Dynamic Hand Gesture Recognition

Using Ultrasonic Sonar Sensors and Deep Learning

Author:

Chiao-Shing Lin

lnxchi026@myuct.ac.za

Supervisors:

Dr Yunus Abdul Gaffar

&

Mr Jarryd Son

Department of Electrical Engineering

University of Cape Town

South Africa

MSc(Eng) thesis submitted in fulfilment of the requirements for the degree of Master of Science in the Department of Electrical Engineering at the University of Cape Town

January 2021

Keywords: Sonar, hand gesture recognition; LSTM; CNN; human-computer interaction

Declaration

I, Chiao-Shing Lin, hereby:

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.
5. I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Signed by candidate

Chiao-Shing Lin

Department of Electrical Engineering

University of Cape Town

Saturday 27th March, 2021

Abstract

The space of hand gesture recognition using radar and sonar is dominated mostly by radar applications. In addition, the machine learning algorithms used by these systems are typically based on convolutional neural networks with some applications exploring the use of long short term memory networks. The goal of this study was to build and design a Sonar system that can classify hand gestures using a machine learning approach. Secondly, the study aims to compare convolutional neural networks to long short term memory networks as a means to classify hand gestures using sonar.

A Doppler Sonar system was designed and built to be able to sense hand gestures. The Sonar system is a multi-static system containing one transmitter and three receivers. The sonar system can measure the Doppler frequency shifts caused by dynamic hand gestures. Since the system uses three receivers, three different Doppler frequency channels are measured. Three additional differential frequency channels are formed by computing the differences between the frequency of each of the receivers. These six channels are used as inputs to the deep learning models.

Two different deep learning algorithms were used to classify the hand gestures; a Doppler bi-LSTM network [1] and a CNN [2]. Six basic hand gestures, two in each x- y- and z-axis, and two rotational hand gestures are recorded using both left and right hand at different distances. The gestures were also recorded using both left and right hands. Ten-Fold cross-validation is used to evaluate the networks' performance and classification accuracy. The LSTM was able to classify the six basic gestures with an accuracy of at least 96% but with the addition of the two rotational gestures, the accuracy drops to 47%. This result is acceptable since the basic gestures are more commonly used gestures than rotational gestures. The CNN was able to classify all the gestures with an accuracy of at least 98%. Additionally, The LSTM network is also able to classify separate left and right-hand gestures with an accuracy of 80% and The CNN with an accuracy of 83%.

The study shows that CNN is the most widely used algorithm for hand gesture recognition as it can consistently classify gestures with various degrees of complexity. The study also shows that the LSTM network can also classify hand gestures with a high degree of accuracy. More experimentation, however, needs to be done in order to increase the complexity of recognisable gestures.

Acknowledgements

Firstly, I would like to thank my supervisors, Dr Yunus Abdul Gaffar and Mr Jarryd Son. Without the constant guidance and advice, I would not have been able to complete this dissertation. The regular emails and meeting helped guide me on the right path to the completion of this dissertation.

Secondly, I would like to thank the Donors of The Max Price & Deborah Posel Scholarship for funding my studies. Without this aid, I would not have been able to complete my final year of research. The funding aided in relieving the financial burden caused by the COVID-19 pandemic allowing me to focus on research.

I would also like to thank my fellow postgraduate research fellows, Tauriq Latief, Luka Wolpe, and other members of the University of Cape Town Radar Remote Sensing Group. It was a pleasure working alongside all of you, bouncing ideas off one another, sharing advice and motivating each other to work hard. You all made this period of research an enjoyable one. I would also like to extend my thanks to the electrical engineering staff that I worked alongside.

Next, I would like to thank my friends Emma Gibson, Anna Mulder, Joshua van Zyl, Jamie Jacobson and friends from CoM. All your support allowed me to achieve greater heights. Thank you for being an emotional pillar during this stressful time.

Lastly, I would like to thank my family. I would not be the person I am in the position that I am in without you all. The constant belief in me and encouragement propelled me to where I am today.

Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background to the study

With the rate that technology is evolving, it has become apparent that traditional human-computer interactions (HCI) have become inadequate and unintuitive. Technologies such as eye-tracking, voice recognition and other touchless forms of HCI are becoming more popular and replacing the more traditional forms of HCI. Gesture recognition is one of the areas of touchless HCI research that has become increasingly popular since the age of virtual reality (VR) and augmented reality (AR) has become more mainstream[3]. The COVID-19 pandemic has also shown the importance of touchless technologies.[4]

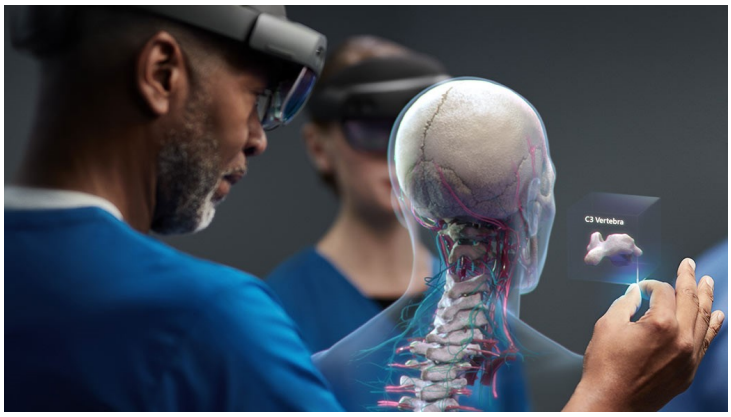


Figure 1.1: Microsoft Hololens AR headset [5]

Radar systems[6], sonar systems[7], camera systems and infrared systems[8] are just some of the systems that have been investigated in gesture recognition research. Commercially available such as the Leap Motion Controller by Ultraleap, the Azure Kinect by Microsoft Azure and Soli by Google are just some of the solutions for touchless HCI. These devices are popular in research that involves gesture recognition.

Research in the field of gesture recognition commonly tries to solve the problem with deep neural networks (DNN) such as convolutional neural networks (CNN)[9]. Recently, the use of long short term memory (LSTM)[10, 11] networks in gesture recognition research have shown comparable results and in some scenarios, better than CNNs.[10]

1.2 Problems to be Investigated

In the field of touchless hand gesture recognition, most of the work has been focusing on vision and infrared depth-based gesture recognition. Vision and infrared sensors are very susceptible to environmental noise such as visible and infrared light pollution. Radar and sonar sensing are , however, immune to irregular lighting conditions unlike vision and infrared-based sensors. [12, 13]

The work that is done in both radar and sonar hand gesture recognition systems typically use continuous wave or frequency modulated continuous wave systems to extract the Doppler velocity associated with a hand gesture or movement. Radar and sonar systems most often make use of convolutional neural networks to solve the problem of gesture recognition and classification.[14, 2, 15, 16] While some work has been done using long short term memory networks to classify gestures using radar Doppler data with very respectable results, almost no work has been done using the same technique on sonar systems. These radar systems try to classify the gestures by training long short term memory networks with spectrogram data[10, 1] and in some cases, layering the long short term memory network on top of a convolutional neural network[13, 11].

There is currently a gap in the literature involving classifying hand gestures captured by sonar sensors using long short term memory networks. Some of the benefits for using sonar over radar include:

- Sonar operates at much lower frequencies (around 40 kHz) compared to radar (GHz) allowing direct sampling of the signals which reduce the overall circuit complexity.
- Sonar systems can operate using regular analogue electronics whereas radar systems require special RF components.
- Radar is more effective over extremely long ranges but for short-range low power applications, sonar offers very similar performance.

1.3 Objectives of the Study

The following objectives for the study are formed based on the problems and gaps found in literature with regards to touchless hand gesture recognition and the potential of using Sonar as a solution:

- Formulate the requirements for a sonar-based gesture recognition system.
- Design a system that uses sonar to sense dynamic hand gestures in three-dimensional space.
- Construct and test the sonar system to ensure that it meets the requirements
- Design a dynamic hand gesture classification system using machine learning and deep learning techniques
- Create and test the gesture classification system to ensure that it meets the requirements

1.4 Scope and Limitations

There was no available hardware to use to perform the experiments needed for this study. Additionally, This study was done during the COVID-19 pandemic. As such, the availability of components and equipment was limited. The data used for training and validating the networks only contain samples collected from one person. A MATLAB licence was made available by the university and as such it will be used in this study because of its ease of use.

The scope of the study described by the objectives are as follows:

- Designing and implementing analogue circuitry for the multiple receiver quasi-monostatic Doppler sonar system.
- Embedded C/C++ code to generate the signal to be transmitted, sample the Doppler echoes and interface with the PC.
- Writing MATLAB code to extract the Doppler signals from the raw ultrasonic recordings.
- Using the MATLAB Deep Learning Toolbox to implement the LSTM network and the CNN to compare the results.
- Using the chosen network to classify gestures

1.5 Overview of Dissertation

This section discusses how the dissertation is broken down. Following the introduction, is the literature review which will investigate previously and currently used technology and research in the field of gesture recognition and machine learning that applies to this study. The literature is followed by a requirements analysis which will form the basis of the goals of this dissertation. An overview of the design of the system is followed by the design and testing of the device to acquire data for the gesture recognition portion of the system. The dissertation is finalised with the results and discussion of the experiments, the conclusions and recommendations for any future work.

1.5.1 Literature Review

This study begins with a literature review detailing work that has been done in the field of hand gesture recognition. This includes the devices that have been used to detect hand gestures and the algorithms used to recognise the gestures. The literature also discusses the theory behind the radar and sonar methods that are relevant to this study as well as the relevant LSTM and CNN theory.

1.5.2 Requirements

The requirements chapter contains the user requirements document which details all the requirements for this study as well as the acceptance test procedures (ATP), which is later used as experiments for the system to gauge whether or not the study was successful.

1.5.3 System Design Overview

The system design overview chapter begins by describing the operational flow of the system as a whole. The system is then broken down into subsystems with a description of the building blocks that make up the subsystems. Each of the building blocks is expanded further upon in-depth in the following design chapter.

1.5.4 System Design and Testing

The system design and testing chapter begin with an overview of the system which is followed by microcontroller and sensor selection. Next, the design and implementation and testing of the data acquisition of the system are discussed in detail followed by the design of the data logging of the system. The chapter also describes the signal processing algorithms that are used.

1.5.5 Experimental Setup and Data

The experimental setup that is used to collect the data that will be used for training the deep learning networks is described in this chapter along with the details of each data set that will be used to test the performance of the deep learning networks that are discussed in the next chapter.

1.5.6 Deep Learning Design

This chapter discusses the specific network structured that was used for each of the deep learning networks. The selection and optimisation of certain hyperparameters are also discussed in this chapter

1.5.7 Results and Discussion

This chapter shows the classification accuracy from training the deep learning networks with each of the data sets and discussed the performance of the deep learning network on the data sets recorded for this study. The results are compared to the conditions set in the ATP.

1.5.8 Conclusion, Recommendations

The final chapter concludes on the success of the study based on the ATP evaluation along with what the study was able to achieve aside from the ATP's. This chapter also discusses how the study can be improved and the work that can be done to further the research.

Chapter 2: Literature Review

The review of literature begins with a broad discussion on gesture recognition followed by the hand gesture recognition sensing hardware that has previously been used or is currently in use. Following is a discussion on the Radar and Sonar theory from the literature that will be used in this dissertation. Convolutional neural networks and long short term memory networks are then discussed in terms of how they work and previous work involving these two algorithms in the field of gesture recognition. The section is finalised with a critical review of the literature explored in this literature review.

2.1 Gesture recognition

Gestures are one of the oldest and most integral forms of communication. Gestures are created by positioning one's body or moving one's body to convey meaningful information to another or to interact with the environment. Gestures are means of compressing information where it is then decoded and reconstructed by the recipient. Research in the field of human-computer interactions (HCI), and more specifically gesture recognition, allows for a more immersive and intuitive experience interacting with computers. With the advancements of technology and computation, traditional forms of HCI such as mouse and keyboards become insufficient and unintuitive especially in the areas of virtual and augmented realities.[17, 18, 8]

Gestures can be either static or dynamic or a combination of each, such as sign language. Gestures can be broadly categorised as follows[17]:

- Hand and arm gestures
- Head and face gestures
- Body gestures

Static gestures are usually classified by its spatial, symbolic and emotional information in an instant of time while dynamic gestures are classified mostly by its spatial information over a specific period.[17, 18, 8]

The following sections will focus specifically on literature covering dynamic hand gestures. The sections are organised by first discussing currently used hand gesture sensing devices used in gesture recognition followed by relevant radar and sonar sensing theory. The final section discusses the recognition aspect of hand gesture recognition including the algorithms that are typically used in radar applications.

2.2 Hand Gesture Sensing

For the computer to be able to classify hand gestures, there needs to be a device or devices that will be able to feed information about the physical characteristics of the hand gesture to the computer. This part of the system architecture is known as detection. Several parameters of the devices will affect the quality of detection. These include:

- Accuracy:
  - Resolution: having a higher resolution will allow the device to sample data at smaller discrete points giving a more accurate representation of the received signal.
  - Drift: Some devices are subject to drift. The error between the measurement and the real data will increase due to environmental effects such as temperature.
  - Noise figure: All devices have sources of noise in its internal circuity. Noise will cause the measurement to deviate from the real data. Noise is usually specified statistically.
- Sampling speed: Having a high sampling speed allows the device to send data more often, which will give more meaning to a gesture in a given period.

This section will cover the following gesture sensing devices that are currently being used for gesture recognition systems along with how the systems operate: data glove, vision-based sensors, time of flight camera, Azure Kinect, Leap Motion Controller, radar sensors and sonar sensors.

2.2.1 Data Glove

Data gloves are gloves that a user wears with embedded sensors that track the position and angle of fingers and the hand. In most cases, the sensors that make up a data glove consists of an array of flex sensors and an inertial measurement unit (IMU).[19, 20]

Flex sensors operate by changing its resistance as it stretches. By placing these sensors above each finger, they can sense how much each finger is bending.[19, 21, 20] Another method of detecting finger position is using capacitive touch sensors mounted to the tip and in between the fingers.[22] Flex sensors are more commonly used instead of the capacitive touch sensors.

Inertial measurement units are sensors that consist of a combination of multi-axis sensors that include gyroscopes, accelerometers, magnetometers and pressure sensors. The IMU can measure the movement and angular rate of the hand. The IMU allows the glove to track the trajectory and the orientation of the hand.[19, 20]

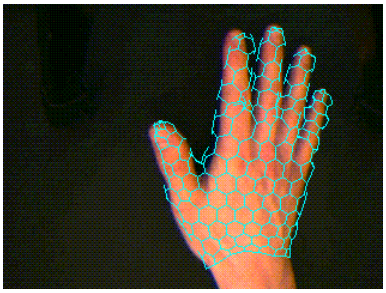
The combination of the flex sensors and the IMU allows the glove to form a model of the hand's movements. Hand gestures are created by recording this data. The gloves often offer a very accurate representation of the orientation and the movement of the hand. The issue with the gloves, however, is that they are often bulky and uncomfortable to wear. Figure 2.1 is an example of what a typical data glove would look like.

Figure 2.1: Data glove [20]

The work done by Pławiak *et al.* [20] uses a data glove that makes use of a support vector machine to classify hand gestures. The system was able to achieve 98.24% accuracy on 22 hand gestures.

2.2.2 Vision based Sensing

Vision-based hand gesture recognition is the most commonly used forms of gesture recognition. This method uses one or more cameras to capture a video feed of the gesture. Vision-based sensors offer a much larger working area to perform gestures when compared to touchscreens. [8]



(a) 3D hand tracker [23]

(b) Colour glove based hand gesture recognition[24]

Figure 2.2: Vision based hand gesture recognition methods

Vision-based gesture recognition is often computationally expensive as it relies heavily on image processing and computer vision.[17, 8] These systems work by detecting and isolating the hand from the rest of the image feed. This data is used to generate three-dimensional models of the hand.[23, 24, 25] Figure 2.2a is an example of a 3d hand tracker. Gloves with coloured markers, as seen in figure 2.2b, are occasionally used to reduce the computational complexity of the system.[8, 24] Some challenges that vision-based gesture recognition faces are that complicated backgrounds, illumination changes and occlusion make the tracking of the hand very difficult.[23, 25]

The work done by Singha *et al.*[25] explores using artificial neural networks (ANN), support vector machines (SVM), k nearest neighbours (kNN), and a fusion of the methods to classify 40 gestures captured on cameras. Table 2.1 shows the success rates of each of the types of classifiers from the study.

Classifier	Success rate (%)	Computational Time (s)
ANN	90.58	0.560
SVM	88.31	0.420
kNN	87.82	0.496
Classifier Fusion	92.23	1.570

Table 2.1: Comparison of success rates from different classifiers [25]

2.2.3 Time-of-Flight (ToF) Cameras

Time-of-flight cameras gather three-dimensional information of the environment by making use of the infrared light spectrum. The system works by illuminating a scene with a modulated infrared light source and then measuring the phase shift of the reflected light that is received by

the image sensor. Based on the phase shift of the received signal, the system can calculate the distance of objects in the scene because the speed of light is constant in a uniform propagation medium.[26]

Figure 2.3: DOMI ToF Sensor[27]

The hand is segmented from the depth information and used to generate a model of the hand. The computational complexity is a lot less for ToF systems when compared to vision-based systems because the data is already in three dimensions.[28, 29]. Figure 2.4 shows an example of hand segmentation.

Figure 2.4: Hand extraction and segmentation [28]

One of the methods used to classify gestures from ToF cameras extracts features from the depth data finds the geodesic centre of the hand and then plots the distance between that point and the border of the silhouette, as seen in figure 2.5. Using a support vector machine, this method can classify 13 gestures with 90% accuracy.[28]

Another paper investigates projecting the image of the extracted hand onto the x- and y-axis. This transformation is shown in figure 2.6. This considerably increases the speed of classification. The system can classify 12 gestures with 94.61% accuracy.[29]

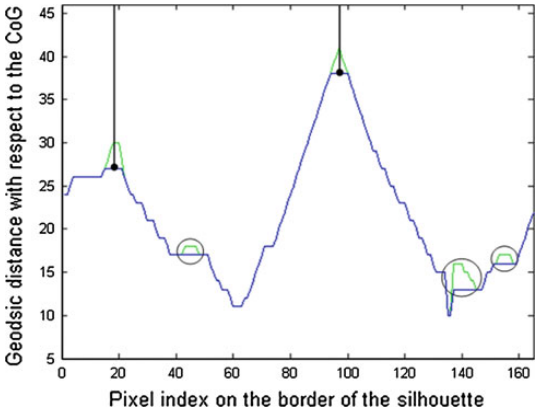


Figure 2.5: Distance from Geodesic Centre to silhouette border[28]

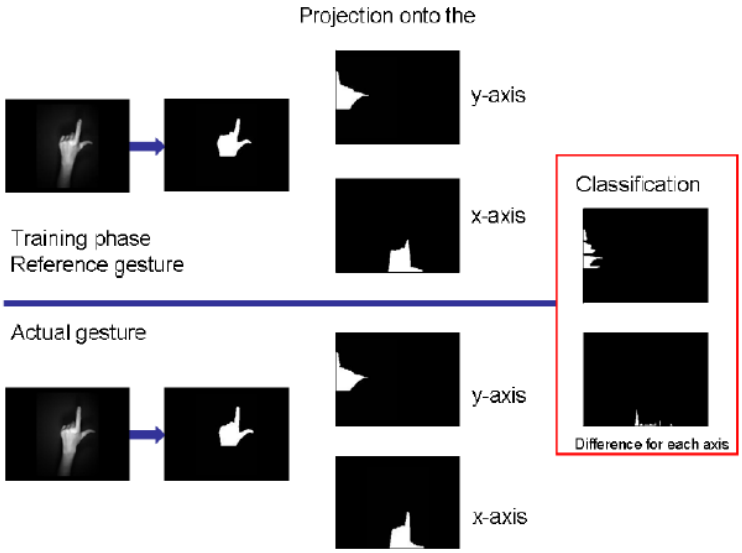


Figure 2.6: x- y-axis projection[29]

2.2.4 Azure Kinect

The Kinect is an integrated system that uses a ToF depth sensor. It has an RGB video camera, a 7-microphone array and an IMU along with the ToF sensor. The camera and the ToF sensor are aligned in a way that coloured pixel has a depth value associated with it.[30]

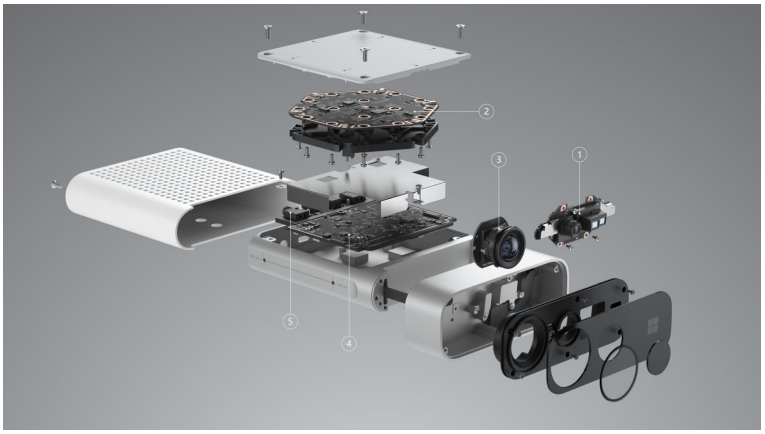


Figure 2.7: Exploded view of the Azure Kinect

Most of the research done on gesture recognition with the Kinect is similar to that done with the ToF camera sensors in terms of hand segmentation and feature extraction.[31, 32, 33]

A study that was done by Ren *et al.*[31] use a metric called the Finger-Earth Mover's Distance to measure the dissimilarity between hand gestures. Using that metric, the system can classify 10 gestures with 93.2% accuracy.

The work done by Li [32] uses a three-step classifier to classify hand gestures. First, the gesture is classified by the number of extended fingers. Secondly, the gestures are classified by the name of the fingers that are extended. Lastly, the gestures are classified by the direction vectors of the extended fingers. This system can classify 9 gestures with 89% accuracy.

2.2.5 Leap Motion Controller (LMC)

The Leap Motion Controller is an implementation of the ToF sensor that uses two image sensors, as seen in figure 2.8a. The use of two image sensors allows the controller to be more accurate and robust than a regular ToF camera [34]. Unlike the other systems that output data frames that contain the depth of each point, the LMC outputs the angle and position of the palm, finger and joints, which is shown in figure 2.8b. That implies is that there is no processing required to segment, extract and model the hand. Feature selection becomes as simple as choosing which

data points to use.[35, 36]

(a) Exploded view of the LMC [37]

(b) Illustration of the output data frames of the LMC[38]

Figure 2.8: Leap Motion Controller

A study by Lu *et al.* [35] use a Hidden Conditional Neural Field to classify dynamic hand gestures using the LMC. This system achieved recognition accuracy of 89.5% using the LeapMotion-Gesture3D data set and 95% with the Handicraft-Gesture data set.

2.2.6 Radar

Radar operates using the same principle as with ToF cameras. ToF cameras use infrared light to illuminate the scene while radar uses radio waves, both of which belong to the electromagnetic (EM) spectrum. Radar is at a much lower frequency (3 MHz to 300 GHz) when compared to infrared light (300 GHz to 1 THz) [39] and thus has a much longer wavelength. Due to radar having a longer wavelength, the transmitters and receivers need to be a lot larger and spaced further apart making solutions such as image sensors not viable. Thus radar uses different detection techniques compared to ToF cameras.

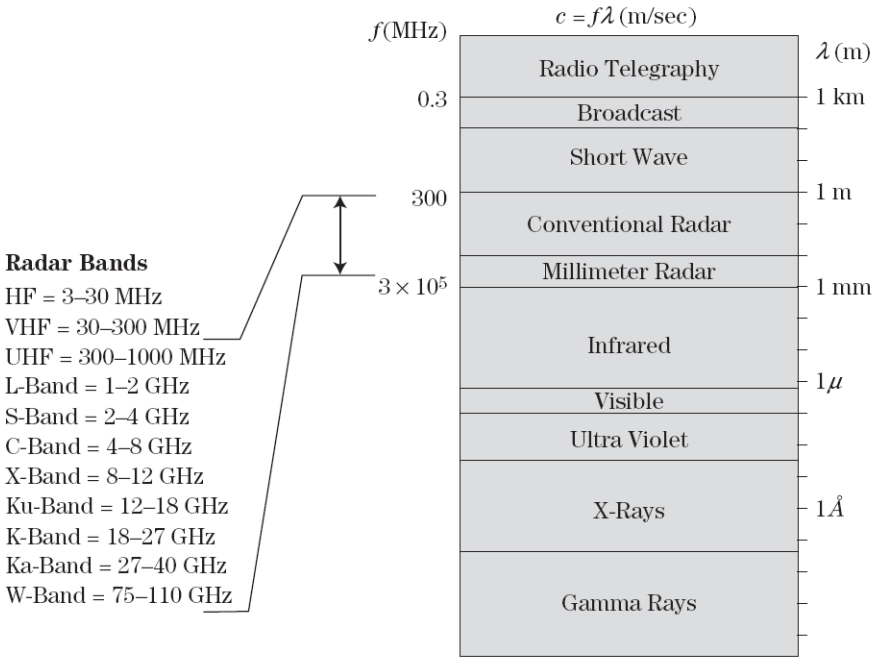


Figure 2.9: Electromagnetic Spectrum [39]

There are three main types of radar have been used to research hand gesture recognition.

- Pulsed radar [11, 6, 40]
- Frequency modulated continuous wave (FMCW) radar [9, 14, 41, 13, 42, 43, 44, 1, 45]
- Continuous-wave (CW) radar [2, 9, 46, 15]

Pulsed radars work by transmitting a pulse and using the delay of the echo to calculate the distance of a target. The pulse is commonly linearly modulated. A finer range resolution can be achieved through the process of modulating the pulse and using a method called matched filtering. The received signal is transformed into a Range Doppler map using radar signal processing techniques. This plot shows the position and velocity of a target at an instant in time. The received signal can also generate individual range and velocity plots.[39]

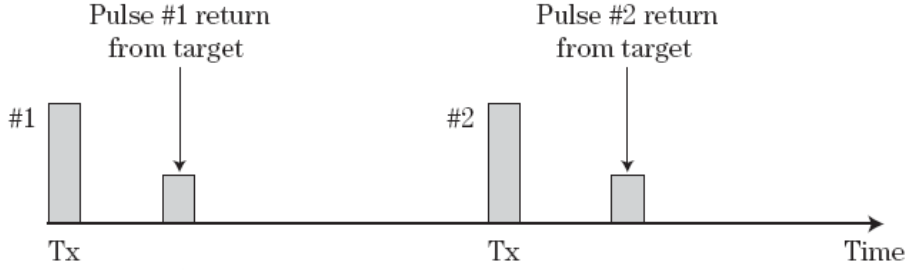


Figure 2.10: Pulsed radar Illustration

As the name suggests, FMCW radar uses a modulated frequency to resolve the range and velocity of objects in the scene. The most common modulation pattern used is the sawtooth linear FMCW waveform. The difference between the frequency of the return echo and the transmitted signal is known as the beat frequency.[47] The range of the object can be calculated using the beat frequency and thus the velocity. As with the pulsed radar, the received signal can generate Range Doppler maps, range plots and velocity plots.

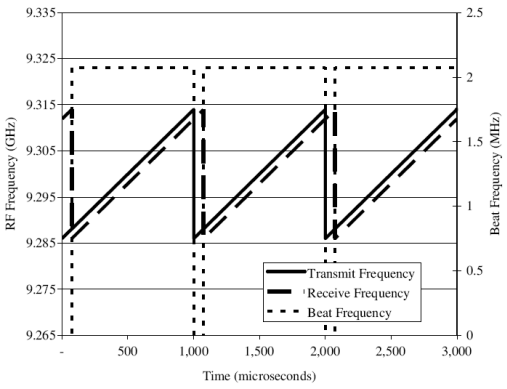


Figure 2.11: Transmit and receive frequency with beat frequency [47]

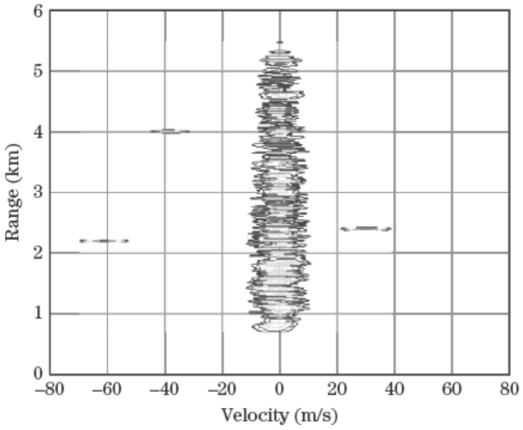
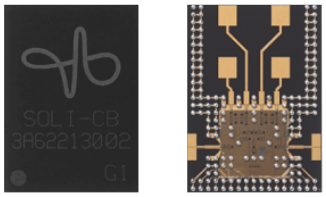


Figure 2.12: Range Doppler Map [39]

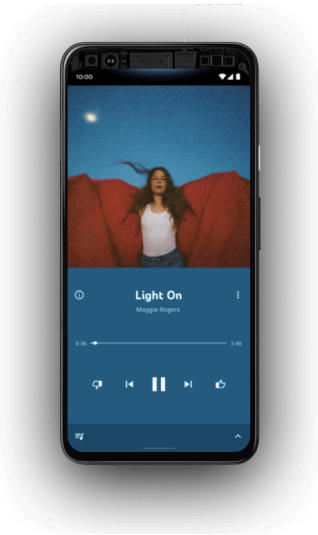
CW radar systems are a lot simpler than FMCW radar systems. The radar transmits a continuous unmodulated waveform. The velocity of the target of interest can be measured based on the change in frequency of the echo. This change in frequency is known as the Doppler effect. Unlike the previously mentioned radars, CW radars can only generate velocity plots. It is, however, a simpler system and less computationally intensive.[39]

Small PCB patch antennas are the most commonly used type of antennas for gesture recognition as the system does not have high power requirements. Both off-the-shelf radar systems and custom made radars are used to research hand gesture recognition. One of the more recent notable radars is the Soli radar created by Google. The whole system is integrated into a single surface mount chip, as seen in figure 2.13a, allowing it to be embedded into devices such as the recently released Google Pixel 4 smartphone, shown in figure 2.13b.[43, 44]. The technical specifications of the Soli radar are shown in table 2.2.

Studies that were done by Wang *et al.* [11] use frames of range-Doppler maps to feed into a machine learning network consisting of a convolutional network coupled with a long short term memory network. The system is able to classify 11 gestures gathered by 10 users with a recognition rate of 87%.



(a) Soli radar Chip [43]



(b) Google pixel phone with Soli [43]

Technical specifications	
Type	FMCW/DSSS
Number of receivers	4
Number of transmitters	2
Carrier	60 GHz
Bandwidth	7 GHz
Range resolution	2 cm
Range	0.05-15 m
Beam width	180°

Table 2.2: Technical specifications of Soli by Google [48]

2.2.7 Sonar

Similar to radar and Infrared cameras, sonar operates by transmitting a wave into the environment and finding the range and velocity of objects based on the return echo with the difference being that sonar transmits sound waves instead of EM waves.

Ultrasonic sonar is preferred over audible sonar because it is inaudible and that there is less background noise at that frequency. Ultrasonic transducers typically have an operating frequency of 40 kHz with a very narrow band of operation (around 2-4 kHz).[49] Wideband microphones exist that have an operational frequency range from 100 Hz to 80kHz[50] which are every useful for when a wideband receiver is needed.

The two most commonly used methods for gesture detection are CW sonar and Pulsed sonar[7]. These two methods operate in the same way as their radar counterparts. CW sonar is sometimes also known as Doppler sonar because it mainly operates using the Doppler effect. FMCW is not used very often in sonar because the narrow bandwidth makes it difficult to modulate the frequency.

Pulsed sonar can very accurately estimate the range and velocity of an object but is limited to very basic movements. Some systems use multiple receivers to estimate the angle of arrival of objects to map the object in three dimensions by calculating the phase difference between adjacent receivers.[51]

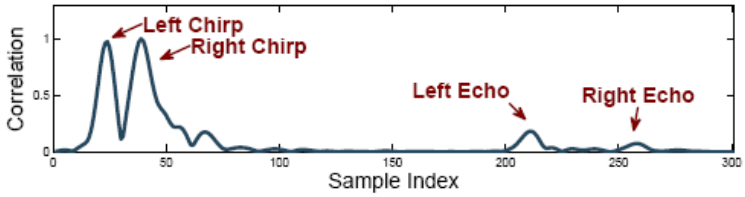


Figure 2.14: Pulsed sonar with two transmitters [51]

Doppler sonars can give very accurate measurements of the velocity of an object as a function of time. Micro-Doppler signatures are smaller Doppler reflections from objects moving on the main object relative to the main object such as fingers on a hand or propellers on a vehicle[52] as seen in figure 2.15. Micro-Doppler signatures are what makes Doppler sonars so popular for gesture recognition applications.

Multiple receivers pointing in different planes are used to map the velocity of the object in three dimensions. A specific study using this principle uses three receivers, one in each of the x-y-z

axis, classifies 8 gestures with an 88.42% accuracy. This study uses a Gaussian Mixture Model to classify the gestures[49]

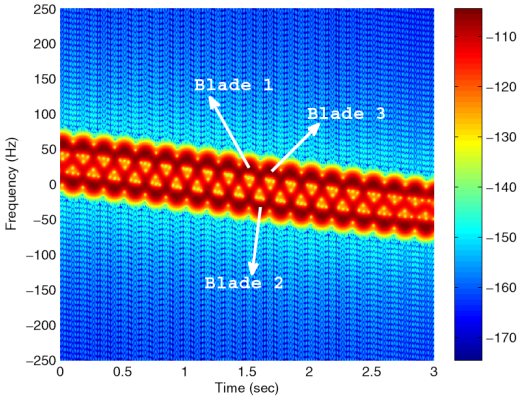


Figure 2.15: Micro-Doppler signatures of a LUAV [52]

A study by Ling *et al.* [16] uses multiple receivers to calculate the differential channel impulse response (dCIR) of the system. The impulse response corresponds to any changes between the transmitter and the receiver. Calculating the dCIR of the system means that any changes caused by static background targets are ignored by the system. The system uses the dCIR as inputs into the CNN yielding an accuracy of 97% with 12 gestures.

2.3 Radar and Sonar Theory

Radar and sonar detection share very similar characteristics with one another. Both use a transmitter that propagates a signal through space and a receiver that receives the echoes of the transmitted signal that have been reflected by objects within the propagation medium. The main difference between radar and sonar being that radar uses electromagnetic waves and sonar using sound waves. This section only covers theory from literature about unmodulated continuous wave radar as the theory is the same for sonar applications. Frequency modulated continuous waves can also achieve the same goal as unmodulated continuous waves in sensing Doppler velocity shifts as well as providing range information but only the theory of unmodulated continuous waves are covered because this study is only interested in Doppler velocity shifts caused by dynamic hand gestures.

2.3.1 Unmodulated Continuous Wave

Unmodulated continuous wave (CW) radars continuously transmit a signal to illuminate the target object while simultaneously continuously receiving echo reflections from the scatters of the illuminated target objects. The frequency from the echoes of stationary objects will have no change from the transmitted illumination signal. If the illuminated target and the radar are not stationary with respect to one another, the frequency of the received echo will differ from the transmitted frequency. This phenomenon is also known as the Doppler effect.[39, 47]

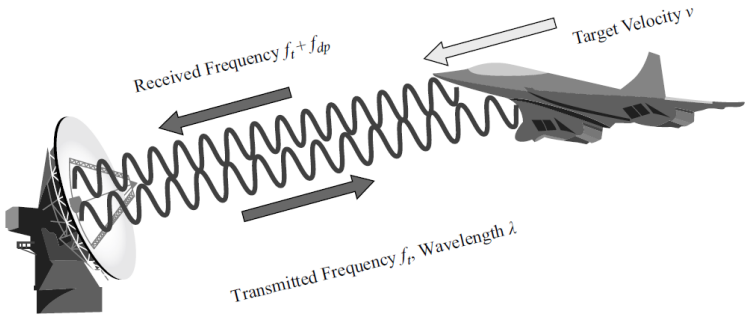


Figure 2.16: CW radar operation [47]

Equation 2.1 shows the relationship between the frequency of an echo of an illuminated target to the original transmitted frequency. From equation 2.1 we also find that for a target moving towards the radar (positive velocity) will result in a higher frequency whereas a target moving away from the radar (negative velocity) will result in a lower frequency.[39]

$$f_r = \left(\frac{1 + v/c}{1 - v/c}\right) f \tag{2.1}$$

Where:

- f_r: received frequency
- f: transmitted frequency
- v: target velocity relative to the radar
- c: propagation speed (speed of light for radar and speed of sound for sonar)

Equation 2.1 can be simplified by expanding the denominator in a binomial series as follows:

$$\begin{aligned} f_r &= (1 + v/c)(1 - v/c)^{-1} f \\ &= (1 + v/c)[1 + (v/c) + (v/c)^2 + \dots]f \\ &= [1 + 2(v/c) + 2(v/c)^2 + \dots]f \end{aligned} \tag{2.2}$$

Equation 2.2 can be simplified because the speed of propagation is a lot higher than the speed of the illuminated target by discarding all terms of (v/c) that are second order and higher which leaves:

$$f_r = [1 + 2(v/c)]f] \tag{2.3}$$

The Doppler frequency is the difference in the transmitted frequency and the received frequency. This can be estimated by rearranging equation 2.3.

$$\begin{aligned} f_r &= [1 + 2(v/c)]f] \\ f_r &= f + 2\frac{v}{c}f \\ (f_r - f) &= 2\frac{v}{c}f \\ f_d &= 2\frac{v}{c}f \end{aligned} \tag{2.4}$$

Where:

- f_d: Doppler frequency

Analysing equation 2.4 we find that the Doppler frequency is directly proportional to the velocity of the target. The equation can further be simplified by replacing (c/f) with λ.[39]

$$f_d = 2\frac{v}{c}f = 2\frac{v}{\lambda} \tag{2.5}$$

For a monostatic radar configuration, the Doppler shift is proportional to the velocity along the line of sight(LOS) between the target and the radar, also known as the radial velocity. Consider a target moving in the direction that is not directly in the LOS, the radial velocity of the target would be related to the angle between the velocity vector of the target and the LOS(ψ). The Doppler velocity would be as follows:[39]

$$f_d = 2\frac{v}{\lambda}cos\psi \tag{2.6}$$

Figure 2.17 is a visual illustration of equation 2.6

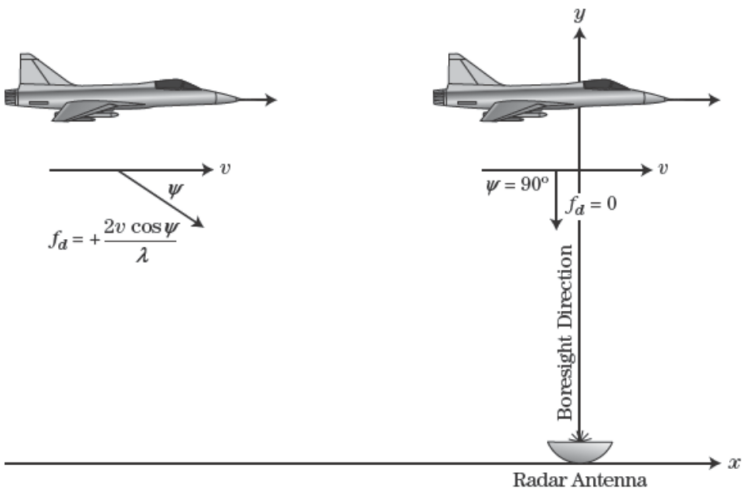


Figure 2.17: Radial component of velocity causing the Doppler shift[39]

A spectrogram is used to visualise the Doppler frequency of the received signal. A spectrogram is a series of overlapping, short-time Fourier transforms. This allows the received signal to be visualised as a function of Doppler frequency against time.

2.4 Gesture Recognition Algorithms

There is one main category of machine learning that has been dominating the current area of research with regards to radar/sonar gesture recognition; deep learning.[53] Majority of the research on gesture recognition using deep learning specifically use Convolutional Neural Networks (CNN).[53] A category of deep learning that is becoming increasingly popular is Long Short Term Memory networks (LSTM) which is a special type of Recurrent Neural Networks (RNN).[53] There are also some machine learning algorithms such as support vector machines (SVM)[40] and computer vision techniques such as video optical flow[45] that have been used for gesture recognition.

Deep learning and machine learning allows computers to recognise data using a statistical approach. There are two very important concepts in deep learning, which are training, validation and testing. Deep learning networks are trained by taking a large, unknown data set with known labels, also known as supervised learning, and trains its neurons so that over time, the probability of the network arriving at the right answer increases. The network improves its probability of arriving at the correct answer by using a validation data set. The validation data set in deep learning is a data set which was not used for training and used only to see

how accurate the network is. Once a network is trained another data set that was not used for training and validation is used to see how accurate the network is with unseen data.[54]

There is a term in machine learning known as hyperparameters. These are parameters that are set external to the model parameters that cannot be estimated by the data. Typically, hyperparameters are chosen using some form of a searching algorithm such as a grid search. Parameters such as the learning rate and the number of hidden units are examples of hyperparameters. [55]

Cross-validation is a concept where a single data set is used for training and validation. The data set is split up into n amount of smaller data sets. One of the n amounts of data sets is reserved for validation and the other data sets are used for training. This process is repeated until all the data sets have been used both for training and validation. Cross-validation is often used when the amount of data available for training and testing is limited [56]. Figure 2.18 illustrates how cross-validation divides the data set.

