rate of 0.1 seconds, was used to split each 60-second spectrogram into 3-second spectrograms. This technique resulted in over 40,000 individual spectrograms. Although 40,000 is a large number of spectrograms, there is very little difference between two consecutively generated spectrograms. This poses the risk of misleading results when evaluating the models if not addressed correctly. For example, if the test set was randomly sampled then the model will have already been trained on almost identical data making the classification results unrepresentative of the actual model capabilities. Therefore, a requirement for this project was to ensure evaluation sets were chosen meaningfully.

A common method in the image classification field to increase dataset sizes is to employ data augmentation to modify the original images by use of operations such as rotations, blurring and mirroring. This can create a much larger dataset and in turn, help the trained model generalise better. Unfortunately, due to the nature of the radar data, any standard augmentation transforms would render the data meaningless. To perform augmentation correctly, significant radar domain knowledge would be required which was considered outside the scope for this project.

With the dataset, an assumption is made that every 3-second window of the 60-second spectrogram contains enough information to distinguish one action from another.

Towards the end of the project, it was discovered that the quality of recordings varied significantly. It was found that for subject F, approximately the last 8 seconds of every recording contained only zero values. It is unclear the cause of this. This corrupted data was removed for the final evaluations.

When interpreting the classification accuracy results from this project it should be taken into account that the data was recorded in a lab environment. A system deployed for a real-world problem will likely be subject to significantly more noise and variation in the data. This is hypothesised to result in significant performance degradation.

3.2 Project Specification

As discussed in section 1.3 the aim of this project is to investigate the application of deep learning to the rich time series data that is available before processing into micro-Doppler signatures. This following section discusses how this was constrained to a feasible project for the given timescale.

3.2.1 Dataset

To allow more time for the development and experimentation with different model architectures and hyperparameter configurations only the dataset produced by Angelov et al. (2017) was used for this project. Using this dataset provided a constraint on this project due to its limited size. This was not anticipated to be an issue at the start of the project and so no additional data was recorded.

The high temporal resolution data used for this project was the range profiles (section 2.1.2). It was decided to focus on this data without applying an MTI filter (section 2.1.3). Similarly to the Doppler processing, applying the MTI filter results in some loss of information from the original data. The MTI filter requires a human decided parameter choice which this project aims to minimise. It was hypothesised that if the model could successfully learn which data is important and in doing so effectively filter out the noise then it may give better results than a humanly designed filter. Furthermore, a model architecture that requires minimal pre-processing is more general and could be directly applied to other research domains given sufficient training data.

The decision to use the range profiles over the raw data directly from the radar (whether shaped into chirps or as one vector) was made based on the inherent meaning behind the data. The way the FMCW radar system is designed, it is intended to be converted into range profiles before interpreted. Not applying this standard processing was thought to hinder the model too greatly, especially due to the limited size of the training set. Furthermore, applying the processing to form the range profiles does not reduce the temporal dimension of the data, unlike the Doppler transform.

3.2.2 Baseline

To evaluate how significant the results from this project were, a baseline was required to compare with. To provide this, a CNN model applied to the micro-Doppler signatures using the same dataset was also developed and evaluated. As the focus for this project was on classifying the range profiles, the architecture for the CNN was not be altered from the original model developed by Angelov et al. (2017) as this was shown to be effective. The parameters for this model were optimised in an attempt to create a fair comparison.

3.2.3 Requirements

This project aimed to develop a model to classify the human actions, based on the range profiles, from the dataset created by Angelov et al. (2017). The intent was to obtain comparable or higher performance than the current leading techniques of classifying the actions using the micro-Doppler spectrograms combined with a CNN classifier. The model should not rely on this engineering prior knowledge and instead implicitly learn a new set of features from scratch. To evaluate how the performance of the range data model compared, a CNN model based on the work conducted by Angelov et al. (2017) applied to the micro-Doppler signatures was tuned and evaluated on the same dataset.

4 Design

This section covers the experimental design for the investigation.

4.1 Dataset Creation

For this project, the dataset was generated from the raw measurements based on a processing method designed by Angelov et al. (2017). The theory behind the processing is discussed in more detail in section 2.1. To process a 60-second recording, first, the data was loaded in and reshaped into the number of pulses and range cells forming a 128×60000 matrix. Next, an FFT was applied over the range cells for each pulse. This gave the range profiles which was the high temporal resolution data that this project aimed to develop a model for. For generating the range dataset, the processing stopped here. The data was divided into three-second chunks by stepping a window of three seconds over the data. The step size equated to 0.1 seconds. Therefore, from one 60 second recording, $(60 - 3)/0.1 = 570$ samples were created. At this stage, the data is represented as complex values. To allow the model to process this data the absolute value was taken.

To generate the micro-Doppler spectrograms an MTI filter (section 2.1.3) was then applied to the range profiles. To generate micro-Doppler signatures from this, a second FFT was applied, this time the FFT computes over the sweeps. Only the range cells between bins 5 and 25 were used for this. The choice to limit the processing to these bins was determined from a visual inspection of the filtered range profiles. A Hamming window was used when applying the FFT with a window of size 200, pad factor of 4 and an overlap of 0.95. A spectrogram was created from each of the 20 range cells. To get the final spectrogram the absolute value of each of the generated spectrograms were summed together. This allowed the spectrogram to be converted into one image. Thresholding was used to remove noise. The value to threshold was decided manually by inspection. As with the range data, the final spectrogram was divided into three-second chunks by stepping a window of three seconds over the spectrogram with a step size of 0.1 seconds. The three-second spectrograms were saved as images and converted to 75×75 resolution. When fed into the CNN, greyscale versions of the spectrogram images were used.

4.2 Test Set

To provide a clear comparison between the model developed for the range data and the CNN model applied to the micro-Doppler signatures, the data from one full subject was withheld from the investigation during the development of the models. This allowed the final models developed to be evaluated on this unseen data and be compared directly. This technique prevented the introduction of bias into the model during the tuning of hyperparameters. It also addressed the concern discussed in section 3.1 of concurrently generated spectrograms being almost identical. Subject C was chosen to be this test subject.

4.3 Baseline Development and Evaluation

To provide a reference for how effective the model applied to the range profiles was, a CNN classification model applied to the micro-Doppler signatures was implemented. As mentioned in section 3.2.2, the same model architecture developed by Angelov et al. (2017) was used. This model was based on the VGG16 architecture proposed by Simonyan and Zisserman (2014). A diagram for this model is shown in figure 4.1. Although Angelov et al. (2017) provided results from the use of this CNN model on the dataset it was decided to repeat this process from scratch to ensure a reliable baseline.

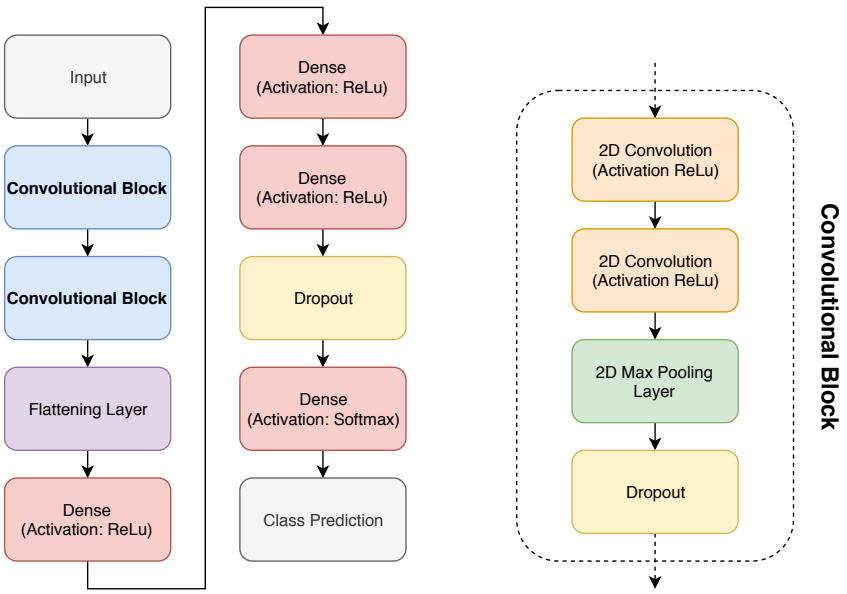


Figure 4.1: The CNN network architecture used for classifying the micro-Doppler signatures. Developed by Angelov et al. (2017) as a downscaled version of the VGG16 model (Simonyan and Zisserman 2014).

4.3.1 Filter Tuning

The first step towards tuning the CNN model was to confirm that the model architecture was complex enough to model the data. This was done by training the model for a large number of epochs using the data from subjects A, D, E and F then examining the training accuracy over the epochs. It was established that the model could achieve almost 100% training accuracy and so was sufficiently complex to model the data. During the training process, the model was validated by the data from subject B. By observing the number of epochs it took for the validation accuracy to converge, the number of epochs to train the remaining models during tuning was decided.

The next step of the investigation compared the effect of various filter combinations for this architecture. Table 4.1 shows the filter combinations evaluated. All other model hyperparameters were kept fixed during this process. These values were chosen to be the original ones used by Angelov et al. (2017). Table 4.2 shows these hyperparameter values. To compare the different filter values, five-fold cross-validation was used. Cross-validation was used to increase the confidence in the model's performance, providing a more reliable set of results at the acceptable cost of an increase in time per evaluation. For each fold, one subject of the five total subjects (subject C being reserved for final evaluation) was withheld for evaluation whilst the model was trained on

the remaining four subjects. The average accuracy of all folds was then computed and used to compare the filter performance. The accuracy for each individual fold was also saved for analysis.

The filter combination to continue the investigation with was selected primarily on the average accuracy however the accuracy distribution across the folds was also compared to ensure the chosen model maintained consistency across the subjects and did not, for example, perform exceptionally well on one fold (subject) but also significantly poorly on another and therefore lack reliability and not be likely to generalise well.

Table 4.1: *The number of filters used for each convolutional layer of the CNN model. Six different combinations were evaluated.*

Model Name	**Number of Filters**			
	Layer 1	Layer 2	Layer 3	Layer 4
'2-4'	2	2	4	4
'4-8'	4	4	8	8
'8-16'	8	8	16	16
'16-32'	16	16	32	32
'32-64'	32	32	64	64
'64-128'	64	64	128	128

Table 4.2: *The fixed hyperparameters used for the CNN model during the filter comparison.*

Parameter	**Value**
Epochs	30
Kernel Size	3×3
Dropout (in conv blocks)	0.25
Dropout (before final dense layer)	0.5
Pooling Size	2×2
Number of Dense Layers	1
Number of Dense Nodes (per Dense Layer)	512
Activation Function	ReLu

4.3.2 Hyperparameter Tuning

Using the chosen filter combination, the remaining model hyperparameters were then tuned to maximise the average accuracy based on the same five-fold cross-validation method as used for comparing the filters (described in section 4.3.1). To minimise the searches required, Bayesian optimisation using Gaussian Processes was used. This optimisation strategy has been shown to be empirically effective when the training time for a model is costly and therefore evaluations are limited. Snoek et al. (2012) is recommended for more information. The objective function to minimise was the function that returned the average accuracy from the five-fold evaluation.