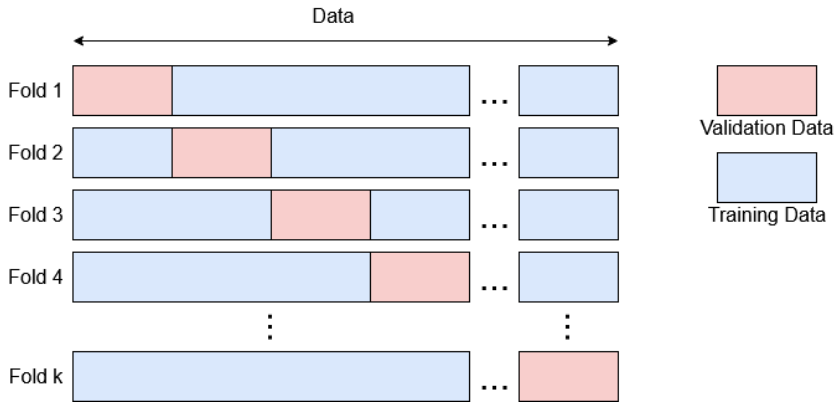


Figure 2.18: Cross-validation

The basic theory and operation of CNN and LSTM networks will be discussed in the following sections.

2.4.1 Convolutional Neural Networks (CNN)

CNN's make use of convolutions, as the name suggests, which uses filters to filter the input that results in an activation function. All the activation functions form a feature map which is used to predict the information contained in the data. [57] The simplest way to describe CNN is a neural network that uses many copies of the same group of neurons or filters. A data point and the neighbouring data points forming a small segment of data is looked at by a group of neurons which computes certain features of the data segment. The layer of repeating neuron groups is

called a convolutional layer. The output of the convolutional layer is fed into a fully connected layer. The fully connected layer is a group of different neurons where every input is connected to every neuron. Figure 2.19 shows segments of data being fed into groups of neurons 'A' and the output of the convolutional layer being fed into a fully connected layer 'F'. This concept can be applied in more than one dimension. It is commonly used in 2 dimensions on images for image recognition.[58]

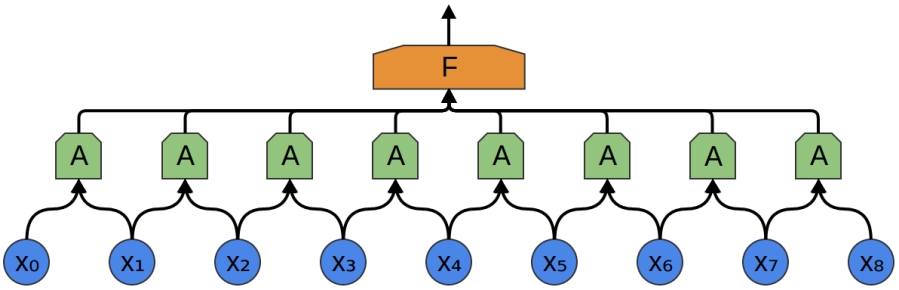


Figure 2.19: Basic representation of a CNN. "A" represents a group of neurons or a filter and "F" is a fully connected layer[58]

Each neuron, or filter, in a group of neurons, is used to detect a different feature in the data-set. Traditional layers have these neurons in parallel but there have cases where each group of neurons are arranged in layers of parallel neurons.[58]

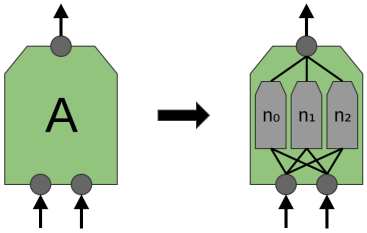


Figure 2.20: Traditional arrangement of neurons in a filter[58]

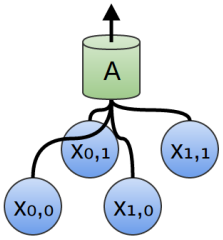


Figure 2.21: Two dimensional application of CNN used for image recognition [58]

More layers can be stacked onto the previous layer allowing the network to detect more high level, abstract features. It is only necessary to detect the presence of a feature instead of its exact location or the time where it occurs for more high-level feature detection. In these cases, max-pooling layers are very popular. These layers take the output of a few blocks from the previous layer and use the largest block as the output. This is a form of decimation that allows neural networks to work on larger data sets.[58]

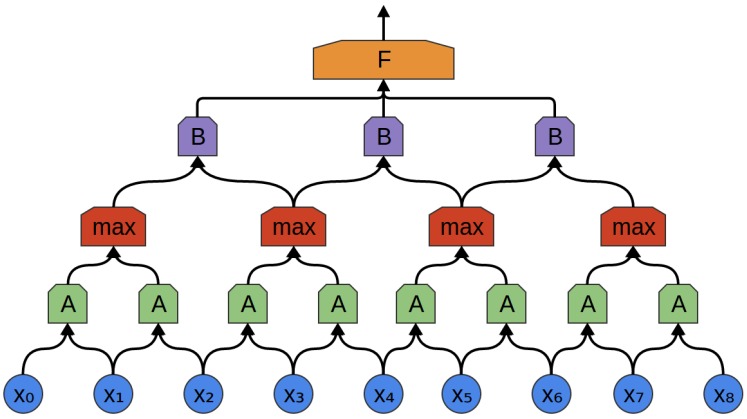


Figure 2.22: CNN with multiple layers [58]

The work that was done by Kulhandjian *et al.* [2] studies the use of deep learning on sign language gestures. The system uses an X-band Doppler radar to classify 10 hand gestures with Doppler spectrogram data. Using a deep convolutional neural network (DCNN) of 3 convolutional layers, the first with 8 filters, the second with 16 filters and the third with 32 filters, a classification accuracy of 87.5% was achieved. Using the VGG-16 algorithm, which is a deep neural network with 16 convolutional layers that are typically used on large scale image recognition tasks, the system was able to achieve an accuracy of 98%. Figure 2.23 and 2.24 show the training results using the DCNN and VGG-16

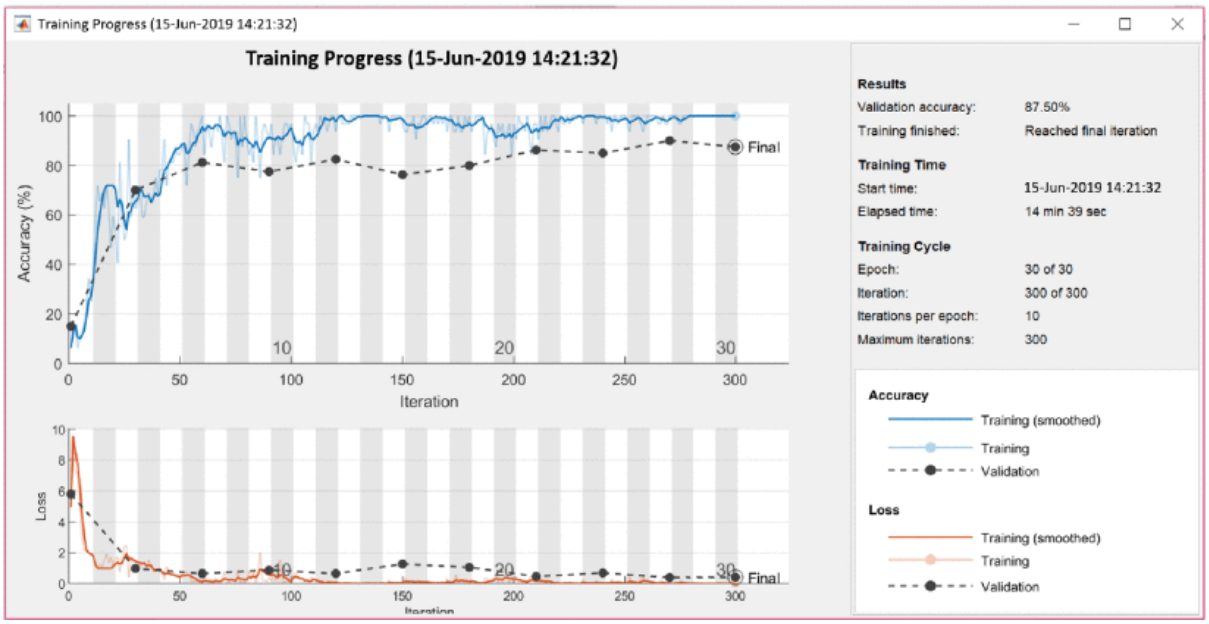


Figure 2.23: Training results using DCNN [2]

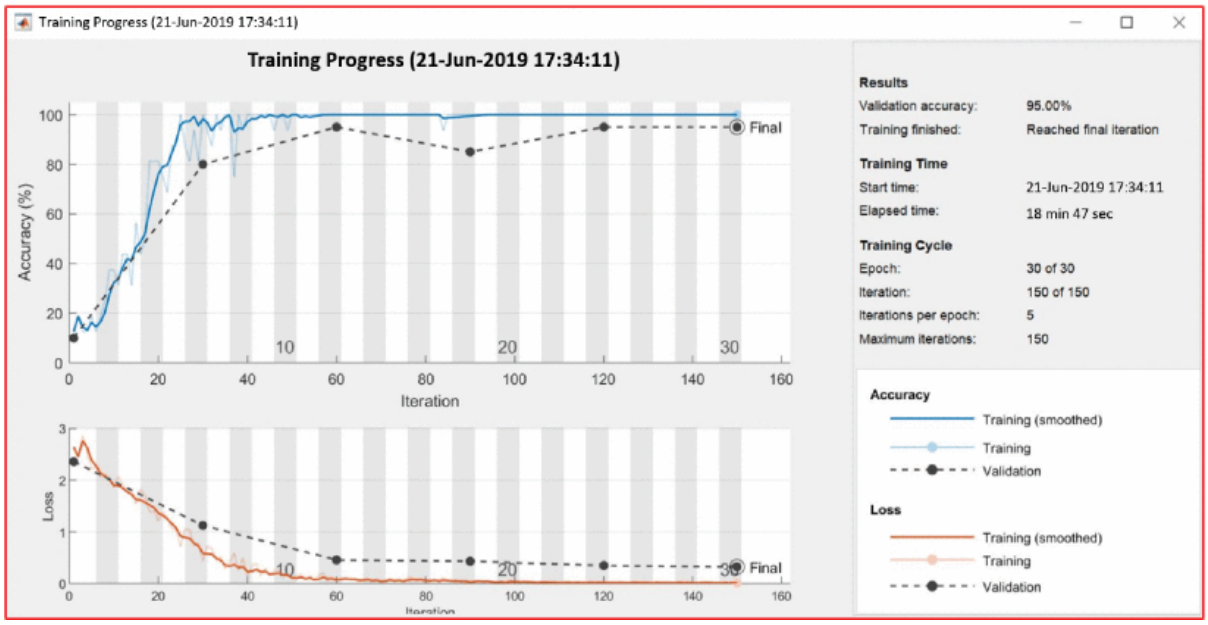


Figure 2.24: Training results using VGG-16 [2]

A study that was done by Kim *et al.* [9] uses a 5.8 GHz continuous wave Bumblebee Doppler radar to classify 10 gestures using CNN. The system uses 3 convolutional layers and 5 fold cross-validation and can achieve an accuracy of 85.6% which is shown in figure 2.25. The system struggles to distinguish between similar gestures such as swiping up (a), down (b), left (c) and right (d) as seen in figure 2.26. With those gestures removed from the data set, the recognition rate increased to 93.1%.

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
88%	84%	90%	84%	82%	85.6%

Figure 2.25: Classification accuracy from cross-validation [9]

Ac Es	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)
(a)	76	20	0	0	0	0	0	0	4	0
(b)	8	48	8	0	0	0	0	0	4	4
(c)	4	16	80	0	0	0	0	0	0	0
(d)	4	0	0	92	12	0	0	0	0	0
(e)	0	0	4	4	88	0	0	0	0	0
(f)	0	0	0	0	0	100	0	0	0	0
(g)	0	0	0	0	0	0	100	0	0	0
(h)	4	0	0	4	0	0	0	100	4	0
(i)	4	0	4	0	0	0	0	0	80	4
(j)	0	16	4	0	0	0	0	0	8	92

Figure 2.26: Confusion matrix of gesture classification results with 1 antenna [9]

The system mentioned previously only use 1 receiver. A study done by Skaria *et al.* [15], uses a 24 GHz continuous-wave Doppler radar with 2 antennas. The system sends 3 features into the CNN, 1 spectrogram from each of the receiving antennas and the angle of arrival calculated using the two antennas. The network consists of 3 convolutional layers, the first with 6 filters, the second with 4 filters and the third with 2 filters. The study shows a 10% increase in accuracy over systems that only use 1 receiver and can achieve an accuracy of 95%. Analysing the diagonal values in figure 2.27, it shows that using 2 antennas increases the systems ability to distinguish similar gestures compared to the study that was done by Kim *et al.* [9].

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	92.16	2.08	0	0	0.96	0	0.56	0.16	0.88	0.08	3.04	0.08	0	0
2	2.8	93.04	0	0	0.4	0.40	0.24	0.56	0.08	0.08	0.32	1.84	0.24	0
3	0.16	0.40	97.92	0.64	0	0.24	0	0	0.32	0	0.24	0.08	0	0
4	0	0	0.56	97.6	0	0.40	0.16	0.16	0.40	0.40	0	0.24	0	0.08
5	0.56	0.24	0.08	0	95.84	1.68	0.24	0	0.08	0.40	0.32	0.40	0.08	0.08
6	0	0.48	0.08	0	1.52	96.72	0	0.24	0	0	0	0.96	0	0
7	0.80	.64	0	0	1.12	0	91.76	0.48	0.56	0.24	3.52	0.16	0.72	0
8	0.64	0.96	0	0.32	0.08	0.16	2	92.96	0	0.08	0.24	2.16	0.08	0.32
9	0.24	0	0	0	0.08	0	0.24	0	98.24	0.80	0	0	0.40	0
10	0	0.32	0.16	0.48	0.16	0	0.08	0.40	1.12	96.4	0	0	0.88	0
11	1.76	0.64	0	0	0.24	0.08	4.64	0.08	0	0	92	0.56	0	0
12	0.56	2.16	0	0	0.24	0.08	0.16	1.36	0	0	0.56	94.88	0	0
13	0	0.08	0	0	0.08	0	0.88	0	0.32	1.76	0	0	96.32	0.56
14	0	0.08	0	0	0	0	0	0	0	0.24	0	0	0.32	99.36

Figure 2.27: Confusion matrix of gesture classification results with 2 antennas [15]

Another study done by Chen and Li *et al.* [14] uses a 24 GHz FMWC radar with 4 receivers to classify 6 gestures. The Doppler spectrograms are fed into a fusion layer consisting of 7 convolutional layers which are then fed into a global mean pooling layer. Using only 1 receiver, the system achieved an average accuracy of 62.89%. With 2 receivers, the system achieved an average accuracy of 83.32%. With all 4 receivers, the system achieved an average accuracy of 98.79%. Figure 2.28 shows the increase in classification accuracy as the number of receiver increases.

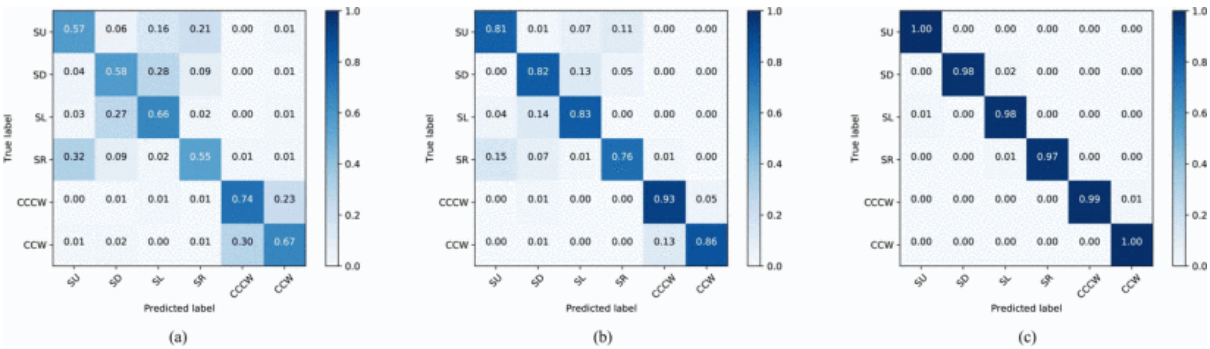


Figure 2.28: Confusion matrix of gesture classification results using 1 (a), 2 (b), and 4 (c) antennas [14]

A study that was done by Ling *et al.* [16] used a 20 kHz sonar system to classify 12 gestures using the differential channel impulse response of the system. The network consists of 2 convolutional layers with pooling which achieves an accuracy of 97%. The confusion matrix showing the classification accuracy of the system is shown in figure 2.29.

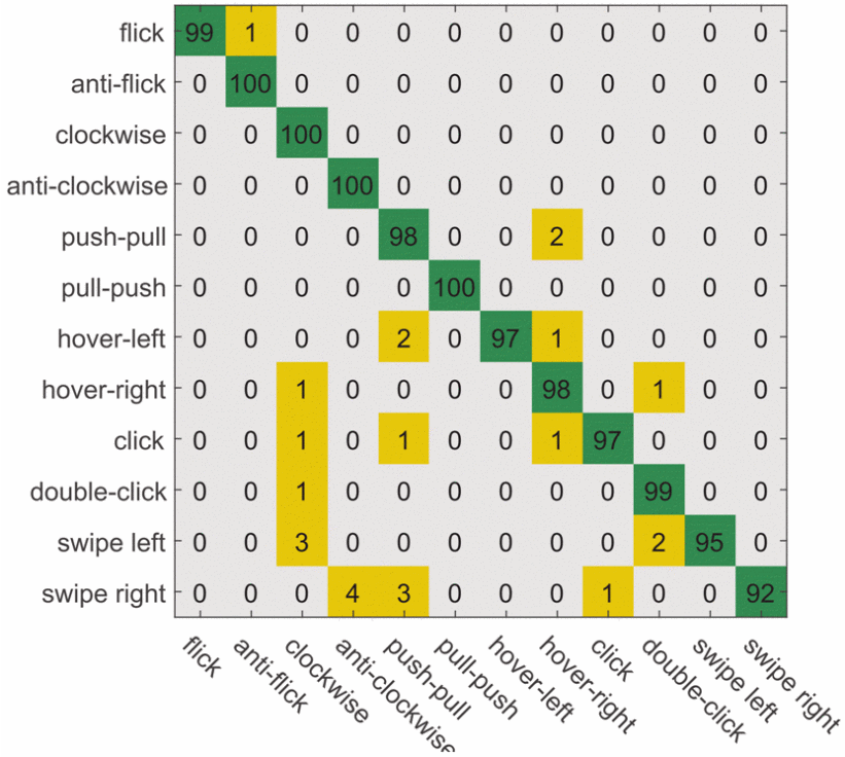


Figure 2.29: Confusion matrix of gesture classification results using dCIR [16]

2.4.2 Long Short Term Memory networks (LSTM)

Traditional neural networks extract features from a select section of data and do not hold any information from the past. This downfall makes it difficult for traditional neural networks to

learn sequential data where past data affects future data such as human speech.[59]

A recurrent neural network is a chain of the same copy of a neural network where each network passes information onto the next network. Basic RNN's have a single tanh layer that allows past information to persist. This allows for RNN's to address the issue that traditional neural networks face of not having the persistence of past information. RNN's are, however, unable to learn data-sets that have long term data dependencies.[59]

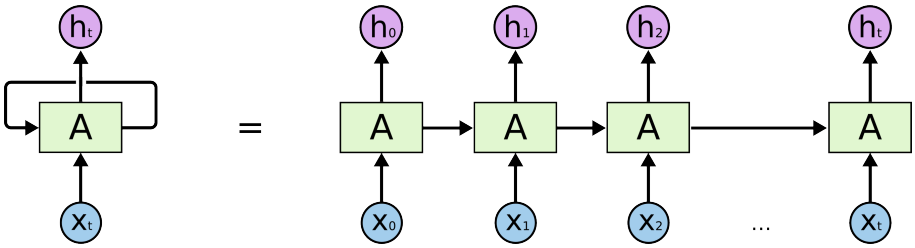


Figure 2.30: Expanded view of an RNN [59]

A Long Short Term Memory network is a special type of RNN. Instead of having a single tanh layer, the basic structure of an LSTM has four gates which allow the LSTM to choose what information from the past to remember and what to forget. This gives the LSTM the ability to learn from data with long term dependencies.[59] LSTM networks first showed success in natural language processing which has heavy temporal dependencies.[53]

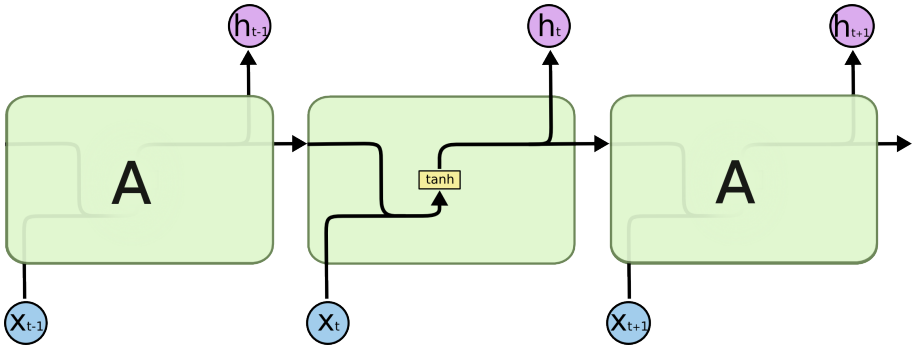


Figure 2.31: Structure inside a standard RNN [59]

The key to LSTM's having the ability to learn from data with long term dependencies is called the cell state. The cell state is what carries past information. The LSTM can control what information is held in the cell state using gates. The LSTM has three gates which regulate the data in the cell state.[59]

The first gate is a sigmoid layer known as the 'forget gate layer'. This layer looks at the output from the previous network and the input to the current network and decides how much

information from the previous cell state to forget.[59]

The next gate is a sigmoid layer called the 'input gate layer' which decides what information in the cell state to update. This layer is coupled with a tanh layer which will decide what values of the cell state to update.[59]

The updated cell state is used to decide what the output of the current copy of the network is. The cell state is put through a tanh layer to push the values to 1 and -1. This output of the tanh layer is multiplied by a sigmoid gate to filter the output. The output of this network and the cell state is passed onto the next copy of the network and the process repeats.[59]

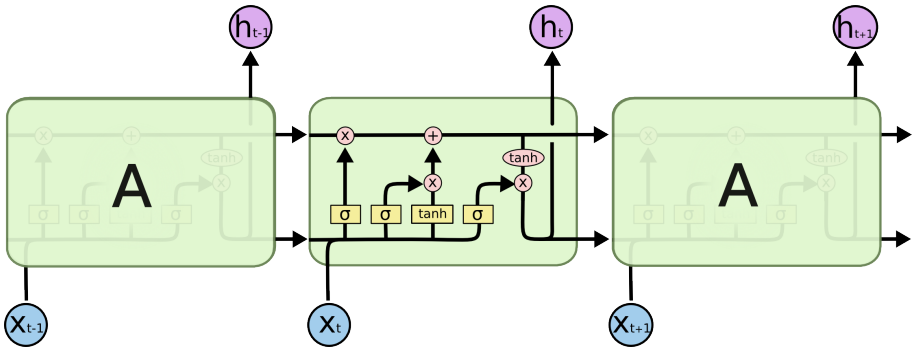


Figure 2.32: Structure of a standard LSTM [59]

Figure 2.32 is the basic structure of an LSTM. There are many variants of the LSTM that work better in different scenarios. One of the variants allows the gates to look at what is in the cell state. There is a variant of the LSTM called the Gated Recurrent Unit (GRU) which merges the forget gate and the input gate into one update gate as well as merging the cell state and the hidden state which allows for a simpler structure.[59] The bidirectional LSTM (bi-LSTM) is a variant where the original data is fed into the network followed by a temporally flipped version of the data.[1]

A paper by Choi *et al.* [10] shows a radar-based hand gesture recognition system using the Google soli[43] radar system and LSTM to achieve an average accuracy of 99.10% with 10 different hand gestures. The study uses 5-fold cross-validation to train the network. The classification accuracy for each fold is shown in figure 2.33 comparing a 2D CNN, traditional RNN encoder and an LSTM encoder.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg. (Std.)
2D CNN	93.51	95.93	96.54	95.49	95.48	95.39 (±1.01)
RNN encoder	90.41	93.90	94.48	93.35	92.39	92.90 (±1.42)
LSTM encoder	98.96	99.51	99.24	99.09	98.73	99.10 (±0.26)

Figure 2.33: Classification accuracy from cross-validation [10]

Shrestha *et al.* [1] use an FMCW radar system to detect different Doppler data on 6 human activities and show the improvement in accuracy by using bi-LSTM instead of LSTM from 78% up to 91% as seen in figure 2.34. This study also shows that optimising the learning rate of the system can greatly impact the classification accuracy of the network with sub-optimal learning rates showing an accuracy of less than 40%.

Classifier	Mean	Standard deviation	Maximum	Minimum
Bi-LSTM	91	5	98	69
LSTM	78	9	92	54
Range BiLSTM	76	7	87	54

Figure 2.34: Classification accuracy of range-LSTM vs LSTM vs bi-LSTM [1]

Zhang *et al.* [13] and Wang *et al.*[11] show interesting work by stacking an LSTM on to of a CNN layer. Both studies use 3 convolutional layers stacked by an additional LSTM layer. The first study uses 64, 128 and 256 filters in each of the layers respectfully in the convolutional layers while the second paper uses 32, 64 and 128 filters. The paper by Zhang *et al.* also investigates the effect of different parameters on recognition accuracy. The paper shows that, for their specific data set, the accuracy of the network stops increasing after 25 layers and maintains a constant accuracy. The paper also shows that increasing the data set improves the overall recognition accuracy. The paper also shows that the range, magnitude and duration of the gesture also impact the recognition accuracy. The final parameter that affects the accuracy of the network is the number of participants. Having more participants ensures that the system does not overfit resulting in better recognition accuracy. Figure 2.35 shows the network structure of the system.

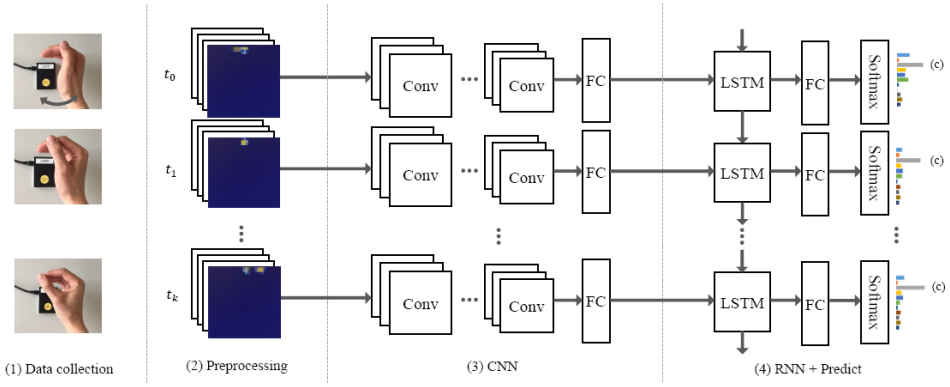


Figure 2.35: LSTM layer on top of CNN layer [11]

The use of LSTM for hand gesture recognition is extremely attractive as the hand gestures have a strong temporal dependency. [53]

2.5 Critical Review

In literature about sonar and radar sensing, there seems to be a gap in signal processing after the Doppler spectrogram data has been acquired. The study by Ling *et al.* [16] estimate the differential channel impulse response of the system before feeding it into the deep learning network which improves the signal to clutter ratio (SCR) of the system. Typically improving the SCR or signal to noise ratio (SNR) of a system improves the overall performance of the system.[39]

While literature shows that both CNN and LSTM networks achieve acceptable recognition results for radar systems, there is only literature showing the use of CNN in sonar systems and almost no LSTM networks. LSTM networks address the issue of deep learning networks struggling to learn from data with temporal dependencies. Dynamic hand gestures are classified by its spatial information over time [17] which is a strong motivation to use LSTM networks. [53]

This study will aim to fill the gap in the literature by including signal post-processing before machine learning as well as implementing and comparing an LSTM network and a CNN on a multi-channel sonar system.

Chapter 3: Requirements

3.1 Baseline User Requirement Description

From the gap in literature, the baseline user requirement of the system is to design and build a sonar system that can classify a basic set of hand gestures. The system should use machine learning as a base for gesture recognition. The system should also include some form of post-processing of the data before it is used for machine learning. The system should be able to classify the following gestures at a reasonable distance away from the sensors:

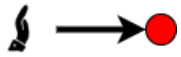
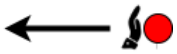
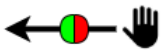
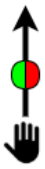
Gesture no.	Gesture Description	Gesture Illustration
1	Hand moving towards sensors	
2	Hand moving away from sensors	
3	Hand swiping from right to left in front of sensors	
4	Hand swiping from left to right in front of sensors	
5	Hand swiping from top to bottom in front of sensors	
6	Hand swiping from bottom to top in front of sensors	
7	Hand moving in a clockwise circle in the z axis in front of sensors	
8	Hand moving in a counter clockwise circle in the z axis in front of sensors	

Table 3.1: Minimal gestures that the system needs to be able to classify

3.2 Technical Requirements

Index	Requirement
T1	The system should operate at ultrasonic frequencies
T2	The system should be able to sense hand gestures up to one metre away
T3	The system should have some form of post-processing of the data prior to using it for machine learning
T4	The system should be able to classify the basic dynamic hand gestures defined in table 3.1
T5	The system should classify hand gestures using a machine learning approach

Table 3.2: Technical requirements for the hand gesture recognition system

3.3 Acceptance Test Procedures

ATP001	Deep Network Performance
Mapping	T4, T5
Test Protocol	Train the network of the system using k fold cross validation on the gestures defined in table 3.1 and the compare the results of the LSTM network to the CNN
Pass Condition	System can classify gestures with at least 80% accuracy with either the LSTM network or the CNN
Fail Condition	System is unable to classify gestures with at least 80% accuracy

Table 3.3: Acceptance test for deep network performance

ATP002	Range Test
Mapping	T2
Test Protocol	Measure classification accuracy of the system at ranges up to 1 metre away from the sensor with the participant directly in the line of sight of the transmitter while always facing the transmitter
Pass Condition	Classification accuracy greater than 80% for gestures up to a metre away from the sensor
Fail Condition	Classification accuracy less than 80% for gestures up to a metre away from the sensor

Table 3.4: Acceptance test for device range

Chapter 4: System Design Overview

4.1 System Operation

The system operates by first transmitting an ultrasonic signal from the ultrasonic transmitter. The sound waves of the signal propagate through the air and will reflect off objects in the scene. Reflections from the dynamic hand gestures will experience a Doppler shift in frequency. The reflections from objects in the scene including the hand gestures are captured by the receivers. These signals are digitised and sampled to a computer where the signal is converted to the frequency-time domain to observe the Doppler frequency shifts caused by the dynamic hand gestures. The Doppler data is then fed into the deep learning algorithm where the hand gestures are classified. The flow of the system is illustrated in figure 4.1.

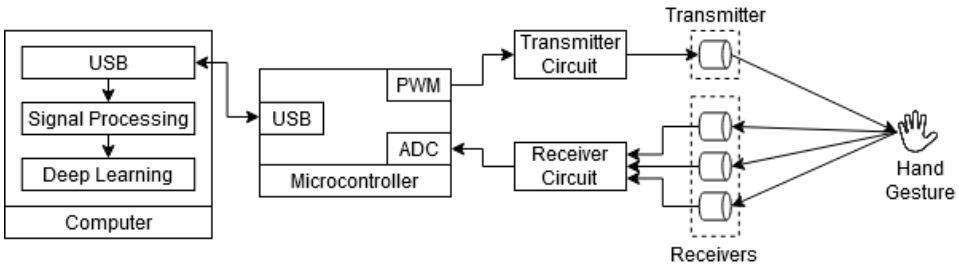


Figure 4.1: System operation

4.2 Design Methodology

The design of the gesture recognition system falls into four major subsystems shown in figure 4.2; data acquisition, data logging, signal processing design and deep learning algorithm design. The data acquisition design involves the circuit design to interface the ultrasonic sensors to the microcontroller, transmitting signals from and sampling signals to the microcontroller, PCB design and the mounting system that the sensors attach to. The subsystems in the data acquisition system were simulated and tested in software before they were implemented. The data logging design involves communicating the computer with the microcontroller which includes sending commands to the microcontroller, streaming and saving data to the computer. The signal processing design involves processing the data into a form that is more usable for deep

learning purposes. Finally, the deep learning design section involves designing and optimising the LSTM network and CNN.

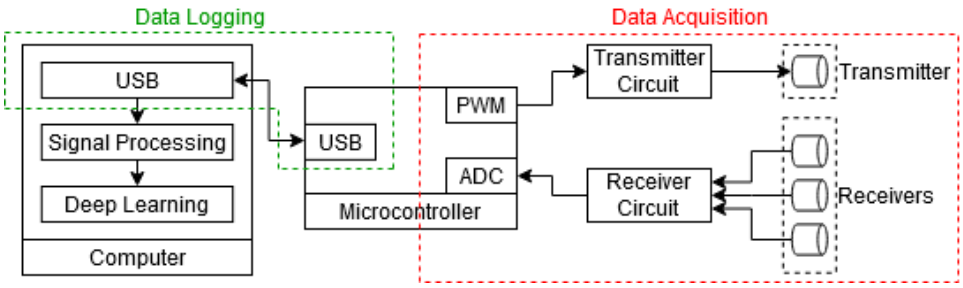


Figure 4.2: Design subsystems

Figure 4.3 illustrates the different subsystems that make up the whole system. The design of the subsystems are centred around the choice of the microcontroller and the sonar sensors. As such, the component selection for these devices will be discussed before designing the subsystems.

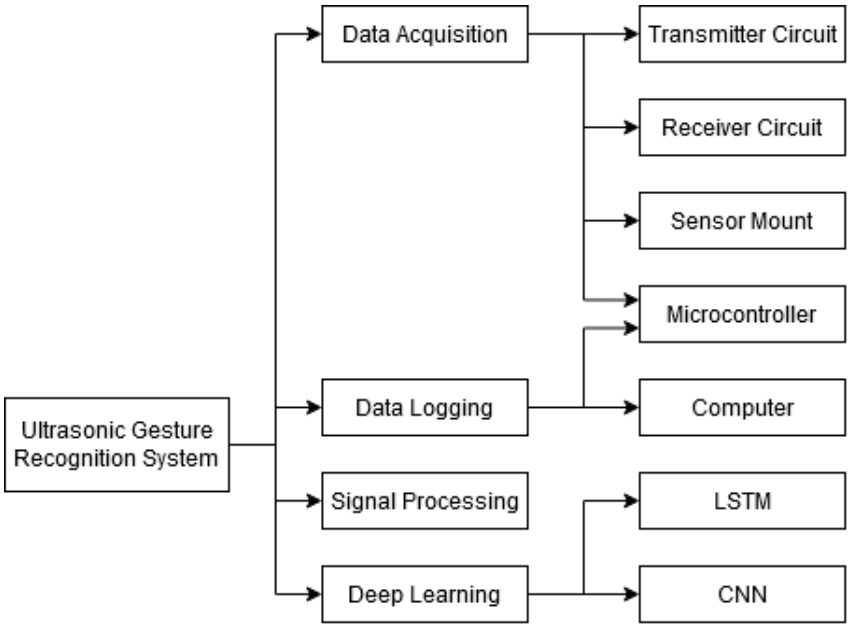


Figure 4.3: Design subsystem breakdown

Chapter 5: System Design and Testing

5.1 Sonar Sensor Selection

As discussed in the literature review ultrasonic transducers are the preferred sensors for hand gesture recognition using sonar because it is inaudible to humans. The transducers can be used as both transmitter and receiver but are often tuned to do one or the other. While the options for ultrasonic transmitters are limited, there are a few options available with regards to ultrasonic receivers. The specifications for the ultrasonic barrel transmitter are shown in table 5.1.

Technical Specifications	
Centre Frequency f_0	40 kHz ± 1 kHz
Bandwidth (-6dB)	2 kHz
Directivity	80°
Sound Pressure Level	120 dB @ 20 V_{pp}
Capacitance	2400 pF ± 20%
Impedance	400 Ω@ 40 kHz

Table 5.1: Technical specifications of the ultrasonic barrel transmitter

The two options for ultrasonic receivers are the barrel receivers similar to the ultrasonic transmitters, shown in figure 5.1, and ultrasonic microphone receivers shown in figure 5.2. The technical specifications of the receivers are shown in table 5.2 and table 5.3. From the technical specifications, it shows that the ultrasonic microphone performs better overall. The microphone has a wider usable bandwidth, has a wider directivity and is more sensitive than the ultrasonic barrel receiver. The specifications for the ultrasonic receivers are however sufficient for this design. Due to the availability of components, the ultrasonic barrel receivers will be used for this design.

Technical Specifications	
Centre Frequency	40kHz ± 1kHz
Bandwidth (-6dB)	2kHz
Directivity	80°
Sensitivity	-61dBV/Pa (minimum)

Table 5.2: Technical specifications of the ultrasonic barrel receiver

Figure 5.1: Ultrasonic Transducer [60]

Technical Specifications	
Frequency response	20Hz - 10kHz ± 2dB 10kHz - 80kHz ± 10dB
Directivity	Omnidirectional
Sensitivity	-41dBV/Pa (minimum)

Table 5.3: Technical specifications of Knowles Ultrasonic Mic SPU0410LR5H-QB [50]

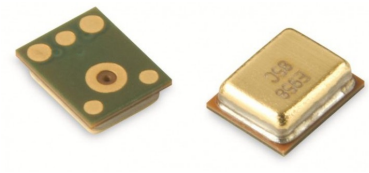


Figure 5.2: Knowles Ultrasonic Mic SPU0410LR5H-QB [50]

The ultrasonic transducers are commonly used for sonar range applications are readily available. Figure 5.3 shows the frequency response analysis from the ultrasonic transmitter to receiver. This frequency response was acquired by pointing the transmitter and receiver directly at one another with no spacing between them and sweeping the frequency from 10 kHz to 100 kHz. As seen in the figure the frequency response flattens out at around -60 dBV because of the limited resolution of the digital oscilloscope. The transmitters and receivers have very similar frequency responses so it can be assumed that there is at least -30 dBV of attenuation outside of the main lobe of operation for the sensors. The datasheets for the ultrasonic transducers can

be found in appendix D.

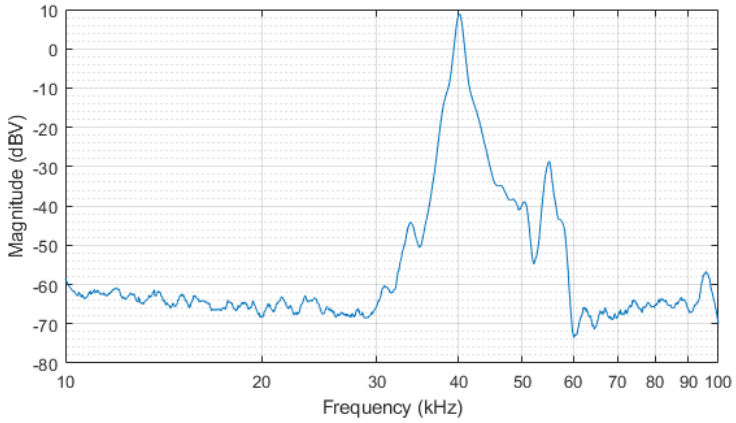


Figure 5.3: Frequency response analysis from ultrasonic transmitter to receiver

5.2 Microcontroller Selection

The two microcontrollers in consideration are the Teensy 4.0 by PJRC and the STM32F4Discovery board by STMicroelectronics. The technical specifications of these two boards are shown below in table 5.4. These two microcontrollers were considered due to there abundance of IO allowing for easy prototyping.

Technical Specifications		
Device	Teensy 4.0	STM32F4Discovery
Processor	32-bit 600 MHz ARM Cortex-M7	32-bit 168 MHz ARM Cortex-M4
USB	480 MBit/s USB port	12 MBit/second USB port
PWM enabled	Yes	Yes
ADC	2 12-bit ADCs on chip with 14 analogue pins	3 12-bit ADCs on chip with 16 analogue pins
DAC	No	2 12-bit DAC's on chip
Footprint	18 x 26 mm	66 x 97 mm

Table 5.4: Comparison between Teensy 4.0 and STM32F4Discovery microcontrollers [61, 62]

The Teensy 4.0 was chosen over the STM32F4Discovery for its performance and its ease of use. It uses a modified version of the Arduino library which makes the device very simple to program compared to the STM32F4Discovery which uses HAL. The Teensy 4.0 can also be programmed using regular C/C++ libraries. While using the DAC's on the STM32F4Discovery to generate a transmit signal may simplify the system, using a PWM pin on the Teensy 4.0 with some form of filtering will achieve the same task. The biggest factor in choosing the Teensy 4.0 over the STM32F4Discovery is the USB baud rates. Sampling 3 ADC channels at 12-bits and streaming the data through to the PC would overload the 12MBit/s USB bus on the STM32F4Discovery (the minimum USB baud rate is discussed further in section 5.4).

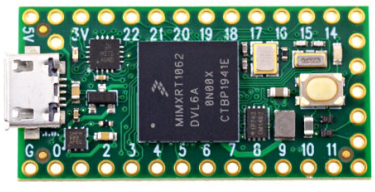


Figure 5.4: Teensy 4.0 [61]

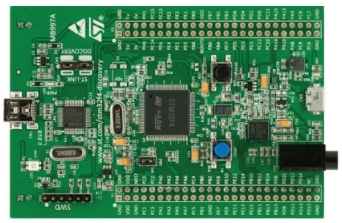


Figure 5.5: STM32F4Discovery [62]

5.3 Data Acquisition

To be able to process any data, the data needs first be acquired. In this section, the design and implementation of the data acquisition subsystem, as seen in figure 5.6, will be discussed in detail. The data acquisition can be broken up into several subsystems; microcontroller signal generation, microcontroller signal sampling, the transmitter circuit, the receiver circuit, PCB design and sensor mount design. The link between the subsystems can be seen in figure 5.7.

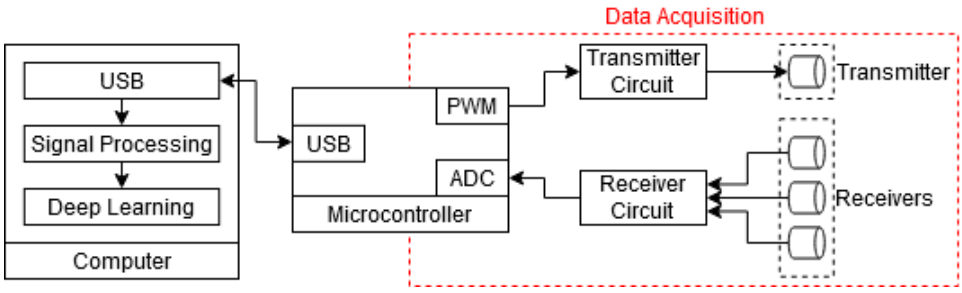


Figure 5.6: Data acquisition overview

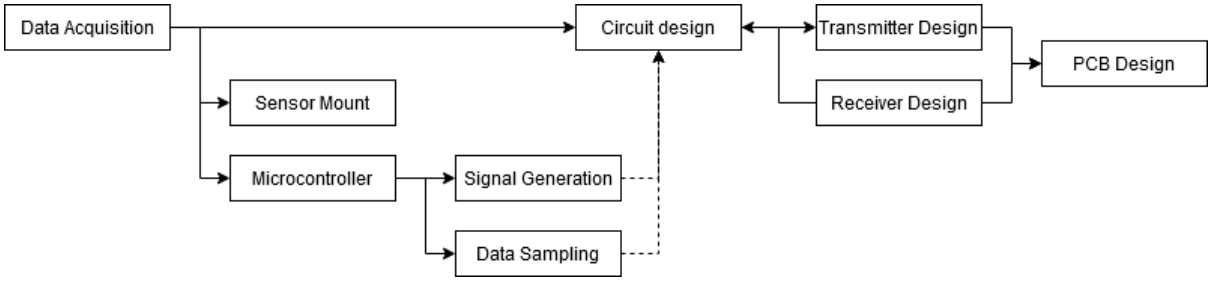


Figure 5.7: Data acquisition functional design breakdown

5.3.1 Microcontroller Signal Generation

The Teensy 4.0 microcontroller does not have a built-in digital to analogue converter (DAC). Thus, the waveform was generated using a PWM pin to create a square wave of the desired frequency. The Teensy library allows this square wave to be easily generated using a build-in tone function. The function takes in a specific PWM enabled pin and a frequency as parameters and outputs a square wave at that specific pin with the desired frequency. Figure 5.8 and shows the square wave generated by the Teensy 4.0 on a digital oscilloscope. For this study, a 40 kHz square wave was generated.

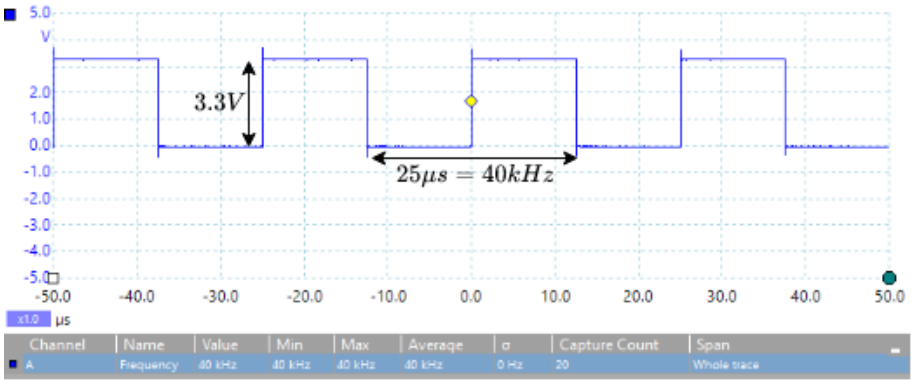


Figure 5.8: Square wave generated on the Teensy 4.0

5.3.2 Microcontroller Signal Sampling

From literature, it is shown that using more than one receiver greatly improves the classification accuracy of the system. As such, the design of this system will use 3 receivers. For this reason, the Teensy 4.0 will have 3 analogue input pins configured. The ADC is configured to 12-bits to achieve the highest dynamic range available. The Nyquist sampling theorem says that to properly sample a signal without aliasing, the sampling frequency needs to be at least twice that of highest frequency in the signal.[63] Figure 5.9 is an example of aliasing due to an insufficient sampling frequency and this is shown by illustrating how the spectrum of the signal is overlapping. Assuming that only the frequencies in the bandwidth of the receivers are of interest, the highest frequency would be 42 kHz meaning the highest sampling frequency should be at least 84 kHz. The highest frequency that the Teensy 4.0 can sample 3 channels on one ADC without any errors is 200 kHz which satisfies the Nyquist requirement. An interrupt timer is used to sample the ADC channels at 200 kHz. An interrupt is triggered every 5µs telling the ADC to sample all 3 channels. Figure 5.10 shows the 3 ADC channels sampling a 42 kHz signal generated by a digital oscilloscope. Figure 5.11 shows the spectrum of data from one of the ADC channels with a peak at 42 kHz.

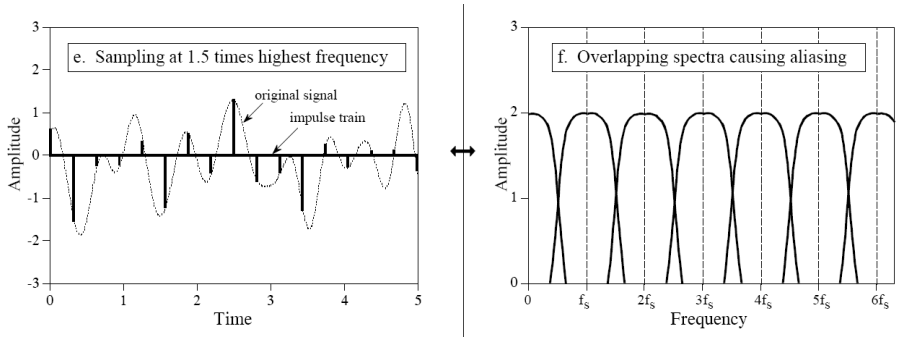


Figure 5.9: Illustration of aliasing[63]

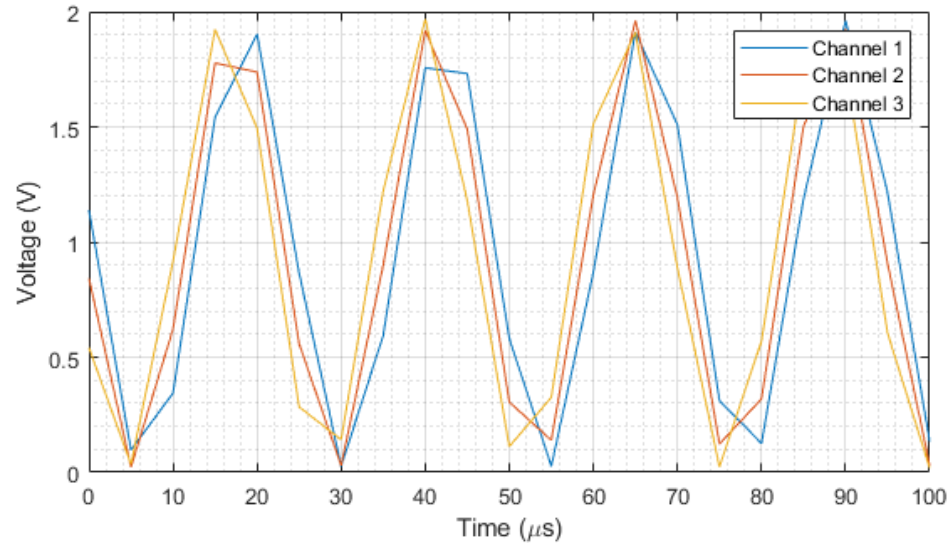


Figure 5.10: 42 kHz sine wave sampled on 3 channels on the Teensy 4.0 at 200 kHz

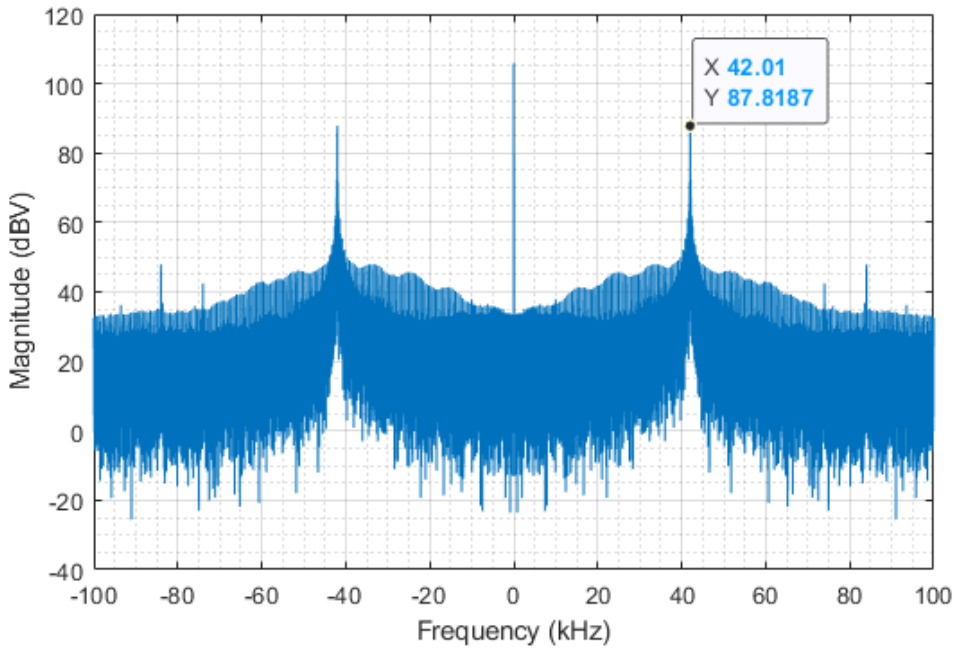


Figure 5.11: Spectrum of a 42 kHz sine wave sampled on the Teensy 4.0 at 200 kHz

5.3.3 Transmitter Circuit Design

The function of the transmitter circuit is to perform filtering and amplification on the input signal to be sent to the transmitter. The operation of the circuit is illustrated in figure 5.12

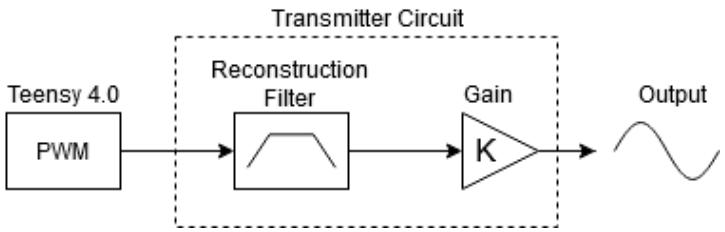


Figure 5.12: Block diagram illustration the operation of the transmitter circuit

The square wave that the Teensy 4.0 is generating can be expressed mathematically by its Fourier expansion:

$$x_{square}(t) = \sum_{n=1,3,5,...}^{\infty} \frac{4}{n\pi} sin(n2\pi f_0 t) \tag{5.1}$$

Assuming all the higher harmonic components are filtered out from equation 5.1, the signal becomes:

$$x_{filtered}(t) = \frac{4}{\pi} sin(2\pi f_0 t) \tag{5.2}$$

Amplifying the filtered signal from equation 5.2, the signal becomes:

$$A \times x_{filtered}(t) = A\frac{4}{\pi} sin(2\pi f_0 t) = Bsin(2\pi f_0 t) \tag{5.3}$$

Equation 5.3 is the desired signal to transmit.

5.3.3.1 Reconstruction Filter Requirements

Ideally, the transmitted waveform should be a sine wave. As seen in the signal generation section however, the signal generated is a square wave. A filter can be used to filter out the high-frequency components of the square wave to transform it into a sine wave. This filter is also called a reconstruction filter.[63] It takes the digitised signal and reconstructs it into a square wave. Figure 5.13 shows the spectrum of the square wave. Filtering out the frequency harmonic components that are higher than 40 kHz as seen in figure 5.13 should reconstruct the square wave into a sine wave. From the data logging section, it is shown that the ADC has a 12-bit resolution. The reconstruction filter needs to filter the harmonics so that they are lower

than the least significant bit of the ADC.

$$A_{LSB} = 20log\left(\frac{1}{2^{12}}\right) = -72.24dBV \tag{5.4}$$

Equation 5.4 shows that the harmonics need to be below -72 dBV for the signal to seem like a pure sine wave to the ADC. Figure 5.13 is a plot of the spectrum of the square wave generated from the Teensy 4.0. The largest harmonic at 120 kHz is 10 dBV lower than the main 40 kHz meaning an additional 62 dBV of attenuation is required from the reconstruction filter.

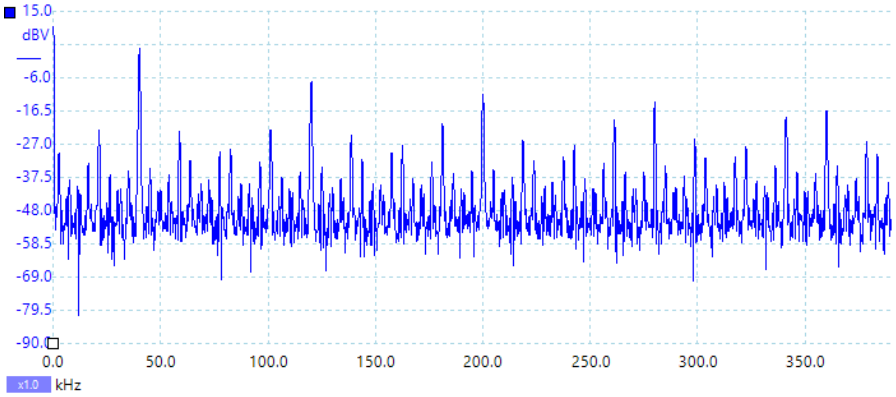


Figure 5.13: Spectrum of the square wave generated on the Teensy 4.0

The type of filter chosen for the reconstruction filter is a Butterworth filter. Butterworth filters have a maximally flat passband which is desirable for signal processing purposes as there is minimal signal distortion in the pass band.[63]. The generalised frequency response of an nth order Butterworth filter is shown in equation 5.5.[64]

$$H_{(j\omega)} = \frac{1}{\sqrt{1 + \epsilon^2\left(\frac{\omega}{\omega_p}\right)^{2n}}} \tag{5.5}$$

Where:

- ω: 2πf
- ε: maximum pass band gain
- n: order of the filter

With the maximum passband defined at the cutoff frequency, ε becomes 1 and ω_p becomes ω_c.

The order of the filter can be calculated by substituting these values into equation 5.5.

$$\begin{aligned} H_{(j\omega)} &= \frac{1}{\sqrt{1+\left(\frac{\omega}{\omega_c}\right)^{2n}}} \\ A_{min} &= \frac{1}{\sqrt{1+\left(\frac{\omega_s}{\omega_c}\right)^{2n}}} \\ {A_{min}}^{-1} &= \sqrt{1+\left(\frac{\omega_s}{\omega_c}\right)^{2n}} \\ {A_{min}}^{-2} &= 1+\left(\frac{\omega_s}{\omega_c}\right)^{2n} \\ \left(\frac{\omega_s}{\omega_c}\right)^{2n} &= {A_{min}}^{-2}-1 \\ \left(\frac{\omega_s}{\omega_c}\right)^{n} &= \sqrt{{A_{min}}^{-2}-1} \\ n &= \frac{log(\sqrt{{A_{min}}^{-2}-1})}{log\left(\frac{\omega_s}{\omega_c}\right)} \end{aligned} \tag{5.6}$$

As discussed earlier, at 120kHz, there needs to be a 62 dBV attenuation for the harmonics to be completely filtered out. A cutoff frequency of 45 kHz is chosen to ensure that the 40 kHz component does not fall in the knee region of the filter. Substituting these values into equation 5.6 yields an order of 7.3 required to filter out the harmonic components. This order can be rounded up to an 8^{th} order filter that is required to filter out the harmonic components.

$$\begin{aligned} n &= \frac{log(\sqrt{{A_{min}}^{-2}-1})}{log\left(\frac{\omega_s}{\omega_c}\right)} \\ n &= \frac{log(\sqrt{10^{\frac{-62}{20}^{-2}}-1})}{log\left(\frac{120000}{45000}\right)} \\ n &= 7.2775 \approx 8 \end{aligned}$$

It needs to be noted, however, that this filter is overcompensating as the ultrasonic transducers already have a narrowband frequency response as discussed in the sonar sensor section.

5.3.3.2 Reconstruction Filter Simulation

A model of the square wave and the filter was simulated in MATLAB SIMULINK. Figure 5.14 shows this model. The filter is an 8^{th} order Butterworth filter with its cutoff frequency set to

45 kHz. Figure 5.15 shows the spectrum of the square wave and the filtered square wave. It shows that there is 68 dBV of attenuation of the largest harmonic validating the calculations. Figure 5.16 shows the square wave before and after filtering.

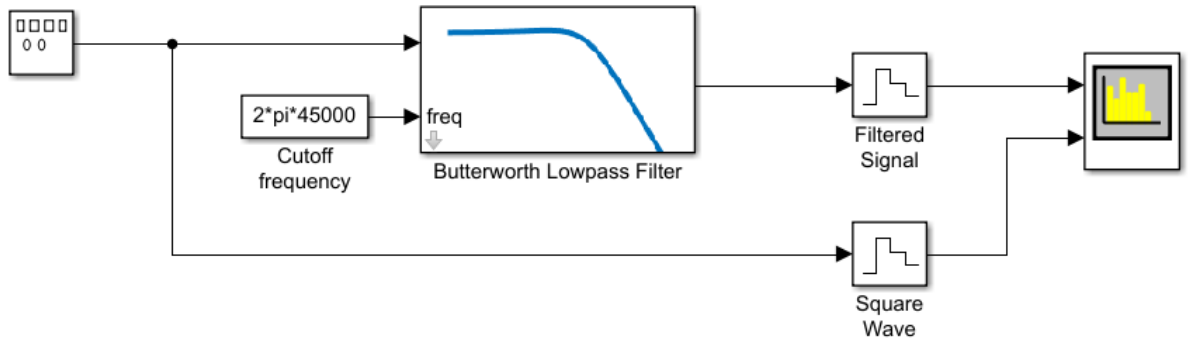


Figure 5.14: SIMULINK simulation of the reconstruction filter

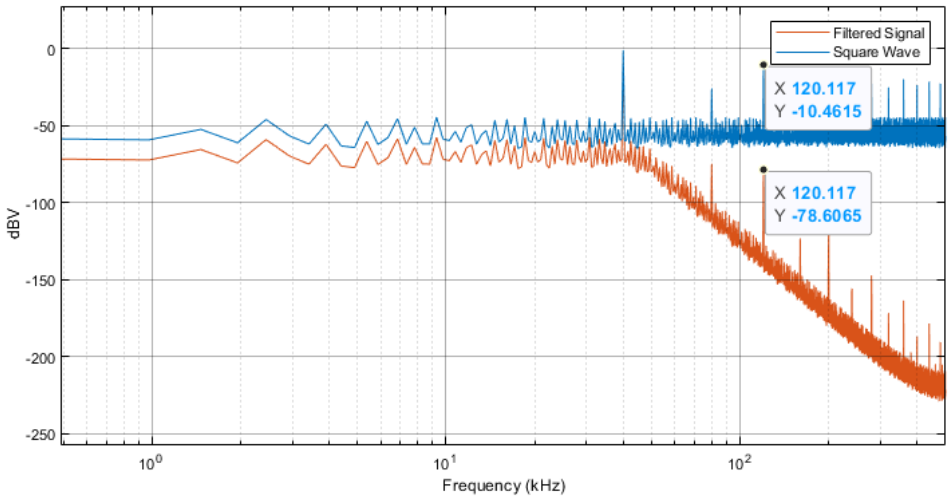


Figure 5.15: Spectrum of the square wave vs the filtered signal

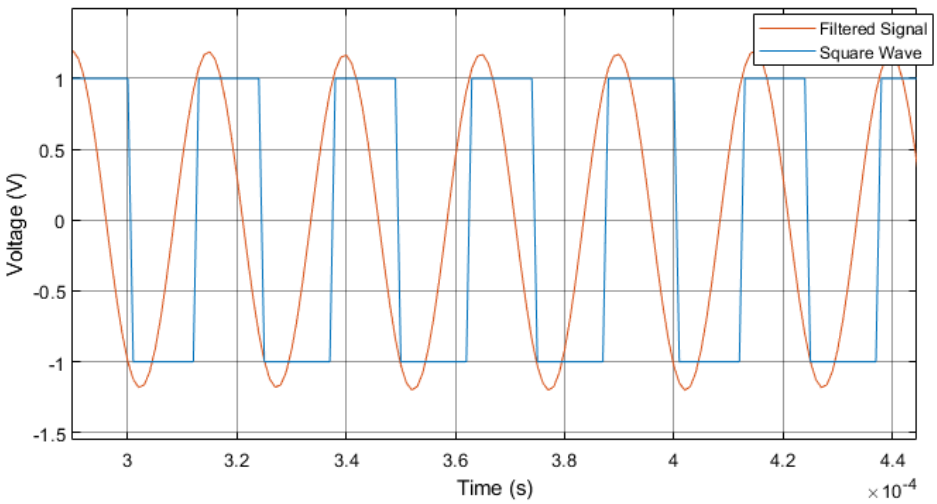


Figure 5.16: Square wave vs the filtered signal

5.3.3.3 Reconstruction Filter Implementation

The filter used in the circuit is the LTC1064-2 8^{th} order Butterworth filter with adjustable cutoff frequency. The maximum attainable attenuation on the filter is 80 dB which for the application of this study is sufficient. The cutoff frequency is adjusted by a clock signal. The ratio of the clock frequency to the cutoff frequency is either 50:1 or 100:1. The specifications of the filter have are shown in table 5.5.

Technical Specifications	
Filter type	Lowpass Butterworth
Package type	14 pin PDIP
Maximum cutoff frequency	140 kHz
Clock to cutoff frequency ration	50:1 or 100:1
Maximum THD	0.03%
Operating input voltage range	±2.37V to ±8V

Table 5.5: Technical specifications of LTC1064-2 filter

The clock signal is also generated on the Teensy 4.0. It, however, cannot use the tone function like the 40 kHz square wave due to the clock frequency being much higher. The 50:1 ratio is used for the clock to cutoff frequency ratio to keep the frequencies generated from the Teensy 4.0 as low as possible because high-frequency square waves experience a lot of ringing on the Teensy 4.0. For a cutoff frequency of 45 kHz the clock frequency that is generated needs to

be 50 times higher which is 2.25 MHz. The square wave is manually generated from a PWM signal with a 50% duty cycle. The pin is configured in two steps. First, the frequency of the signal is set on the PWM enabled pin. The next step is to set the duty cycle of the pin. Figure 5.17 shows the clock frequency generated from the Teensy 4.0. The frequency is less accurate at high frequencies. As seen in figure 5.17, the average frequency is 2.239 MHz which gives a cutoff frequency of 44.8 kHz.

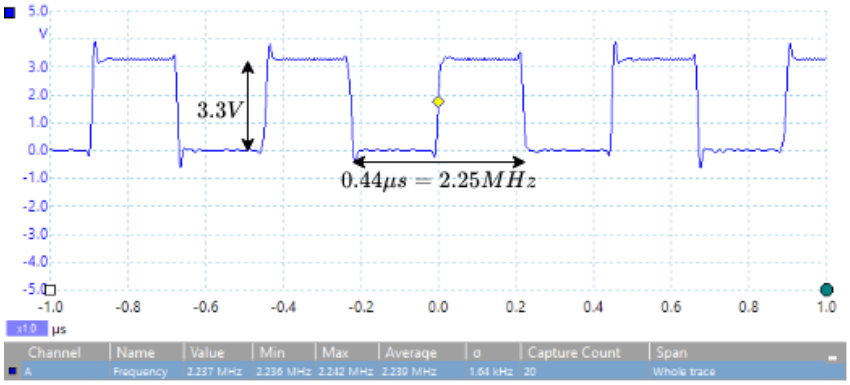


Figure 5.17: Clock signal for the filter generated by the Teensy 4.0 board

The frequency response of the LTC1064-2 filter was obtained by sweeping the input frequency from 10 kHz to 100 kHz. The frequency response is shown in figure 5.18. Substituting -40 dBV and 80 kHz into equation 5.6 shows us that the filter is in fact, an 8^{th} order filter.

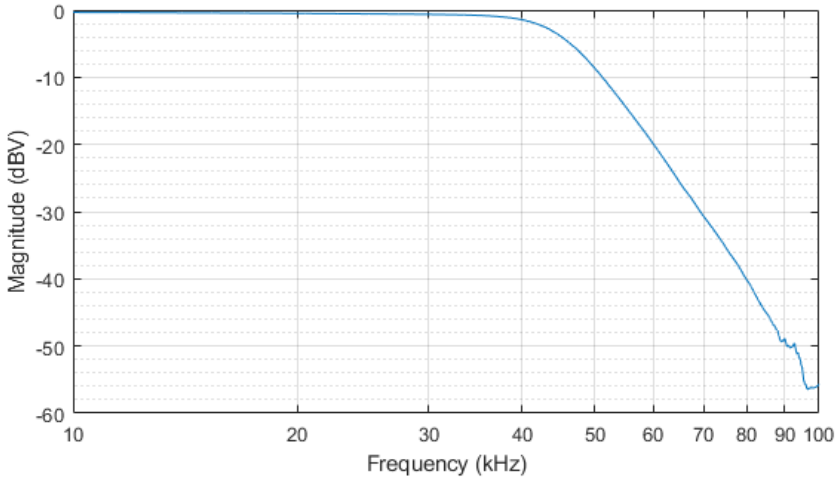


Figure 5.18: Frequency response analysis of the LTC1064-2

To test the filter, a 40 kHz square wave and 2.25 MHz clock signal was generated on the Teensy 4.0 and fed into the filter. The filter was powered with a ±8V power supply and the clock to cutoff frequency ratio was set to 50:1. As seen in figure 5.19, the harmonics present in the square wave has been successfully filtered out. Figure 5.20 shows the square wave generated

from the Teensy 4.0 has been successfully reconstructed into a sine wave.

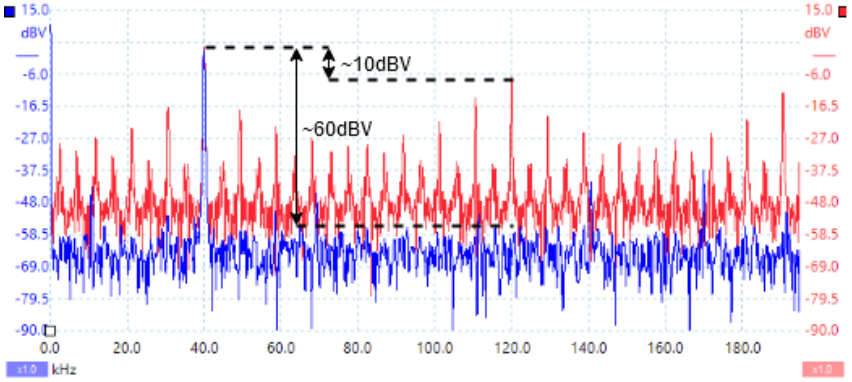


Figure 5.19: Frequency response of the square wave(red) vs the filtered signal(blue)

As seen in figure 5.20, there is still a DC component present in the filtered signal. This can be removed by using an AC coupling capacitor shown in figure 5.21. The cutoff frequency of the AC coupling capacitor circuit is determined by equation 5.7. Choosing a resistor value of 10 kΩ and a capacitor value of 100 nF allows frequencies above 159 Hz to pass the AC coupling capacitor.

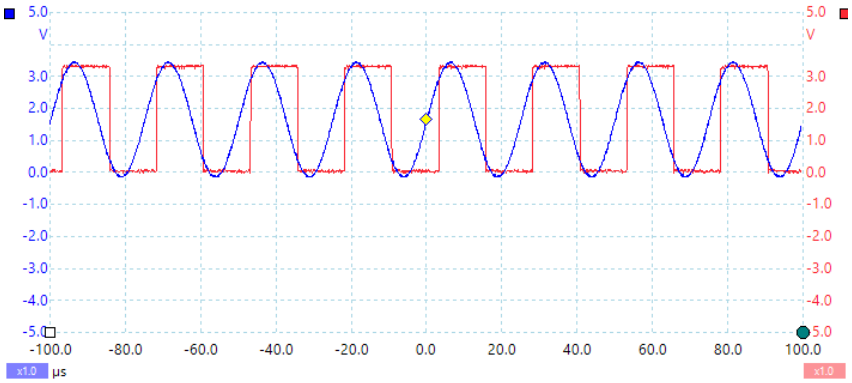


Figure 5.20: Square wave(red) generated from the Teensy 4.0 vs the filtered signal(blue)

$$f_c = \frac{1}{2\pi RC} \tag{5.7}$$

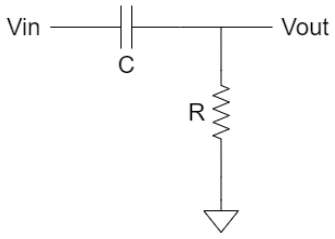


Figure 5.21: AC coupling capacitor

5.3.3.4 Signal Amplification Requirements

The system needs to be able to pick up gestures from up to 2 metres away. To ensure that the transmitter has enough power, an audio operational amplifier is used to amplify the signal. An audio amplifier was chosen because it has low noise characteristics. The amplifier is configured as a non-inverting amplifier with an adjustable gain.

There are two parameters to consider when choosing the correct operational amplifier which are the slew rate and the gain-bandwidth product (GBW).

The slew rate is the rate of change of the output signal. The slew rate required can be calculated using equation 5.8.

$$SlewRate = 2\pi \times f \times V_{pp} \times 10^{-6}\ V/\mu s \tag{5.8}$$

Since the system has a maximum operating voltage of ±8V the peak to peak voltage is 16V. This yields a minimum slew rate requirement of 4.02 V/µs with a 40 kHz signal.

$$SlewRate = 2\pi \times 40000 \times 16 \times 10^{-6} = 4.02\ V/\mu s$$

The GBW is the maximum closed-loop gain that the operational amplifier can achieve at The bandwidth of the system. The GBW required can be calculated using equation 5.9. Typically the GBW should be ten times higher than the required value.

$$GBW = 10 \times A_{cl} \times B\ Hz \tag{5.9}$$

Since the maximum possible peak to peak voltage is 16V with a peak to peak input voltage of 3.3V, the maximum linear gain that can be achieved is 4.84. This yields a minimum GBW requirement of 1.94 MHz.

$$GBW = 10 \times 4.84 \times 40000 = 1.94\ MHz$$

5.3.3.5 Signal Amplification Implementation

The operational amplifier used is the LM833 audio operational amplifier. The amplifier has a minimum slew rate of 5 V/µs and a minimum GBW of 10 MHz which is sufficient for the use case of this study. The LM833 can also output a current of 29 mA which is sufficient to drive the ultrasonic transmitter. The specifications of the LM833 are shown in table 5.6.

Technical Specifications	
Maximum supply voltage	±18V
Typical output current	29 mA
Minimum slew rate	5 V/µs
Minimum GBW	10 MHz
Noise voltage	4.5 nV/$\sqrt{Hz}$
THD	0.002%

Table 5.6: Technical specifications of LM833 audio operational amplifier

The adjustable gain on the system is set using a potentiometer. The gain of a non-inverting is set using equation 5.10. Figure 5.22 shows the setup for a non-inverting amplifier. R1 is 10 kΩ and R2 is a 100 kΩ potentiometer giving a maximum attainable linear gain of 11. As this is not a rail to rail operational amplifier, the output voltage of the amplifier cannot reach that of the supply voltage. The LM833 can reach ±6 V when powered with ±8 V. Outputs voltages larger than ±6 V will simple be saturated at ±6 V.

$$Gain = \left(1 + \frac{R2}{R1}\right) \tag{5.10}$$

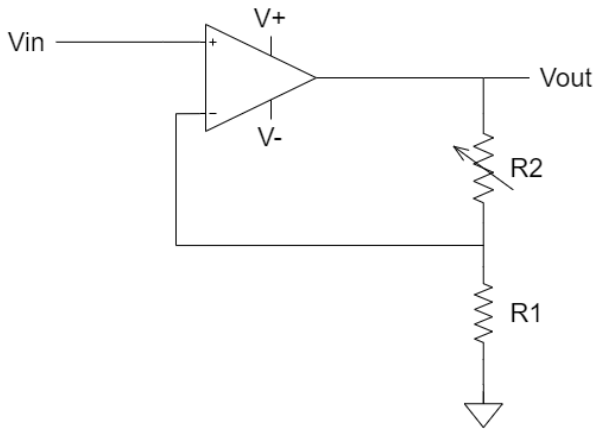


Figure 5.22: Non-inverting amplifier

Figure 5.23 shows the filtered signal with the DC component removed and amplified to the point before saturation. This is the signal that will be fed into the ultrasonic transmitter. The ultrasonic transmitter has an operating voltage of ±10 V so an input voltage of ±6 V falls into an acceptable operational range.

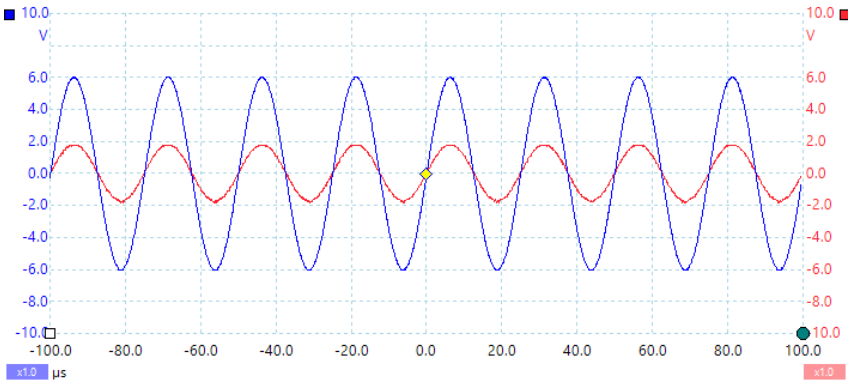


Figure 5.23: Filtered signal(red) amplified to the point right before saturation(blue)

5.3.4 Receiver Circuit Design

The purpose of the receiver circuit is to take the echoes of the propagated signal, modify it to the desired specifications and then send it to the microcontroller where it is sampled. A DC biasing need to be applied on the received signal since the microcontroller can not sample negative signals. The received signal is then amplified and clipped as to not damage the microcontroller. The signal is then put through an anti-aliasing filter before it is fed into the ADC. This flow is illustrated in figure 5.24. As previously mentioned, three receivers are required for the system sense gestures in three dimensions. Thus the receiver circuit will have three copies of the same circuit.

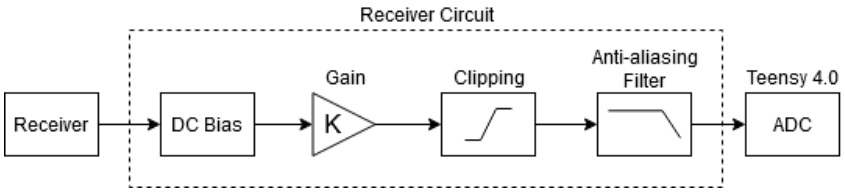


Figure 5.24: Block diagram illustration the operation of the receiver circuit

5.3.4.1 Signal Conditioning

It is expected that the sound pressure level of the echoes received from the transmitted signal is going to be lower than that of the actual transmitted signal due to the target reflectivity and

propagation losses. As such, the echoes received by the ultrasonic receivers need to be amplified. Any signals going to the Teensy 4.0 needs to be between then range of -0.5 and 3.6 V. Signals with levels outside this range will damage the Teensy 4.0. The ADC will only sample levels in the range of 0 to 3.3 V. Values outside of this range will be saturated.

Since the Teensy 4.0 cannot read negative voltages, the signal needs to be biased to half of the maximum voltage of the ADC which is 1.65 V. This can be done by coupling the AC component of the received signal to a resistor voltage divider network as shown in figure 5.25. The capacitor couples the AC signal to the circuit and removes any DC components from the signal. Equation 5.11 shows how to calculate the resistor divider network. The resistor divider network is set to half of Vcc by using the same resistance for R1 and R2.

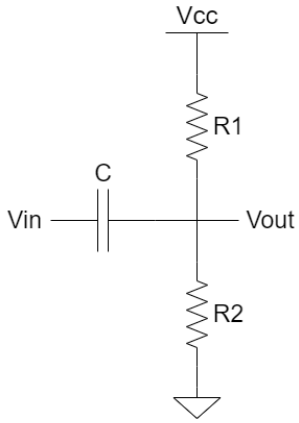


Figure 5.25: DC Biasing circuit for AC circuits

$$V_{out} = V_{cc} \times \frac{R2}{R1 + R2} \tag{5.11}$$

The DC biasing circuit was first simulated in LTSpice using 10kΩ resistors and a 100nF coupling capacitor. The simulation result is seen in figure 5.26. The input signal is forced to oscillate about 1.65 V. The circuit was then tested and the results are shown on figure 5.27. Figure 5.27 shows the blue input signal with no DC biasing has been biased by 1.65V as seen on the red output signal.

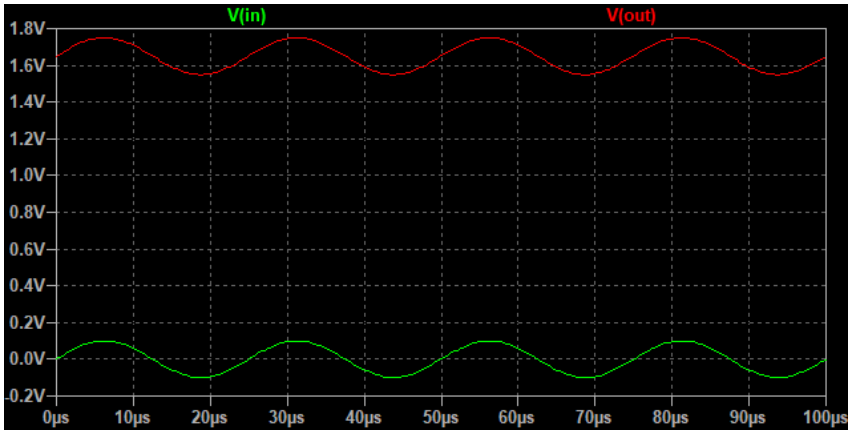


Figure 5.26: DC Biasing circuit simulated in LTSpice

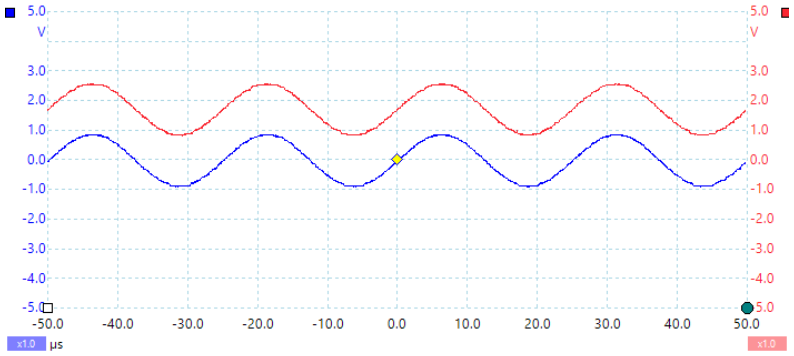


Figure 5.27: DC Biasing circuit with unbiased input signal(blue) and biased output signal(red)

Once the signal has been biased, it needs to be amplified so that the ADC can use as much as the usable range as possible. The amplifier needs to only amplify the AC component of the signal since there is the correct DC bias has already been applied on the signal. Additionally, the amplified signal has to be within the range of 0 to 3.3 V. To achieve this, a rail to rail operational amplifier configured in AC coupled non-inverting amplifier mode is used. The AC coupled amplifier will only amplify the AC component of the signal which maintaining the same bias voltage level. The rail to rail operational amplifier powered by 3.3 V and ground will ensure that the output of the signal will never be outside the range of 0 to 3.3 V while being able to reach that range unlike the audio operational amplifier used in the transmitter. The circuit was simulated in LTSpice as shown in figure 5.28. Examining figure 5.29, it is shown that the unbiased input signal is amplified and biased to the right voltage level with the values that are out of the 0 to 3.3 V range saturated.

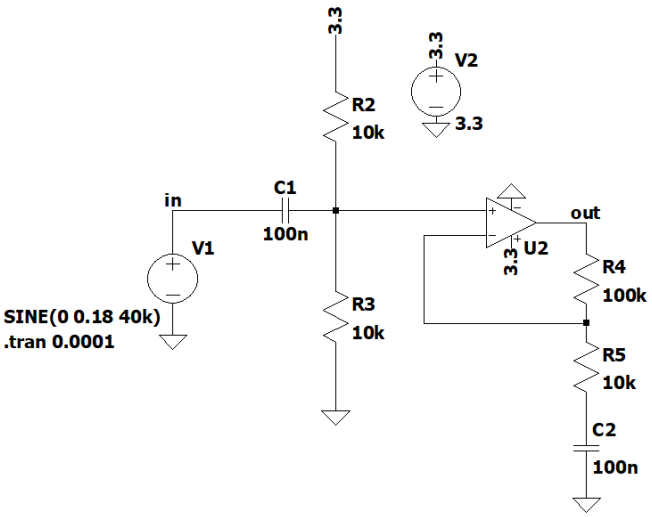


Figure 5.28: AC coupled non-inverting amplifier with DC biasing LTSpice circuit

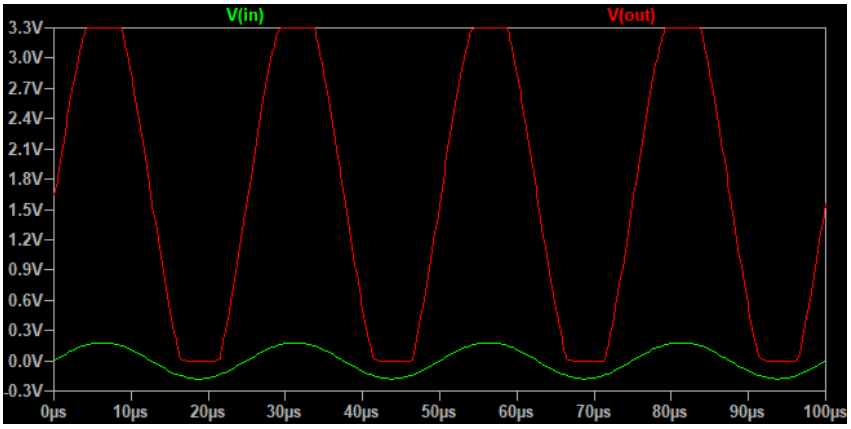


Figure 5.29: AC coupled non-inverting amplifier with DC biasing simulation output

Similar to the audio operational amplifier, the slew rate and GBW need to be calculated. The highest frequency of interest is 42 kHz which is the upper limit of the bandwidth for the ultrasonic receivers. The same resistors and potentiometers used in the transmitter circuit are used in the receiver circuit so the maximum achievable linear gain is 11. This yields a slew rate of 0.87 V/µs and a GBW of 4.62 MHz.

$$SlewRate = 2\pi \times 42000 \times 3.3 \times 10^{-6} = 0.87\ V/\mu s$$

$$GBW = 10 \times 11 \times 42000 = 4.62\ MHz$$

The rail to rail operational amplifier used is the MCP6292. It has a slew rate of 7 V/µs and a

GBW of 10 MHz, which is sufficient for this study. The circuit was built and tested and the results are shown in figure 5.30. The figure shows that the blue unbiased input signal has been biased to 1.65 V and amplified. It is also seen that the voltages above 3.3 V and below 0 V have been clipped off as designed.