The hyperparameter space searched over is listed in table 4.3. The search used 10 random restarts before approximating the function using a Gaussian process estimator. From this, followed 112 additional parameter configuration evaluations. It is expected that a better parameter set is possible and may be reached by increasing the number of random starts before making the first estimation of the function. During the search, the number of epochs the model was trained for each fold was kept constant at 30.

Table 4.3: *Hyperparameter space searched over for the CNN model. Square brackets are used to represent a range (inclusive) whilst braces are used to show a categorical list. SGD is used to represent Stochastic Gradient Descent. For the kernel size, the value represents both the width and height of the kernel as it is a square.*

Hyperparameter	Space	Distribution
Optimiser	{Adam, SGD, SGD w/ Nesterov Momentum}	N/A
Learning Rate	[0.0001, 0.1]	Log Uniform
Activation Function	{ReLu, Sigmoid, TanH}	N/A
Number of Dense Nodes (per Layer)	[16, 1024]	Linear
Number of Dense Layers	[1, 3]	Linear
Kernel Size	[2, 5]	Linear
Pooling Size	[2, 4]	Linear
Batch Size	[8, 1024]	Linear

4.3.3 Early Stopping Patience Tuning

For training the model up until this point the same number of epochs had been used. This may not be the optimal number, potentially causing overfitting or even underfitting. In an attempt to address this and therefore help to improve performance on an unseen dataset, early stopping was employed. Early stopping works by stopping the training of the model when the performance on a validation set starts degrading (indicating over-fitting). For early stopping, a patience value can be set to define how many epochs after the validation performance starts degrading to continue before stopping the training. The patience value can be tuned to prevent the training halting from random fluctuations whilst still reacting to signs of overfitting. To tune the patience value, values from 0 to 10 (inclusive) were considered and compared with the performance based on no early stopping. For this experiment, the maximum number of epochs was increased to 100. For comparison, the same average accuracy from the five-fold cross-validation as used in the hyperparameter search and filter comparison was used. For the early stopping, a training, validation and test set was needed. For each fold, the test set was composed of all the data from one of the subjects. To produce the validation set, 20% of the data, taken equally from the remaining subjects, was used. When selecting the validation set it was ensured that the 20% was taken as consecutive blocks of spectrograms to reduce the impact of consecutive spectrograms being almost identical.

4.3.4 Evaluation Strategy

After the early stopping patience had been selected, the final CNN model was then evaluated. The parameters used for the model were the optimum parameters found from the hyperparameter search. Three different evaluations of the model were conducted. Once the results had been collected there was an indication that the corrupt data of subject F (section 3.1) had significantly affected the model's performance. Therefore, the full evaluation was re-run with the final 20% of all recordings from subject F removed. Only the results from the re-evaluation have been included in this investigation.

For the first evaluation the model was trained on subjects A, B, D, E and F then the model was evaluated on subject C. During the training process, to provide the validation data for the early stopping, 20% of the data, from all training subjects, was used taken in the same consecutive method as the early stopping tuning section. Up until this point, the model had not been exposed to the data from subject C. This was intended to simulate a real-world scenario as it is not practical

in many tasks to collect data on every user of the system. Furthermore, this nullified the issue of consecutively generated spectrograms being too similar.

The second evaluation exposed the model to a set of data from all subjects during the training/validation process then used a different set, also containing data from all the subjects for testing. The data was split 60% training, 20% validation and 20% testing. To prevent the issue of the similar consecutive spectrograms having already been seen, whole consecutive chunks were extracted for evaluation. To do this, all spectrograms generated from the same 60-second recording were split into five equal groups. This was done consecutively not randomly. This was done for every recording group. For the first fold, the first group was withheld from the training set, the second group used for validation the remaining three groups used for the evaluation. For the second fold, the second group was used for the test set, the third group used for validation and the remaining groups for training. This process was continued for all five folds. The average accuracy from the five folds was then taken. As the parameters for the model had been selected based on a large amount of this data during the development phase the model will have had a degree of bias. However, this still provides a reasonable metric for how well the model performs on actions for a subject it has already been exposed to.

The final evaluation method involved using six-fold cross-validation. Each fold represented one of the subjects (this time including subject C). Similar to the first evaluation on subject C, 20% of the subjects not being used for evaluation were used to form the validation set. For this evaluation, it should be noted that as with the second evaluation method, the model will have developed a degree of bias towards five of the six subjects during development however this metric does allow an extra comparison with the range model which increases the reliability of the overall results.

To evaluate how effective the model development and tuning process had been, the three evaluations were repeated, this time using the original model parameters discovered by Angelov et al. (2017).

4.4 Range Data Model Development and Evaluation

This section describes the process used for developing the model to classify based directly on the range profile data.

4.4.1 Initial Model Comparison

Initially, three different model architectures were considered, all based around the WaveNet dilated CNN classifier architecture (see section 2.3). The first model (model 1) was designed according to the classification adaption to the WaveNet model proposed by Oord et al. (2016a). Figure 4.2 shows the structure of this network. As described in Oord et al. (2016a), an average pooling layer is added after the stacks of dilated blocks, followed by several non-dilated 1D convolution layers. The output of this is then flattened and passed through two dense layers, the latter of which has the same number of outputs as the number of classes (seven).

The second architecture (model 2) investigated was developed by Kevin Mader (Mader 2018) for classifying the Google 'Quick, Draw!' dataset (Quick, Draw! 2018). The model comprises of the same stacks of dilated convolutions as the WaveNet model however it then has a 1D non-dilated convolution layer followed by an average pooling layer. This is then passed through two more non-dilated 1D convolution layers followed by a second average pooling layer. One final non-dilated 1D convolution is added followed by a global average pooling layer and lastly a softmax activation layer to allow prediction.

The final model (model 3) was a simplified version of model 1 with the dilated block component changed to only consist of layers of dilated 1D convolution layers without the residual connections, skip connections or gated activation units. Each dilated layer was followed by a ReLu activation layer.

The data input to these models was the full range profile data with shape 63×3000. As this data is complex, the absolute value was taken.

It was found that these models took significantly longer to train than the CNN models with one epoch taking over an hour using an Nvidia GTX 1080 graphics card. By comparison, the CNN model only took a couple of minutes per epoch. As a result, five-fold cross-validation was not feasible. Instead, subject B was used for the validation set.

Each model was trained for five epochs on subjects A, D, E and F then evaluated on subject B. For each model, three different variations of the number of stacks and dilated blocks were compared. Table 4.4 shows the combinations used. For models one and two, only causal dilations were used however for the third model both causal and non-causal convolutions were tested. The remaining parameters for the models remained fixed and are shown in table 4.5.

Based on the highest classification accuracy on subject B, one model architecture was selected for further tuning.

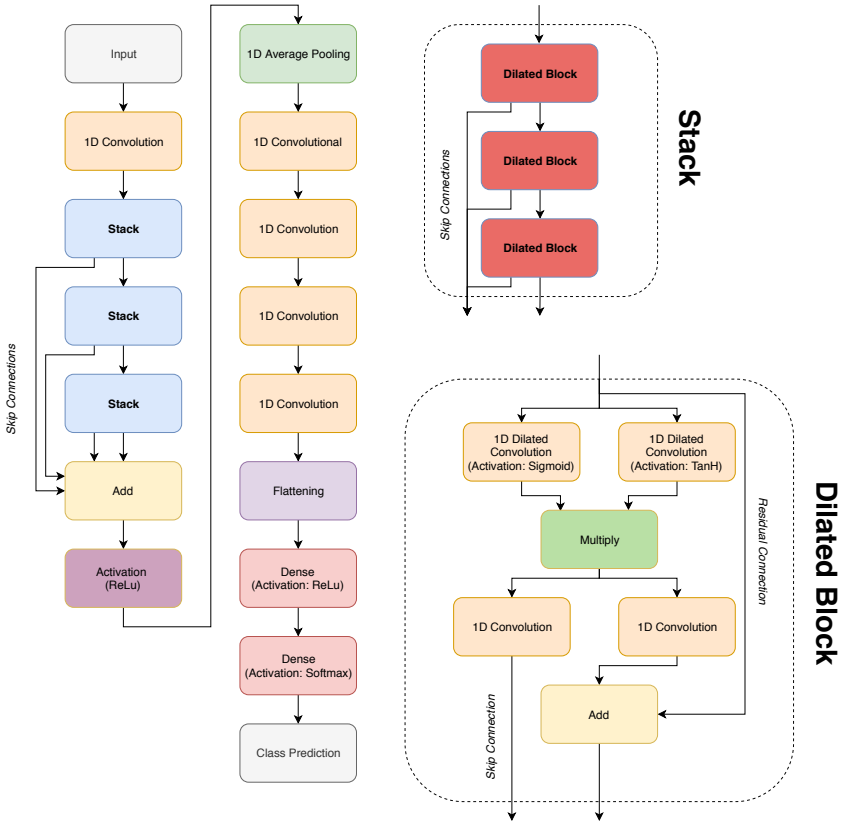


Figure 4.2: *WaveNet based classification architecture used to classify the range profile data. The figure shows three stacks with three dilated blocks per stack. Both the number of stacks and dilated blocks can be altered. The dilation rate for each dilated block in a stack is doubled for each layer added starting from a dilation of 1 (equivalent to no dilation).*

Table 4.4: *The combinations of stacks and dilated blocks investigated in the initial range model comparison (section 4.4.1). Figure 4.2 shows how the stacks and dilated blocks are used in the range model. The receptive field is calculated using the equation derived in section 4.4.1*

Number of Stacks	Number of Dilated Blocks per Stack	Receptive Field
1	4	16
3	8	766
5	8	1276

Table 4.5: *The fixed model parameters used in the initial range model comparison (section 4.4.1). The 'models' column is used to refer to which model the parameter applies to. Models 1, 2 and 3 are described in section 4.4.1.*

Parameter	Model(s)	Value
Epochs	1, 2, 3	5
Bath size	1, 2, 3	16
Optimizer	1, 2, 3	Adam
Kernel size for first conv layer and dilated conv layers	1, 2, 3	2
Pooling size for average pooling layer after dilated stacks	1, 2, 3	4
Number of filters for conv layers in a dilated stack	1, 2, 3	64
Number of filters for first two conv layers after dilated stacks	1, 3	64
Number of filters for last two conv layers	1, 3	7
Kernel size for first two conv layers after dilated stacks	1, 3	4
Kernel size for last two conv layers	1, 3	8
Number of dense nodes in second last dense layer	1, 3	512
Number of filters for first two conv layers after dilated stacks	2	64
Kernel size for first conv layer after dilated stacks	2	4
Kernel size for second and third conv layers after dilated stacks	2	8
Number of filters for third and final conv layers after dilated stacks	2	7
Pooling size for second average pooling layer	2	8

Receptive Field Formula: To compute the effective receptive field of the nodes in the final dilated convolutional layer of a model based on the number of stacks and dilated blocks, the following equation has been developed.