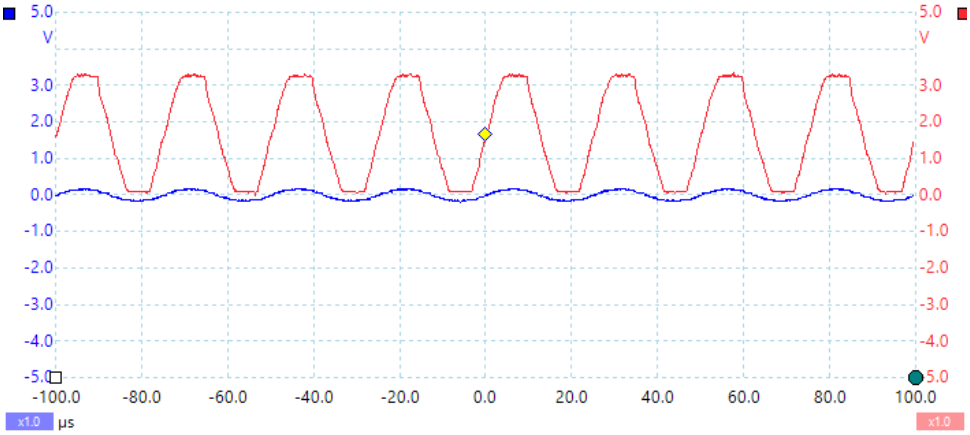


Figure 5.30: DC Biasing circuit simulated in LTSpice

5.3.4.2 Anti-Aliasing Filter Requirements

In the signal sampling section, it was discussed that the sampling frequency of the ADC needed to be higher than twice that of the highest frequency to prevent the signal of interest from aliasing. A sampling frequency of 200 kHz was decided upon. There is however still a possibility that signals higher than 100 kHz may be present in the system which will cause aliasing. For this reason, an anti-aliasing filter is required in the receiver circuitry. As seen in equation 5.4 from the reconstruction filter section, around 72 dBV is needed to ensure that any aliased signals are not present to the ADC. Using equation 5.6 from the reconstruction filter section we need an 11^{th} order to filter out any signals that may cause aliasing. If the frequency response of the receiver is included in the calculation, which is approximately -30 dBV at 100 kHz, only 42 dBV of attenuation is required resulting in a 7^{th} order filter.

$$n = \frac{log(\sqrt{{A_{min}}^{-2} - 1})}{log\left(\frac{\omega_s}{\omega_c}\right)}$$

$$n = \frac{log(\sqrt{10^{\frac{-72}{20}^{-2}} - 1})}{log\left(\frac{100000}{45000}\right)}$$

$$n = 10.38 \approx 11$$

$$n = \frac{log(\sqrt{{A_{min}}^{-2} - 1})}{log\left(\frac{\omega_s}{\omega_c}\right)}$$

$$n = \frac{log(\sqrt{10^{\frac{-42}{20}^{-2}} - 1})}{log\left(\frac{100000}{45000}\right)}$$

$$n = 6.05 \approx 7$$

5.3.4.3 Anti-Aliasing Filter Simulation

A SIMULINK simulation of an 8^{th} order filter with white noise as an input was simulated. The result of the simulation is shown in figure 5.31 which show a total of 55 dBV of attenuation at 100 kHz.

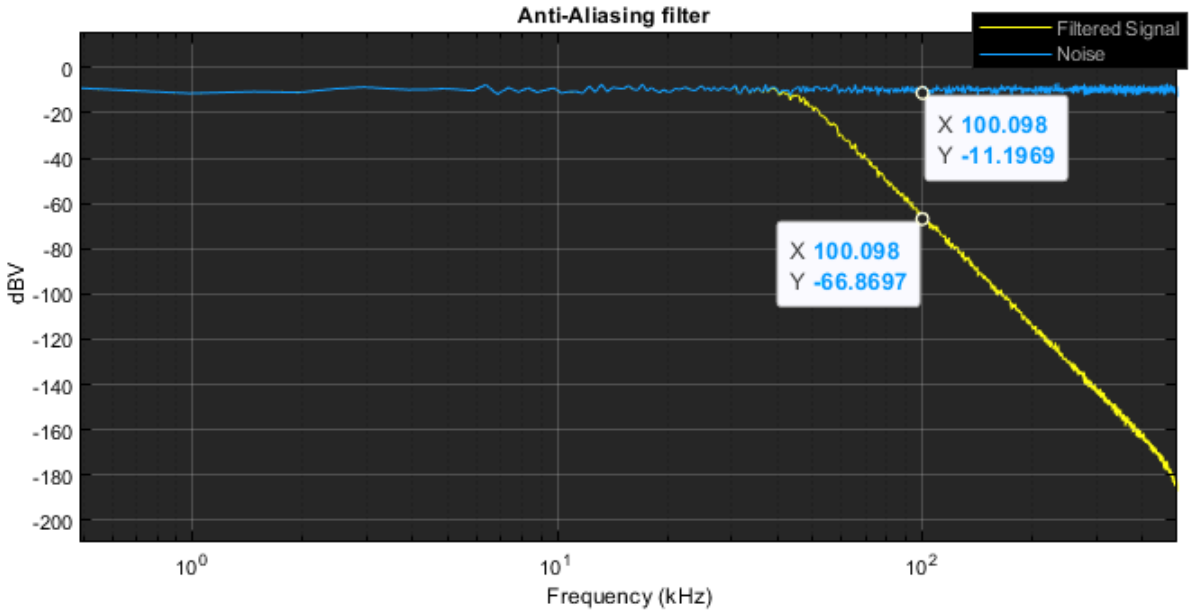


Figure 5.31: Caption

5.3.4.4 Anti-Aliasing Filter Implementation

The same 8^{th} order Butterworth filter used in the reconstruction filter is used here in the anti-aliasing filter. A 100 kHz signal was injected into the filter. The results are shown in figure 5.32. The 100kHz is attenuated by approximately 51 dBV by the filter this is satisfactory for this study.

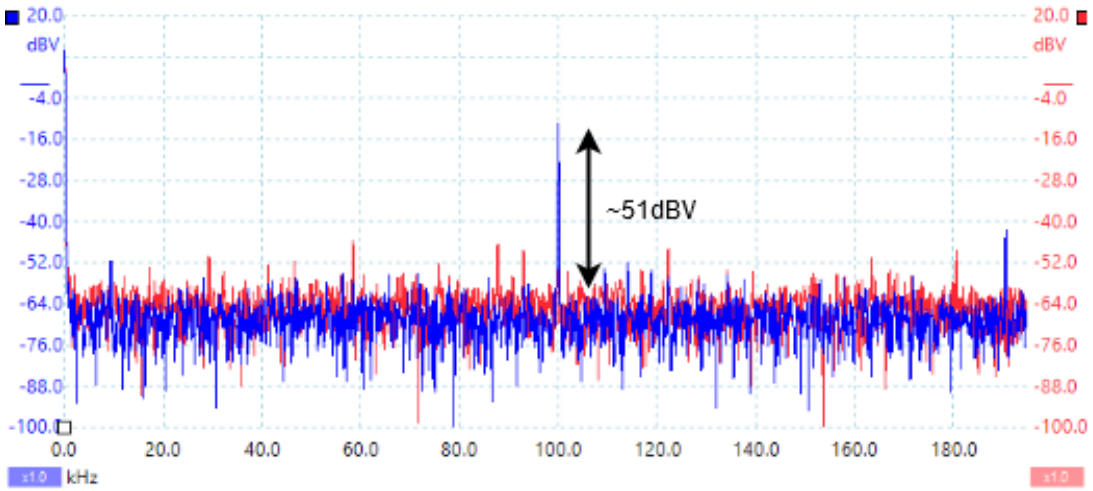
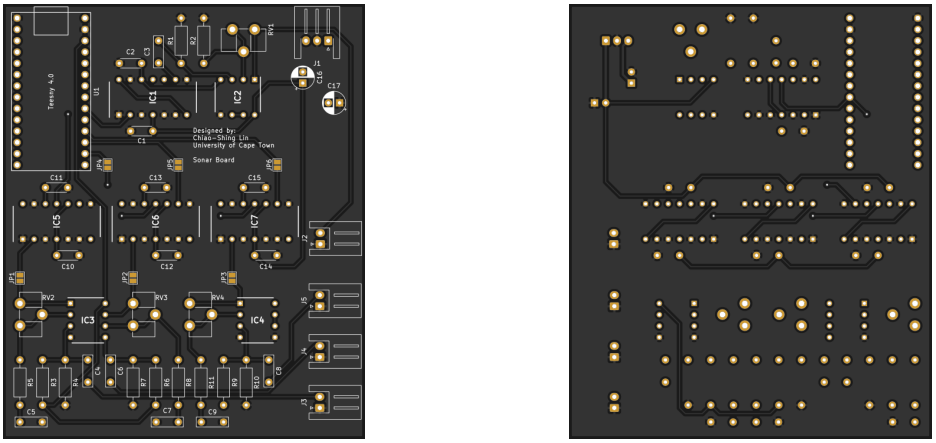


Figure 5.32: Anti-alias filter output

5.3.5 PCB Design

The circuit schematic for the sonar PCB board can be found in appendix A. The PCB was designed using KiCAD. The components used for this design are all through-hole components to allow for easier debugging.



(a) Top view of PCB **(b)** Bottom view of PCB

Figure 5.33: PCB

The transmitter section of the PCB was tested by probing the output of the transmitter circuit with a digital oscilloscope and analysing whether the signal going to the output is as expected. The receiver section of the circuit was tested by injecting a 1 V, 40 kHz sine wave at the input and probing the output of the circuit to analyse whether the signal going to the output is as expected. The results of the testing can be found in appendix B

5.3.6 Sensor Mount Design

As discussed in section 5.3.2, there are three receivers and one transmitter in the system. The transmitter is placed in the centre with the receivers on the vertices of an equilateral triangle. The receivers have a beamwidth of 80 degrees, 40 degrees on either side, as discussed in section 5.1. The receivers are angled 20 degrees away from the transmitter. This ensures that the normal of each of the receivers are in a different plane while still having overlapping beams. An angle of 20 degrees was arbitrarily chosen as the goal of this study is to be able to classify hand gestures and not to detect the precise position of the hand. The mechanical drawing of the sensor mount can be found in appendix C.

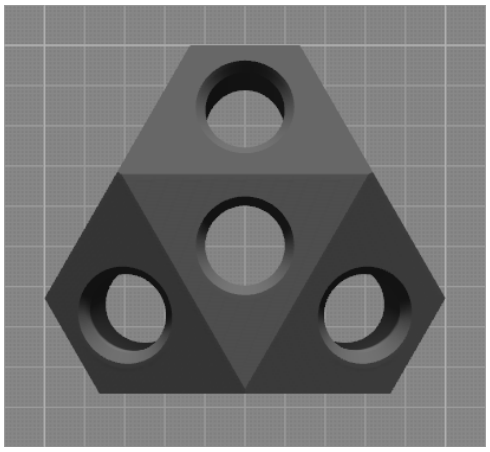


Figure 5.34: Sensor Mount

5.3.7 Fully Assembled Hardware

Figure 5.35 shows the sonar data acquisition hardware fully assembled.

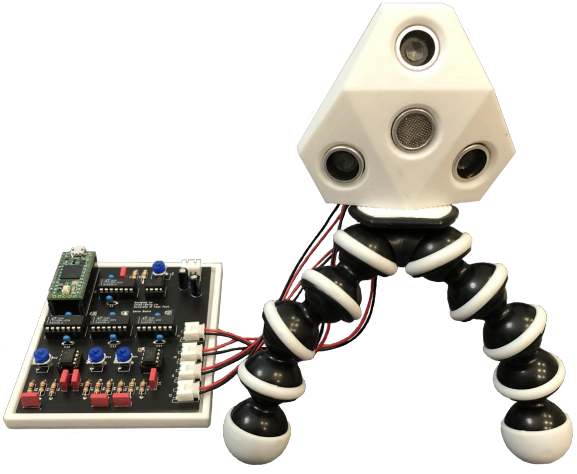


Figure 5.35: Fully Assembled Hardware

5.4 Data Logging

The microcontroller needs to receive commands so that it knows when to begin transmitting a signal and when it needs to start sampling data. For this system, the 480 Mbit/s baud rate USB port on the Teensy 4.0 will be used to communicate between the computer and the microcontroller. MATLAB is used to send the serial commands to the Teensy 4.0 and to receive the sampled data from the Teensy 4.0. The design of the data logging subsystem is linked to the other subsystems as illustrated in figure 5.36.

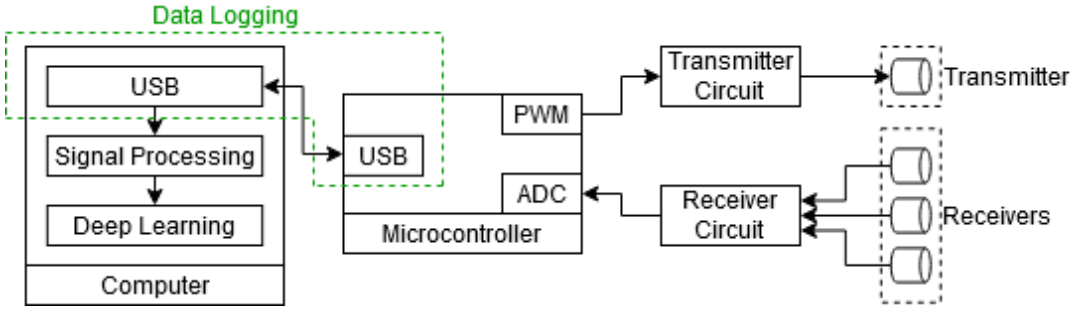


Figure 5.36: Data Logging overview

The data logging sequence begins in MATLAB. The sampling frequency and duration of the recording is first defined. These values are used to calculate how many data points need to be sampled per channel. In the case of this system design, the sampling frequency is set to 200 kHz. A serial connection between MATLAB and the microcontroller is then established with a baud rate of 480 Mbit/s. The data structure of the USB protocol includes a start bit, 8 data bits, an optional parity bit, and up to 2 stop bits for a maximum of 12 bits. Knowing that the 3 ADC channels are sampling 12 bits at 200 kHz, the minimum baud rate required would be 14.4 Mbit/s. This maximum baud rate of the microcontroller is more than 30 times larger than the minimum required baud rate so the USB interface should not be a bottleneck.

Once the connection has been established, a string command of how many seconds the microcontroller needs to transmit and record are sent to the microcontroller. The microcontroller reads this value and calculates how many data points to sample and send to the computer. The microcontroller then starts transmitting the signal and begins to sample the 3 channels at 200 kHz. Each of the values sampled from each channel is stored in a 16-bit array with a size of 3. This array is transformed into an 8-bit array of size 6 since USB can only send 8-bit values. These 6 values are then sent to a MATLAB array. MATLAB waits until the expected amount of data points from all three channels have been sent to the PC. MATLAB will timeout if it doesn't receive the expected amount of data. MATLAB can read the values sent from the

microcontroller as 16 bits so the received 8-bit data does not need to be recombined into 16-bit data.

The received data is split up into 3 separate arrays, one for each channel. These 3 arrays are stored into a MATLAB cell array and the corresponding label assigned to the specific gesture performed is stored in a MATLAB categorical array. These 2 arrays are then saved as a MATLAB workspace.

The operational flow of the data logging system is illustrated in figure 5.37.

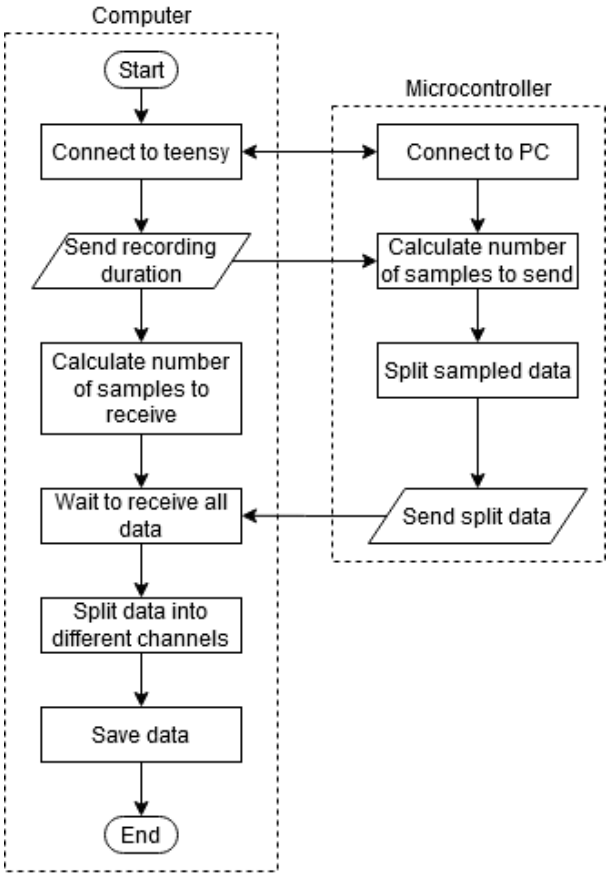


Figure 5.37: Data Logging operational flow

5.5 Signal Processing

The signal sampled by the data logger is processed into a form that is more meaningful that a raw signal. The signal processing happens after the data has been sampled and before the deep learning stage is illustrated by figure 5.38.

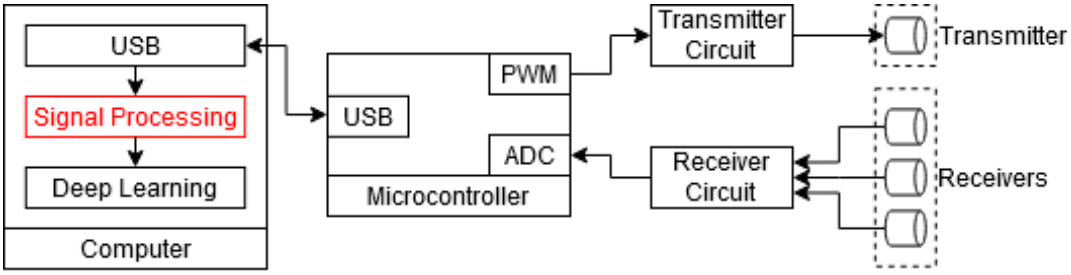


Figure 5.38: Signal processing overview

The signal processing is implemented using MATLAB. The raw data of the gesture recordings are first loaded into the MATLAB workspace. Iterating through each recording, the three received channels are filtered by a notch filter centred at 40 kHz to ensure that the Doppler echoes are stronger than stationary clutter. The stationery clutter is represented by 40 kHz echoes. The notch filter filters out velocities less than 0.1715 m/s.

The filtered signals are multiplied by the complex exponential of 40kHz to shift the signal to baseband. Doing this, however, also shifts a copy of the signal to 80 kHz so the base banded signal is filtered by a low pass filter to remove the 80 kHz copy. The signal before and after the baseband conversion are shown in figure 5.39.

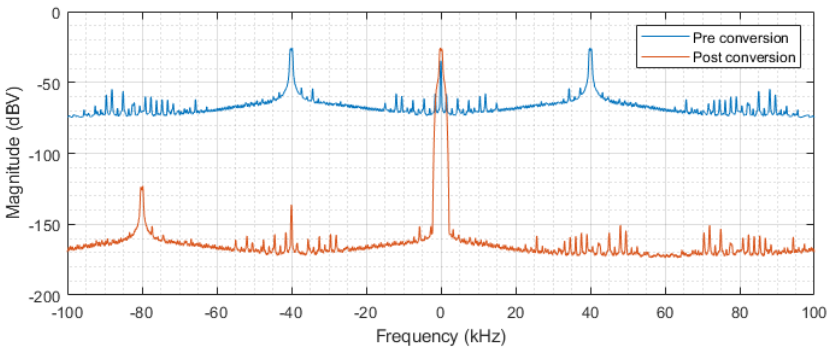


Figure 5.39: Base band conversion and filtering

The signal is then down-sampled since the signal is now at baseband and the frequencies present are lower than the original signal. This reduces the overall size of the data while retaining all the important features of the raw signal. The down-sampled base banded data can be included in the data logging section to reduce the amount of data that is saved. Figure 5.40 shows that

all the important information is still retained after the signal has been downsampled. The effect of the notch filter is also present.

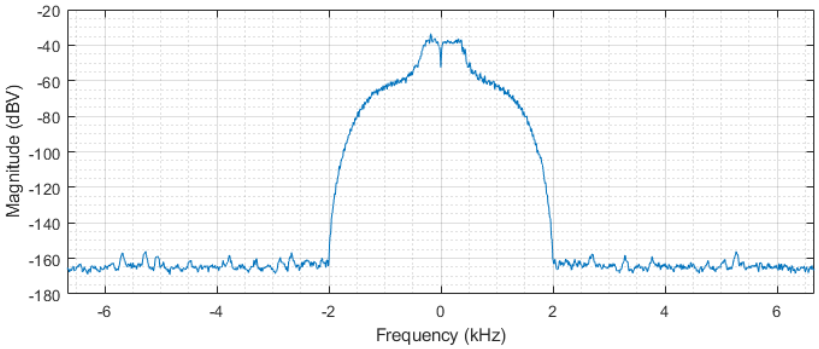


Figure 5.40: Signal down sampling

The time-frequency representation of the signal can be extracted by creating a spectrogram of the signal. The spectrogram is a three-dimensional plot showing the power spectrum of the data as a function of time. The "pspectrum" function in MATLAB was used to create the spectrogram with a frequency resolution of 128 Hz with 50% overlap and a coherent processing time of 9.5995 ms. Almost all the papers in literature end the signal processing at this stage. For testing purposes, the gesture used in this scenario was a hand waving left and right repeatedly in front of the sensors. This gesture can be seen represented by the spectrogram in figure 5.41.

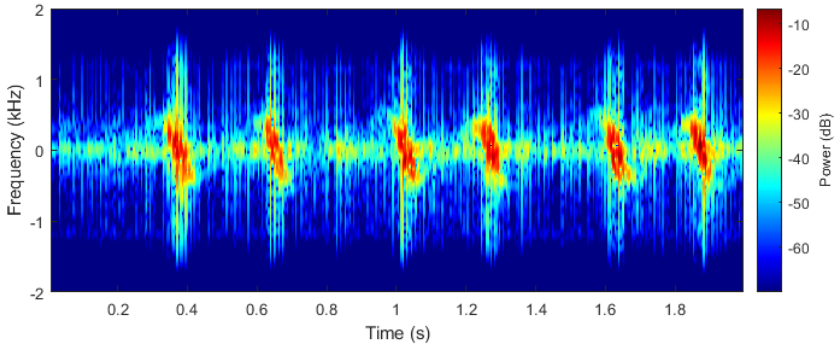


Figure 5.41: Spectrogram data of gesture

The median frequency of the power spectrum for each time instant of the spectrogram is computed. With the stationary clutter attenuated by the notch filter, the Doppler frequencies should be more prevalent in the spectrum meaning the median frequency that is computed for a time instance should correspond to the Doppler frequency present in the signal. This extracted frequency plot is filtered with a moving mean filter with a window size of 10. An example of the median frequency extracted from each channel from the spectrograms is shown in figure 5.42. One of the reasons for extracting the median frequency from the spectrogram is that it reduces the overall dimension by one which decreases the size of the data that is used for deep learning.

One downside to this method is that since it only extracts one frequency, it would not work on a system where multiple moving targets moving at different velocities are present in the signal.

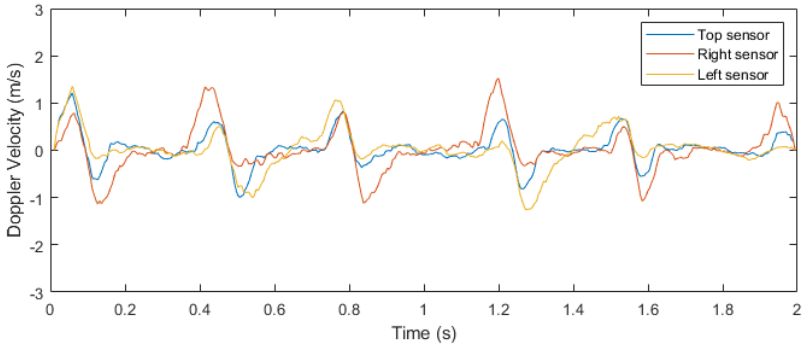


Figure 5.42: Doppler frequency from each of the receivers

Once the three frequency channels have been computed, three differential frequency channels are computed. This being the difference between channel one and two, one and three, and two and three respectively. These differential channels represent the subtle time and magnitude differences between the different channels as an amplitude. The three Doppler frequency channels and the differential frequency channels are saved in an array and saved in a MATLAB workspace. The three differential channels are illustrated in figure 5.43.

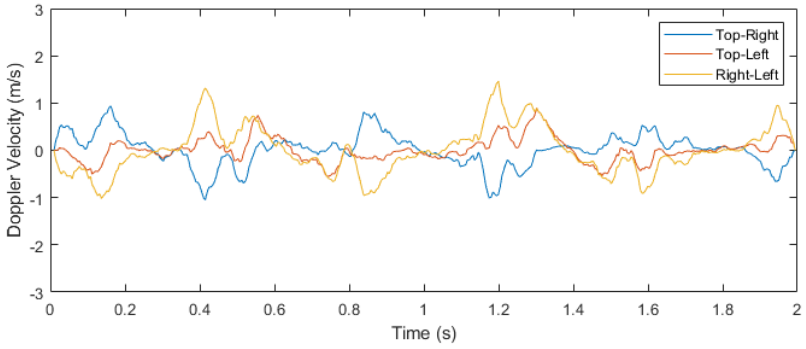


Figure 5.43: Doppler frequency differential channels

The operational flow of the signal processing in illustrated in figure 5.44.

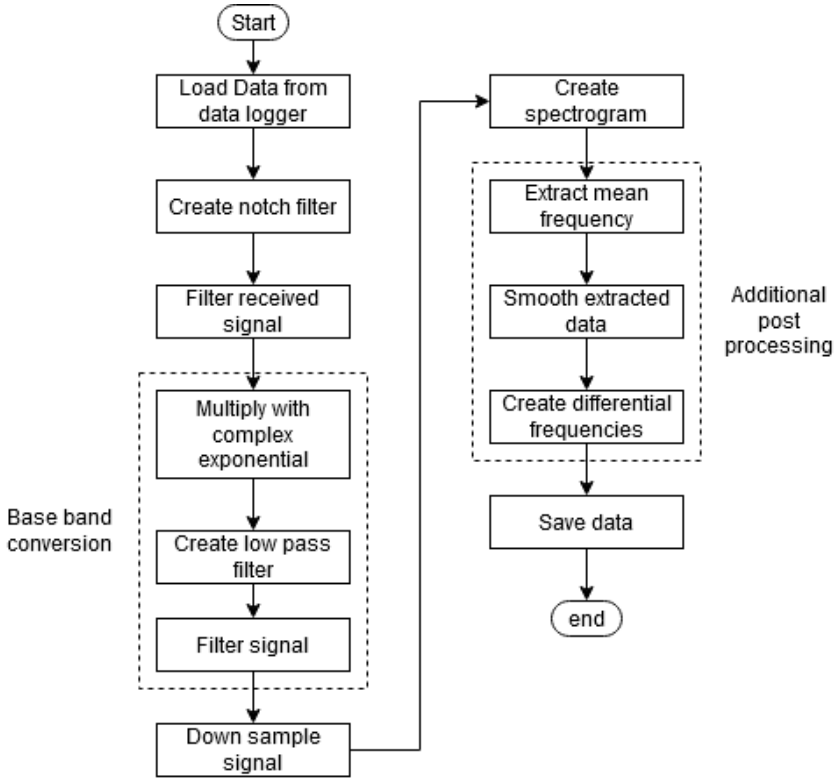


Figure 5.44: Signal processing operational flow

Chapter 6: Experimental Setup and Data

This chapter details the geometry of the experimental setup with the specifications of the computer used to collect and process all the data. A breakdown of the specific details of each of the data sets that were used in each experiment is also included in this chapter.

6.1 Experimental Setup

The circuits were powered using a dual supply bench power supply and connected via USB to the PC got communication with the microcontroller. The sensors were placed on a desk in front of the user with the user sitting on a chair at the same desk. The gestures recorded were withing half a meter between the seated user and the sensors with the user's body in the direct line of sight of the transmitter illustrated by figure 6.1. The physical setup can be seen in figure 6.2. Additional gestures were recorded at a distance of one metre to test the range performance of the system.

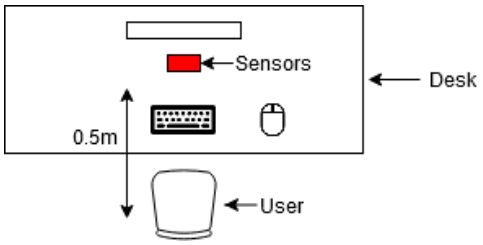


Figure 6.1: Illustration of experimental setup

The specifications for the computer used for the data logging, signal processing and deep learning is shown in table 6.1. The deep learning networks were trained using the GPU.

Computer Specifications	
CPU	AMD Ryzen 5 2600X Six-Core Processor @ 3.6 GHz
RAM	16 GB DDR4 @ 2666 MHz
GPU	Nvidia GeForce GTX 1060
GPU VRAM	6 GB

Table 6.1: Computer Specifications

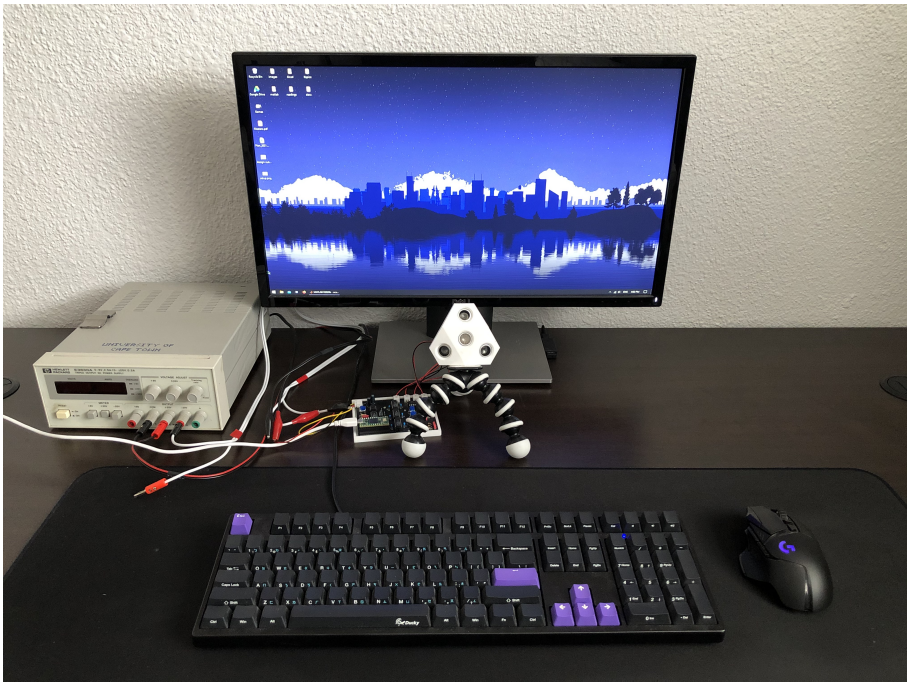


Figure 6.2: Experimental setup

6.2 Data sets

The gestures included in the data set are the gestures shown in table 3.1 that were defined in the requirements. 6 different data sets are being used to test the system. The specific details of the data sets are shown in table 6.2.

Set no.	Hand	Separate labels for each hand	Distance	Circular gestures
1	Right	No	0m-0.5m	No
2	Right	No	0m-0.5m	Yes
3	Right	No	0.5m-1m	No
4	Right	No	0m-1m	No
5	Both	No	0m-0.5m	No
6	Both	Yes	0m-0.5m	No

Table 6.2: Gesture data sets used for testing

Each gesture is recorded 300 times for each data set in table 6.2 for a total of 15000 individually recorded gestures. Gestures on different hands are seen as different gestures. The recordings are divided into 3 equal subsets; two sets for training and validation and one for testing.

Chapter 7: Deep Learning Design

After signal processing operations have been performed on on the raw data, it is in a form that is ready to used to train the deep learning models. This section describes the design of the deep learning algorithms. Figure 7.1 shows how the deep learning subsystem is connected to the rest of the system.

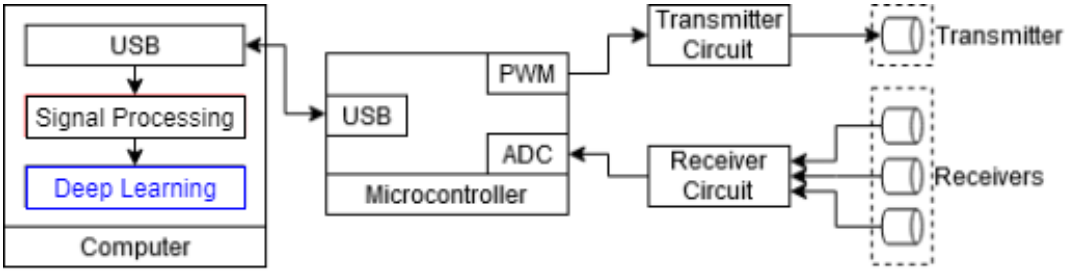


Figure 7.1: Deep Learning overview

7.1 Long Short Term Memory (LSTM)

The first network to be investigated is based on the Doppler bi-LSTM network proposed in the work done by Shrestha *et al.* to classify human activities using an FMCW Radar. The proposed network takes spectrograms as inputs and consists of 2 stacked LSTM layers with the second layer being a bi-LSTM layer. Stacking layers allow the network to identify higher-level abstractions from the input data. Stacking a bi-LSTM after the LSTM layer allows the network to search for forwarding temporal dependencies, as well as bidirectional temporal dependencies [1].

7.1.1 Network Structure

The network takes in 6 sequential inputs as there are 6 separate channels per gesture. The input is fed into the LSTM layer which searches for forward temporal dependencies. The outputs from the activations of the LSTM layer are fed into a batch normalisation layer. The batch normalisation layer normalises the outputs form an activation layer to stabilise the network and to prevent overfitting. The normalised outputs are then fed into the bi-LSTM layer which

searches for forwards and backwards temporal dependencies that were extracted from the LSTM layer. Once again, the outputs from the bi-LSTM layer are fed into a batch normalisation layer. The normalised output is fed into a fully connected layer which connects all the outputs from the normalisation layer to the Softmax layer. The Softmax layer computes the probability distribution of the data to predict which output class it might belong to. The final layer is a classification layer which takes the probability distribution from the Softmax layer and outputs a class label prediction. The network is illustrated in figure 7.2. The difference between this network and the Doppler bi-LSTM network is the change in the input data type from three dimensions to two dimensions as well as the introduction of batch normalisation layers.

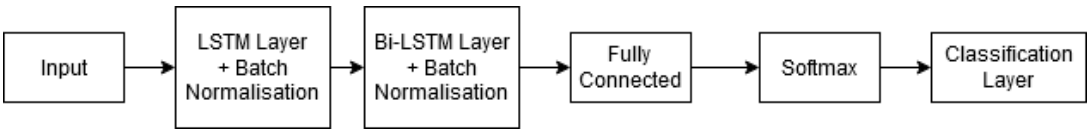


Figure 7.2: Bi-LSTM network structure

7.1.2 Hyperparameter Selection and Optimisation

7.1.2.1 Hidden Units

In an LSTM or bi-LSTM layer, the number of hidden units corresponds to the amount of information that is remembered each time step. The number of hidden units is set to the same length as the input to the network since all the gestures are single gestures instead of a combination of gestures such as in sign language. in the case of this experiment, the number of hidden units it set to 203. To verify this, the number of hidden units was swept from 1 to 300 using MATLAB's built-in Bayesian optimisation function. A mini-batch size of 32 was used due to system memory and power limitations.

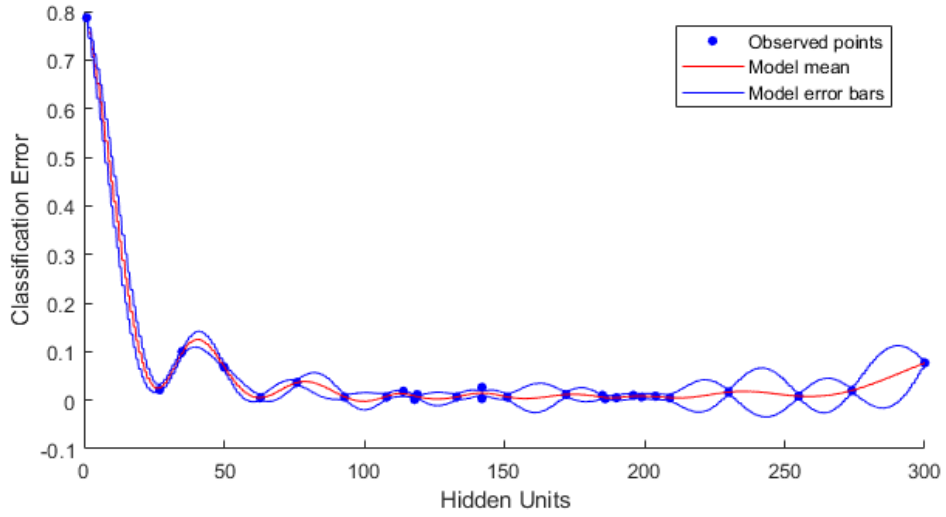


Figure 7.3: Bayesian optimisation result for the number of hidden units for the LSTM network

The Bayesian optimisation function takes in a variable that is to be optimised and the range of values which the optimiser should optimise the variable over. Data set number 1 from table 6.2 is used for the optimisation. The network is optimised by minimising the classification errors. The plot illustrating the observed points and the model of the learning rate against the classification error that was formed from the observed points is shown in figure 7.3. The observed points are points that were used by the optimiser to calculate the error. The model mean is the estimated learning rate versus classification error curve that was calculated using the observed points. The model error bar shows the computed uncertainty for the model mean. The classification error is minimised between a hidden unit count of 100 and 250 with a small range of uncertainty. This shows that the choice of 203 hidden units is viable.

7.1.2.2 Learning Rate

The learning rate determines how long you have to train the network for it to find the optimum solution. If the learning rate is too low, the network will take longer to train. If it is too high, however, the network will overshoot the optimal solution and won't converge to the optimal solution.

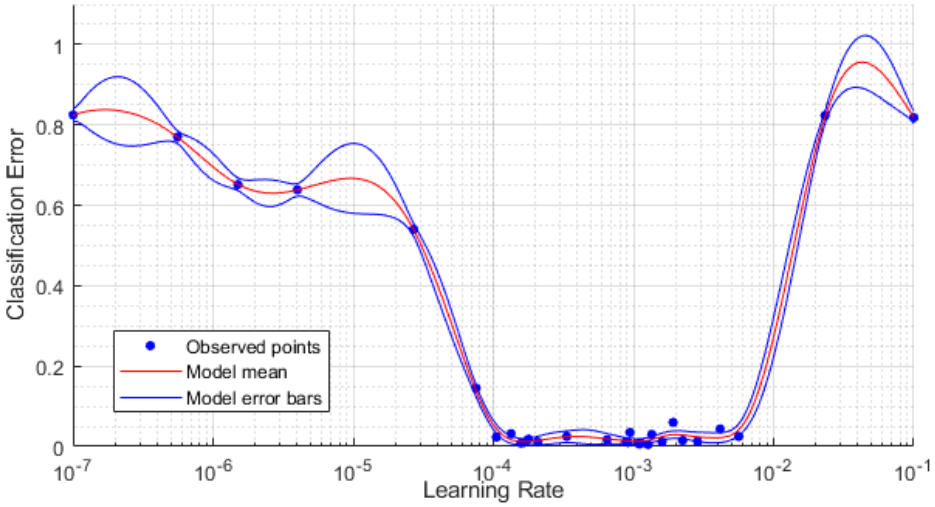


Figure 7.4: Bayesian optimisation result for the learning rate for LSTM network

The optimum learning rate for the network was searched for using MATLAB's built-in Bayesian optimisation function sweeping from learning rates between 10^{-7} to 10^{-1}. A mini-batch size of 32 was used due to system memory and power limitations.

Data set number 1 from table 6.2 is used for the optimisation. The network is optimised by minimising the classification errors. The plot illustrating the observed points and the model of the learning rate against the classification error that was formed from the observed points is shown in figure 7.4. The classification error is minimised between a learning rate of 10^{-4} and $5x10^{-3}$ with a small range of uncertainty. The best-observed learning rate and the best-estimated learning rate according to the model formed from the optimisation are shown in table 7.1. The best estimated model learning rate is used to train the LSTM network.

	Best Learning Rate	Error (%)
Observed	0.0012709	0.5
Model	0.0010949	1.2747

Table 7.1: Bayesian optimisation result for LSTM network

7.2 Epoch

The number of epochs was chosen so that the training would run long enough for the validation loss to settle. Figure 7.5 shows that the validation loss from the training process for the LSTM network has settled for a sufficient amount of time by 15 epochs.

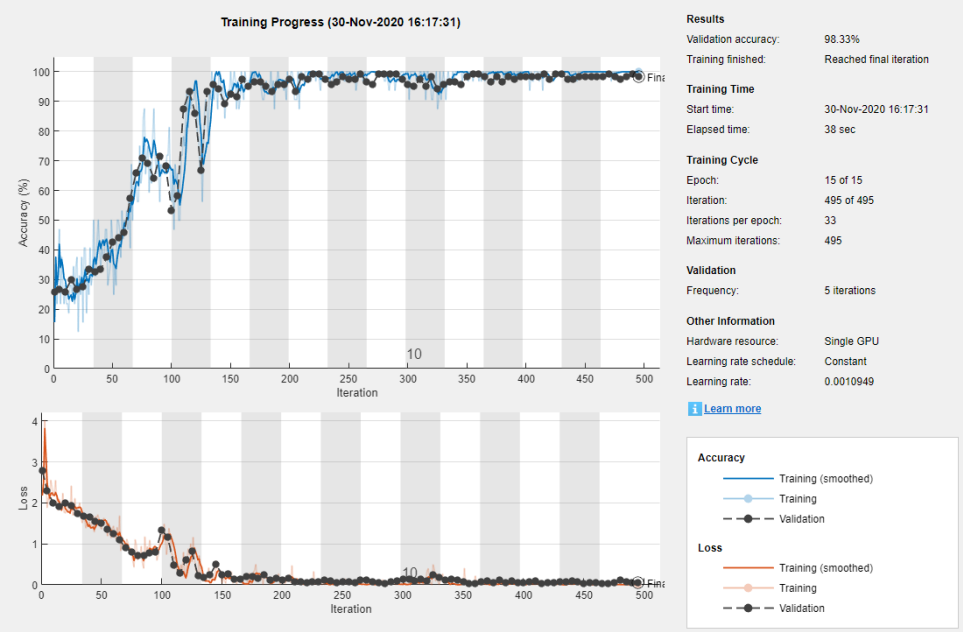


Figure 7.5: Training process of the LSTM network

7.3 Convolutional Neural Network (CNN)

The CNN used to compare the LSTM network discussed in the previous section is based on the DCNN algorithm used in the study by Kulhandjian *et al.* [2] to classify sign language gestures.

7.3.1 Network Structure

The network takes in the 6 different features in the form of an image with a height of 1 and a depth of 6 since CNN's work predominantly with image data. The network consists of 3 convolutional layers. Each of these layers proceeds with a batch normalisation layer, a rectified linear unit (ReLU) layer and finally a max-pooling layer. ReLU is an activation function that is often used in multi-layered CNN's. It is a non-linear activation function that greatly improves the computational throughput of the neural network when compared to traditional activation functions such as sigmoid functions and hyperbolic tangent functions. The max-pooling layers have a max-pooling size of 2.

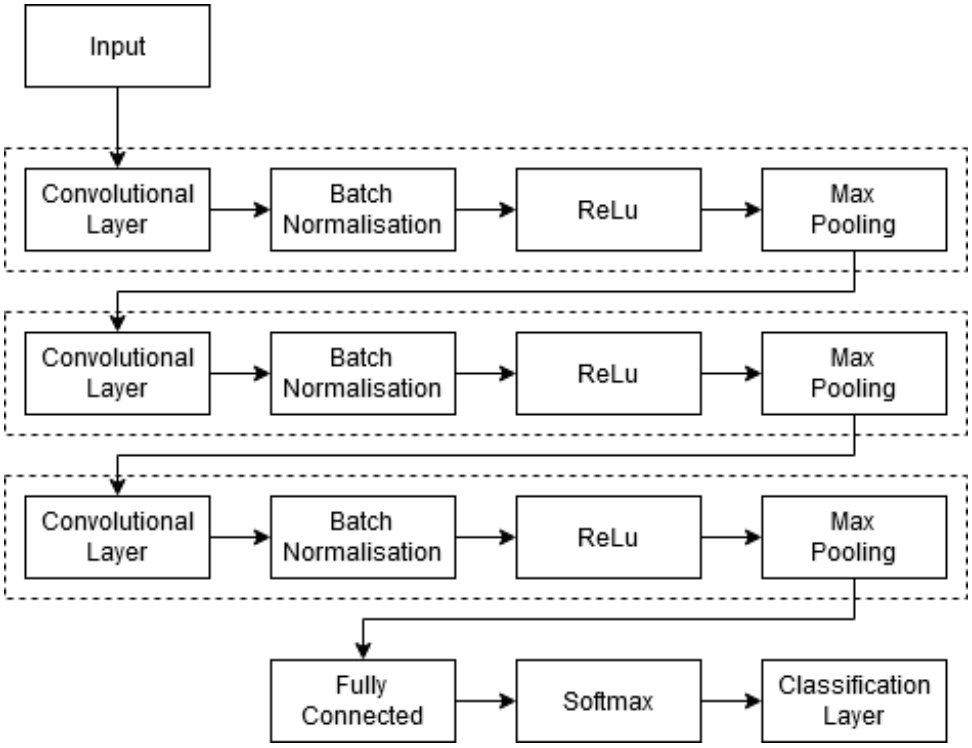


Figure 7.6: CNN network structure

7.3.2 Hyperparameter Selection and Optimisation

7.3.2.1 Convolutional Layer Configuration

The filters in the convolutional layers used in the work by Kulhandjian *et al.* are configured as follows. The first convolutional layer consists of 8 filters with a size of 20. The second convolutional layer consists of 16 filters with a size of 10. The third convolutional layer consists of 32 filters with a size of 5.

The work that was done by Zhang *et al.* [13] also use 3 convolutional layers with the layers having 64, 128 and 256 filters in their layers. Another work done by Wang *et al.* [65] use 32, 64 and 128 filters in their layers.

Table 7.2 shows the mean accuracy using the different convolutional layer configurations. From this we can see that for the case of the data in this study, the filter numbers from the work done by Kulhandjian *et al.* [2], which is 8, 16 and 32, is the most feasible. Data set 1 was used for this test.

Filter Sizes	Mean Accuracy (%)
8, 16, 32	99.75
32, 64, 128	99.31
64, 128, 256	98.89

Table 7.2: Mean accuracy from different number of filters

7.3.2.2 Learning Rate

Using a mini-batch size of 64, the optimum learning rate for the network were searched for using MATLAB's built-in Bayesian optimisation function sweeping from learning rates between 10^{-7} to 10^{-1}.

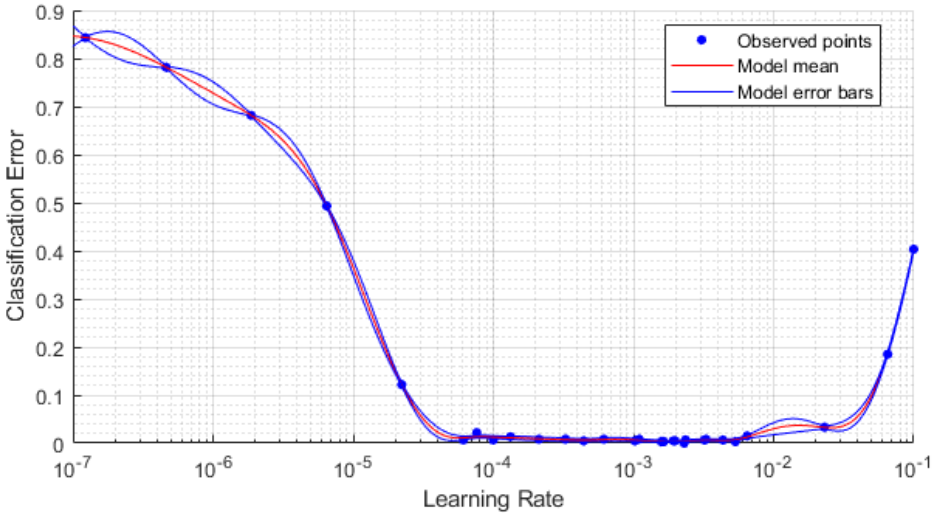


Figure 7.7: Bayesian optimisation result for CNN

The optimum learning rate for the network was searched for using MATLAB's built-in Bayesian optimisation function sweeping from learning rates between 10^{-7} to 10^{-1}. A mini-batch size of 64 was used due to system memory and power limitations.

Data set number 1 from table 6.2 is used for the optimisation. The network is optimised by minimising the classification errors. The plot illustrating the observed points and the model of the learning rate against the classification error that was formed from the observed points is shown in figure 7.4. The classification error is minimised between a learning rate of $4x10^{-5}$ and $5x10^{-3}$ with a small range of uncertainty. The best-observed learning rate and the best-estimated learning rate according to the model formed from the optimisation are shown in table

7.3. The best estimated model learning rate is used to train the CNN.

	Best Learning Rate	Error (%)
Observed	0.0022942	0
Model	0.0019276	0.37

Table 7.3: Bayesian optimisation result for CNN

7.4 Epoch

The number of epochs was chosen so that the training would run long enough for the validation loss to settle. Figure 7.8 shows that the validation loss from the training process for the CNN has settled for a sufficient amount of time by 10 epochs.

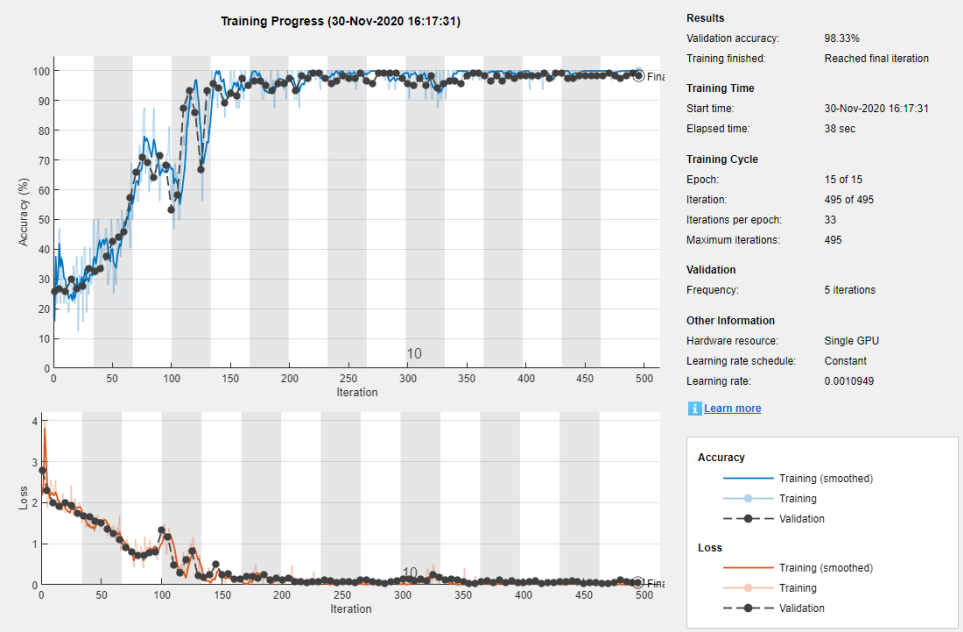


Figure 7.8: Training process of the CNN

Chapter 8: Results and Discussion

8.1 Single Direction Gestures

This section covers the results for the classification of data set 1 from table 6.2 which includes gestures 1 through to 6 that is shown in table 3.1. The results from ten-fold cross-validation training will be shown comparing the LSTM network to the CNN discussed in the previous chapter.

8.1.1 LSTM

Figure 8.1 is a box and whisker plot of the classification accuracy of ten-fold cross-validation of the LSTM network for each gesture. It shows that the lowest classification accuracy for any of the gestures to be 96% with the highest being 100%.

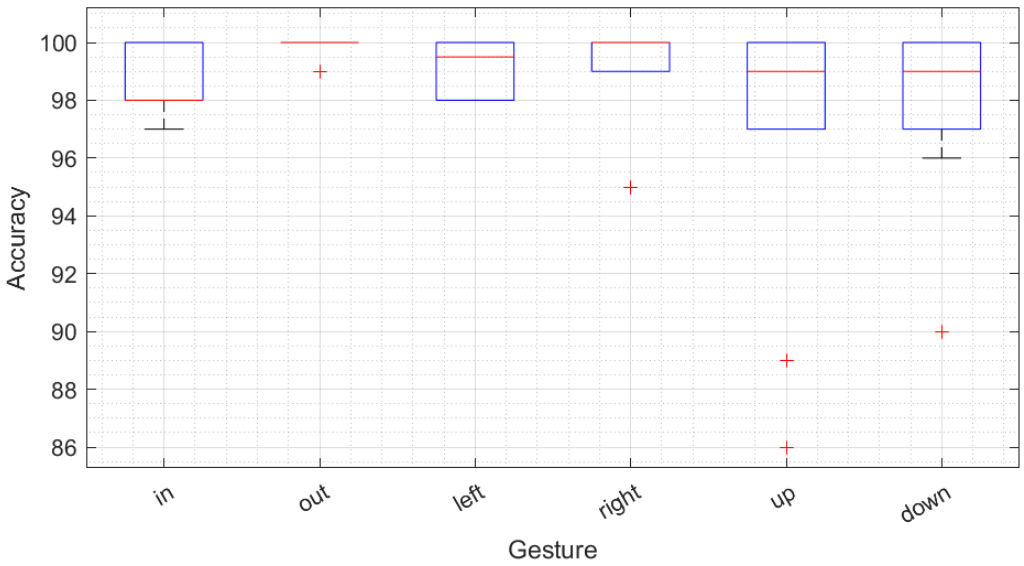


Figure 8.1: Box and whisker plot ten-fold cross-validation for the LSTM network for data set 1

Figure 8.2 and 8.3 show the performance of the network using the test data. The networks with the best and worst mean accuracy from the ten-fold cross-validation were used to illustrate the performance of the network.

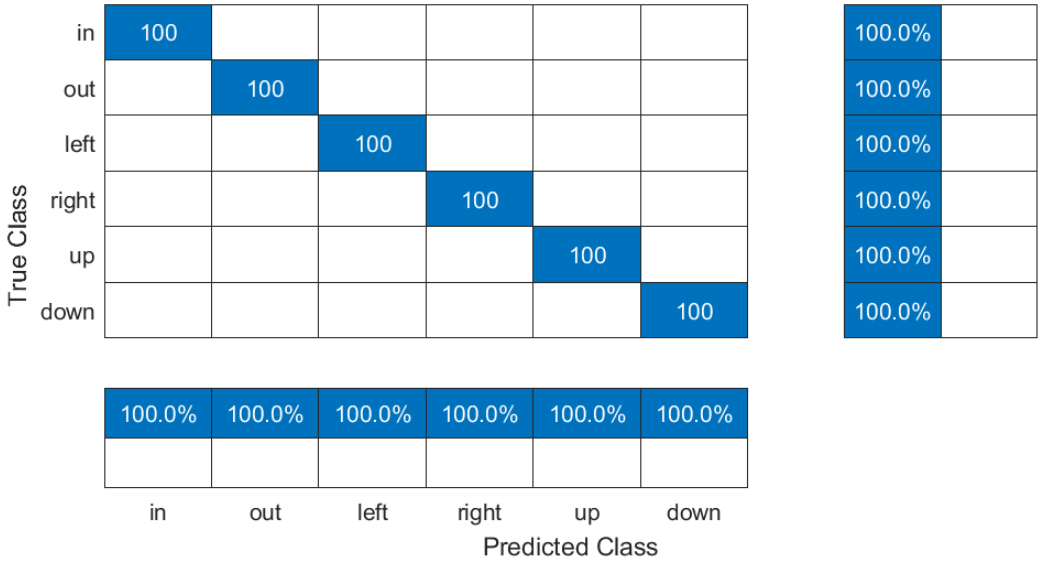


Figure 8.2: Performance of the network for the test data with the best classification accuracy from the ten-fold cross-validation for the LSTM network for data set 1

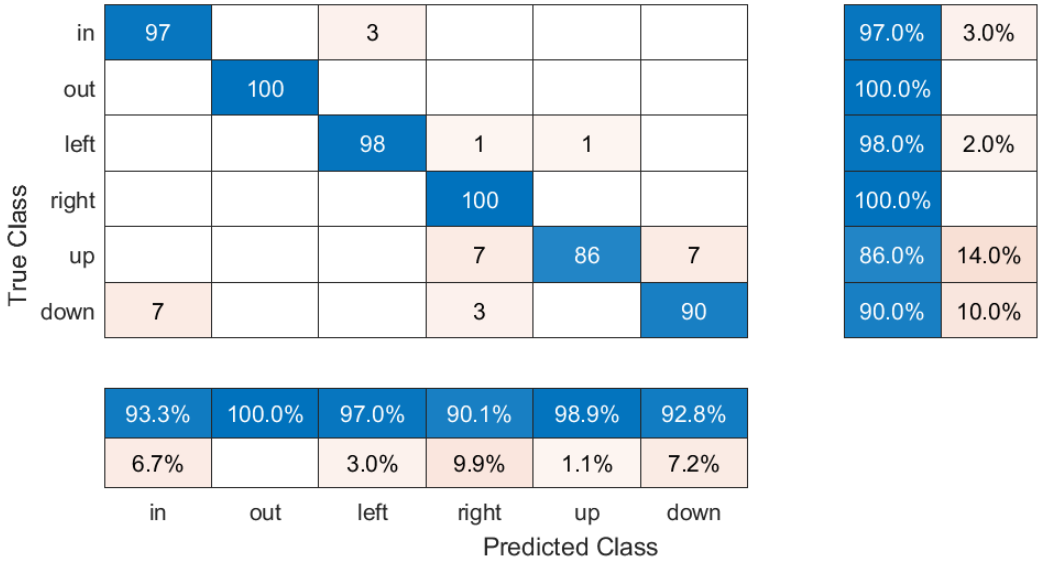


Figure 8.3: Performance of the network for the test data with the worst classification accuracy from the ten-fold cross-validation for the LSTM network for data set 1

Figure 8.4 shows the mean classification accuracy for each fold in the cross-validation as well as the mean classification accuracy across all the folds.

The network is trained again using both training and validation data as training data. There is no need for validation data since the performance of the network has already been validated. The network achieved a classification accuracy of 99.67% with more training data.

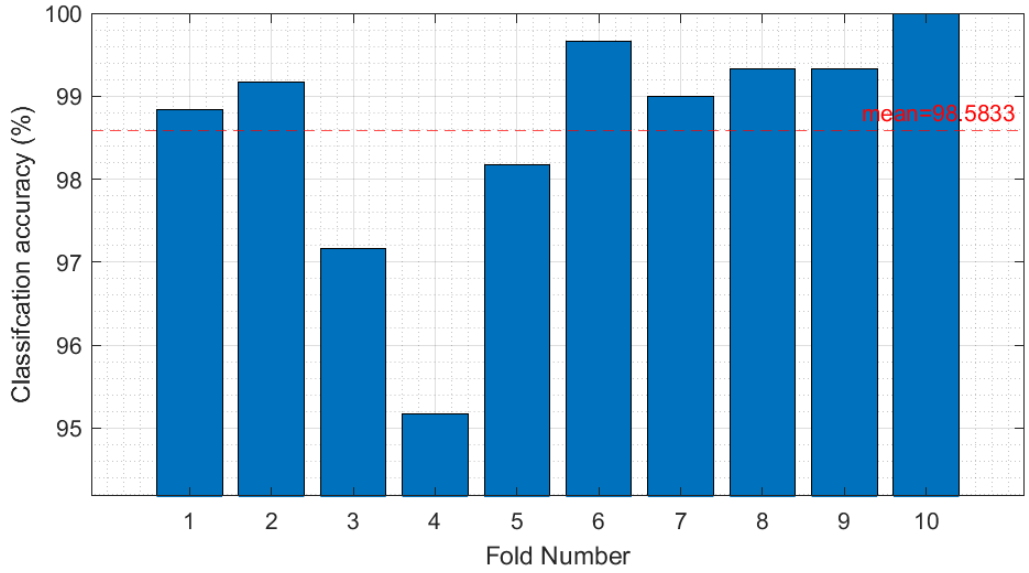


Figure 8.4: Mean classification accuracy for each cross-validation fold for the LSTM network for data set 1

8.1.2 CNN

Figure 8.5 is a box and whisker plot of the classification accuracy of ten-fold cross-validation of the CNN for each gesture. It shows that the lowest classification accuracy for any of the gestures to be 99% with the highest being 100%.

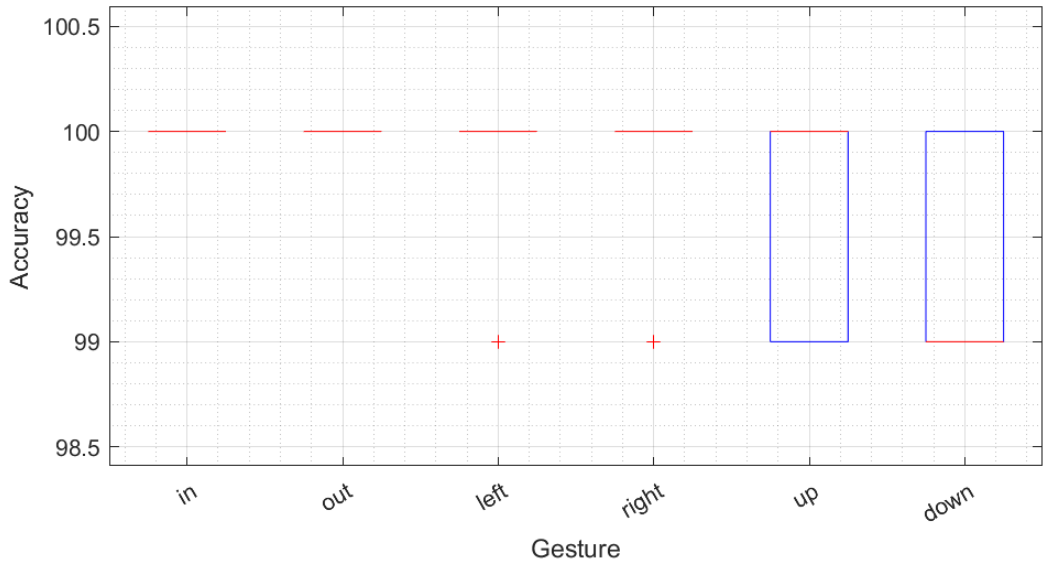


Figure 8.5: Box and whisker plot ten-fold cross-validation for the CNN for data set 1

Figure 8.6 and 8.7 show the performance of the network using the test data. The networks with

the best and worst mean accuracy from the ten-fold cross-validation were used to illustrate the performance of the network.

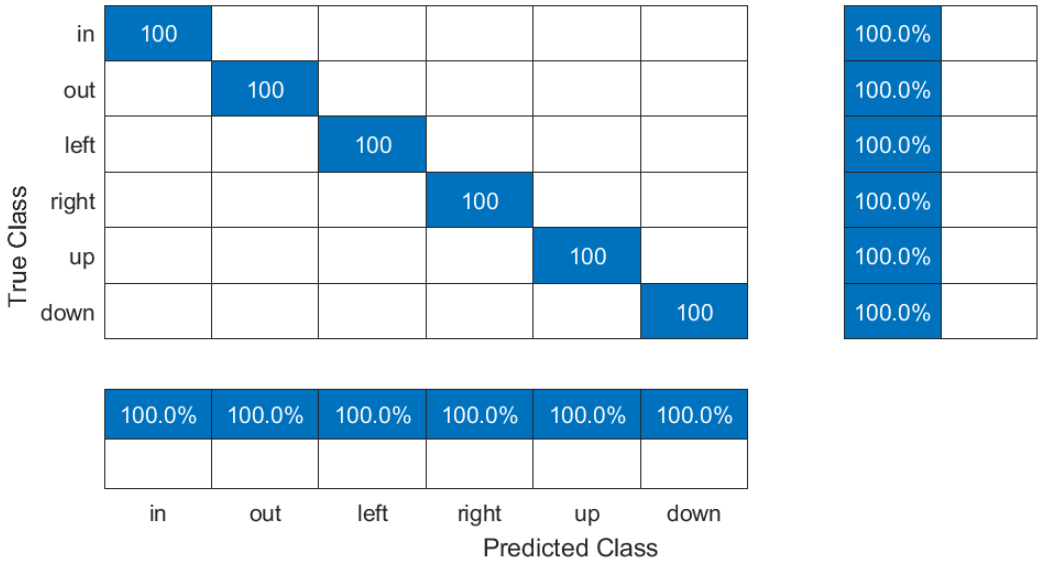


Figure 8.6: Performance of the network for the test data with the best classification accuracy from the ten-fold cross-validation for the CNN for data set 1

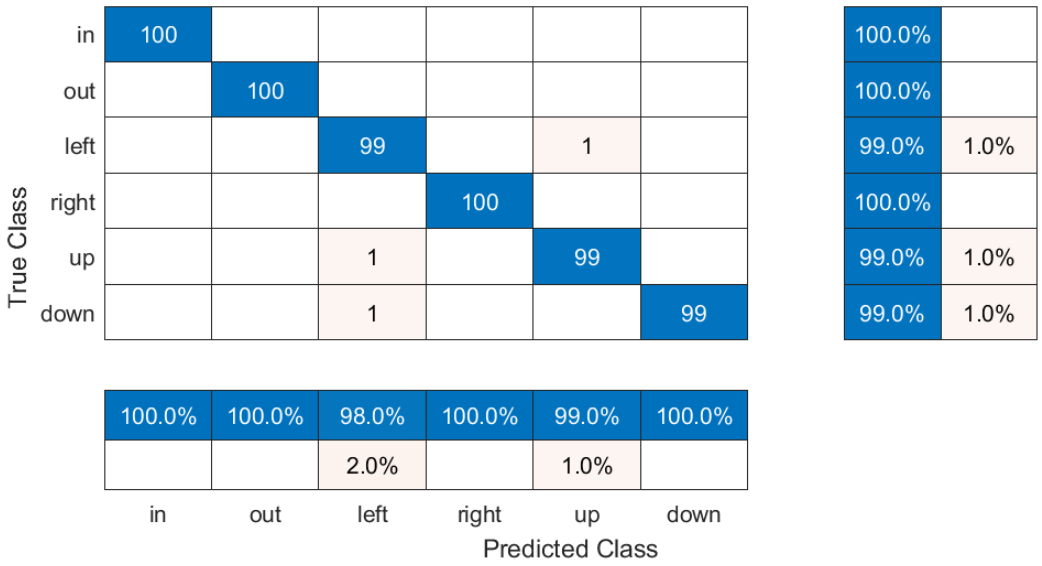


Figure 8.7: Performance of the network for the test data with the worst classification accuracy from the ten-fold cross-validation for the CNN for data set 1

Figure 8.8 shows the mean classification accuracy for each fold in the cross-validation as well as the mean classification accuracy across all the folds.

The network is trained again using both training and validation data as training data. There is no need for validation data since the performance of the network has already been validated. The network achieved a classification accuracy of 100% with more training data.

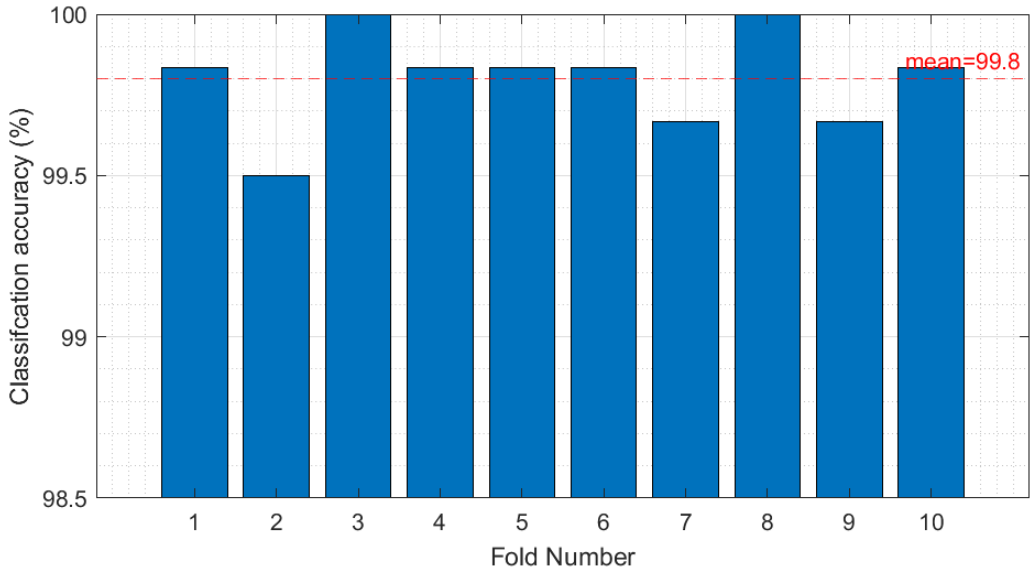


Figure 8.8: Mean classification accuracy for each cross-validation fold for the CNN for data set 1

8.2 Full Gesture Set

This section covers the results for the classification of data set 2 from table 6.2 which includes gestures 1 through to 8 that is shown in table 3.1. The results from ten-fold cross-validation training will be shown comparing the LSTM network to the CNN discussed in the previous chapter.

8.2.1 LSTM

Figure 8.9 is a box and whisker plot of the classification accuracy of ten-fold cross-validation of the LSTM network for each gesture. It shows that the lowest classification accuracy for any of the gestures to be 0% with the highest being 100%.

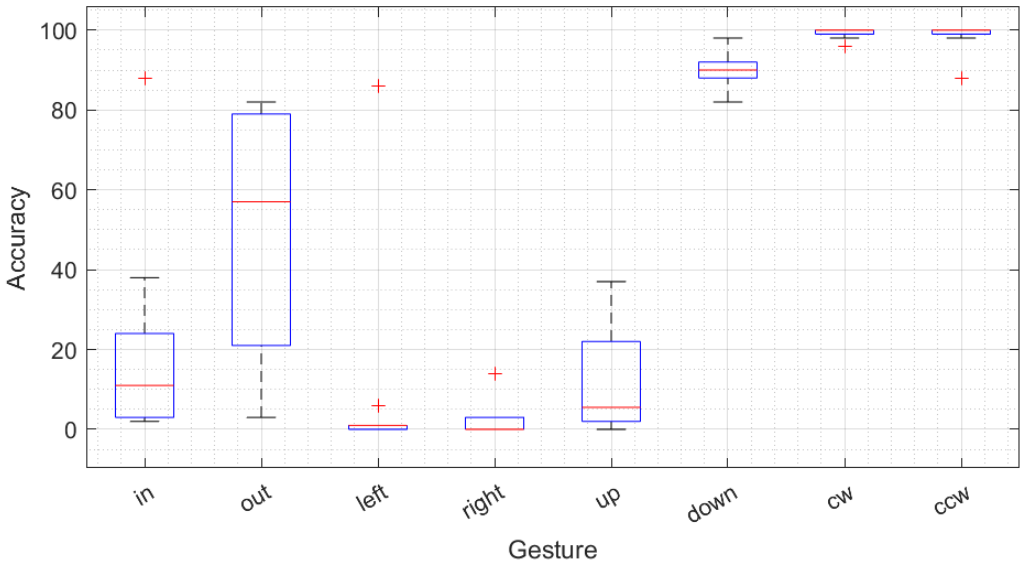


Figure 8.9: Box and whisker plot ten-fold cross-validation for the LSTM network for data set 2

Figure 8.10 and 8.11 show the performance of the network using the test data. The networks with the best and worst mean accuracy from the ten-fold cross-validation were used to illustrate the performance of the network.

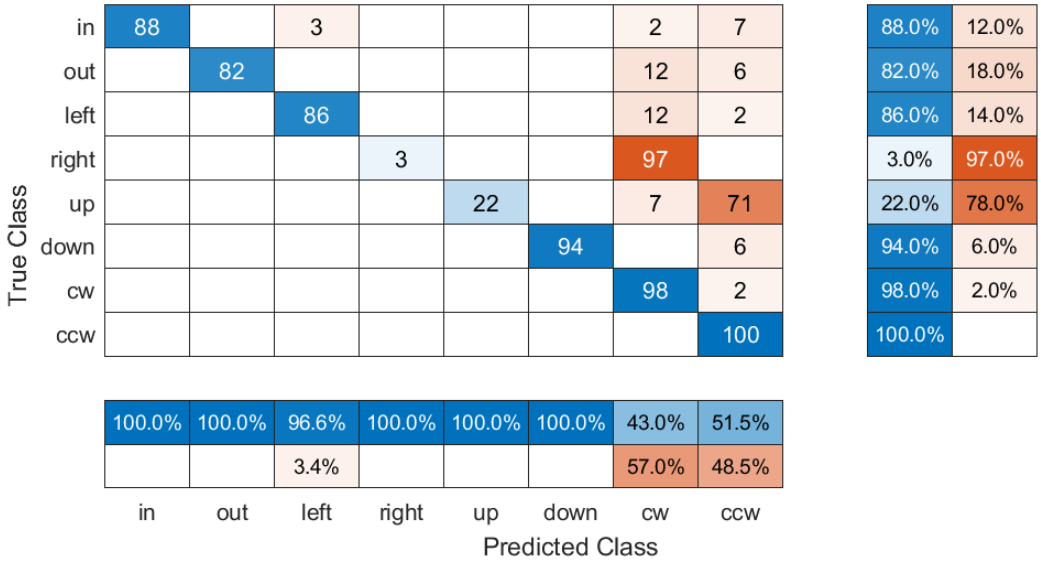


Figure 8.10: Performance of the network for the test data with the best classification accuracy from the ten-fold cross-validation for the LSTM network for data set 2

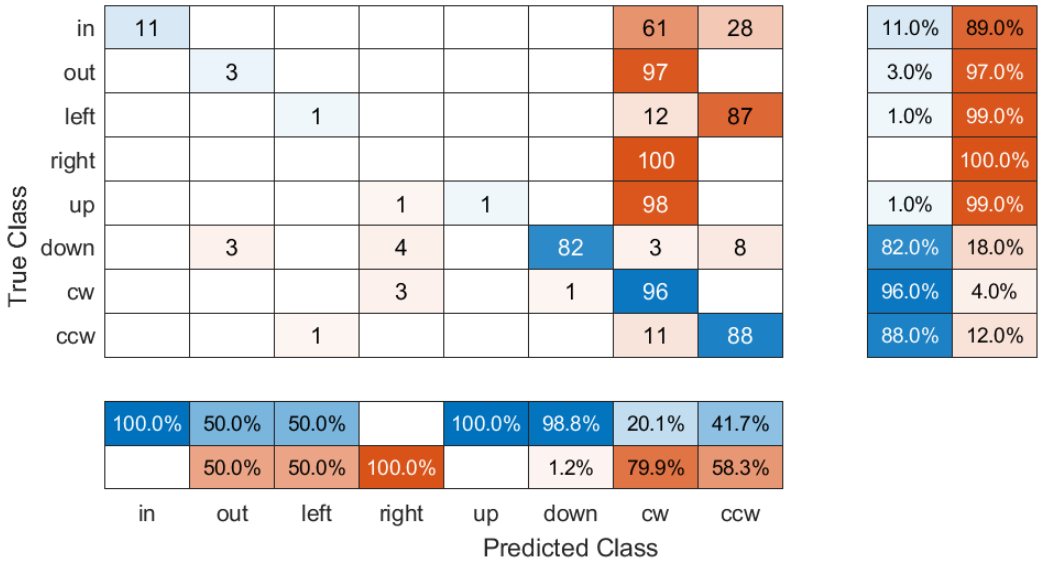


Figure 8.11: Performance of the network for the test data with the worst classification accuracy from the ten-fold cross-validation for the LSTM network for data set 2

Figure 8.12 shows the mean classification accuracy for each fold in the cross-validation as well as the mean classification accuracy across all the folds.

The network is trained again using both training and validation data as training data. There is no need for validation data since the performance of the network has already been validated. The network achieved a classification accuracy of 41.75% with more training data.

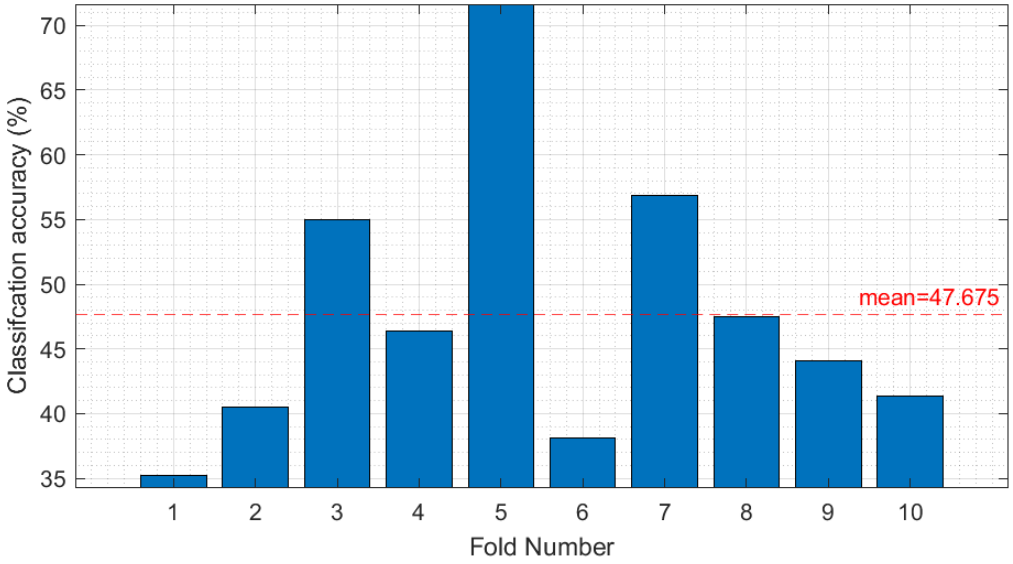


Figure 8.12: Mean classification accuracy for each cross-validation fold for the LSTM network for data set 2

8.2.2 CNN

Figure 8.13 is a box and whisker plot of the classification accuracy of ten-fold cross-validation of the CNN for each gesture. It shows that the lowest classification accuracy for any of the gestures to be 97% with the highest being 100%.

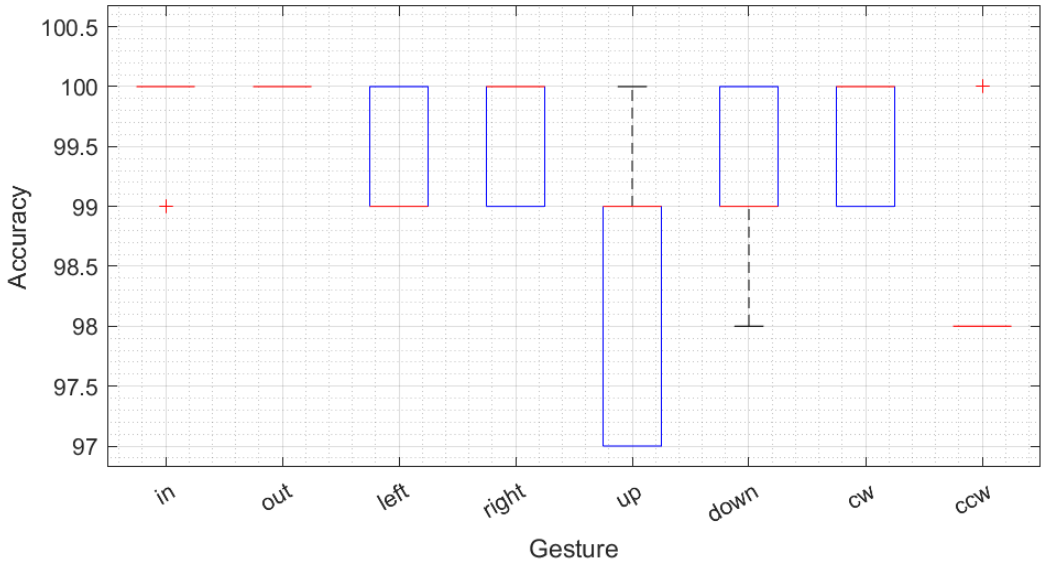


Figure 8.13: Box and whisker plot ten-fold cross-validation for the CNN for data set 2

Figure 8.14 and 8.15 show the performance of the network using the test data. The networks

with the best and worst mean accuracy from the ten-fold cross-validation were used to illustrate the performance of the network.