When considering one stack of dilated blocks, the receptive field of the final nodes can be computed using:

$$r_s = k + 2(k-1)(2^{n-1} - 1), \tag{4.1}$$

where r_s is the receptive field of the top nodes in one stack, k is the kernel size and n is the number of dilated blocks in the stack.

When these stacks are layered on top of each other it is found that the receptive field of the final nodes in this structure can be computed using the following equation:

$$r = s(r_s) - (s-1), \tag{4.2}$$

where r is the receptive field of the final nodes in this structure, s is the number of stacks and r_s is the receptive field of an individual stack.

The terminology of stacks and dilated blocks is in direct reference to the labels of figure 4.2.

4.4.2 Data Representation Investigation

Ideally, all models would be evaluated through each stage however, due to the training time, it was necessary to limit the total combination of experiments to run. Therefore, only the best model from the initial comparison was used for the following experiments.

By plotting the individual cells of the range profiles, it was found that two consecutive range cells appeared to contain very similar information. The second step of the investigation looked at comparing different resolutions of data inputs to the model. Five different inputs were tested. The first data representation evaluated was the average of all range cells which reduced the dimension of the input from the original size of 63×3000 down to 1×3000. The second representation used individual range cells as the input. For this, cells 0, 30 and 62 were investigated. Again, this limited the data size to 1×3000. The final representation took every second range cell which reduced the size down to 32×3000. The same parameters as used in the initial investigation were used.

For evaluating the different combinations, the model was trained for five epochs on subjects A, D, E and F then evaluated on subject B. From this, the data representation was chosen.

4.4.3 Causal vs Non-Causal Investigation

The model selected from the initial model investigation combined with the data representation selected from the previous stage formed the basis for this stage of the investigation. In this stage, the causal version of the model was compared with the non-causal version. The number of epochs was increased from five to ten to get a better understanding of how the model performed in relation to epochs. As with the previous two stages, the model was trained on subjects A, D, E and F then evaluated on subject B.

Two receptive field sizes were evaluated for both the causal and non-causal versions. The first combination used three stacks with eight dilated blocks per stack. The second combination investigated a much smaller receptive field, using one stack with eight dilated blocks per stack. At this stage, a slight variant of the original model which used the same kernel size for the last four convolutional layers was also evaluated. This model used a kernel size of four, one stack and eight dilated blocks. The variant was introduced to try and limit the number of variables in the model.

4.4.4 Regularization Investigation

Based on the results from the causal/non-causal comparison an architecture was chosen. From inspection of the validation accuracy over epochs from the previous investigation, it was suspected that the model was overfitting. To address this, regularization techniques were investigated.

The first technique implemented was L2 regularization. L2 regularization penalises the loss function to encourage the use of small weights. Smaller weights result in a less complex model which is then less likely to overfit the data. Three different levels of L2 regularization were investigated: 0.1, 0.01 and 0.001. For this experiment, the epochs were increased up to 20. Again, the model was trained on subjects A, D, E and F and validated on subject B.

The second technique looked at the effect of batch normalisation layers. A batch normalisation layer normalises the output from an activation function layer then sets the mean and standard deviation by multiplication and addition. These values used to set the mean and standard deviation are learned during training. This keeps the weights balanced which reduces the impact one particular weight has on the classification. It has also been found to speed up training. Batch normalisation layers were placed after each 1D non-dilated convolutional layer, excluding the first 1D convolutional layer.

The last experiment for regularization used the best performing L2 value and combined it with the use of batch normalisation layers.

Based on the results, a suitable regulation technique was added to the model.

4.4.5 Hyperparameter Tuning

From the previous four sections of tuning, the model will likely have developed a strong bias towards subject B as this was the only subject used for validation. To reduce this imbalance, the hyperparameter search used to tune the model further was changed to use 20% of the data from every subject for validation. It was made sure that this data was selected in consecutive chunks to negate the issue of consecutive range profiles being almost identical.

To perform the search, the same method of Bayesian optimisation using Gaussian Processes as used for the CNN model (section 4.3.2) was used. The objective function to minimise returned the validation loss.

The hyperparameter space searched over is listed in table 4.6. From observation of the previous results, it was found that the model did not seem to improve when training past ten epochs. Therefore, to allow a broader search in the limited time available, the number of epochs was limited to ten. Early stopping was also introduced with the patience being included in the search. Due to time restrictions, only five random restarts were used before making the first approximation of the objective function. Followed by seven additional searches. Unfortunately, the number of searches had to be kept very low due to the length of time one evaluation would take. For future work, it would be recommended to evaluate many more parameter configurations.

To evaluate if the search had been effective, the original model parameters were also evaluated and compared.

After the search had been conducted it was discovered that there was a flaw in the design. When using early stopping, the validation set for early stopping was the same data that the model was evaluated with. This meant that the model would always stop at much closer to the optimum epoch than what would normally be the case and so the results have been skewed significantly. Unfortunately, due to the time constraints of the project, it was not feasible to re-run the search.

Table 4.6: *Hyperparameter space searched over for the range data model. Square brackets are used to represent a range of integers (inclusive). In the final row, an early stopping patience of -1 is used to represent no early stopping.*

Hyperparameter	Space	Distribution
Number of filters for conv layers	[32, 128]	Linear
Kernel size for dilated conv layers	[2, 5]	Linear
Number of dilated blocks	[2, 9]	Linear
Number of stacks	[1, 4]	Linear
Pooling size	[4, 10]	Linear
Kernel size for final four conv layers	[2, 8]	Linear
Early stopping patience	[-1, 4]	Linear

4.4.6 Evaluation Strategy

Using the hyperparameters found from the search the final model was then evaluated using the exact same process as used for the CNN model described in section 4.3.4. This allowed a direct comparison to be made.

5 Implementation

5.1 Tools and Technologies

5.1.1 Python

The codebase for this project was written exclusively in Python (Python 2019). Python was chosen as it has been adopted strongly by the data science community and therefore has a wide selection of developed packages that cover common low-level data science tasks such a data manipulation and visualisation as well as packages to make machine learning accessible such as Keras (Chollet et al. 2015), PyTorch (Paszke et al. 2017) and TensorFlow (MartÃŋn Abadi et al. 2015). This allows more time to focus on experimentation and model design.

The choice of Python did come with some disadvantages. As the language is interpreted and dynamically typed, many errors were only detectable during runtime which, due to the long runtime of the programs being developed, could be very costly. To overcome this, it was ensured that before running a full evaluation, a test version of the script, limited to one epoch, was executed. As it is an interpreted language, Python is not as efficient as many compiled languages. However, as the main costly computations for deep learning tasks are performed using the GPU and low latency was not a requirement for this project, this was not an issue.

5.1.2 Jupyter Notebooks

The code for this project was written in the format of Jupyter notebooks (Kluyver et al. 2016). Jupyter notebooks allow the combination of executable Python code with markdown in the same document. This provides the ability to clearly document code in a more aesthetically pleasing format. The Python code can be run live in individual blocks which allows rapid prototyping as the developer gets instant feedback as the code is written. Furthermore, with the use of graphing packages, results from an investigation can be displayed in the same document as the code that trained and evaluated the model. This makes the experiment transparent and more accessible to future researchers.

The typical development environments for creating these notebooks only provide very limited error checking as the code blocks can be executed in any order. This is not a significant issue as this format provides the ability to run code as you work which is very useful for testing code snippets and debugging. The main drawback is there is no clear way to share code written in one notebook with another. This creates significant amounts of code duplication. Initially, in an attempt to overcome this, shared code was written in a Python package and modules imported into the notebooks when needed. This technique was abandoned halfway through the project as it was found that these packages constantly needed to be modified due to the slight differences between investigations which hindered the ability to rapidly prototype. It was also found that importing custom modules reduced the transparency of the notebooks.

5.1.3 Cookiecutter Data Science

To structure the project, the 'Cookiecutter Data Science' template was used (Cookiecutter Data Science 2019). This template style is used by many data science projects and provides a clear structure to the project. As it is commonly used, it allows easier interpretation to any other researchers as all files are in their expected locations.

5.1.4 Keras

To create and train the models for this project, the Keras library was chosen. Keras is a high-level machine learning library written in Python. The high-level nature of Keras was one of the main reasons for its selection. This allowed more focus to be made on model architecture design without having to write large amounts of boilerplate code to implement the chosen models. This allowed for rapid prototyping and development. It was also found that the use of the high-level library made the code very legible due to its syntactic simplicity which coupled well with the use of Jupyter notebooks.

Keras also has the advantage of a swappable backend including the options of TensorFlow and Theano (Al-Rfou et al. 2016). This uncouples the developed code from a particular implementation which helps to future proof the developed code. For this project, the TensorFlow backend was used.

Two alternatives to Keras were considered, TensorFlow and PyTorch. Both provide a lower level interface for developing models which allows greater flexibility for creating non-standard model layers. Both TensorFlow and PyTorch are more efficient than Keras, however, as latency was not a requirement for this project, the faster development time for Keras was considered to significantly outweigh potential performance gains.

5.1.5 Hardware

As resource requirements increased with the development of the project, a variety of hardware was used for this project. To begin with, an Nvidia 940m GPU was used. This was satisfactory for initial experiments with the CNN model. As the required number of epochs increased, the time to train made this impractical. Therefore, the Google Colaboratory (Google Colaboratory 2019) facilities were utilised. Google Colaboratory is a Jupyter notebook environment that runs in the cloud. The service provides free access to an Nvidia Tesla K80 GPU. Although this significantly sped up experiments, the session would expire if inactive. This made it unfeasible for long experiments and was only used for development. To overcome this issue, access was granted to the University's 'High Performance Compute Cluster' which provided access to an Nvidia GTX 1080 GPU and allowed jobs to be run for a maximum duration of one week. Despite the access to a very competent GPU, training time was still a significant limitation for this project. For the WaveNet classifier model, training one epoch would take over one hour.

Unfortunately, the cluster could not run the Jupyter notebook files directly and so first the notebooks had to be converted to standard Python files before execution. The ability to convert the files was provided by the development environments used.

5.2 Data Processing

The initial code for producing the spectrograms was written by Angelov et al. (2017) as a MATLAB script. To allow better compatibility with the project and to improve the understanding

of the processing, the decision was made to port this file into Python. In doing this it was found that the complex data was represented in the format $a + b\mathbf{i}$, however, Python requires the syntax $a + b\mathbf{j}$. Initially, this conversion was done in memory for each file when required however this process took several minutes per file. To speed up development time, the decision was made to convert all files to use the '**j**' format, saving to storage.

To perform the FFT transformations, the NumPy package was used (Walt et al. 2011). For creating and applying the MTI filter, the SciPy signal processing library was used (Jones et al. 2001–). For loading in the data, the pandas library was used (McKinney 2010). The availability of these packages turned what could have been time-consuming development tasks into one-line imports. One notable issue when porting the code was the difference in indexing and axis ordering between MATLAB and Python.

5.3 Data Handling

To store the processed spectrogram files, only 241MB was needed. This meant that the files could comfortably be loaded entirely into RAM during the training and evaluation of the CNN models. Issues arose however when dealing with the range profile data. After processing, the range data consumed 126GB's of storage. Due to the large size, loading the dataset directly into memory was not an option. To handle this, the use of Keras's data generator system was employed.

Data generators allow the loading of data as needed directly from storage when training a model. This can be done for each batch in an epoch. The built-in versions of the Data Generator class provide very limited flexibility for partitioning data into training, validation and testing sets. As multiple different evaluations were to be conducted, each requiring a different format of train, validation and test sets, a custom Data Generator class was implemented based on an example from Amidi and Amidi (2018). As the data needed to be loaded from storage there was a significant performance hit when compared with RAM. The delay will be a function of the hardware used but will be at least an order of magnitude slower. Combined with the increased complexity of the range model, this accounts for the significantly slower training time of the model (one hour per epoch compared with under one minute). As low latency was not a requirement, no experiments were conducted to investigate the bottleneck of the system.

5.4 Model Implementation

The implementation of the CNN model was taken directly from the code developed by Angelov et al. (2017).

The base code for the range model was taken from a popular Keras implementation of the WaveNet model (Veeling 2018) combined with the classification adaption implemented by Kevin Mader (Mader 2018) and Myeongjang Pyeon (Pyeon 2018). By utilising this pre-existing code, significant development time was saved which allowed a greater focus on design and evaluation.

5.5 Hyperparameter Optimisation

As stated in sections 4.3.2 and 4.4.5, the hyperparameter optimisation algorithm utilised was Bayesian optimisation using Gaussian Processes. The implementation provided by Head et al. (2018) was used. However, it was the unreleased, in-development, version that was used to access the newly added 'CheckpointSaver' callback. This callback was needed to save progress after each evaluation to protect against the program crashing and losing all progress.

6 Evaluation

6.1 CNN Model Development

This section discusses the results that guided the development of the CNN model used to classify the micro-Doppler signatures.

6.1.1 Number of Filters Comparison

As described in section 4.3.1, the first step towards developing the CNN model was deciding on the optimum combination of filters to use. Each filter combination was evaluated using five-fold cross-validation with each fold representing a full subject. The validation accuracy for each fold was recorded. Using this data, a boxplot for each filter combination was created, shown in figure 6.1. For brevity, the abbreviation of the combination, taken from table 4.1 is used to distinguish between the combinations. As there are only five measurements per combination (five folds), the five-figure summary represents the actual values recorded for each fold. The figure shows that all filter combinations performed similarly across the folds however the filter combination '2-4' was the least consistent with both the highest and lowest accuracies. Due to this instability, the combination was discarded. From the remaining combinations, it was chosen to go forward with '8-16' due to its higher average performance. The combination '64-128' does appear to be more consistent across the folds than '8-16' however was not selected due to its lower average performance.

6.1.2 CNN Hyperparameter Search

Figure 6.2 shows the results from the hyperparameter search for the CNN model as described in section 4.3.2. As the figure shows, over 120 different parameter configurations were evaluated. The highest accuracy achieved was 78.4%. The corresponding parameters for this result are shown in table 6.1. The figure shows that the performance started to converge from the 60^{th} parameter set evaluation. This convergence is a result of using the Gaussian process estimation for the search. It is possible that with a greater number of random starts before approximating the function that a more optimum parameter configuration could have been achieved.

6.1.3 Early Stopping Patience Tuning

As described in section 4.3.3, to tune the early stopping patience, five-fold cross-validation was used with one fold for each subject. Figure 6.3 shows the average accuracy from the evaluation for each patience value used. The graph shows that there is very little effect of patience value on accuracy with less than 6% accuracy difference between the lowest and highest average accuracy. This suggests that the model is not drastically overfitting. The graph does show that a patience value of 7 produces the highest average accuracy.

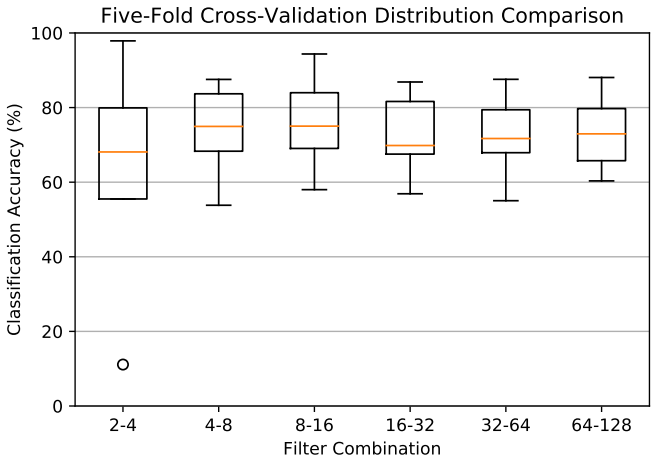


Figure 6.1: *Filter combination comparison based on the results from a five-fold cross-validation evaluation with each fold representing an entire subject. As there are only five-folds, the five-figure summary of the data corresponds directly to the recorded values. A breakdown of the filter combinations is shown in table 4.1. The figure shows that the combination of '2-4' is the least consistent. Combination '8-16' was chosen as it achieved the highest average performance.*

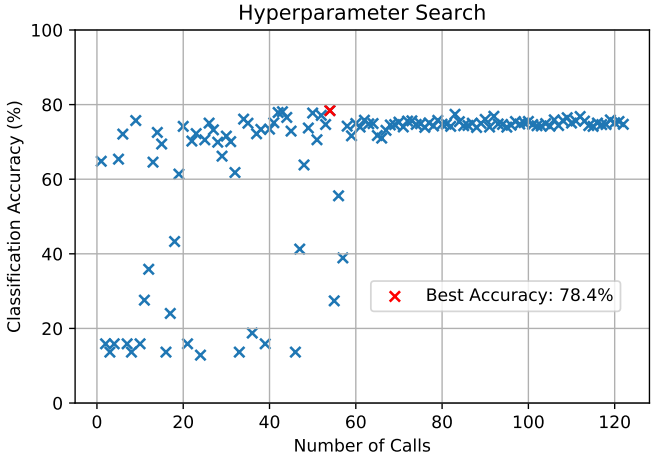


Figure 6.2: *Classification accuracy for each hyperparameter configuration during the search for the optimum model parameters. The maximum accuracy achieved is highlighted in red. The figure shows that the performance started to converge from the 60^{th} parameter set evaluation.*

To ensure that the patience of 7 was the best choice, the accuracy distribution across the five folds was compared for the top three values (7, 8 and no early stopping). This was done in the form of a boxplot and is shown in figure 6.4. The figure shows that each patience had a similar distribution however the patience of 8 had lower maximum accuracy. It was, however, the most consistent across the folds with a smaller box. However, due to the overall higher average performance of patience 7, it was selected for the final model.

Table 6.1: *Optimum hyperparameters found for the CNN model based on the search of the hyperparameter space.*

Parameter	Value
Optimizer	Adam
Learning Rate	0.001
Activation Function	ReLu
Dropout Rate	59.7%
Number of Dense Nodes per Fully-Connected Layer	1024
Number of Dense Layers	3
Kernel Size	3×3
Pooling Size	2×2
Batch Size	576

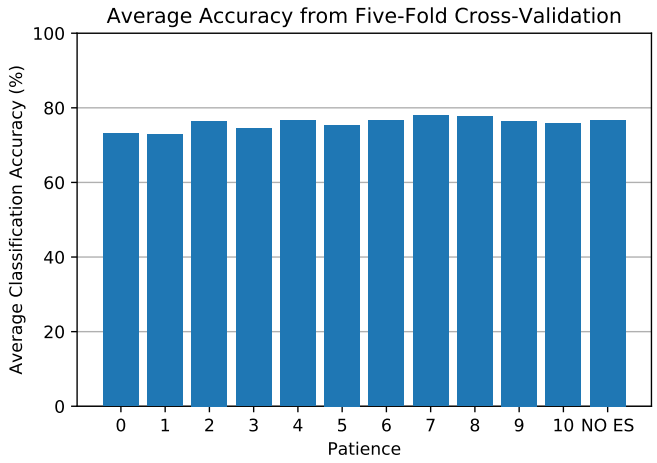


Figure 6.3: *Early stopping patience comparison for the CNN model using the average accuracy from five-fold cross-validation using one subject for each fold. 'NO ES' is used to represent training without any early stopping. The figure shows the patience value to have very little effect on classification accuracy suggesting the model is not overfitting. The patience value of 7 is shown to achieve the highest accuracy.*

6.2 WaveNet-Based Classifier Development

This section discusses the results that guided the development of the WaveNet-based classification model used to classify the range profiles.

6.2.1 Initial Model Comparison

The setup for the initial model comparison is described in section 4.4.1. Each model combination was trained on subjects A, D, E and F whilst being validated on subject B. Figure 6.5 shows the validation performance for the various combinations of the number of stacks and the number of dilated blocks per stack for models 1 (6.5a) and 2 (6.5b). The results from the third model have been excluded as it only achieved the equivalent of random guessing (14%) for all combinations evaluated. Model 1 refers to the adaption suggested by Oord et al. (2016a), model 2 for the design by Kevin Mader (Mader 2018) and model 3, a simplified version of model 1 with no gated activation units or skip connections.