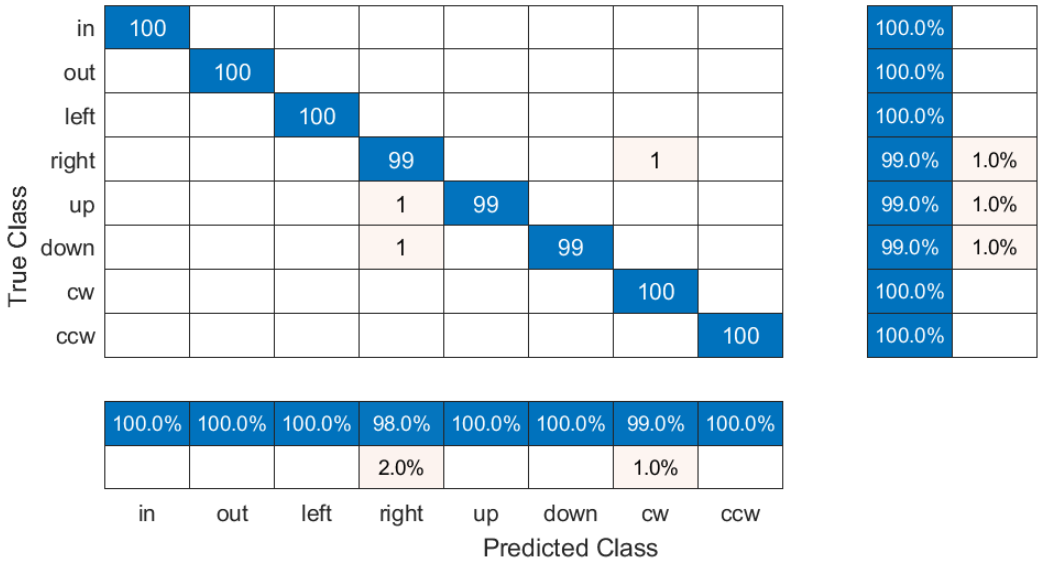


Figure 8.14: Performance of the network for the test data with the best classification accuracy from the ten-fold cross-validation for the CNN for data set 2

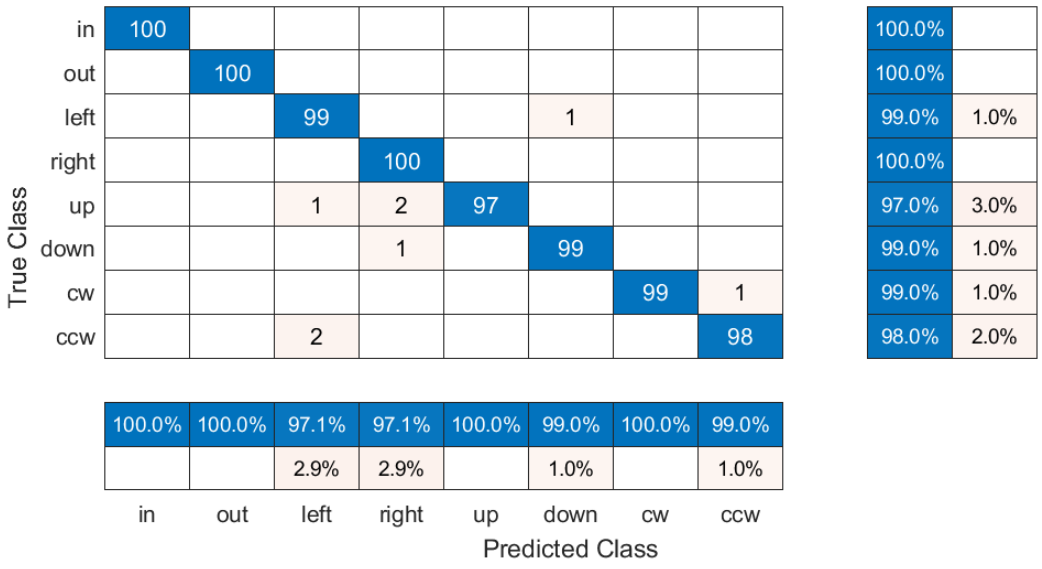


Figure 8.15: Performance of the network for the test data with the worst classification accuracy from the ten-fold cross-validation for the CNN for data set 2

Figure 8.16 shows the mean classification accuracy for each fold in the cross-validation as well as the mean classification accuracy across all the folds.

The network is trained again using both training and validation data as training data. There is no need for validation data since the performance of the network has already been validated.

The network achieved a classification accuracy of 99.75% with more training data.

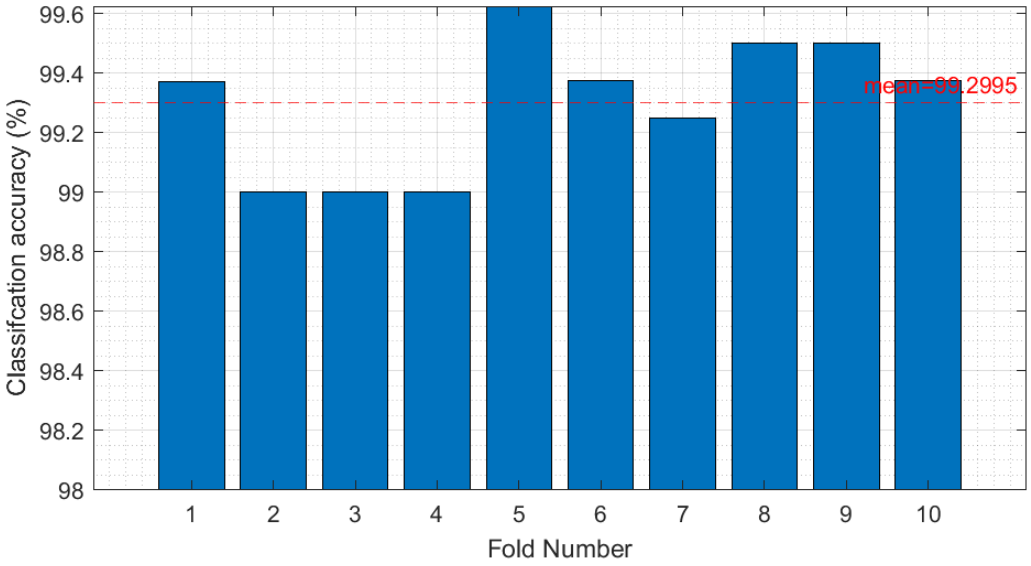


Figure 8.16: Mean classification accuracy for each cross-validation fold for the CNN for data set 2

8.3 Discussion on types of gestures

For the single direction gestures test, the test results show that the LSTM network and CNN have very similar classification accuracy with the CNN having an accuracy of 1.22% higher than the LSTM network. This result, however, does not hold when the clockwise and counter-clockwise gestures, gesture 7 and 8 from table 3.1, are introduced to the data set as the LSTM network's performance drops from 98.58% to 47.67% while the CNN maintains a mean classification accuracy higher than 99%. Additionally, figure 8.9 shows that certain gestures exhibit high variance for the LSTM network, resulting in the network being unreliable. Almost all of the gestures from data set 1 except for the swiping down gesture are confused as the clockwise and counter-clockwise gestures for the LSTM network. The clockwise and counter-clockwise gestures are essentially composed of many of the single direction gestures. This causes the LSTM network to confuse the single direction gestures as gesture 7 and 8 since the LSTM network is sensitive to temporal sequences instead of patterns such as for the CNN.

Mixing simple single direction gestures with slightly more complex gestures cause the LSTM network's classification accuracy to drop while the CNN continues to classify gestures which a high accuracy.

In literature, Chen *et al.* [14] and Zhang *et al.* [13] use a similar gesture set as the gestures in both data sets 1 and 2. Chen was able to achieve a classification accuracy of 99% using a CNN and Zhang was able to achieve 96% using a CNN followed by an LSTM layer. The CNN from this study shows similar performance to the results achieved in literature with a classification accuracy of 99%. The LSTM network using data set 1 also shows similar results achieved in literature with an accuracy of 98.58%.

8.4 Distance Test (0.5m-1m)

This section covers the results for the classification of data set 3 from table 6.2 which includes gestures 1 through to 6 that is shown in table 3.1. These gestures are measured between 0.5m and 1m away from the sensor compared to data set 1 which are gestures within 0.5m of the sensor. The results from ten-fold cross-validation training will be shown comparing the LSTM network to the CNN discussed in the previous chapter.

8.4.1 LSTM

Figure 8.17 is a box and whisker plot of the classification accuracy of ten-fold cross-validation of the LSTM network for each gesture. It shows that the lowest classification accuracy for any of the gestures to be 71% with the highest being 100%.

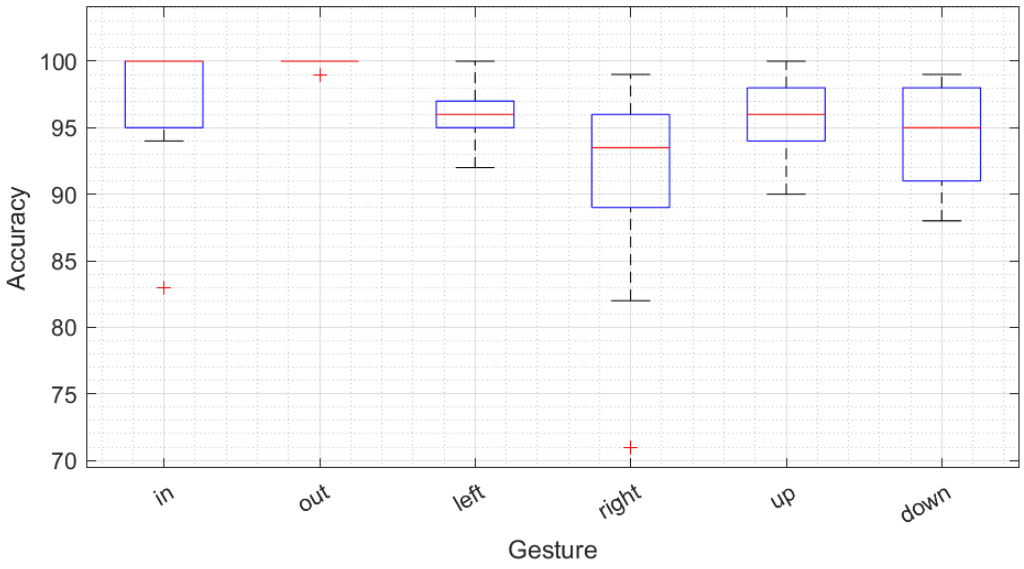


Figure 8.17: Box and whisker plot ten-fold cross-validation for the LSTM network for data set 3

Figure 8.18 and 8.19 show the performance of the network using the test data. The networks with the best and worst mean accuracy from the ten-fold cross-validation were used to illustrate the performance of the network.

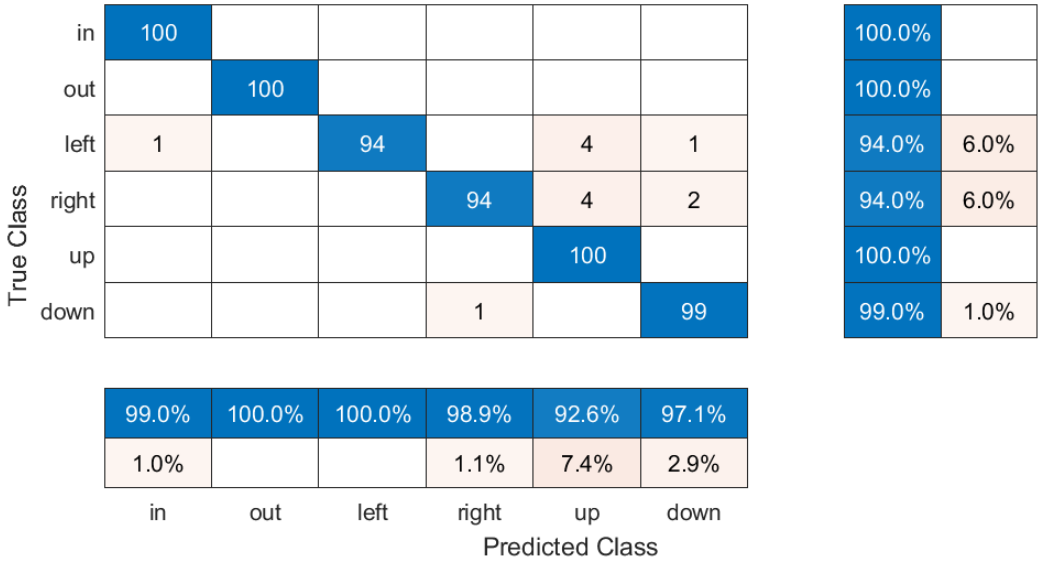


Figure 8.18: Performance of the network for the test data with the best classification accuracy from the ten-fold cross-validation for the LSTM network for data set 3

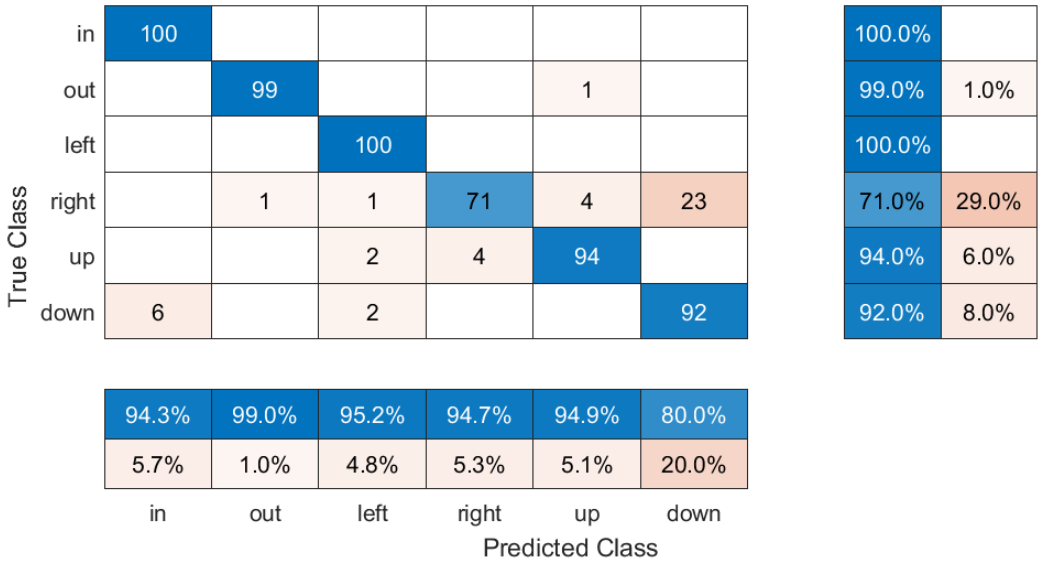


Figure 8.19: Performance of the network for the test data with the worst classification accuracy from the ten-fold cross-validation for the LSTM network for data set 3

Figure 8.20 shows the mean classification accuracy for each fold in the cross-validation as well as the mean classification accuracy across all the folds.

The network is trained again using both training and validation data as training data. There is no need for validation data since the performance of the network has already been validated. The network achieved a classification accuracy of 98.17% with more training data.

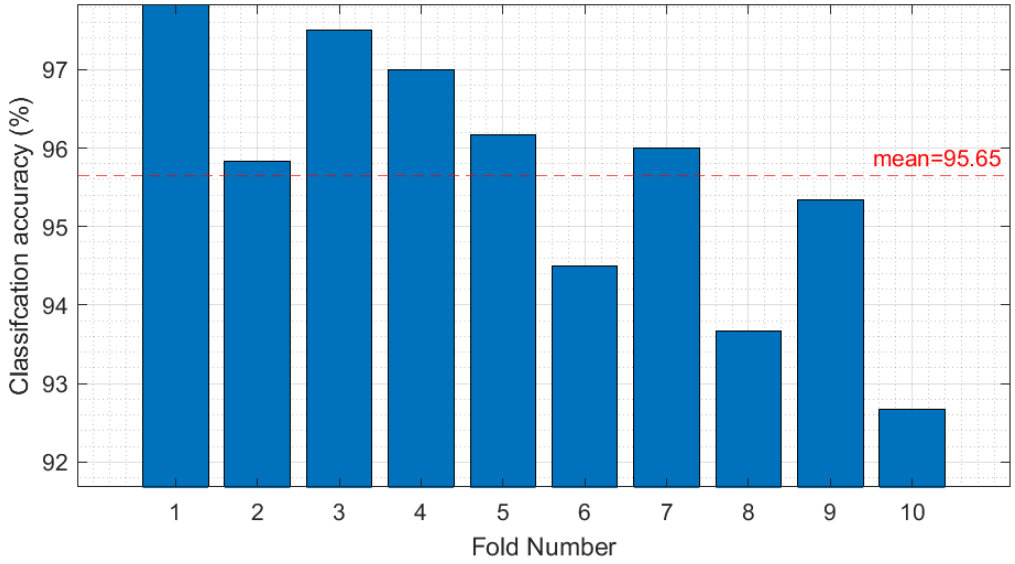


Figure 8.20: Mean classification accuracy for each cross-validation fold for the LSTM network for data set 3

8.4.2 CNN

Figure 8.21 is a box and whisker plot of the classification accuracy of ten-fold cross-validation of the CNN for each gesture. It shows that the lowest classification accuracy for any of the gestures to be 96% with the highest being 100%.

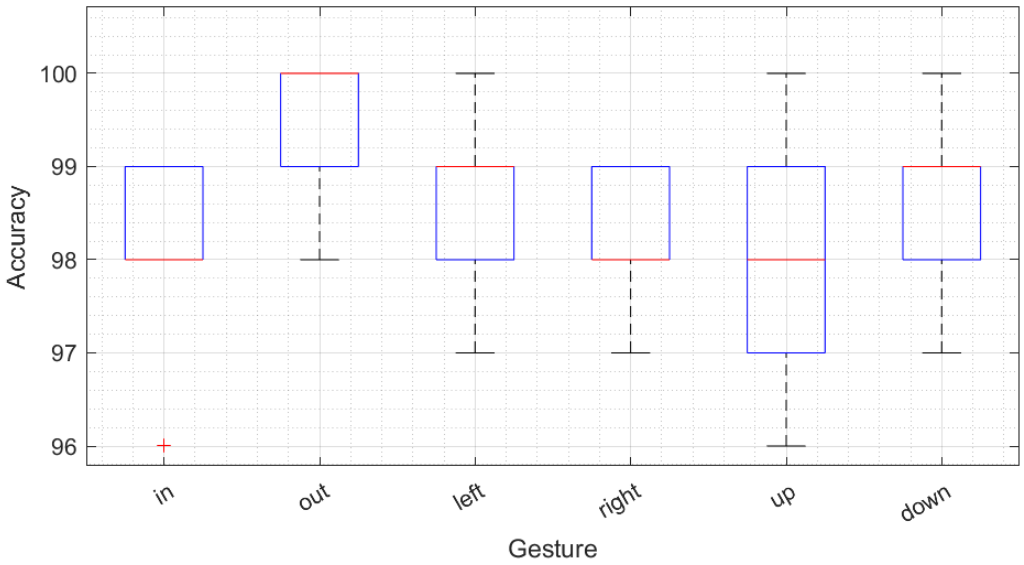


Figure 8.21: Box and whisker plot ten-fold cross-validation for the CNN for data set 3

Figure 8.22 and 8.23 show the performance of the network using the test data. The networks

with the best and worst mean accuracy from the ten-fold cross-validation were used to illustrate the performance of the network.

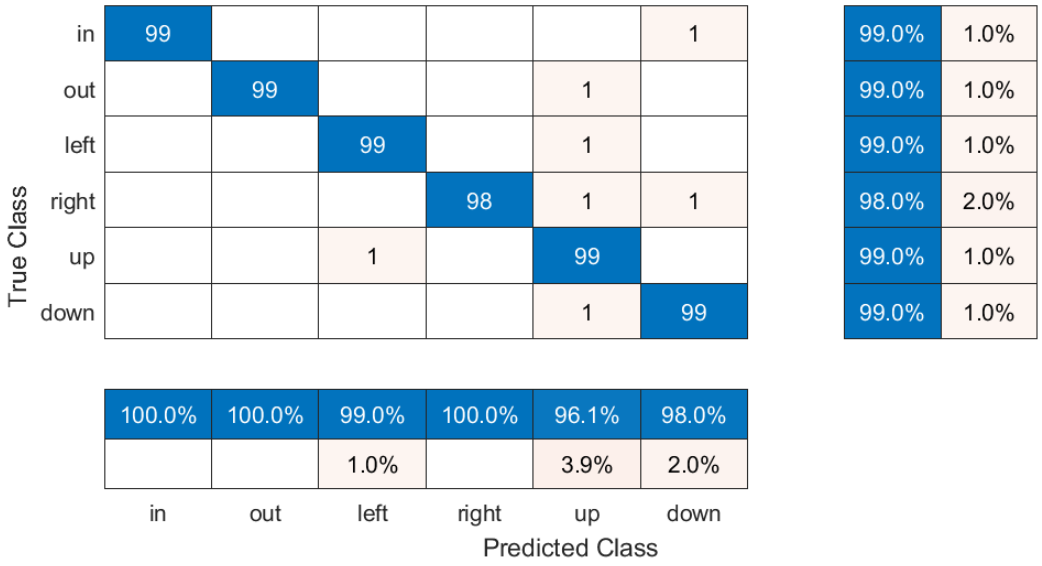


Figure 8.22: Performance of the network for the test data with the best classification accuracy from the ten-fold cross-validation for the CNN for data set 3

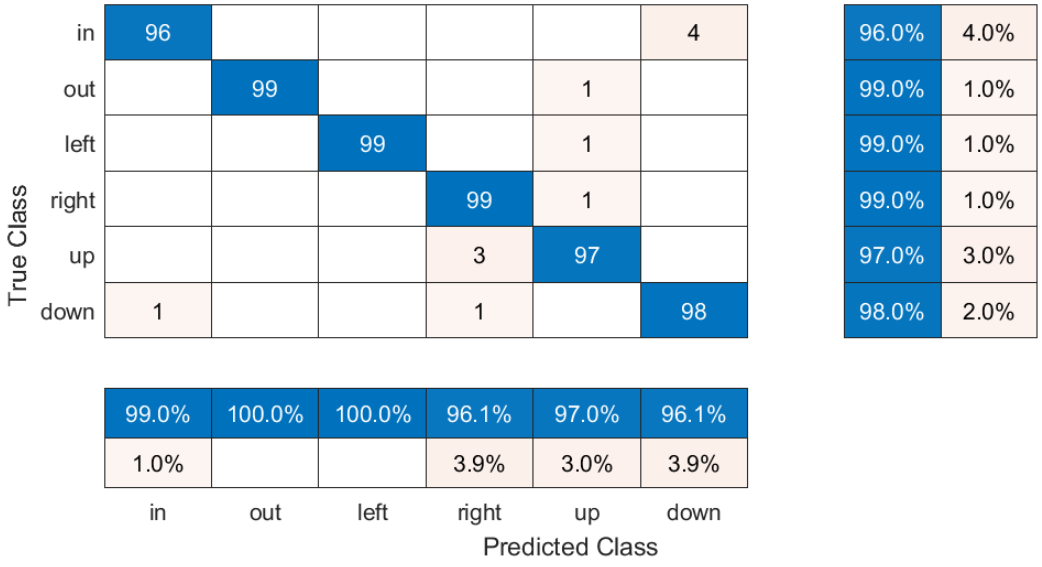


Figure 8.23: Performance of the network for the test data with the worst classification accuracy from the ten-fold cross-validation for the CNN for data set 3

Figure 8.24 shows the mean classification accuracy for each fold in the cross-validation as well as the mean classification accuracy across all the folds.

The network is trained again using both training and validation data as training data. There is no need for validation data since the performance of the network has already been validated.

The network achieved a classification accuracy of 98.83% with more training data.

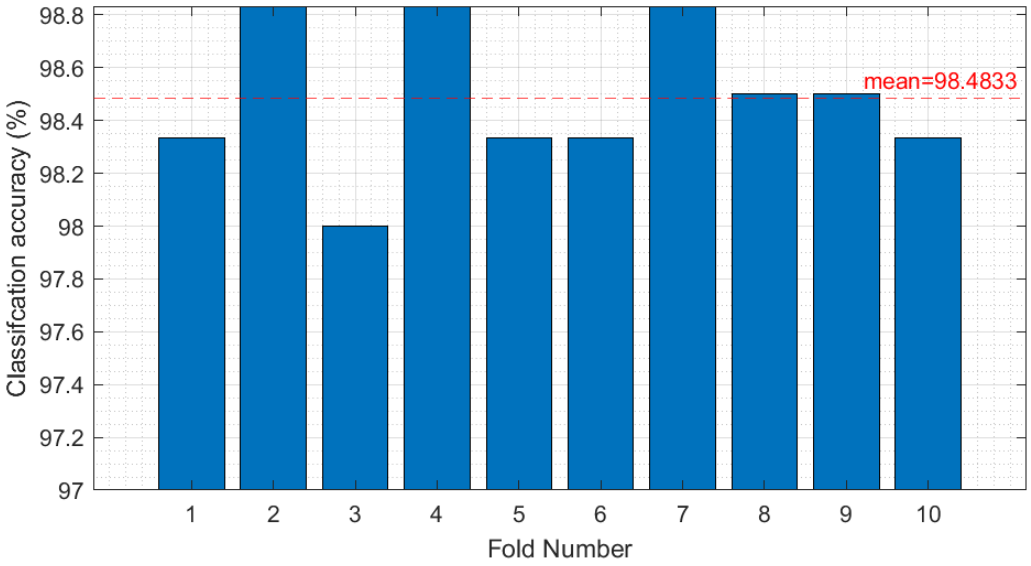


Figure 8.24: Mean classification accuracy for each cross-validation fold for the CNN for data set 3

8.5 Distance Test (0m-1m)

This section covers the results for the classification of data set 4 from table 6.2 which includes gestures 1 through to 6 that is shown in table 3.1. This data set is a combination of data set 1 and 3 which include data from a range of 0m to 0.5m and 0.5m to 1m. The results from ten-fold cross-validation training will be shown comparing the LSTM network to the CNN discussed in the previous chapter.

8.5.1 LSTM

Figure 8.25 is a box and whisker plot of the classification accuracy of ten-fold cross-validation of the LSTM network for each gesture. It shows that the lowest classification accuracy for any of the gestures to be 25.5% with the highest being 100%.

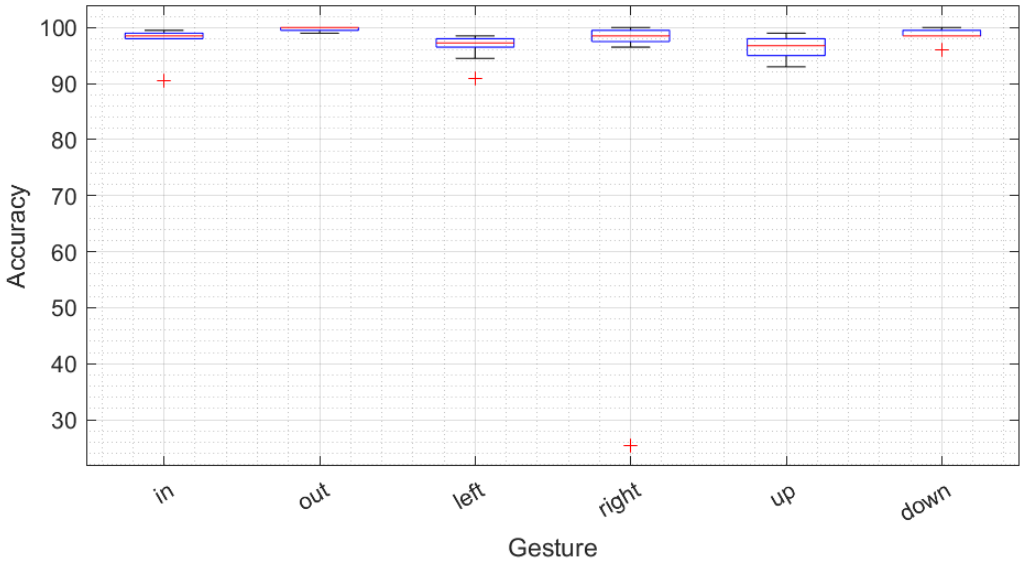


Figure 8.25: Box and whisker plot ten-fold cross-validation for the LSTM network for data set 4

Figure 8.26 and 8.27 show the performance of the network using the test data. The networks with the best and worst mean accuracy from the ten-fold cross-validation were used to illustrate the performance of the network.

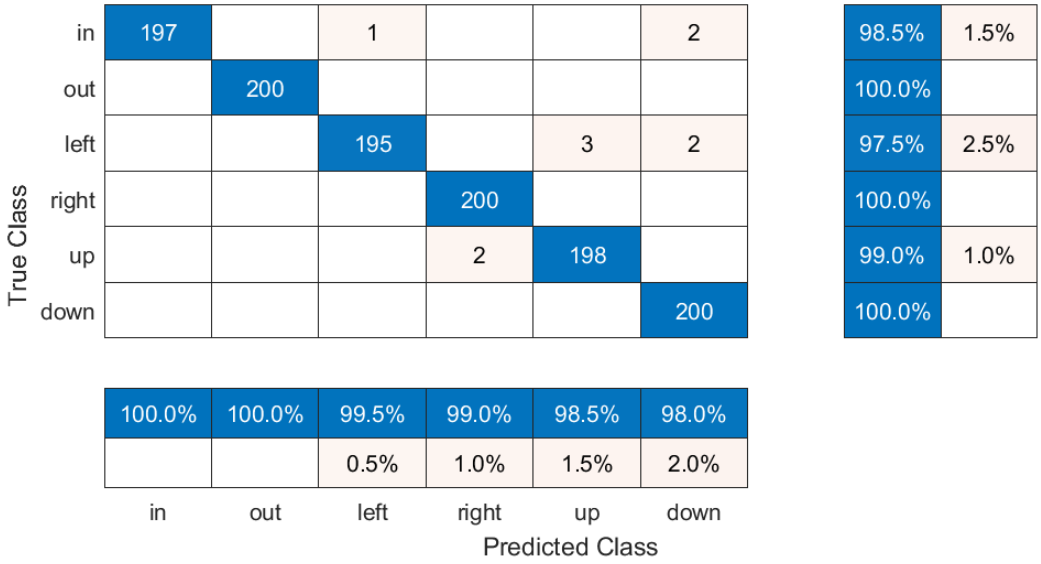


Figure 8.26: Performance of the network for the test data with the best classification accuracy from the ten-fold cross-validation for the LSTM network for data set 4

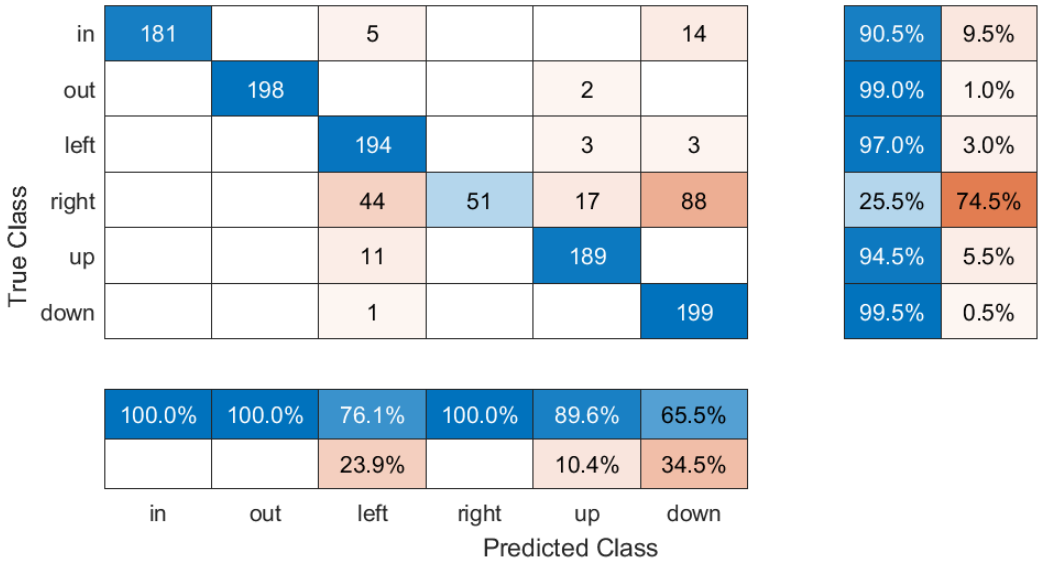


Figure 8.27: Performance of the network for the test data with the worst classification accuracy from the ten-fold cross-validation for the LSTM network for data set 4

Figure 8.28 shows the mean classification accuracy for each fold in the cross-validation as well as the mean classification accuracy across all the folds.

The network is trained again using both training and validation data as training data. There is no need for validation data since the performance of the network has already been validated. The network achieved a classification accuracy of 99.08% with more training data.

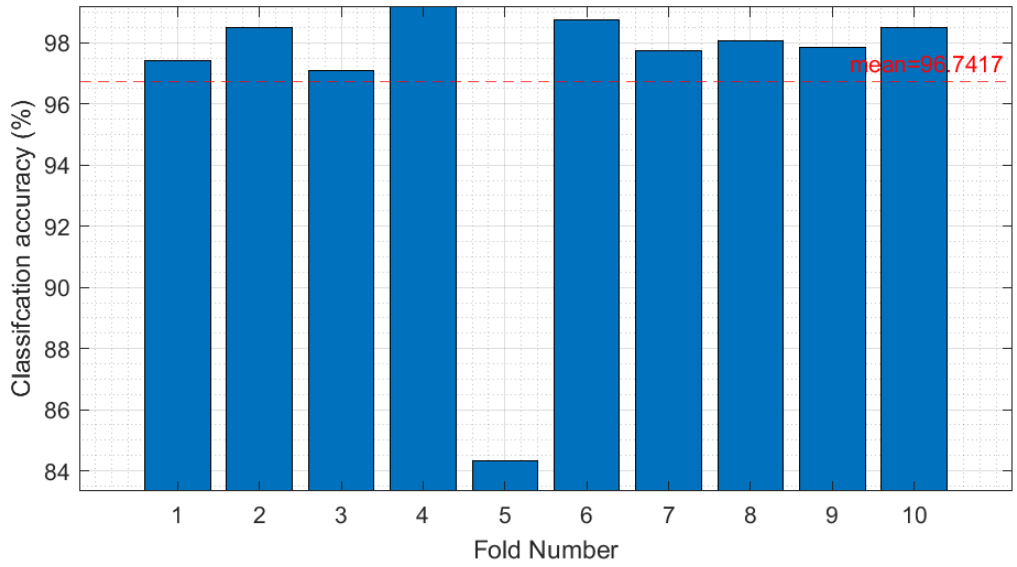


Figure 8.28: Mean classification accuracy for each cross-validation fold for the LSTM network for data set 4

8.5.2 CNN

Figure 8.29 is a box and whisker plot of the classification accuracy of ten-fold cross-validation of the CNN for each gesture. It shows that the lowest classification accuracy for any of the gestures to be 96% with the highest being 100%.

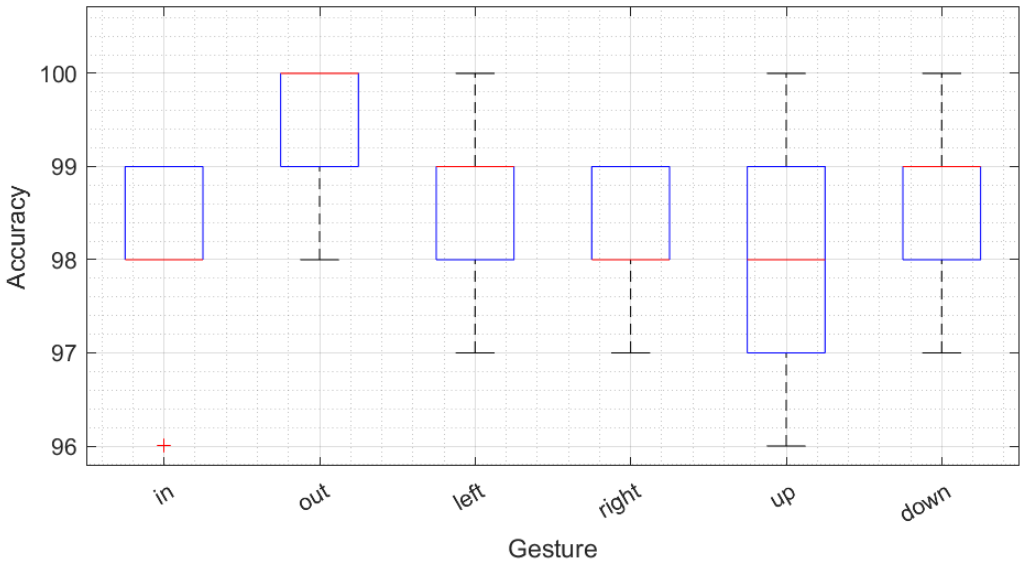


Figure 8.29: Box and whisker plot ten-fold cross-validation for the CNN for data set 4

Figure 8.30 and 8.31 show the performance of the network using the test data. The networks

with the best and worst mean accuracy from the ten-fold cross-validation were used to illustrate the performance of the network.

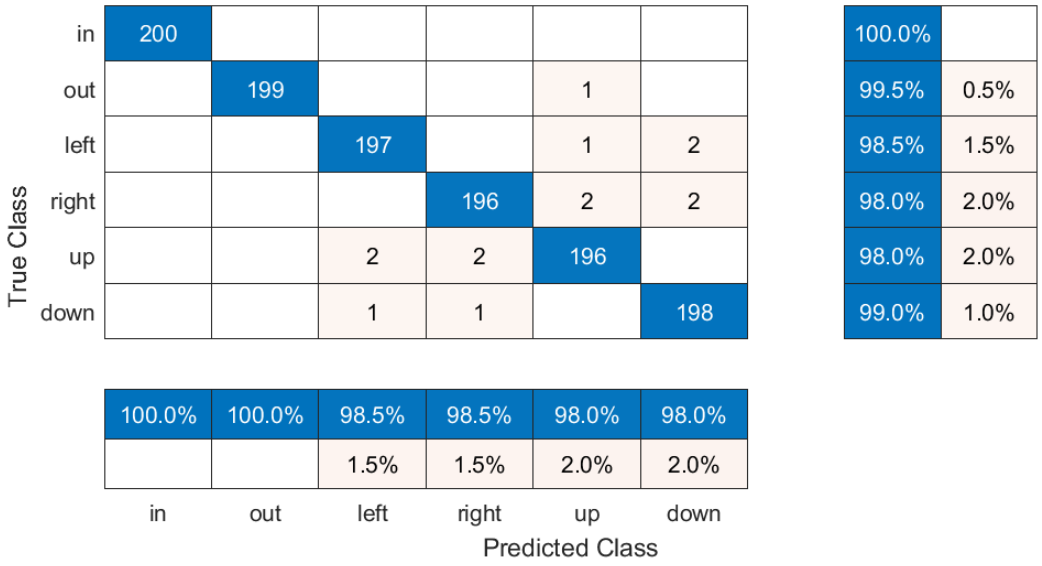


Figure 8.30: Performance of the network for the test data with the best classification accuracy from the ten-fold cross-validation for the CNN for data set 4

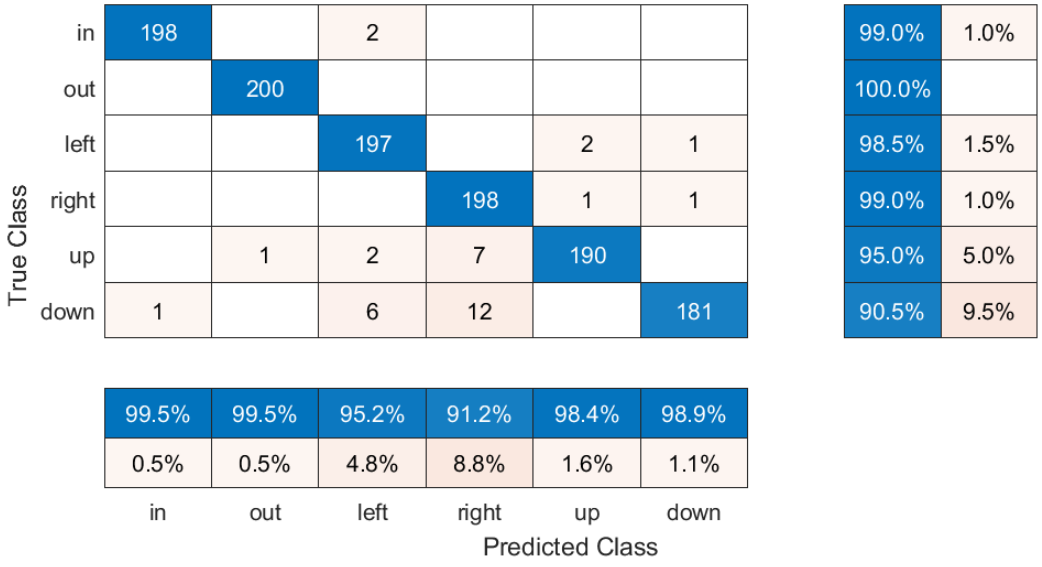


Figure 8.31: Performance of the network for the test data with the worst classification accuracy from the ten-fold cross-validation for the CNN for data set 4

Figure 8.32 shows the mean classification accuracy for each fold in the cross-validation as well as the mean classification accuracy across all the folds.

The network is trained again using both training and validation data as training data. There is no need for validation data since the performance of the network has already been validated.

The network achieved a classification accuracy of 98.75% with more training data.

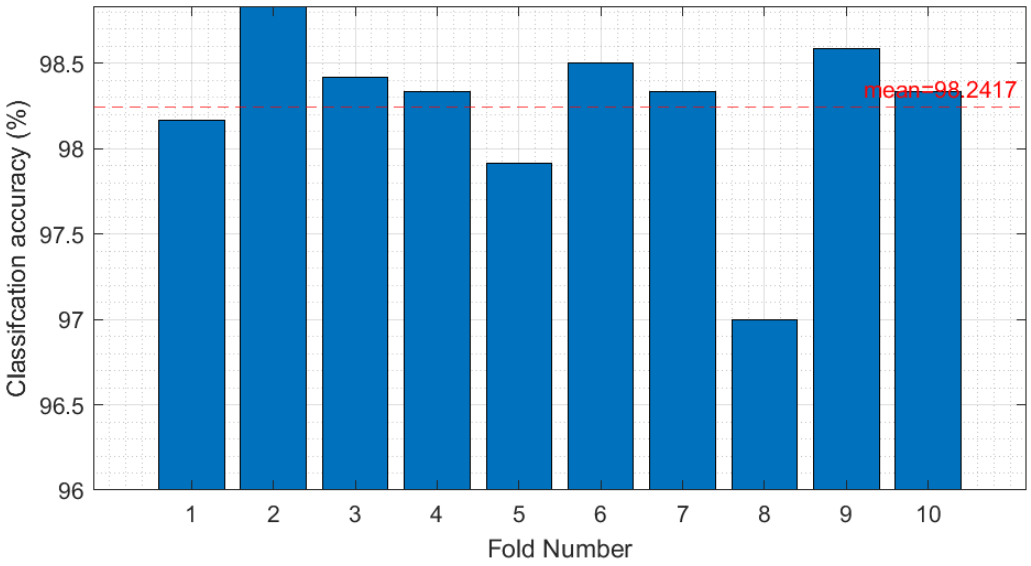


Figure 8.32: Mean classification accuracy for each cross-validation fold for the CNN for data set 4

8.6 Discussion on distance tests

To determine whether or not the system could detect gestures within a 1-metre range from the sensors, 2 tests were conducted and compared to the results of the single directions gesture test. The first test recorded gestures at a range between half a metre and one metre. The mean classification accuracy from the cross-validation of the LSTM network yielded a result of 95.65%. Analysing figure 8.17 we see that The classification accuracy for the gestures is mostly above 80% with an outlier for the swiping right gesture at 71%. The CNN can classify the gestures with a mean accuracy of 98.48%.