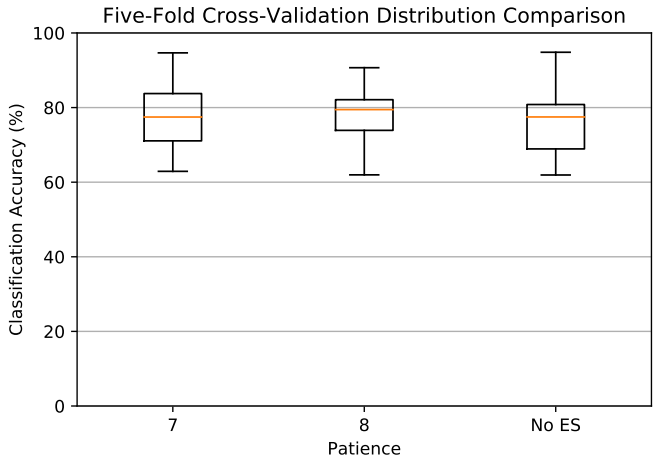
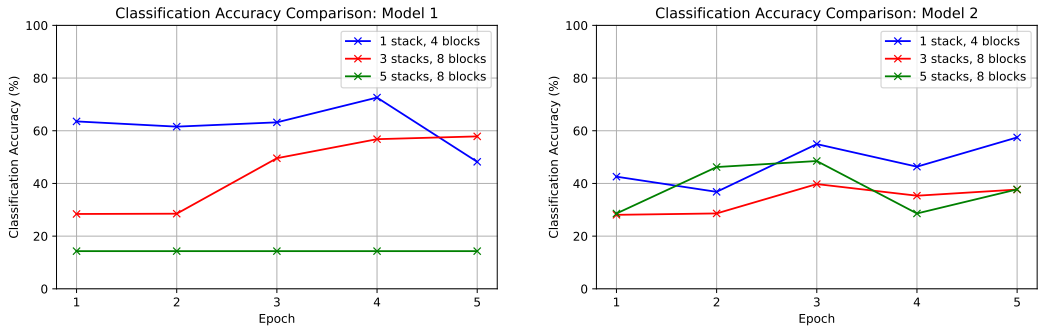


Figure 6.4: *Early stopping patience comparison for the CNN model using a five-fold cross-validation evaluation with each fold representing an entire subject. As there are only five-folds, the five-figure summary of the data corresponds directly to the recorded values. 'NO ES' is used to represent training without any early stopping. The patience value of 8 is shown to achieve the most consistent results however the value of 7 was chosen due to its overall higher average performance.*

Figure 6.5a shows that model 1 achieved the highest classification accuracy with a combination of 1 stack of 4 dilated blocks. This was reached in the fourth epoch. No other combination of either models 1 or 2 came within 10% of this. It should be noted however that by the fifth epoch the accuracy had dropped by over 20% suggesting significant overfitting issues. By comparison, figure 6.5b shows that the combination of 1 stack with 4 dilated blocks was still increasing in performance by the final epoch. Both models suggest that a smaller receptive field performs better.

Due to the potential of achieving significantly higher accuracy, model 1 was chosen for further tuning.



(a) *Model 1: Oord et al. (2016a) adaption.*

(b) *Model 2: Mader (2018) adaption.*

Figure 6.5: *Comparison of the range model architectures. Figures 6.5a and 6.5b show the validation accuracy on subject B for models 1 and 2 respectively. The models were trained on subjects A, D, E and F. Three combinations of stacks and dilated blocks per stack were evaluated for both models. In the legend 'blocks' is used to refer to the number of dilated blocks per stack. Model 1 is shown to achieve the highest classification accuracy with 1 stack and 4 dilated blocks per stack in the 4^{th} epoch.*

Figure 6.6 shows the training accuracy from the same experiment. The figure shows that the combination of 1 stack with 4 dilated blocks managed to reach greater than 90% training accuracy within the five epochs. The figure also suggests that the combination of 3 stacks with 8 dilated blocks per stack had not reached convergence within the limited number of epochs.

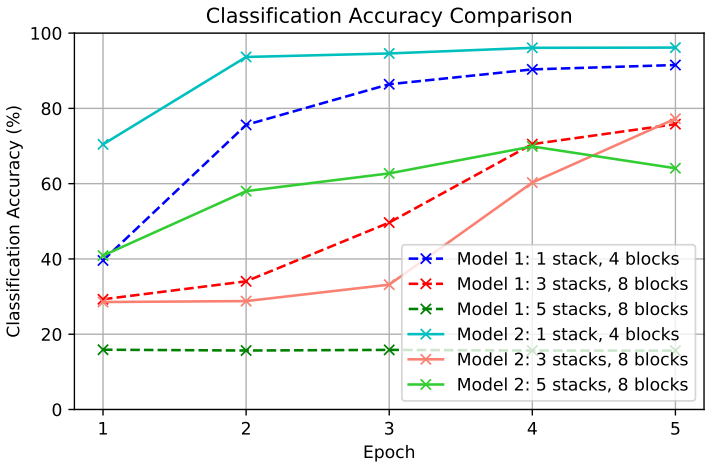


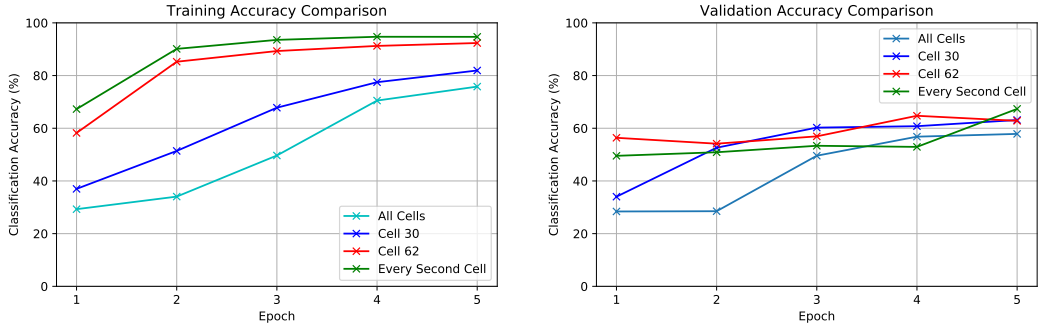
Figure 6.6: *Training accuracy for all combinations of models 1 and 2. Model 1 refers to the adaption suggested by Oord et al. (2016a) whilst model 2 refers to the model by Mader (2018). Subjects A, D, E and F were used for training. Three combinations of stacks and dilated blocks per stack were evaluated for both models. In the legend 'blocks' is used to refer to the number of dilated blocks per stack. As expected, the smaller number of layers is shown to converge much faster. The graph suggests that an insufficient number of epochs was used to allow the deeper models to fully converge.*

6.2.2 Data Type Investigation

This section discusses results from the data type investigation as described in section 4.4.2. Five different representations of the data were investigated. Both the average of all range cells and just cell 0 performed no better than random. Figure 6.7 shows the training and validation accuracy for the remaining three representations along with the original representation of all cells to provide a baseline. The model used 3 stacks with 8 dilated blocks per stack. The model was trained on subjects A, D, E and F and evaluated on subject B.

The validation results, shown in figure 6.7b, suggest that the model can classify just the same, if not better, when the number of range cells are reduced. Out of the three representations, it is shown that the format of every second range cell performed slightly better, however, as these results are only from subject B, the significance of this increase is uncertain. When considering the training graph (figure 6.7a), cell 62 and every second cell followed a very similar pattern which may suggest that cell 62 contains almost as much information as every second cell. As the training performance for cell 30 was considerably lower it was decided that taking every second cell is likely to be a more robust option than manually selecting a cell. Therefore, this data representation was selected for further tuning of the model.

These results suggest that if a task required low latency it may be possible to only use one cell for classification. As one cell only contains 3000 values as opposed to the 62×3000 values from the original representation, it is hypothesised that classification time could be significantly reduced. This reduction of data size would also prove effective if the storage space was limited.

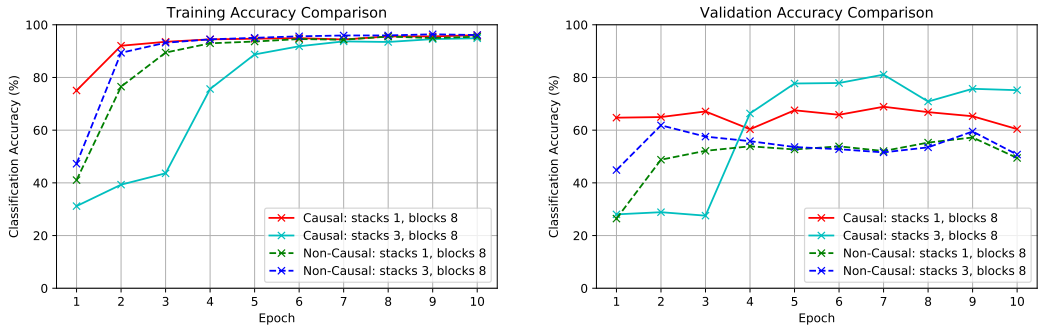


(a) Training Accuracy *(b) Validation Accuracy*

Figure 6.7: *Comparison of data format on classification accuracy. Figure 6.7a shows the accuracy during training from subjects A, D, E and F. Figure 6.7b shows the results from validating on subject B. Figure 6.7a shows that using all the data takes longer to converge whilst figure 6.7b reveals that there is very little difference regarding performance for the different data representations.*

6.2.3 Causal vs Non-Causal Comparison

Following the selection of data representation, the causal and non-causal variants of the model were compared. During the experiment, the number of epochs trained for was increased to 10. Section 4.4.3 discusses in more detail the design of this experiment. Figure 6.8a shows that both the causal and non-causal versions of the model were able to achieve close to 100% training accuracy. However, figure 6.8b reveals that when evaluated on subject B, the causal variant outperforms the non-causal for both depths of the model. Therefore, the causal model was selected for further tuning. By training for an increased number of epochs the experiment also revealed significant signs of overfitting for the causal model from epoch 7. This was the motivation behind the introduction of the regularization techniques evaluated in the following section.



(a) Training Accuracy *(b) Validation Accuracy*

Figure 6.8: *Comparison of causal and non-causal versions of the range model. Figure 6.8a shows the accuracy during training from subjects A, D, E and F. Figure 6.8b shows the results from validating on subject B. In the legend 'blocks' is used to refer to the number of dilated blocks per stack. Figure 6.8a shows that the deeper causal variant takes longer to train however figure 6.8b reveals that this corresponds to the best performing combination. Figure 6.8b also shows both causal versions outperform the non-causal versions.*

6.2.4 Regularization Investigation

To counteract the overfitting issue, regularization techniques were introduced as described in section 4.4.4. Figure 6.9 shows the results of this experiment. The L2 value of 0.1 has been excluded as it only achieved the equivalent of random performance. For the experiment, the number of epochs was increased to 20. As the figure shows, the use of L2 regularization with a value of 0.001 performed significantly better than the other techniques. The model appears to no longer overfit but instead, the accuracy increases with epochs.

Despite the reduced accuracy when compared with the model without regularization it was chosen to implement the L2 regularization with a value of 0.001 to the final model. This decision was made based on the perceived stability of the model's performance over epochs. It is expected that a more stable model is likely to generalise better.

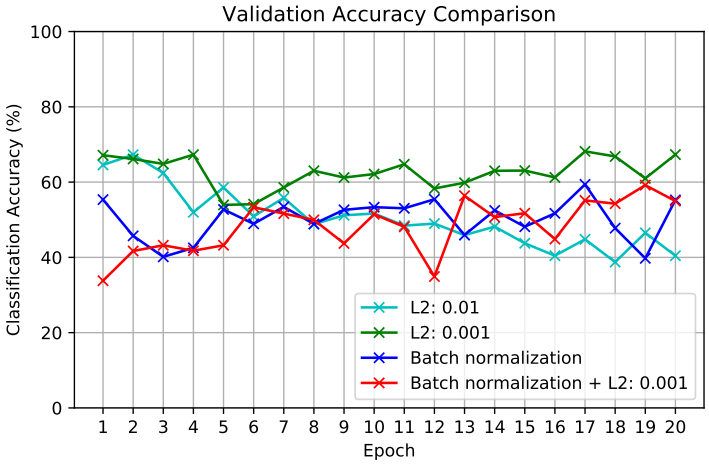


Figure 6.9: *Comparison of regularization techniques based on validation accuracy. Subjects A, D, E and F were used for training with subject B used for validation. The figure shows that the model that used L2 regularization with a value of 0.001 outperformed the other variants. This regularization selection shows no obvious signs of overfitting.*

6.2.5 Hyperparameter Search

The design of the hyperparameter search for the model is discussed in section 4.4.5. For the search the model was trained on 80% of subjects A, B, D, E and F then evaluated on the remaining 20%. For the search, the validation loss was chosen to be minimised. Figure 6.10 shows the results from the 12 parameter configuration evaluations made. The graph appears to show convergence in calls 10–12 however, due to the limited number of evaluations, this cannot be confirmed. The lowest value achieved from the search was 0.15, the parameters for this are listed in table 6.2. To evaluate if the model had improved the performance from the original parameters, the base parameters were also evaluated and achieved a loss of 0.7, suggesting that the search had been successful. However, as the search used the same data for validation of early stopping as well as computing the evaluation loss, (discussed in section 4.4.5), it is unclear how effective the search actually was.

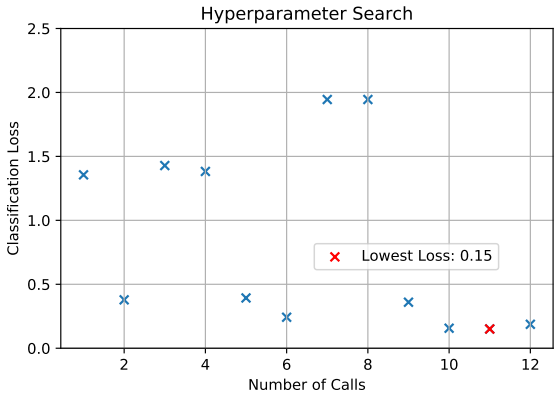


Figure 6.10: *Classification loss for each hyperparameter configuration during the search for the optimum model parameters. The lowest loss achieved is highlighted in red. The search shows signs of convergence from epoch 9 however due to the limited number of evaluations, it is not conclusive.*

Table 6.2: *Optimum hyperparameters found for the range model based on the search of the hyperparameter space.*

Parameter	Value
Kernel size of dilated conv layers	3
Number of stacks	2
Number of dilated blocks per stack	3
Early stopping patience	4
Pooling size	6
Number of filters	32
Kernel size of final conv layers	2

6.3 Final Model Evaluation and Comparison

Table 6.3 shows a summary of the results from the final evaluation of the models and is discussed in the following three sections. The design of the final evaluation is covered in section 4.3.4. In the following sections, 'Base CNN' refers to the CNN developed by Angelov et al. (2017) applied to the micro-Doppler signatures, 'Tuned CNN' refers to the same model but with the optimum parameters selected by this investigation and 'Range Model' refers to the WaveNet based classification model that is applied to the range profiles.

As mentioned in section 3.1, during this investigation it was discovered that the last section of each recording from subject F was corrupt. For the following evaluations, this section of the data was removed.

6.3.1 Evaluation on Subject C

The first evaluation compared the models' performance when trained on subjects A, B, D, E and F then evaluated on subject C. Up until this point, the data from subject C had been kept completely separate from the model development. This experiment was designed to evaluate how suitable the models would be in a system where it would not be feasible to collect data from all users of the system and so the model would have to generalise to unseen subjects. As the results

Table 6.3: *Final evaluation results summary. Values are in % and refer to classification accuracy. For columns 'Test on 20% of All Subjects' and 'Test on Each Subject', the average classification accuracy from the k-fold cross-validation has been displayed. Base CNN refers to the model and parameters developed in Angelov et al. (2017). Tuned CNN refers to the same model as Base CNN however the parameters are the optimum parameters found during this investigation. Range model refers to the WaveNet based classification model. Base CNN and Tuned CNN were applied to the micro-Doppler signatures whilst the range model operated directly on the range profiles.*

Evaluation → ↓ Model	Test on Subject C	Test on 20% of All Subjects (5-Fold Average)	Test on Each Subject (6-Fold Average)
Base CNN	**78.31**	97.11	78.48
Tuned CNN	69.93	94.96	**79.39**
Range Model	74.32	**97.94**	56.23

in column 'Test on Subject C' of table 6.3 show, the range model outperformed the tuned CNN model by over 4%, achieving an accuracy of 74.32%. However, the base CNN was found to perform almost 4% better than the range model and therefore 8% better than the tuned CNN. It is suspected that during the selection of the parameters for the model, Angelov et al. (2017) may have used the data from Subject C. This would then account for the significant performance increase over the tuned CNN. To fully understand the significance of these results, the evaluation would benefit from being repeated several times and the average classification accuracy compared. However, due to the time constraints of this project, it was not possible.

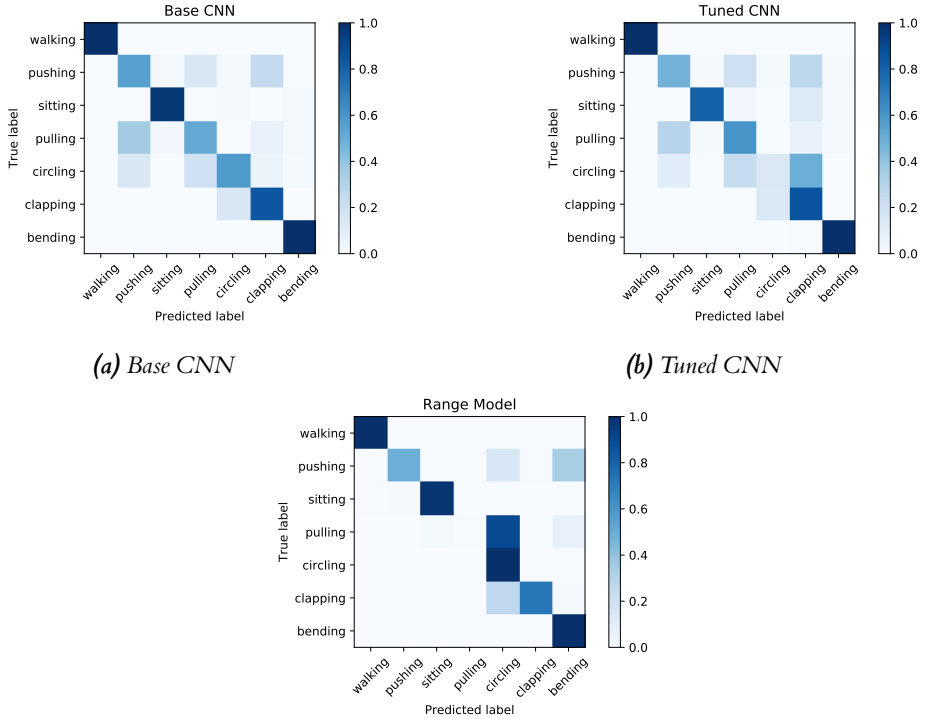
To get a more in-depth understanding of how the models were performing, normalised confusion matrices were plotted and are shown in figure 6.11. Figures 6.11a and 6.11b show that both variants of the CNN were susceptible to misclassifying the pulling action as pushing. This is not surprising given the clear similarity between the motions, especially when only provided with three seconds of data. Additionally, both CNN variants appear to struggle more when classifying the pushing, pulling and circling actions. As the circling action involved rotating the arm forward it would create alternating positive then negative Doppler peaks. This alternating positive then negative Doppler would also be caused by the pulling action as well as the pushing action which may be the source of the confusion. The lower performance of the tuned CNN model appears to be caused by the frequent misclassification of the circling action as clapping. It is unclear why these actions would be confused due to the distinct difference in the primary plane of motion of each action.

Figure 6.11c reveals that the range model has a strong bias towards the circling motion. Notably almost entirely classifying the pulling action as circling. As with the CNN variants, the misclassification of the pushing and pulling actions as circling may be due to the similar overall motion as the circling action moves the arm both towards and away from the radar, similar to the pushing and pulling actions.

6.3.2 Five-Fold Evaluation with Equal Subject Representation

This evaluation was conducted to evaluate how well the models could perform when exposed to data from all subjects during training. This is designed to mimic a small system where it would be feasible to collect data from all subjects. For this evaluation, the model was trained on 60% of the data, 20% was used for validation and the remaining 20% for evaluation. To increase the reliability of the results this was conducted using five-fold cross-validation. The data for validation and testing was extracted as consecutive blocks.

The average accuracy from the cross-validation is shown in table 6.3 under the column 'Test on



(a) Base CNN

(b) Tuned CNN

(c) Range Model

Figure 6.11: *Normalised confusion matrices produced from evaluating the models on subject C. Figures 6.11a and 6.11b show that the CNN variants struggle to classify the pushing, pulling and circling actions. Figure 6.11c shows the range model also struggles with the pushing and pulling actions but also has a strong bias towards the circling action.*

20% of All Subjects'. For this evaluation all models performed significantly better than when evaluated only on subject C, increasing by a minimum of 18%. This suggests that there may be significant variation in the way each subject performed the actions. For this evaluation, the range model was found to outperform both CNN variants. However, the base CNN was only 0.83% worse. Figure 6.12 shows the distribution over the folds. The figure reveals that the range model maintained a more consistent performance over each fold, shown by the small box. The figure also shows that the tuned model performed worse on every fold than the base CNN and the range model. It should be noted that the y-axis for the figure has been set to range from 90 to 100 and therefore the variation between the models is exaggerated. When considered on a scale of 0 to 100 there is only a very small difference between the models.