Comparing the results to the mean classification accuracy from data set 1, we see that the mean accuracy of the LSTM network decreases by 2.93% while the CNN's accuracy decreases by 1.32%.

To determine whether the networks could classify gestures at different ranges, data set 1 was combined with data set 3 to form data set 4 which contains gestures with ranges from 0 metres to 1 metre.

Combining the data sets allowed the LSTM network to classify the gestures more accurately when compared to the results from data set 3. The mean classification accuracy for this test

yielded a result of 96.74% which is a 1.09% increase in accuracy. The CNN was able to correctly classify the gestures with an accuracy of 98.24% which is only a 0.24% decrease in accuracy. There is an outlier for the swiping right gesture for the LSTM at 25.5%.

Analysing the performance of both the networks, by looking at figure 8.18, 8.19, 8.22, 8.23, 8.26, 8.27, 8.30 and 8.31, we can see the most of the confusion occurs within the swiping left, right, up and down gestures. This result is to be expected because as the user gets further and further away from the sensor, the component of velocity of the hand gestures towards the sensors decreases. Figure 8.33 illustrates how moving further away from the sensor causes the angle difference from start to finish of the gesture to seem smaller to the sensors which in turn is perceived as a slower gesture. As seen in figure 8.33, θ2 decreases as the gesture is performed further away from the sensors compared to θ1. As the perceived speed of the gesture decreases, it begins to be masked by stationary clutter and becomes harder for the system to sense.

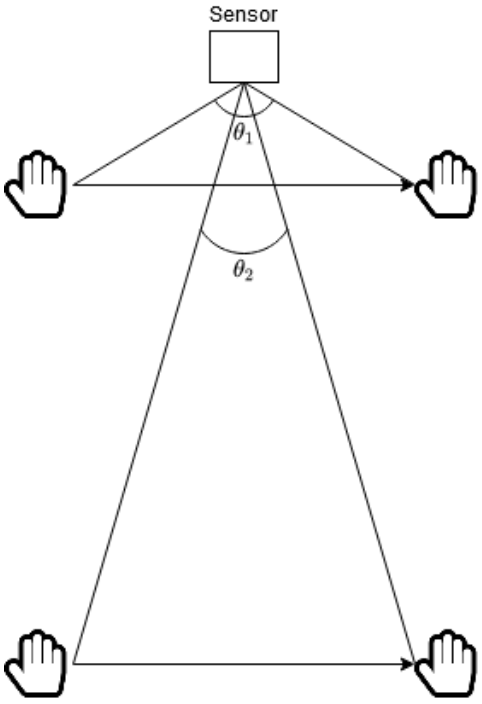


Figure 8.33: Velocity change based on distance

8.7 Gestures with Both Hands

This section covers the results for the classification of data set 5 from table 6.2 which includes gestures 1 through to 6 that is shown in table 3.1 performed using both left and right hands. This data set has the same labels for both left- and right-handed gestures. The results from ten-fold cross-validation training will be shown comparing the LSTM network to the CNN discussed in the previous chapter.

8.7.1 LSTM

Figure 8.34 is a box and whisker plot of the classification accuracy of ten-fold cross-validation of the LSTM network for each gesture. It shows that the lowest classification accuracy for any of the gestures to be 66% with the highest being 100%.

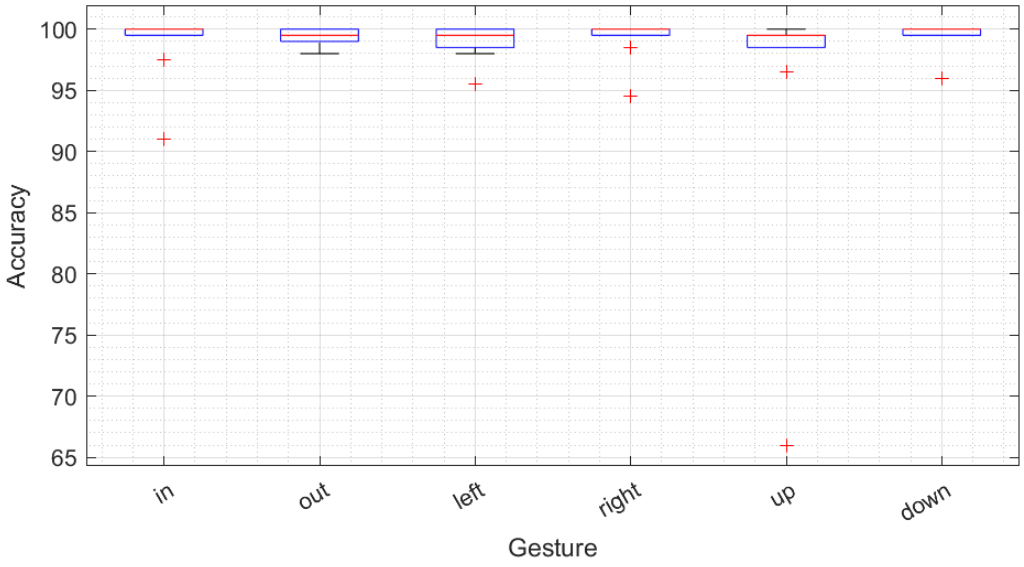


Figure 8.34: Box and whisker plot ten-fold cross-validation for the LSTM network for data set 5

Figure 8.35 and 8.36 show the performance of the network using the test data. The networks with the best and worst mean accuracy from the ten-fold cross-validation were used to illustrate the performance of the network.

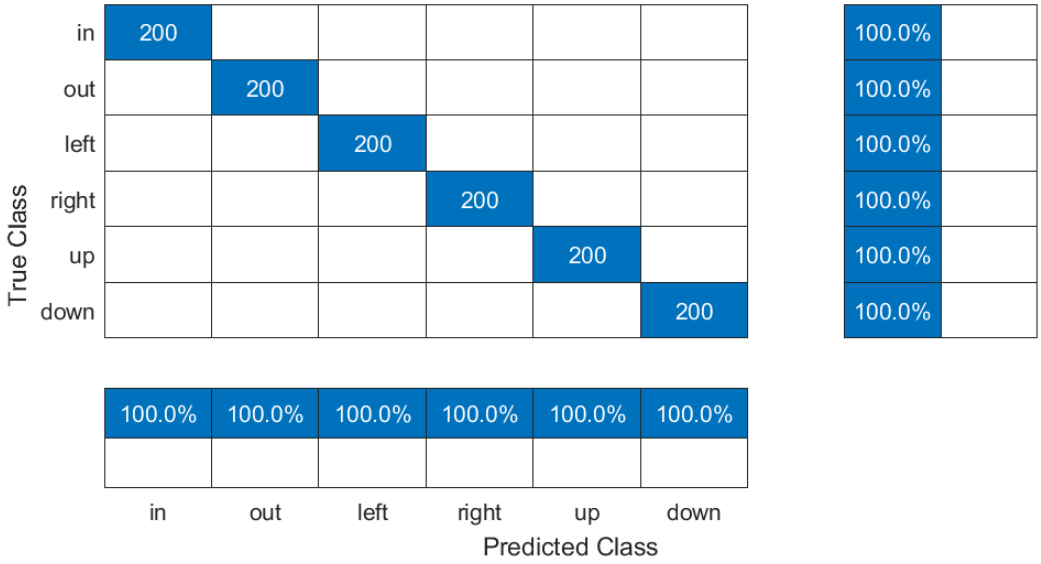


Figure 8.35: Performance of the network for the test data with the best classification accuracy from the ten-fold cross-validation for the LSTM network for data set 5

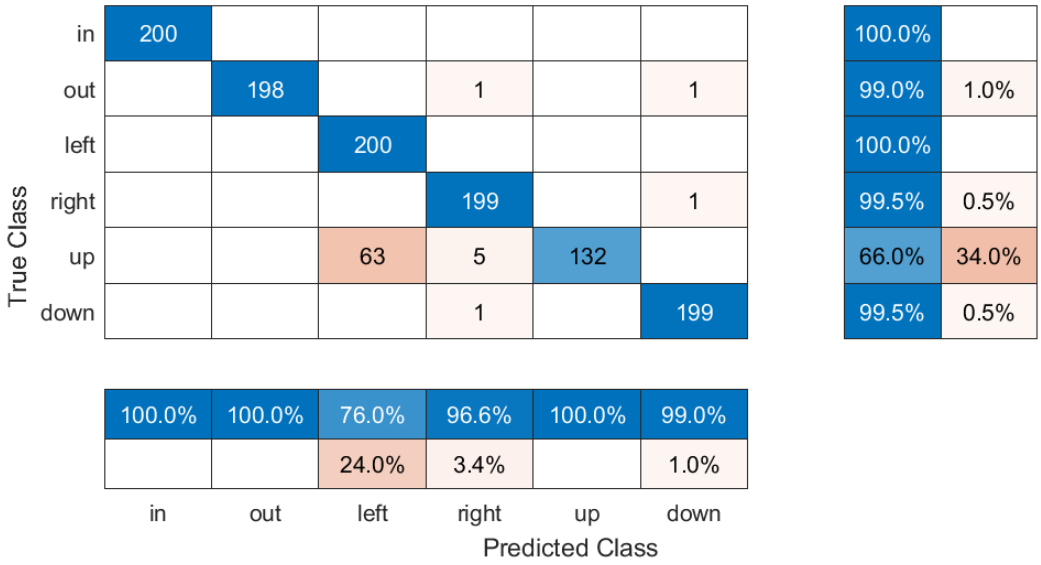


Figure 8.36: Performance of the network for the test data with the worst classification accuracy from the ten-fold cross-validation for the LSTM network for data set 5

Figure 8.37 shows the mean classification accuracy for each fold in the cross-validation as well as the mean classification accuracy across all the folds.

The network is trained again using both training and validation data as training data. There is no need for validation data since the performance of the network has already been validated. The network achieved a classification accuracy of 99.83% with more training data.

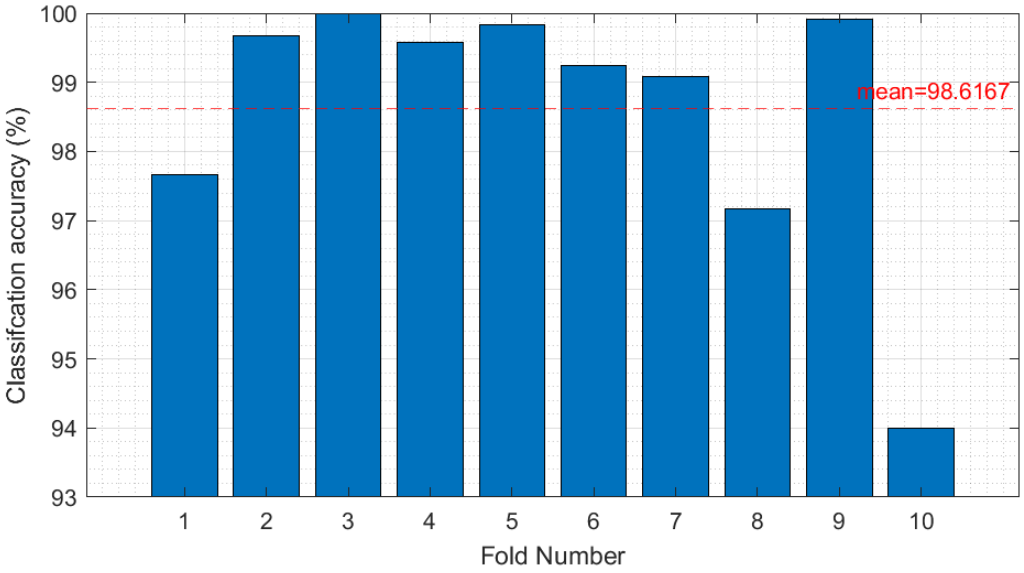


Figure 8.37: Mean classification accuracy for each cross-validation fold for the LSTM network for data set 5

8.7.2 CNN

Figure 8.38 is a box and whisker plot of the classification accuracy of ten-fold cross-validation of the CNN for each gesture. It shows that the lowest classification accuracy for any of the gestures to be 95.5% with the highest being 100%.

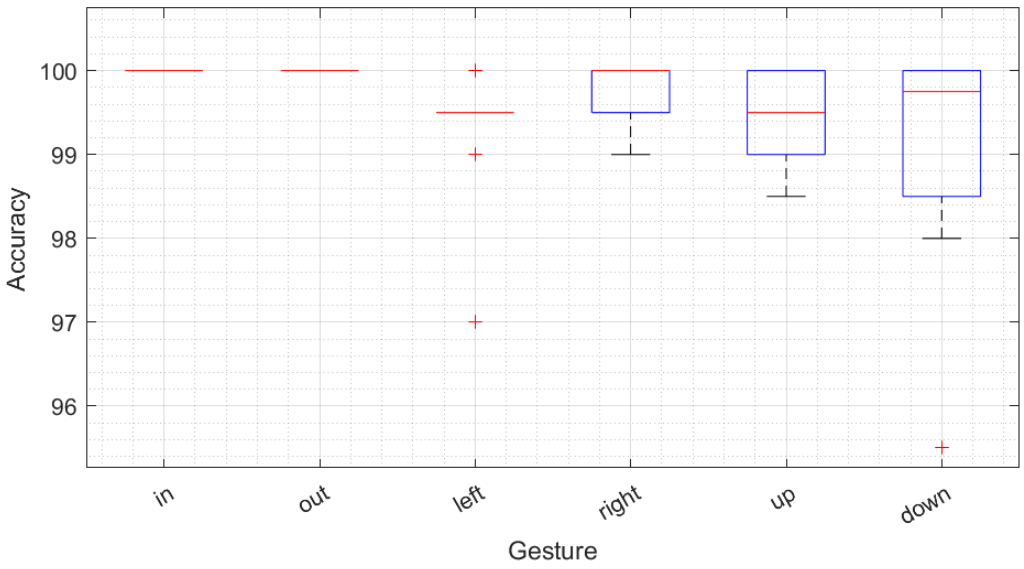


Figure 8.38: Box and whisker plot ten-fold cross-validation for the CNN for data set 5

Figure 8.39 and 8.40 show the performance of the network using the test data. The networks

with the best and worst mean accuracy from the ten-fold cross-validation were used to illustrate the performance of the network.

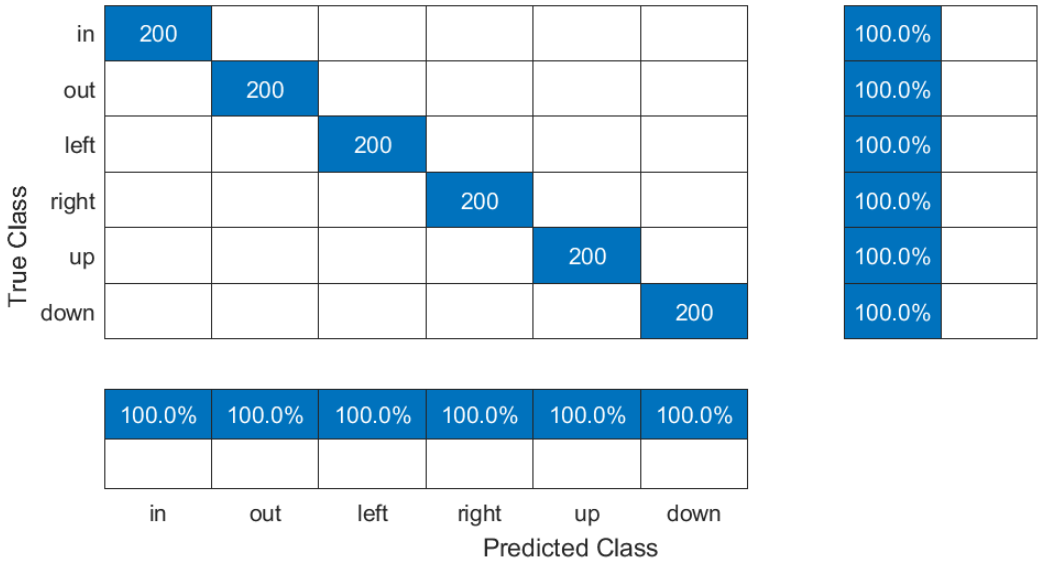


Figure 8.39: Performance of the network for the test data with the best classification accuracy from the ten-fold cross-validation for the CNN for data set 5

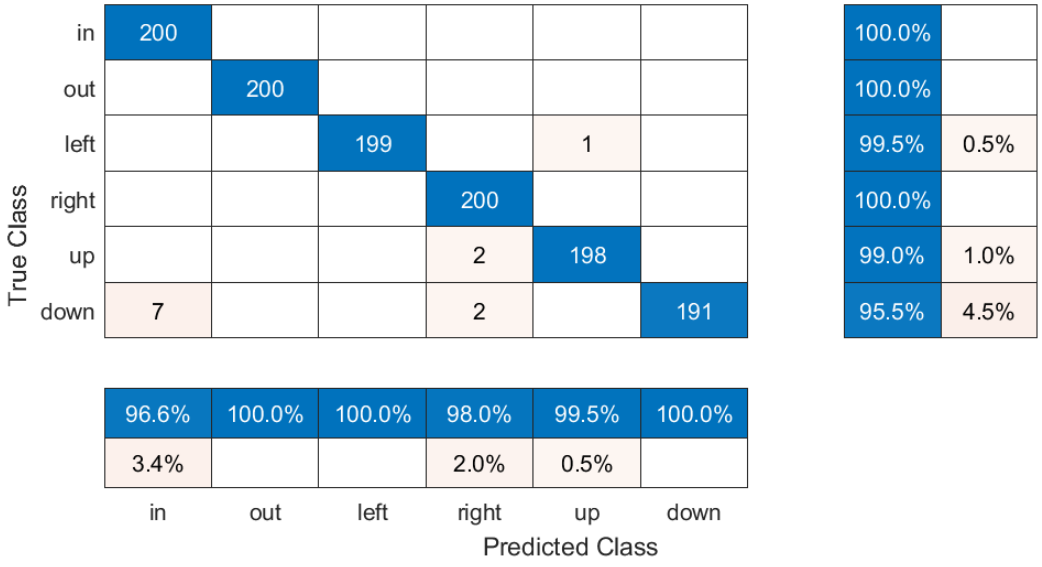


Figure 8.40: Performance of the network for the test data with the worst classification accuracy from the ten-fold cross-validation for the CNN for data set 5

Figure 8.41 shows the mean classification accuracy for each fold in the cross-validation as well as the mean classification accuracy across all the folds.

The network is trained again using both training and validation data as training data. There is no need for validation data since the performance of the network has already been validated.

The network achieved a classification accuracy of 98.17% with more training data.

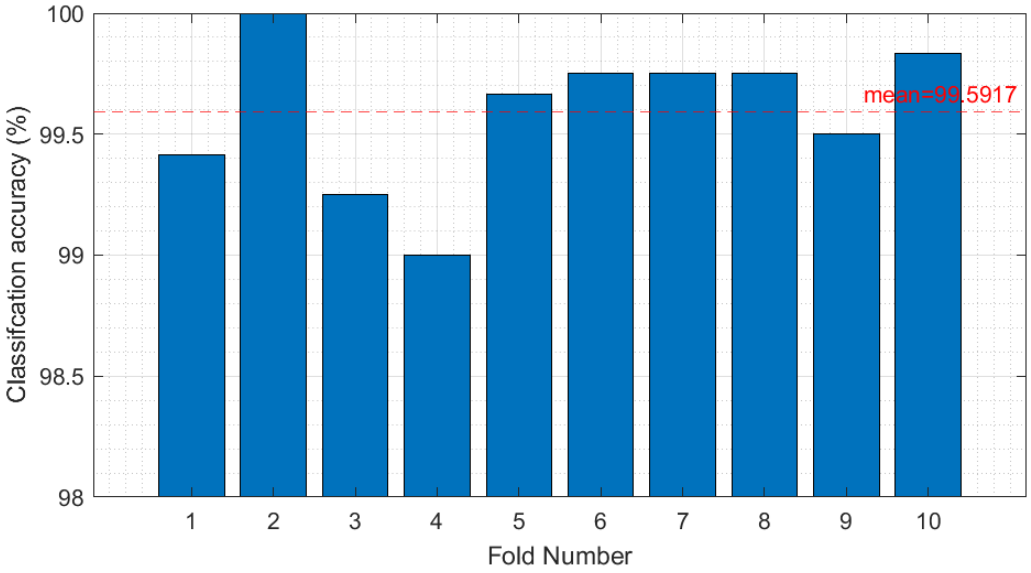


Figure 8.41: Mean classification accuracy for each cross-validation fold for the CNN for data set 5

8.8 Gestures with Both Hands with Separate Labels

This section covers the results for the classification of data set 6 from table 6.2 which includes gestures 1 through to 6 that is shown in table 3.1 performed using both left and right hands. This data set has different labels for left and right-handed gestures. The results from ten-fold cross-validation training will be shown comparing the LSTM network to the CNN discussed in the previous chapter.

8.8.1 LSTM

Figure 8.42 is a box and whisker plot of the classification accuracy of ten-fold cross-validation of the LSTM network for each gesture. It shows that the lowest classification accuracy for any of the gestures to be 2% with the highest being 100%.

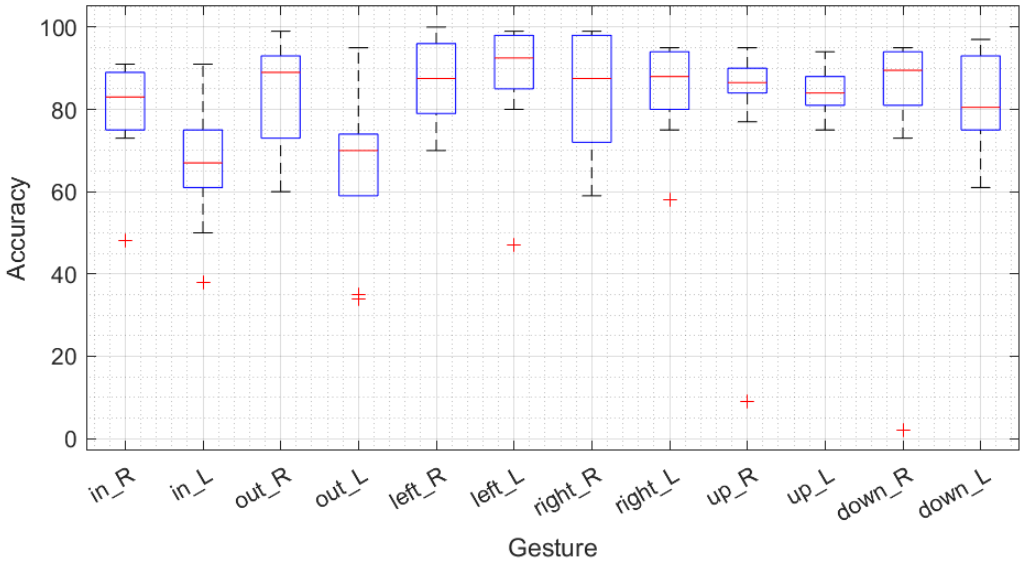


Figure 8.42: Box and whisker plot ten-fold cross-validation for the LSTM network for data set 6

Figure 8.43 and 8.44 show the performance of the network using the test data. The networks with the best and worst mean accuracy from the ten-fold cross-validation were used to illustrate the performance of the network.

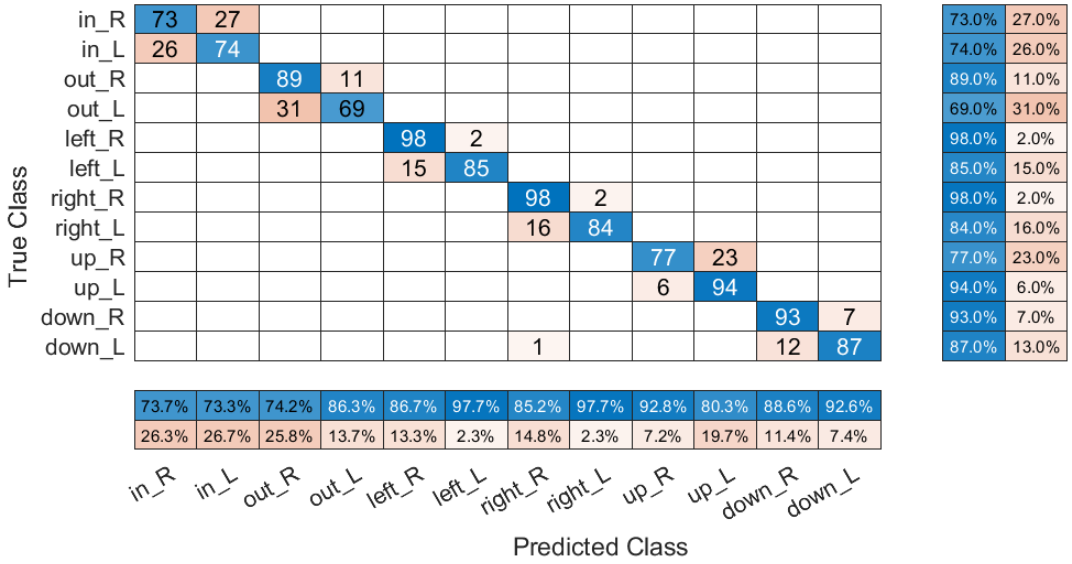


Figure 8.43: Performance of the network for the test data with the best classification accuracy from the ten-fold cross-validation for the LSTM network for data set 6

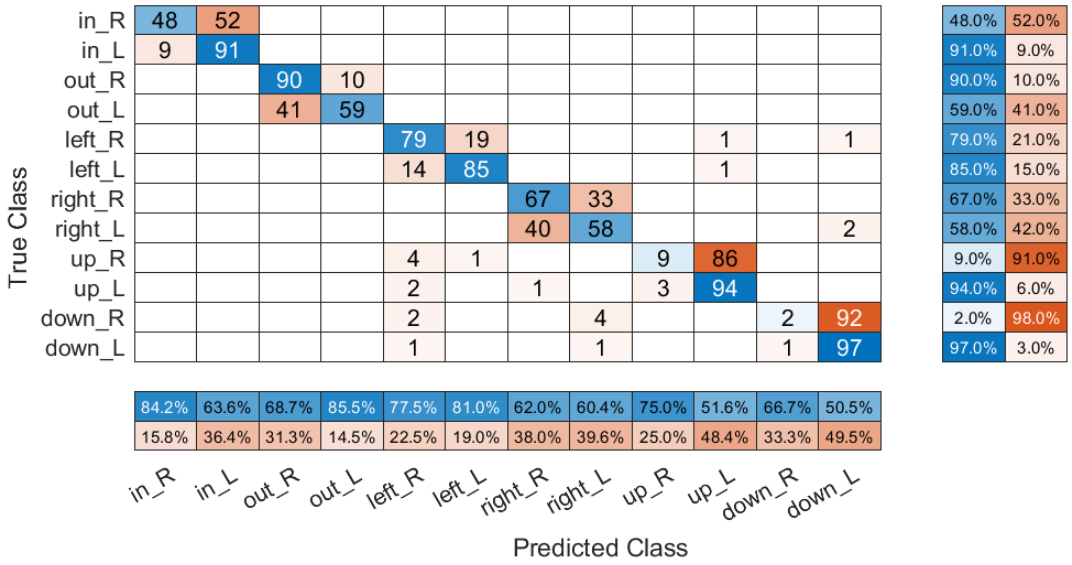


Figure 8.44: Performance of the network for the test data with the worst classification accuracy from the ten-fold cross-validation for the LSTM network for data set 6

Figure 8.45 shows the mean classification accuracy for each fold in the cross-validation as well as the mean classification accuracy across all the folds.

The network is trained again using both training and validation data as training data. There is no need for validation data since the performance of the network has already been validated. The network achieved a classification accuracy of 83.33% with more training data.

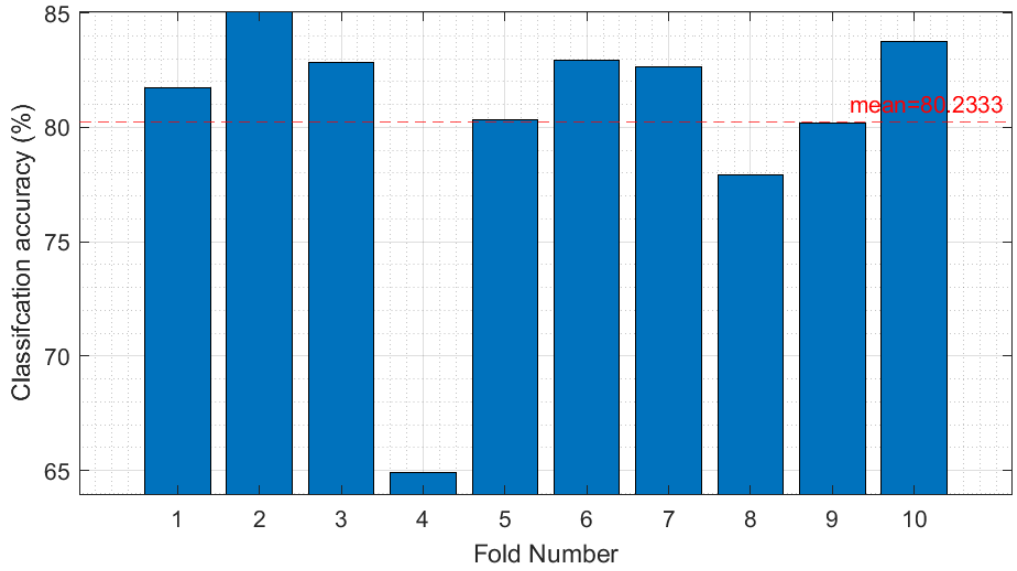


Figure 8.45: Mean classification accuracy for each cross-validation fold for the LSTM network for data set 6

8.8.2 CNN

Figure 8.46 is a box and whisker plot of the classification accuracy of ten-fold cross-validation of the LSTM network for each gesture. It shows that the lowest classification accuracy for any of the gestures to be 40% with the highest being 100%.

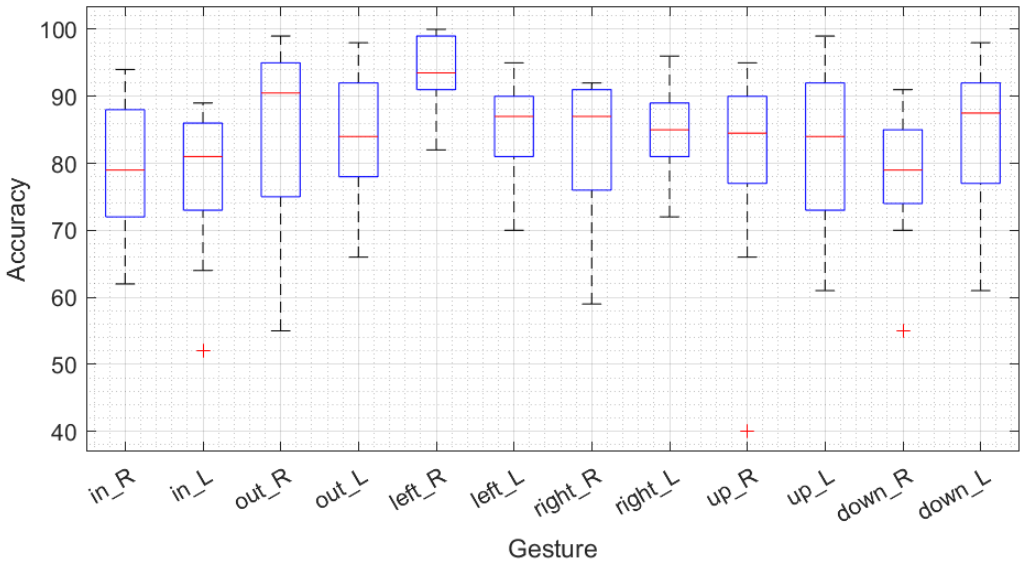


Figure 8.46: Box and whisker plot ten-fold cross-validation for the CNN for data set 6

Figure 8.47 and 8.48 show the performance of the network using the test data. The networks

with the best and worst mean accuracy from the ten-fold cross-validation were used to illustrate the performance of the network.

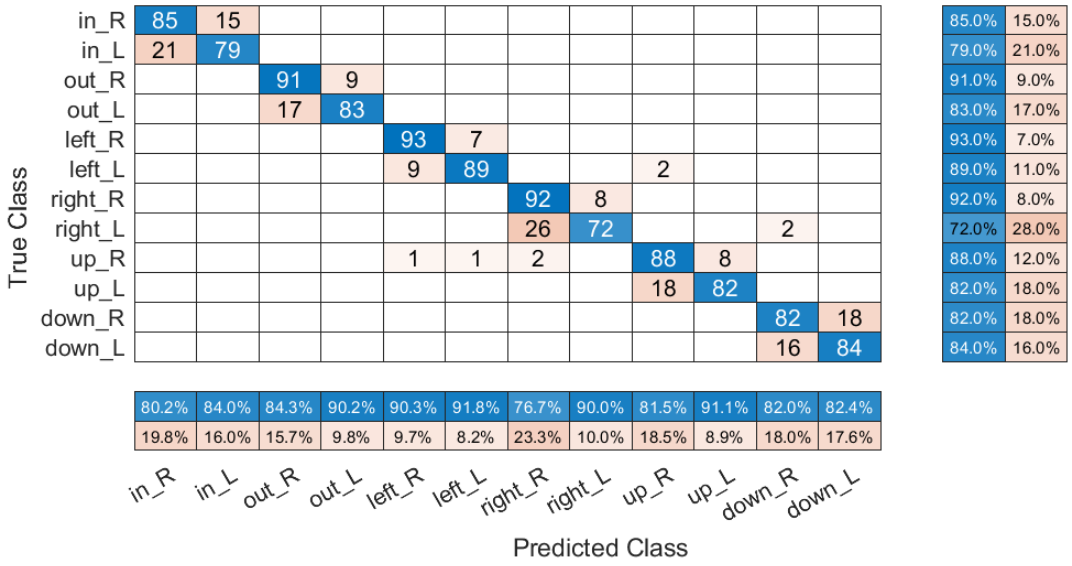


Figure 8.47: Performance of the network for the test data with the best classification accuracy from the ten-fold cross-validation for the CNN for data set 6

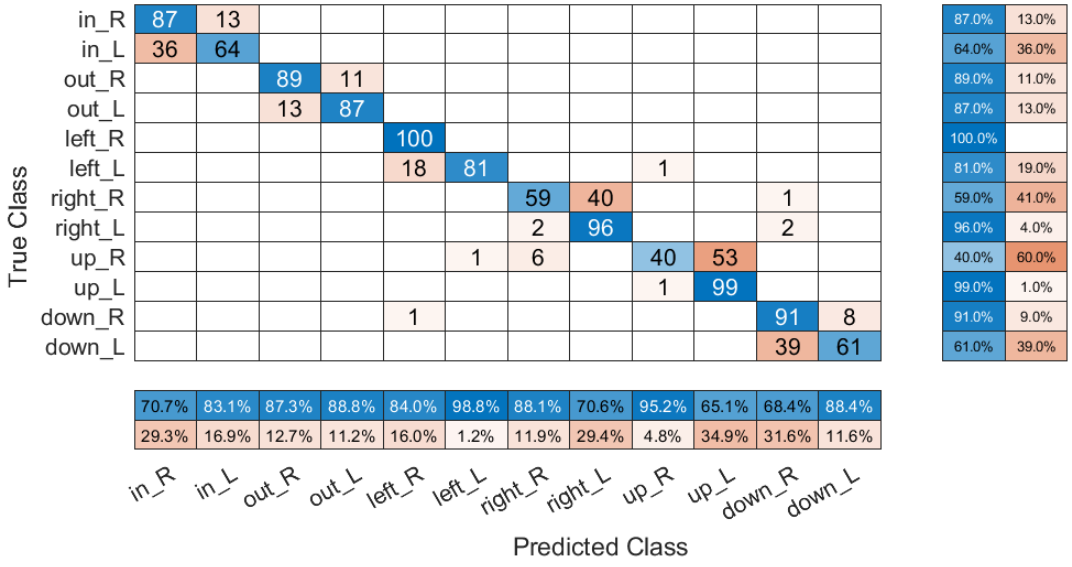


Figure 8.48: Performance of the network for the test data with the worst classification accuracy from the ten-fold cross-validation for the CNN for data set 6

Figure 8.49 shows the mean classification accuracy for each fold in the cross-validation as well as the mean classification accuracy across all the folds.

The network is trained again using both training and validation data as training data. There is no need for validation data since the performance of the network has already been validated.

The network achieved a classification accuracy of 83% with more training data.

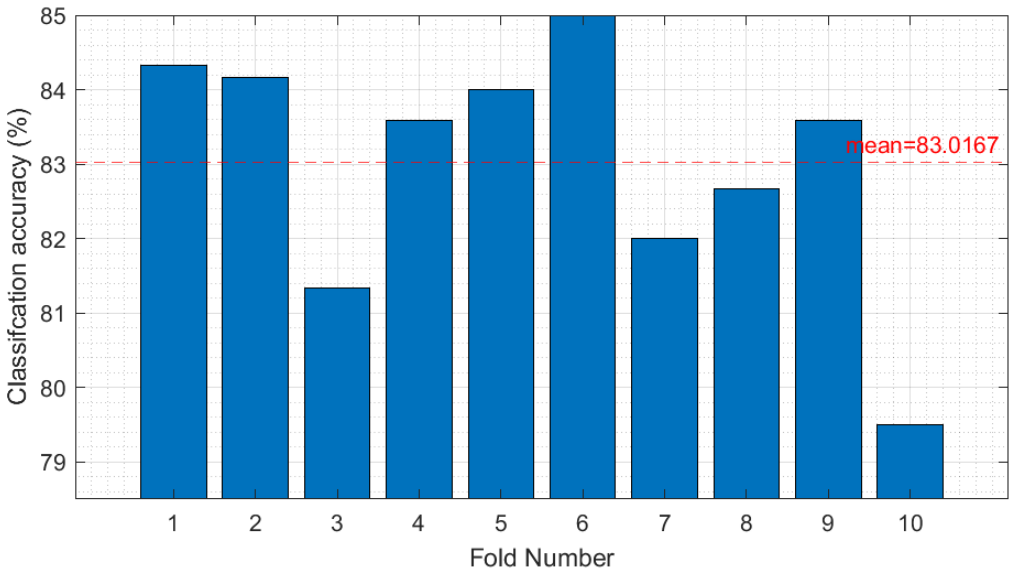


Figure 8.49: Mean classification accuracy for each cross-validation fold for the CNN for data set 6

8.9 Discussion on gestures performed on different hands

The final tests determined the effects of performing the same gesture on different hands. Gestures 1 through to 6 from table 3.1 were performed and recorded using the left hand and added to data set 1 to create data set 5. This data set makes no distinction between which hand performed which gesture. This test shows that using either left or right-hand does not negatively affect the performance of the system with the LSTM network having a mean classification accuracy of 98.62% and the CNN with an accuracy of 99.59%. This data set may actually be better than data set 1 since the slight differences between the left and right-hand gestures prevent the networks from overfitting.

The second test uses the same gesture set like the ones in data set 5 except the data has separate labels for gestures performed on the left and right hand. Analysing figure 8.43, 8.44, 8.47 and 8.48 shows that while the classification accuracy of both the networks decrease when using separating labels for each hand, most of the confusion is between which hand is used to perform the gestures and not with other gestures. The LSTM network was able to classify gestures with a mean accuracy of 80.23% and the CNN with 83.02%.

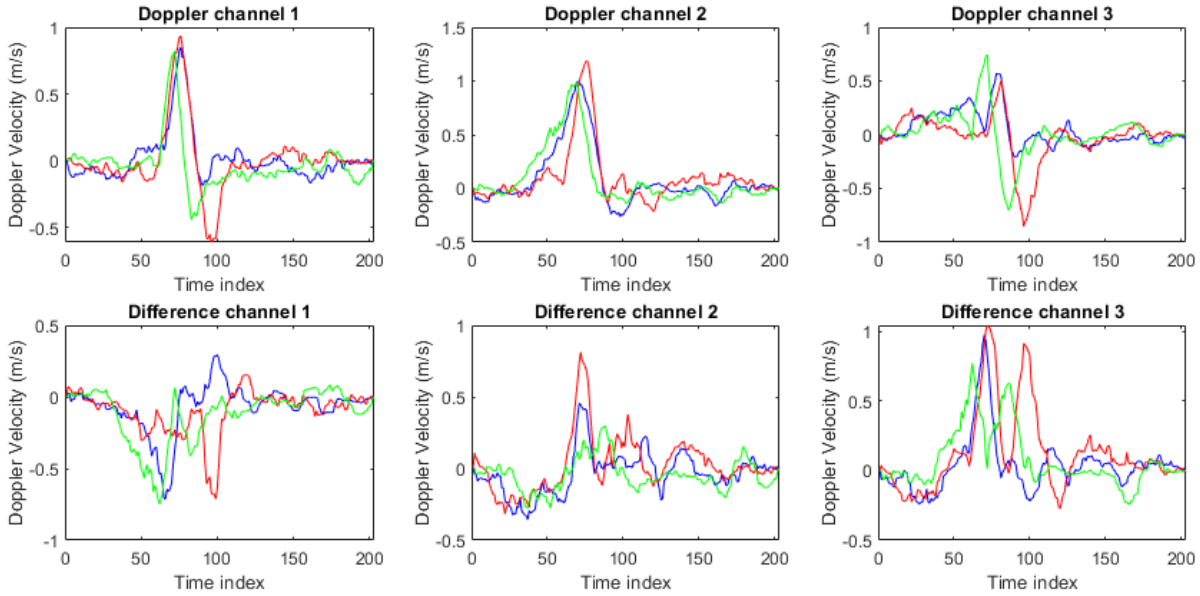


Figure 8.50: Comparison of 3 swiping left gestures; correctly classified using left hand (blue), correctly classified using right hand (red) and incorrectly classified using left hand (green)

Further analysis was performed on why the networks fail to classify certain gestures between left and right hands in figures 8.43, 8.44, 8.47 and 8.48. The 6 input channels of three of the swiping left gestures with classification predictions from the CNN were used to compare the gestures to one another. The three gestures from data set 6 that were analysed are, a correctly classified swiping left gesture with the left hand, correctly classified swiping left gesture with the right hand and a swiping left gesture with the left hand incorrectly classified as swiping left gesture with the right hand, which are shown in figure 8.50. Figure 8.50 shows that Doppler channel 1 and 2 are similar between all three gestures. Doppler channel 3 and difference channel 3 show that the left-handed gesture shared more characteristics with the right-hand gesture while only difference channel 1 shows that the left-hand gestures have similar features. Difference channel 2 shows all 3 of the gestures showing slightly different features. From this we see that out of the 6 features, 2 show that the left-handed gesture is similar to the right-handed gesture while there is only one feature showing that the gesture should be a left-handed gesture causing the incorrect classification. This occurs throughout the data set between left- and right-handed gestures using both the CNN and the LSTM network to classify the gestures which means that the incorrect classifications were not due to the networks but due to the characteristics of the gestures being similar.