6.3.3 Six-Fold Evaluation Split by Individual Subjects

The final evaluation was used to provide a more robust measure of the models' ability to generalise between subjects than the initial evaluation on subject C. For this evaluation, the models were trained on five of the six subjects and tested on the remaining subject. This was repeated for each subject. For example, the first fold would train on subjects B-F then evaluate of subject A, the second fold would train on subjects A, and C-F then evaluate on subject B. The average of the six folds is shown in table 6.3 under the column 'Test on Each Subject'. The results show that the CNN variants performed almost identically however the range model performed significantly worse than the initial evaluation on subject C had suggested.

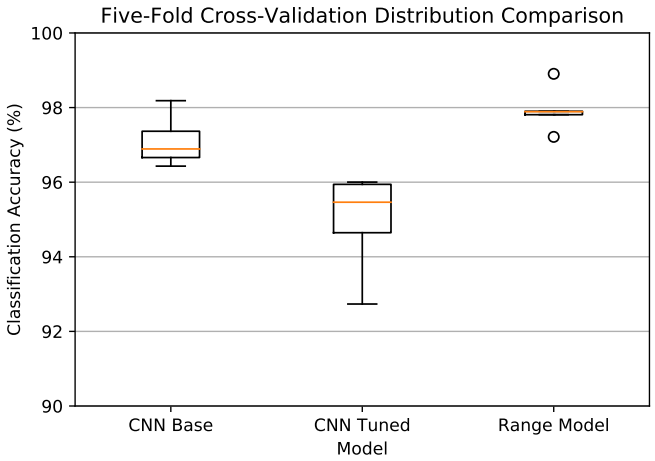


Figure 6.12: *Model comparison results from five-fold cross-validation evaluation using data from all subjects during training. As only five folds were used, the five-figure summary corresponds directly to the individual fold values. The figure shows the range model to perform the most consistent with a very small box. The tuned CNN performed worse on every fold than the other two models. It should be highlighted that the y-axis has been set between 90 and 100 to allow better visualisation. This has had the effect of exaggerating the differences between the models.*

Figure 6.13a shows the distribution across the folds for the models. As shown by the smaller boxes, both CNN variants were significantly more consistent across the subjects than the range model. The figure shows that although the average performance of the CNN variants was similar, the tuned CNN had a much better worst-case performance which may be a more desirable model characteristic depending on the system being developed. The figure also shows that the worst-case performance for the range model achieved very similar to random performance.

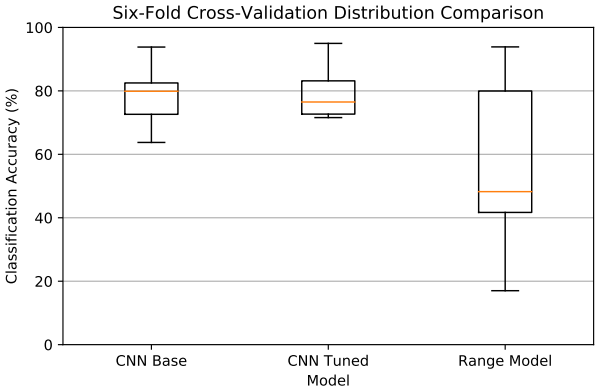
Figure 6.13b shows the performance on each individual subject and reveals that this random performance corresponds to subject F. It is unclear why the model fails to classify subject F correctly however it is hypothesised that the small number of subjects used to create the dataset does not provide sufficient representation of the variation in the movements between subjects.

The figure also shows that the range model is shown to achieve 88.91% accuracy on the fold corresponding to subject C. This is the exact same conditions as the first evaluation method however during the first evaluation the model only reached 74.32%. The only difference between the setups would be the way the training data was shuffled between epochs as the random generator was not seeded. This variability suggests that greater performance may be achievable from the model if it is provided with a larger dataset. Figure 6.13b also shows that for four of the six folds the CNN variants outperformed the range model, however, for folds C and D the range model actually outperformed the CNN models.

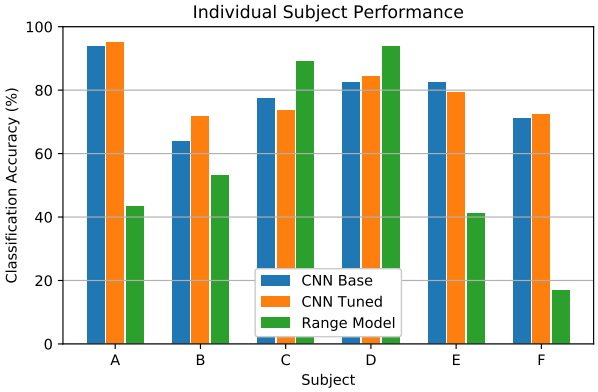
6.4 Discussion

Overall the results from this investigation suggest that the range model does not generalise as well to new subjects as the CNN model. However, it was found to generalise better to new data from subjects it had already seen. When comparing the models, it should be taken into account the difference in their nature.

Firstly, the CNN model benefits from years of previous development and research from the



(a) Boxplot summary.



(b) Individual subject performance.

Figure 6.13: *Figures 6.13a and 6.13b provide an insight into the per fold performance of the six-fold cross-validation evaluation using individual subjects per fold. Figure 6.13a shows that the range model has very little consistency between folds with a very large box. Figure 6.13b shows that the range model was able to outperform the CNN variants on subjects C and D however on the remaining four subjects it performed considerably worse.*

image classification field. In contrast, this is the first application (based on a thorough search of the literature) of a dilated convolutional based classifier to the task of radar classification from range profiles. As a result, only a limited exploration of the model's parameters was feasible.

Secondly, due to the relatively fast training time of the CNN model, it was possible to perform more robust evaluations during the design of the model such as five-fold cross-validation and training for a larger number of epochs. This was not feasible with the range model given the time constraints of the project. Therefore, the model received a very limited development process.

Thirdly, the range model is significantly more complex than the CNN and therefore is likely to require a much larger amount of data to train correctly. It is hypothesised that when applied on a larger dataset the WaveNet model would perform significantly better.

Lastly, the nature of the data that the model has to classify is vastly different. The range model is achieving this level of accuracy on noisy, minimally processed data whilst the CNN requires a highly processed input. However, it should be noted that the CNN model was classifying based on compressed, 75×75 images of the spectrograms and is likely to perform even better if provided higher resolution representations.

7 Conclusion

7.1 Summary

This project has provided a clear demonstration of the feasibility of classifying human actions directly from radar data represented as raw range profiles, instead of applying heavy processing to convert the data into micro-Doppler spectrograms. The developed model utilised dilated convolutions to handle the high temporal resolution of the data. This investigation also evaluated the current leading technique in the field to provide a fair baseline. This method classifies the radar data by processing it into micro-Doppler signatures then applying a CNN.

The results have shown the range model to be marginally better than the baseline approach when classifying new samples from a subject it had already been trained on (section 6.3.2). However, it was found to perform significantly worse than the baseline when evaluated on its ability to generalise to a new subject (sections 6.3.1 and 6.3.3). This is not surprising as only six subjects were used in the dataset. A summary of the final results can be found in table 6.3.

Although tasked with improving radar classification, this investigation has also provided a more general-purpose model that requires very limited domain knowledge to be applied. Such a model reduces the need for manually configured processing stages and therefore facilitates the use of the model in other research areas. As shown, this model has been successfully adapted from WaveNet (designed for audio generation and classification) and therefore may also be successful in other domains that currently rely on spectrograms such as seismology and SONAR.

7.2 Future Work

The main focus for future work should be to develop a larger dataset to train and evaluate the model. Due to the complexity of the network, it is hypothesised that the dataset available was not large enough to reach the models full potential.

There are many degrees of freedom in the model design. Due to the exploratory nature of this project, it was only possible to scratch the surface of the parameter space. With access to a larger dataset, a more thorough exploration of the model's architecture and hyperparameters should be considered.

During this investigation, before inputting the data to the model, the absolute value was taken. Using the complex data directly would provide the model with a much richer data source and therefore provide more information to the model. However, this would result in twice as many parameters to train and as a result, may require a larger dataset. Moran et al. (2018) found that complex weighted models would consistently outperform real-valued alternatives for the inversion of transmission effects in multimode optical fibres.

Finally, work into interpreting what features the trained model is learning may be useful. If the model were to show an increase in performance with a larger dataset, radar engineers would then be able to understand what features it is extracting and justify why it is successful. This may help with further development of the model and reduce the black box nature of the system.

7 Bibliography

R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, Y. Bengio, A. Bergeron, J. Bergstra, V. Bisson, J. B. Snyder, N. Bouchard, N. Boulanger-Lewandowski, X. Bouthillier, A. d. BrÃl'bisson, O. Breuleux, P.-L. Carrier, K. Cho, J. Chorowski, P. Christiano, T. Cooijmans, M.-A. CÃŧtÃl', M. CÃŧtÃl', A. Courville, Y. N. Dauphin, O. Delalleau, J. Demouth, G. Desjardins, S. Dieleman, L. Dinh, M. Ducoffe, V. Dumoulin, S. E. Kahou, D. Erhan, Z. Fan, O. Firat, M. Germain, X. Glorot, I. Goodfellow, M. Graham, C. Gulcehre, P. Hamel, I. Harlouchet, J.-P. Heng, B. Hidasi, S. Honari, A. Jain, S. Jean, K. Jia, M. Korobov, V. Kulkarni, A. Lamb, P. Lamblin, E. Larsen, C. Laurent, S. Lee, S. Lefrancois, S. Lemieux, N. LÃl'onard, Z. Lin, J. A. Livezey, C. Lorenz, J. Lowin, Q. Ma, P.-A. Manzagol, O. Mastropietro, R. T. McGibbon, R. Memisevic, B. v. MerriÃńnboer, V. Michalski, M. Mirza, A. Orlandi, C. Pal, R. Pascanu, M. Pezeshki, C. Raffel, D. Renshaw, M. Rocklin, A. Romero, M. Roth, P. Sadowski, J. Salvatier, F. Savard, J. SchlÃijter, J. Schulman, G. Schwartz, I. V. Serban, D. Serdyuk, S. Shabanian, Ã. Simon, S. Spieckermann, S. R. Subramanyam, J. Sygnowski, J. Tanguay, G. v. Tulder, J. Turian, S. Urban, P. Vincent, F. Visin, H. d. Vries, D. Warde-Farley, D. J. Webb, M. Willson, K. Xu, L. Xue, L. Yao, S. Zhang, and Y. Zhang. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

A. Amidi and S. Amidi. A detailed example of data generators with Keras, 2018. URL `https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly`. accessed: 10-03-2019.