8.10 Summary

Table 8.1 shows the mean classification accuracy using ten-fold cross-validation for each of the data sets in table 6.2. Table 8.2 shows the classification accuracy with no validation for each of the data sets in table 6.2.

	Mean Accuracy (%)	
Gesture set no.	LSTM	CNN
1	98.58	99.80
2	47.67	99.30
3	95.65	98.48
4	96.74	98.24
5	98.62	99.59
6	80.23	83.01

Table 8.1: Mean classification accuracy summary from the different data sets with ten-fold cross-validation

	Accuracy (%)	
Gesture set no.	LSTM	CNN
1	99.67	100.00
2	41.75	99.75
3	98.17	98.83
4	99.08	98.75
5	99.83	98.17
6	83.33	83.00

Table 8.2: Classification accuracy summary from the different data sets with no validation

The overall accuracy of the networks increases after combining the training data and validation data into one larger training data set from table 8.1 to table 8.2. Increasing the amount of data used for training allows for the networks to predict the gestures more accurately. The accuracy for data set 2 is however lower than the mean accuracy for the LSTM. This result is expected since there is a large variance in accuracy from fold to fold in the cross-validation, as seen in figure 8.12.

8.11 Acceptance test procedures Evaluation

8.11.1 ATP001: Deep network performance

8.11.1.1 LSTM

The LSTM only partially passes ATP001. The pass condition was for the network to be able to classify gestures with 80%. The LSTM network only manages to pass 5 out of the 6 tests to evaluate this ATP. The LSTM network fails to classify data set 2 with an accuracy of higher than 80%.

Gesture set no	Mean Accuracy	Pass/Fail
1	98.58	Pass
2	47.67	Fail
3	98.65	Pass
4	96.74	Pass
5	98.62	Pass
6	80.23	Pass

Table 8.3: ATP001 pass fail status for the LSTM network

8.11.1.2 CNN

The CNN passes ATP001. The lowest classification accuracy for the CNN is for data set 6 with an accuracy of 83.01% which is higher than the required 80%.

Gesture set no	Mean Accuracy	Pass/Fail
1	99.80	Pass
2	99.30	Pass
3	98.48	Pass
4	98.24	Pass
5	99.59	Pass
6	83.01	Pass

Table 8.4: ATP001 pass fail status for the CNN

8.11.2 ATP002: Range Test

8.11.2.1 LSTM

The LSTM network passes ATP002. The LSTM network can classify gestures up to 1 metre away with an accuracy higher than 80%.

Gesture set no	Mean Accuracy	Pass/Fail
3	98.65	Pass
4	96.74	Pass

Table 8.5: ATP002 pass fail status for the LSTM network

8.11.2.2 CNN

The CNN passes ATP002. The CNN can classify gestures up to 1 metre away with an accuracy higher than 80%.

Gesture set no	Mean Accuracy	Pass/Fail
3	98.48	Pass
4	98.24	Pass

Table 8.6: ATP002 pass fail status for the CNN

8.12 Additional Testing

8.12.1 Real-time test

Real-time tests were conducted to determine how long the system would take from sampling the hand gesture to classifying the gesture. Using the LSTM network, the system managed to sample a 2-second gesture and classify the gesture with an average time of 4.91 seconds. Using the CNN, the system managed to sample a 2-second gesture and classify the gesture with an average time of 5.02 seconds.

Chapter 9: Conclusion & Recommendations

9.1 Conclusion

A standalone Sonar system was successfully designed from the ground up for this study. The design of the system makes it possible to change the number of transmitters and receivers without having to redesign the whole system.

The signal processing used in this study was able to decrease the computational load as the dimension of the data is reduced while achieving an acceptable classification accuracy. The signal processing used in this study however limits the system to be able to only classify single target hand gestures.

According to the acceptance test procedures set out in section 3.3, the CNN passes all the requirements being able to classify gestures with at least an 80% accuracy at ranges up to 1 metre satisfying both ATP001 and ATP002.

The LSTM only partially passes the requirements as one of the data sets does not pass APT001. The LSTM failed to classify data set 2 with an accuracy of higher than 80% with an accuracy of 47.67% due to the nature of how recurrent neural networks operate. When it can classify the data sets with an accuracy of higher than 80%, it achieves a similar result to the CNN showing that it is a viable deep learning algorithm for specific gesture recognition applications.

The networks both achieve a high classification accuracy with the CNN achieving accuracy on 99.80% in ideal scenarios and the LSTM network achieving 98.58%. With both the networks being able to classify gestures performed on different hands with an accuracy of higher than 80%, the effective number of gestures that the system can classify doubles.

This study shows that deep learning techniques used in Radar applications can be applied to Sonar applications with a similar degree of success. This study also shows why CNN is the most widely used deep learning algorithm for gesture recognition. It can consistently classify gestures with a high level of accuracy. The LSTM network, however, shows a lot of promise as it can classify certain data sets with a level of accuracy similar to that of the CNN.

9.2 Recommendations

9.2.1 Hardware

The choice of the ultrasonic receivers was limited due to the COVID-19 pandemic. Further work can be done on investigating the use of the ultrasonic microphones to increase the receiver bandwidth.

The power supply used for the system is a dual supply bench power supply. A standalone custom power supply should be built for the system to allow for the device to be more portable.

The components for the PCB are all through-hole components. Building a PCB with surface mount components can reduce the overall footprint of the device.

The transmitter and receiver modules can be built onto separate daughter boards. This modularity would allow the system to be easily changed to fit different needs.

More work can be done in investigating the arrangement of the sensors such as changing the number of sensors used, the angle of the sensors and the distance between the sensors.

9.2.2 Sonar system

Additional work can be done investigating using the device with other Radar/Sonar techniques such as using FMCW or a pulse-Doppler.

9.2.3 Software

All of the signal processing and machine learning was done in MATLAB. While it is a versatile tool, the availability is very limited due to the cost of the software. Future work can be done by converting the MATLAB code into a different language such as Python to allow for the system to be more usable and accessible.

9.2.4 Signal processing

As mentioned, the signal processing limits the number of gestures the system can classify since it can only track single targets. Further work can be done to overcome this situation by applying

signal processing techniques other than the one used in this study. There are definite benefits in processing the signal further to extract features.

9.2.5 Deep learning

This work only explored using basic LSTM networks and CNN networks to classify gestures. More work can be done in exploring other machine learning algorithms to classify hand gestures. Other more complicated neural networks have been designed that yield very good classification results such as the VGG16 algorithm designed by the Visual Geometry Group from Oxford [66]. Using transfer learning to reuse the pre-trained VGG16 algorithm and re-tuning with the Sonar data could potentially yield interesting results.

The data for this study were collected from only one individual due to the restrictions from the COVID19-pandemic. The data set should be expanded to include gestures from multiple users to be able to better gauge the performance of the system. The number of gestures along with the complexity of the gestures can also be increased to test the limits of the system.

Using checkpoints to save the network when the training loss is minimised can improve the classification accuracy of the system. Using L2 regularisation may reduce the variance in the model to provide a better more stable network.

Bibliography

[1] A. Shrestha, H. Li, J. Le Kernec, and F. Fioranelli, "Continuous human activity classification from fmcw radar with bi-lstm networks," *IEEE Sensors Journal*, 2020.

[2] H. Kulhandjian, P. Sharma, M. Kulhandjian, and C. D'Amours, "Sign language gesture recognition using doppler radar and deep learning," in *2019 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–6.

[3] M. S. Hasan and H. Yu, "Innovative developments in hci and future trends," *International Journal of Automation and Computing*, vol. 14, no. 1, pp. 10–20, 2017.

[4] M. Z. Iqbal and A. Campbell, "The emerging need for touchless interaction technologies," *Interactions*, vol. 27, no. 4, pp. 51–52, 2020.

[5] "microsoft hololens," (accessed December 3, 2020). [Online]. Available: https://www.microsoft.com/en-us/hololens

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

[7] Z. Wang, Y. Hou, K. Jiang, W. Dou, C. Zhang, Z. Huang, and Y. Guo, "Hand gesture recognition based on active ultrasonic sensing of smartphone: a survey," *IEEE Access*, vol. 7, pp. 111 897–111 922, 2019.

[8] H. Cheng, L. Yang, and Z. Liu, "Survey on 3d hand gesture recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 9, pp. 1659–1673, 2016.

[9] Y. Kim and B. Toomajian, "Hand gesture recognition using micro-doppler signatures with convolutional neural network," *IEEE Access*, vol. 4, pp. 7125–7130, 2016.

[10] J.-W. Choi, S.-J. Ryu, and J.-H. Kim, "Short-range radar based real-time hand gesture recognition using lstm encoder," *IEEE Access*, vol. 7, pp. 33 610–33 618, 2019.

[11] S. Wang, J. Song, J. Lien, I. Poupyrev, and O. Hilliges, "Interacting with soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum," in *Proceedings*

of the 29th Annual Symposium on User Interface Software and Technology, 2016, pp. 851–860.

[12] X. Zabulis, H. Baltzakis, and A. A. Argyros, "Vision-based hand gesture recognition for human-computer interaction." *The universal access handbook*, vol. 34, p. 30, 2009.

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

[14] Z. Chen, G. Li, F. Fioranelli, and H. Griffiths, "Dynamic hand gesture classification based on multistatic radar micro-doppler signatures using convolutional neural network," in *2019 IEEE Radar Conference (RadarConf)*. IEEE, 2019, pp. 1–5.

[15] S. Skaria, A. Al-Hourani, M. Lech, and R. J. Evans, "Hand-gesture recognition using two-antenna doppler radar with deep convolutional neural networks," *IEEE Sensors Journal*, vol. 19, no. 8, pp. 3041–3048, 2019.

[16] K. Ling, H. Dai, Y. Liu, and A. X. Liu, "Ultragesture: Fine-grained gesture sensing and recognition," in *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2018, pp. 1–9.

[17] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311–324, 2007.

[18] A. R. Sarkar, G. Sanyal, and S. Majumder, "Hand gesture recognition systems: a survey," *International Journal of Computer Applications*, vol. 71, no. 15, pp. 26–37, 2013.

[19] A. Dekate, A. Kamal, and K. S. Surekha, "Magic glove - wireless hand gesture hardware controller," in *2014 International Conference on Electronics and Communication Systems (ICECS)*, 2014, pp. 1–4.

[20] P. Pławiak, T. Sośnicki, M. Niedźwiecki, Z. Tabor, and K. Rzecki, "Hand body language gesture recognition based on signals from specialized glove and machine learning algorithms," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 3, pp. 1104–1113, 2016.

[21] C. Preetham, G. Ramakrishnan, S. Kumar, A. Tamse, and N. Krishnapura, "Hand talk-implementation of a gesture recognizing glove," in *2013 Texas Instruments India Educators' Conference*, 2013, pp. 328–331.

[22] K. S. Abhishek, L. C. F. Qubeley, and D. Ho, "Glove-based hand gesture recognition sign language translator using capacitive touch sensor," in *2016 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC)*, 2016, pp. 334–337.

[23] P. Garg, N. Aggarwal, and S. Sofat, "Vision based hand gesture recognition," *World academy of science, engineering and technology*, vol. 49, no. 1, pp. 972–977, 2009.

[24] Y. Zhu, Z. Yang, and B. Yuan, "Vision based hand gesture recognition," in *2013 International Conference on Service Sciences (ICSS)*. IEEE, 2013, pp. 260–265.

[25] J. Singha, A. Roy, and R. H. Laskar, "Dynamic hand gesture recognition using vision-based approach for human–computer interaction," *Neural Computing and Applications*, vol. 29, no. 4, pp. 1129–1141, 2018.

[26] M. Patrick, "Capturing 3d images with time-of-flight camera technology - industry articles," 2018 (accessed August 12, 2020). [Online]. Available: https://www.allaboutcircuits.com/industry-articles/capturing-3d-images-with-tof-camera-technology/

[27] "DOMI ToF Sensor," 2020 (accessed August 12, 2020). [Online]. Available: https://www.domisensor.com/

[28] J. Molina, M. Escudero-Viñolo, A. Signoriello, M. Pardàs, C. Ferrán, J. Bescós, F. Marqués, and J. M. Martínez, "Real-time user independent hand gesture recognition from time-of-flight camera video using static and dynamic models," *Machine vision and applications*, vol. 24, no. 1, pp. 187–204, 2013.

[29] E. Kollorz, J. Penne, J. Hornegger, and A. Barke, "Gesture recognition with a time-of-flight camera," *International Journal of Intelligent Systems Technologies and Applications*, vol. 5, no. 3-4, pp. 334–343, 2008.

[30] "Azure Kinect DK - Develop AI Models — Microsoft Azure," 2020 (accessed August 12, 2020). [Online]. Available: https://azure.microsoft.com/en-us/services/kinect-dk/

[31] Z. Ren, J. Yuan, J. Meng, and Z. Zhang, "Robust part-based hand gesture recognition using kinect sensor," *IEEE transactions on multimedia*, vol. 15, no. 5, pp. 1110–1120, 2013.

[32] Y. Li, "Hand gesture recognition using kinect," in *2012 IEEE International Conference on Computer Science and Automation Engineering*. IEEE, 2012, pp. 196–199.

[33] G. Marin, F. Dominio, and P. Zanuttigh, "Hand gesture recognition with leap motion and kinect devices," in *2014 IEEE International conference on image processing (ICIP)*. IEEE, 2014, pp. 1565–1569.

[34] "World-leading hand tracking: Small. Fast. Accurate. — Ultraleap," 2020 (accessed August 12, 2020). [Online]. Available: https://www.ultraleap.com/tracking/

[35] W. Lu, Z. Tong, and J. Chu, "Dynamic hand gesture recognition with leap motion controller," *IEEE Signal Processing Letters*, vol. 23, no. 9, pp. 1188–1192, 2016.

[36] L. E. Potter, J. Araullo, and L. Carter, "The leap motion controller: a view on sign language," in *Proceedings of the 25th Australian computer-human interaction conference: augmentation, application, innovation, collaboration*, 2013, pp. 175–178.

[37] A. Colgan, "How does the leap motion controller work?" 2014 (accessed August 12, 2020). [Online]. Available: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/

[38] "Tracking — Leap Motion Controller — Ultraleap," 2020 (accessed August 12, 2020). [Online]. Available: https://www.ultraleap.com/product/leap-motion-controller/

[39] M. Richards, W. Holm, and J. Scheer, *Principles of Modern Radar: Basic Principles, Volume 1*, ser. Electromagnetics and Radar. Institution of Engineering and Technology, 2010. [Online]. Available: https://books.google.co.za/books?id=nD7tGAAACAAJ

[40] H. Li, X. Liang, A. Shrestha, Y. Liu, H. Heidari, J. Le Kernec, and F. Fioranelli, "Hierarchical sensor fusion for micro-gesture recognition with pressure sensor array and radar," *IEEE Journal of Electromagnetics, RF and Microwaves in Medicine and Biology*, vol. 4, no. 3, pp. 225–232, 2020.

[41] A. Shrestha, C. Murphy, I. Johnson, A. Anbulselvam, F. Fioranelli, J. Le Kernec, and S. Z. Gurbuz, "Cross-frequency classification of indoor activities with dnn transfer learning," in *2019 IEEE Radar Conference (RadarConf)*. IEEE, 2019, pp. 1–6.

[42] Y. Wang, S. Wang, M. Zhou, Q. Jiang, and Z. Tian, "Ts-i3d based hand gesture recognition method with radar sensor," *IEEE Access*, vol. 7, pp. 22 902–22 913, 2019.

[43] "Soli: Home," 2020 (accessed August 26, 2020). [Online]. Available: https://atap.google.com/soli/

[44] "Soli: Technology," 2020 (accessed August 26, 2020). [Online]. Available: https://atap.google.com/soli/technology/

[45] S. Z. Gurbuz, A. C. Gurbuz, E. A. Malaia, D. J. Griffin, C. Crawford, M. M. Rahman, R. Aksu, E. Kurtoglu, R. Mdrafi, A. Anbuselvam *et al.*, "A linguistic perspective on radar micro-doppler analysis of american sign language," in *2020 IEEE International Radar Conference (RADAR)*. IEEE, 2020, pp. 232–237.

[46] T. Fan, C. Ma, Z. Gu, Q. Lv, J. Chen, D. Ye, J. Huangfu, Y. Sun, C. Li, and L. Ran, "Wireless hand gesture recognition based on continuous-wave doppler radar sensors," *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 11, pp. 4012–4020, 2016.

[47] M. Richards, W. Melvin, J. Scheer, J. Scheer, and W. Holm, *Principles of Modern Radar: Radar Applications, Volume 3*, ser. Electromagnetics and Radar. Institution of Engineering and Technology, 2013. [Online]. Available: https://books.google.co.za/books?id=tYfYAgAAQBAJ

[48] H.-S. Yeo and A. Quigley, "Radar sensing in human-computer interaction," *interactions*, vol. 25, no. 1, pp. 70–73, 2017.

[49] K. Kalgaonkar and B. Raj, "One-handed gesture recognition using ultrasonic doppler sonar," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 1889–1892.

[50] "Ultrasonic mic board knowles spu0410lr5h-qb - micbooster.com," (accessed September 17, 2020). [Online]. Available: https://micbooster.com/ultrasonic-microphones/146-ultrasonic-mic-board.html

[51] H. Chen, F. Li, and Y. Wang, "Echotrack: Acoustic device-free hand tracking on smart phones," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[52] R. Kashyap, I. Singh, and S. S. Ram, "Micro-doppler signatures of underwater vehicles using acoustic radar," in *2015 IEEE Radar Conference (RadarCon)*. IEEE, 2015, pp. 1222–1227.

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

[54] J. Brownlee, "What is the difference between test and validation datasets?" 2017, (accessed October 10, 2020). [Online]. Available: https://machinelearningmastery.com/difference-test-validation-datasets/

[55] ——, "What is the difference between a parameter and a hyperparameter?" 2017, (accessed March 26, 2021). [Online]. Available: https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/

[56] ——, "A gentle introduction to k-fold cross-validation," 2018, (accessed October 10, 2020). [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/#:~:text=Cross%2Dvalidation%20is%20a%20resampling,k%2Dfold%20cross%2Dvalidation.

[57] ——, "How do convolutional layers work in deep learning neural networks?" 2019, (accessed March 26, 2021). [Online]. Available: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

[58] C. Olah, "Conv nets: A modular perspective - colah's blog," 2014 (accessed September 18, 2020). [Online]. Available: http://colah.github.io/posts/2014-07-Conv-Nets-Modular/

[59] ——, "Understanding lstm networks – colah's blog," 2015 (accessed September 18, 2020). [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[60] "ultrasonic transmitter & receiver pair." [Online]. Available: https://www.electronicscomp.com/ultrasonic-transnitter-receiver-40khz-transducer-india

[61] 2019 (accessed September 29, 2020). [Online]. Available: https://www.pjrc.com/store/teensy40.html

[62] "stm32f4discovery - stmicroelectronics." [Online]. Available: https://www.st.com/en/evaluation-tools/stm32f4discovery.html

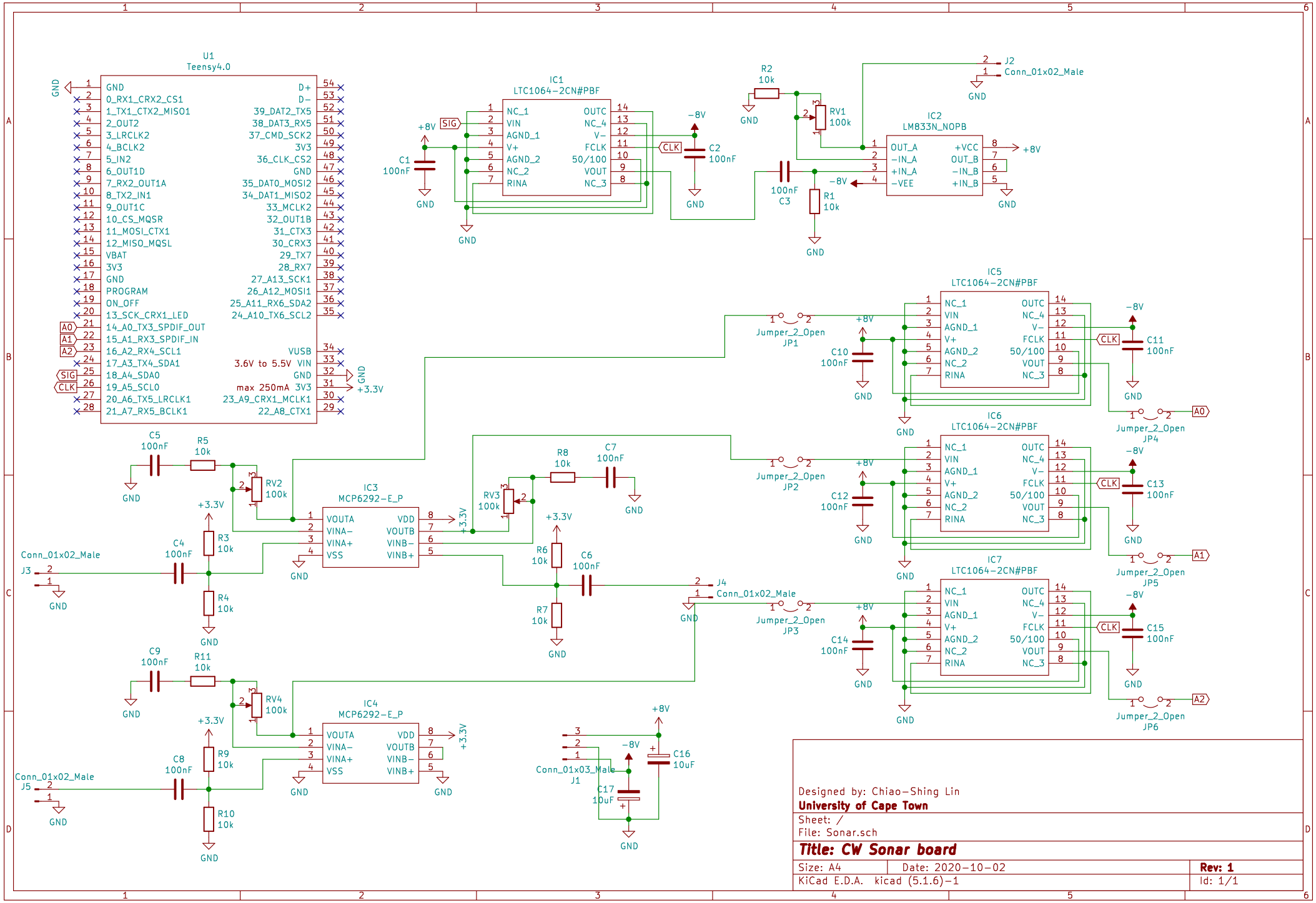
[63] S. W. Smith, *The scientist and engineer's guide to digital signal processing*, 1st ed. California Technical Pub., 1997.

[64] "butterworth filter design," (accessed October 1, 2020). [Online]. Available: https://www.electronics-tutorials.ws/filter/filter_8.html

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

[66] F. Chollet, "How convolutional neural networks see the world," 2016, (accessed November 27, 2020). [Online]. Available: https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html

Appendix A: Circuit Schematic



Designed by: Chiao−Shing Lin
University of Cape Town

Sheet: /
File: Sonar.sch

Title: CW Sonar board

Size: A4	Date: 2020−10−02	**Rev: 1**
KiCad E.D.A. kicad (5.1.6)−1		Id: 1/1

Appendix B: PCB Testing

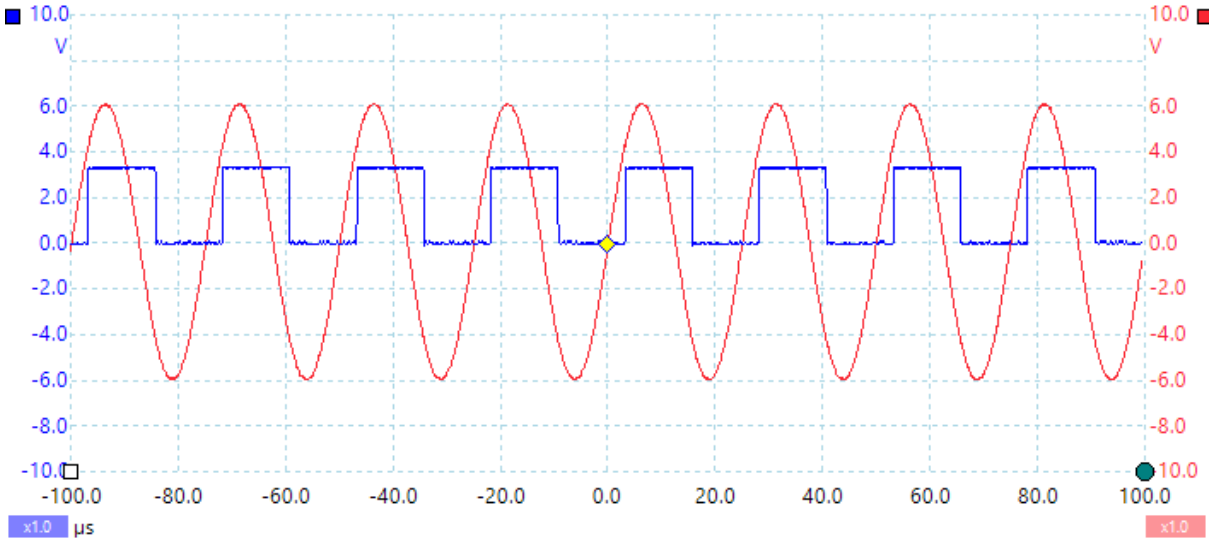


Figure B.1: Transmitter circuit test (input: blue, output: red)

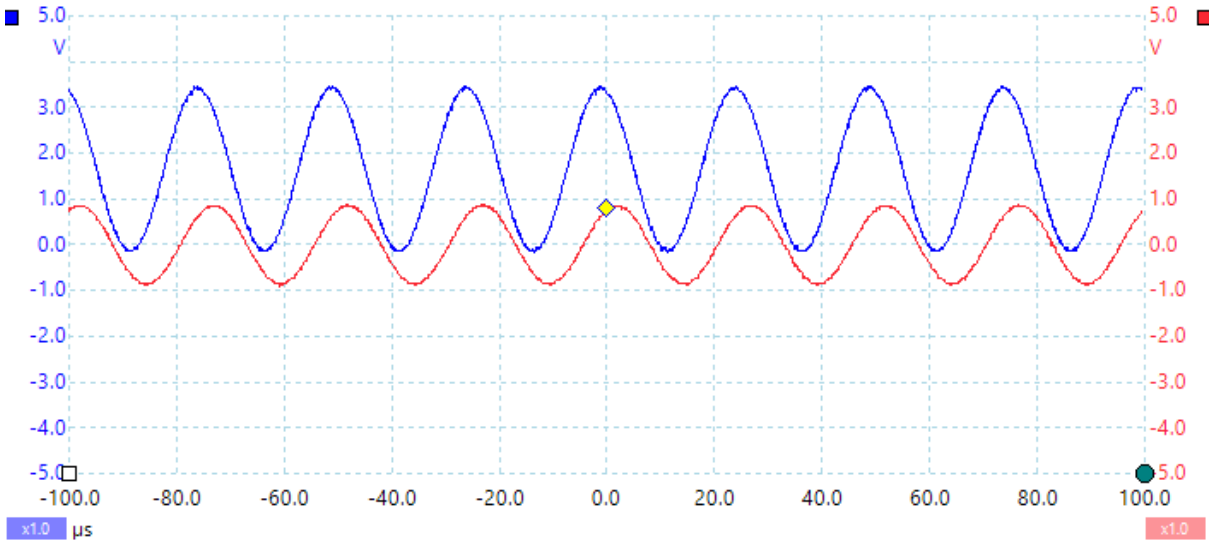


Figure B.2: Receiver circuit 1 test (input: red, output: blue)

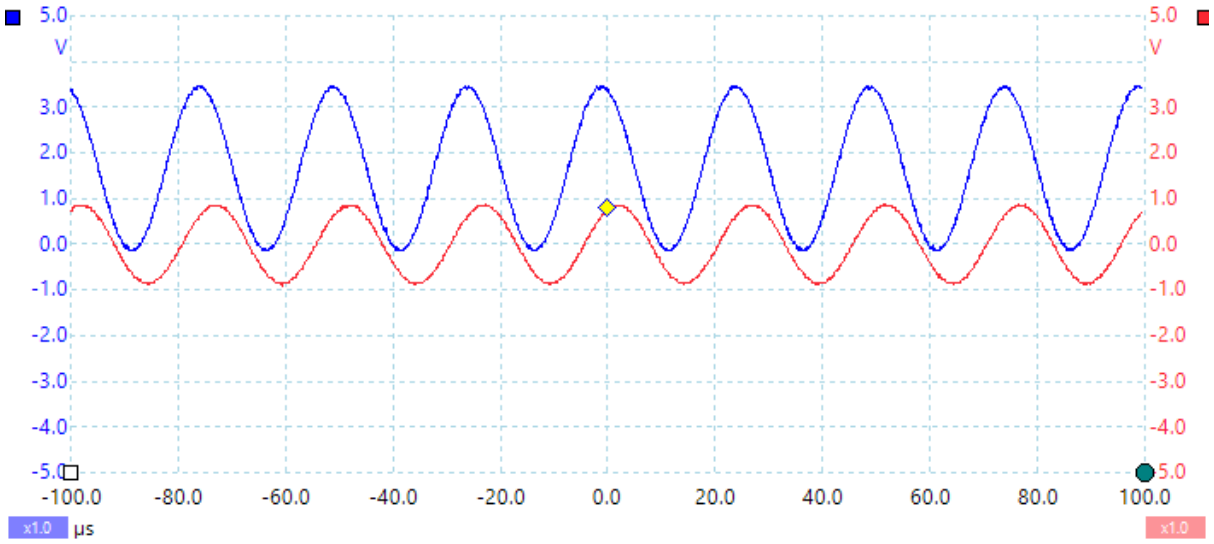


Figure B.3: Receiver circuit 2 test (input: red, output: blue)

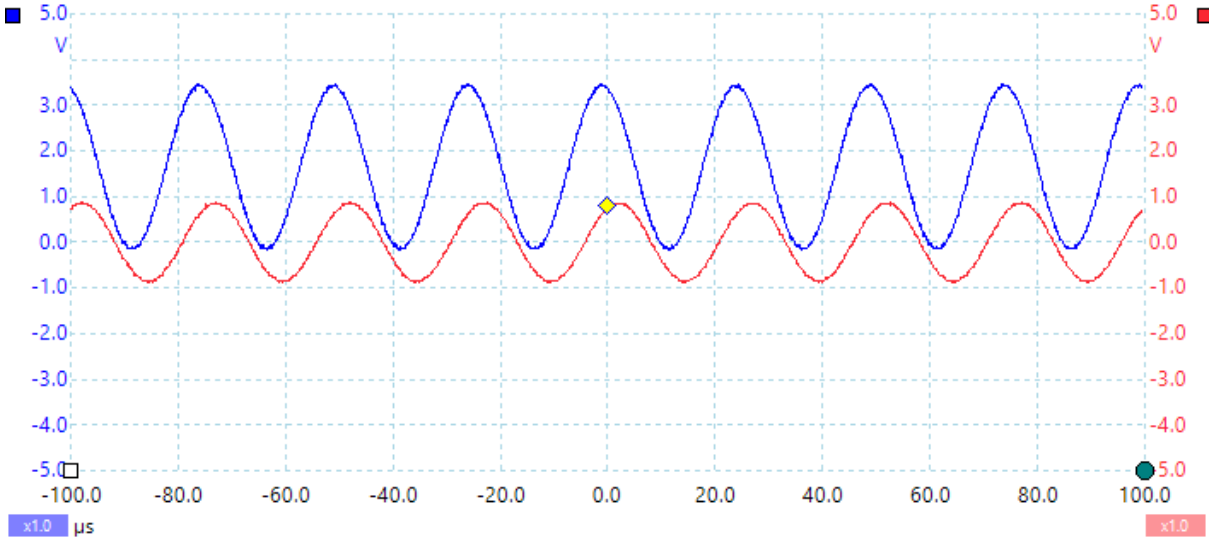
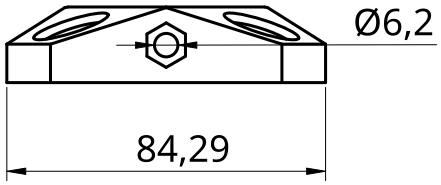
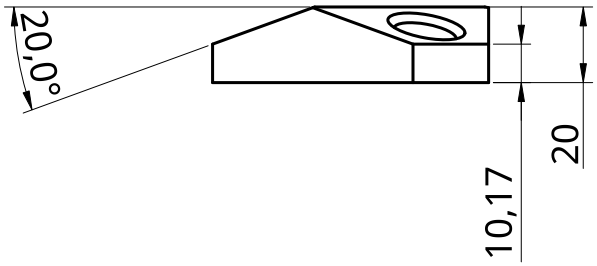
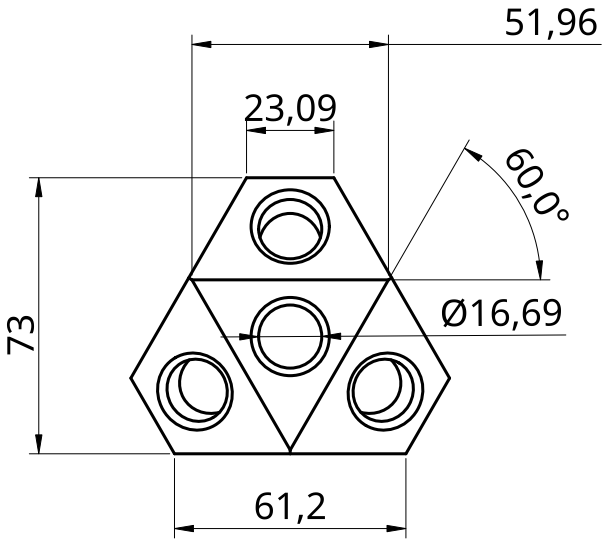


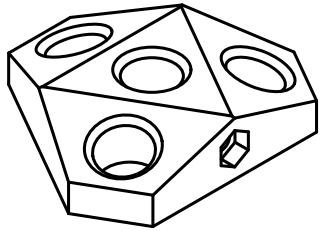
Figure B.4: Receiver circuit 3 test (input: red, output: blue)

Appendix C: Sensor Mount









	NAME	SIGNATURE	DATE
DRAWN	CHIAO-SHING LIN		2020-08-29
CHECKED			
APPROVED			

MATERIAL	FINISH
PLA+	N/A

TITLE: Sonar Sensor Mount

SIZE	DWG NO.	REV.
A4	1	1

SCALE	WEIGHT	SHEET
1:2	50g	1 of 1

Appendix D: Data sheets

KOBITONE™ AUDIO COMPANY

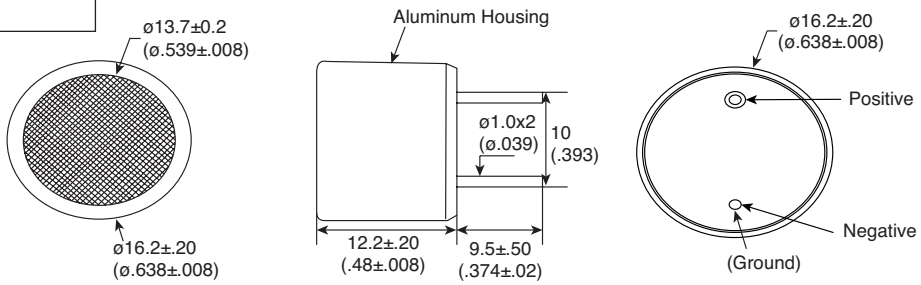
Ultrasonic Transducers

Date: 07/19/18

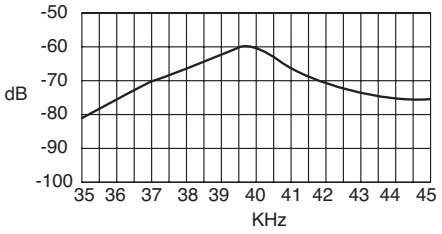
Dimensions: mm (in.)

Part Number:

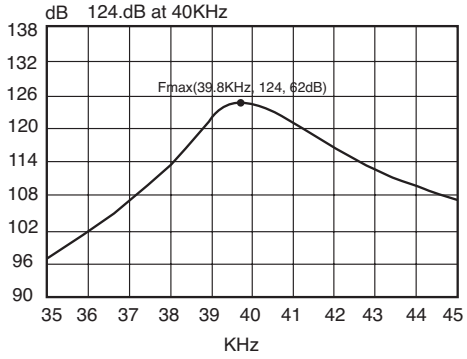
255-400ST16-ROX
255-400SR16-ROX



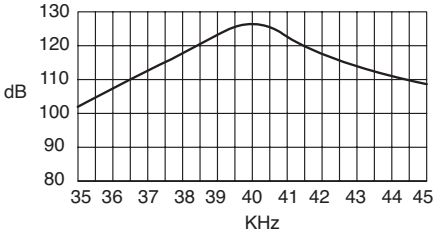
Sensitivity for 255-400SR16-ROX



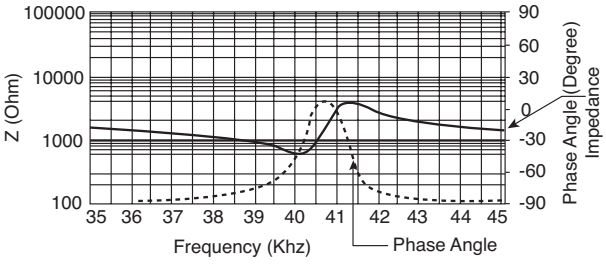
Transmitting Sound Pressure Level (0dB=0.0002μbar)



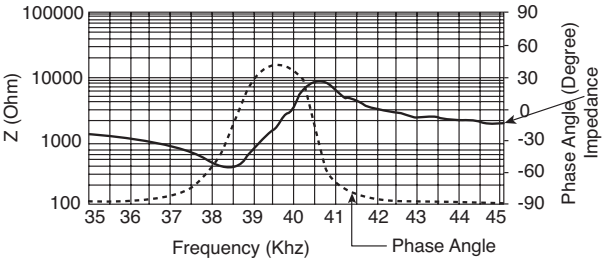
Sound Pressure Level for 255-400ST16-ROX



Impedance/Phase Angle vs. Frequency for 255-400ST16-ROX
Tested under 1Vrms Oscillation Level



Impedance/Phase Angle vs. Frequency for 255-400SR16-ROX
Tested under 1Vrms Oscillation Level



KT-400244

Electrical Specifications:

- Center Frequency (f_0): 40.0KHz±1.0KHz
- SPL @ f_0 for 255-400ST16-ROX: 120dB at 40KHz @ 0dB re 0.0002μ bar)
- Sensitivity @ f_0 for 255-400SR16-ROX: -61dB (0dB=1V/μ bar)
- Bandwidth (-6dB): 255-400ST16-ROX - 2KHz; 255-400SR16-ROX - 2.5KHz
- Capacitance: 2,400pF±20%

Mechanical Specifications:

- Type: Transmitter - 255-400ST16-ROX; Receiver - 255-400SR16-ROX
- Operating Temperature: -30°C ~ +70°C
- Storage Temperature: -40°C ~ +80°C

Note:

- RoHS Compliant by Exemption

MEC

SPECIFICATION OF ULTRASONIC SENSOR UNIT

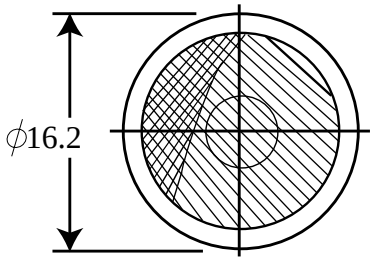
PART NO. : **UR1612MP**

E L E C T R I C A L C H A R A C T E R I S T I C S

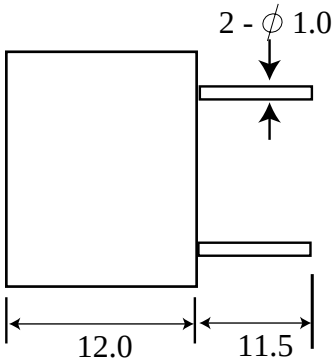
1.	Construction	Open Structure
2.	Function	Receiver
3.	Center Frequency	40kHz +/- 1kHz
4.	Overall Sensitivity	-61 dB min. [0dB = 0.02mPa] at 10Vrms, 30cm
5.	Direction	80 Degree max.
6.	Capacitance	2,000 pF +/- 30 % at 1kHz
7.	Detectable Range	0.2 ~ 4.0 meter
8.	Housing Material	Aluminium
9.	Colour	Silver
10.	Operating Temperature	-20 ~ +70°C
11.	Weight	5.0 gram

D I M E N S I O N s (mm)

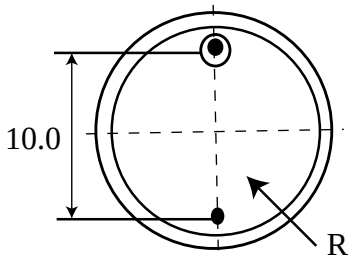
TOP VIEW



SIDE VIEW



BOTTOM VIEW



MOBICON
Electronic Components

Prepared By: Leo Wong
DOC. No: UR1612MP

Appendix E: Code

All the code used for this dissertation can be found at the following GitHub page:
`https://github.com/AngryPingu1996/Masters`