A. Angelov, F. Fioranelli, and R. Murray-Smith. Human activities classification using radar signatures and deep learning. unpublished, 2017.

A. Angelov, A. Robertson, R. Murray-Smith, and F. Fioranelli. Practical classification of different moving targets using automotive radar and deep neural networks. *Sonar Navigation IET Radar*, 12(10):1082–1089, 2018. ISSN 1751-8784. doi: 10.1049/iet-rsn.2018.0103.

V. C. Chen, F. Li, S. Ho, and H. Wechsler. Micro-Doppler effect in radar: phenomenon, model, and simulation study. *IEEE Transactions on Aerospace and Electronic Systems*, 42(1):2–21, Jan. 2006. ISSN 0018-9251. doi: 10.1109/TAES.2006.1603402.

F. Chollet et al. Keras. `https://keras.io`, 2015. accessed: 23-03-2019.

Cookiecutter Data Science, 2019. URL `https://drivendata.github.io/cookiecutter-data-science/`. accessed: 17-03-2019.

Department for Transport. Contributory factors to reported road accidents 2011. Annual Report, Department for Transport, Great Britain, 2011. URL `https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/9277/rrcgb2011-04.pdf`.

E. Ferreira and N. Silva. Google DeepMind's Wavenet for detection of epileptic seizures in raw iEEG data.: projectappia/eegnet, 2016. URL `https://github.com/projectappia/eegnet`. accessed: 17-02-2019.

F. Fioranelli, M. Ritchie, and H. Griffiths. Multistatic human micro-Doppler classification of armed/unarmed personnel. *Sonar Navigation IET Radar*, 9(7):857–865, 2015. ISSN 1751-8784. doi: 10.1049/iet-rsn.2014.0360.

J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett. DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1. *NASA STI/Recon Technical Report N*, 93, Feb. 1993.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts, 2016. ISBN 978-0-262-03561-3.

Google ATAP, 2019. URL `https://atap.google.com/`. accessed: 11-02-2019.

Google Colaboratory. Welcome To Colaboratory - Colaboratory, 2019. URL `https://colab.research.google.com/notebooks/welcome.ipynb`. accessed: 17-03-2019.

A. Gupta and A. M. Rush. Dilated Convolutions for Modeling Long-Distance Genomic Dependencies. *arXiv:1710.01278 [q-bio, stat]*, Oct. 2017. URL `http://arxiv.org/abs/1710.01278`. arXiv: 1710.01278.

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, Dec. 2015. URL `http://arxiv.org/abs/1512.03385`. arXiv: 1512.03385.

T. Head, MechCoder, G. Louppe, I. Shcherbatyi, fcharras, Z. VinÃŋcius, cmmalone, C. SchrÃűder, nel215, N. Campos, T. Young, S. Cereda, T. Fan, rene-rex, K. K. Shi, J. Schwabedal, carlosdanielcsantos, Hvass-Labs, M. Pak, SoManyUsernamesTaken, F. Callaway, L. EstÃĺve, L. Besson, M. Cherti, K. Pfannschmidt, F. Linzberger, C. Cauet, A. Gut, A. Mueller, and A. Fabisch. scikit-optimize/scikit-optimize: v0.5.2, Mar. 2018. URL `https://doi.org/10.5281/zenodo.1207017`.

S. Hochreiter and J. Schmidhuber. Long Short-term Memory. *Neural computation*, 9:1735–80, 1997. doi: 10.1162/neco.1997.9.8.1735.

L. Jackson and R. Cracknell. Road accident casualties in Britain and the world. *House of Commons Library*, Apr. 2018. URL `https://researchbriefings.parliament.uk/ResearchBriefing/Summary/CBP-7615`.

E. Jones, T. Oliphant, P. Peterson, and others. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. accessed: 23-03-2019.

Kaggle. Kaggle: Your Home for Data Science, 2019. URL `https://www.kaggle.com/`. accessed: 20-02-2019.

Y. Kim and H. Ling. Human activity classification based on micro-Doppler signatures using an artificial neural network. In *2008 IEEE Antennas and Propagation Society International Symposium*, pages 1–4, July 2008. doi: 10.1109/APS.2008.4619933.

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*, 47(5):1328–1337, May 2009. ISSN 0196-2892. doi: 10.1109/TGRS.2009.2012849.

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*, 13(1):8–12, Jan. 2016a. ISSN 1545-598X, 1558-0571. doi: 10.1109/LGRS.2015.2491329. URL `http://ieeexplore.ieee.org/document/7314905/`.

Y. Kim and T. Moon. Classification of human activity on water through micro-Dopplers using deep convolutional neural networks. In K. I. Ranney and A. Doerry, editors, *Radar Sensor Technology XX*, page 982917, Baltimore, Maryland, United States, May 2016b. doi: 10.1117/12.2224196. URL http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2224196.

Y. Kim and B. Toomajian. Hand Gesture Recognition Using Micro-Doppler Signatures With Convolutional Neural Network. *IEEE Access*, 4:7125–7130, 2016. ISSN 2169-3536. doi: 10.1109/ACCESS.2016.2617282.

T. Kluyver, B. Ragan-Kelley, F. PÃl'rez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing. Jupyter Notebooks âĂŞ a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*, 25, 2012. doi: 10.1145/3065386.

J. Lei and C. Lu. Target classification based on micro-Doppler signatures. In *IEEE International Radar Conference, 2005.*, pages 179–183, May 2005. doi: 10.1109/RADAR.2005.1435815.

L. Liu, M. Popescu, M. Skubic, M. Rantz, and P. Cuddihy. An automatic in-home fall detection system using Doppler radar signatures. *JAISE*, 8:453–466, 2016. doi: 10.3233/AIS-160388.

K. Mader. QuickDraw with WaveNet Classifier | Kaggle, 2018. URL https://www.kaggle.com/kmader/quickdraw-with-wavenet-classifier. accessed: 17-02-2019.

MartÃŋn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion ManÃl', Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda ViÃl'gas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL https://www.tensorflow.org/.

MATLAB. MATLAB - MathWorks, 2019. URL https://uk.mathworks.com/products/matlab.html. accessed: 17-03-2019.

W. McKinney. Data Structures for Statistical Computing in Python. In S. v. d. Walt and J. Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

J. Michalek and J. Vanek. A Survey of Recent DNN Architectures on the TIMIT Phone Recognition Task. *arXiv:1806.07974 [cs]*, June 2018. URL http://arxiv.org/abs/1806.07974. arXiv: 1806.07974.

O. Moran, P. Caramazza, D. Faccio, and R. Murray-Smith. Deep, complex, invertible networks for inversion of transmission effects in multimode optical fibres. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3280–3291. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7589-deep-complex-invertible-networks-for-inversion-of-transmission-effects-in-mu
pdf.

A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *arXiv:1609.03499 [cs]*, Sept. 2016a. URL http://arxiv.org/abs/1609.03499. arXiv: 1609.03499.

A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. *arXiv:1606.05328 [cs]*, June 2016b. URL http://arxiv.org/abs/1606.05328. arXiv: 1606.05328.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS-W*, 2017.

Project Soli, 2019. URL https://atap.google.com/soli/. accessed: 23-01-2019.

M. Pyeon. Keras Implementation of Deepmind's WaveNet for Supervised Learning Tasks: mjpyeon/wavenet-classifier, 2018. URL https://github.com/mjpyeon/wavenet-classifier. accessed: 23-03-2019.

Python. Welcome to Python.org, 2019. URL https://www.python.org/. accessed: 17-03-2019.

Quick, Draw! Quick, Draw! Doodle Recognition Challenge, 2018. URL https://kaggle.com/c/quickdraw-doodle-recognition. accessed: 17-02-2019.

K. Ramasubramanian. Using a complex-baseband architecture in FMCW radar systems. *Texas Instruments*, page 10, 2017. URL http://www.ti.com/lit/wp/spyy007/spyy007.pdf.

D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin. Backpropagation. chapter Backpropagation: The Basic Theory, pages 1–34. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995. ISBN 0-8058-1259-8. URL http://dl.acm.org/citation.cfm?id=201784.201785.

E. Sebastian. Radar target classification using Support Vector Machines and Mel Frequency Cepstral Coefficients. Master's thesis, KTH Royal Institute of Technology, Stockholm, 2017. URL http://www.diva-portal.se/smash/get/diva2:1143293/FULLTEXT01.pdf.

Seizure Prediction. Melbourne University AES/MathWorks/NIH Seizure Prediction, 2016. URL https://kaggle.com/c/melbourne-university-seizure-prediction. accessed: 17-02-2019.

K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, Sept. 2014. URL http://arxiv.org/abs/1409.1556. arXiv: 1409.1556.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf.

E. Strubell, P. Verga, D. Belanger, and A. McCallum. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions. *arXiv:1702.02098 [cs]*, Feb. 2017. URL http://arxiv.org/abs/1702.02098. arXiv: 1702.02098.

The Tesla Team. Upgrading Autopilot: Seeing the World in Radar, 2016. URL https://www.tesla.com/en_GB/blog/upgrading-autopilot-seeing-world-radar. accessed: 23-01-2019.

United Nations, Department of Economic and Social Affairs, Population Division. World Population Ageing 2017. Ageing Report ST/ESA/SER.A/408, United Nations, Department of Economic and Social Affairs, Population Division, New York, 2017. URL http://www.un.org/en/development/desa/population/publications/pdf/ageing/WPA2017_Report.pdf.

B. Veeling. Wavenet Implementation in Keras, 2018. URL https://github.com/basveeling/wavenet. accessed: 10-03-2019.

S. v. d. Walt, S. C. Colbert, and G. Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*, 13(2):22–30, 2011. doi: 10.1109/MCSE.2011.37. URL https://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37.

C. Wolff. Radar Basics - In-phase & Quadrature Procedure, 2018. URL http://www.radartutorial.eu/10.processing/sp06.en.html. accessed: 09-02-2019.

H.-S. Yeo, R. Minami, K. Rodriguez, G. Shaker, and A. Quigley. Exploring Tangible Interactions with Radar Sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(4):1–25, Dec. 2018. ISSN 24749567. doi: 10.1145/3287078. URL http://dl.acm.org/citation.cfm?doid=3301777.3287078.

F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. *arXiv:1511.07122 [cs]*, Nov. 2015. URL http://arxiv.org/abs/1511.07122. arXiv: 1511.07122.