

HAND ACTIVITY RECOGNITION USING A WEARABLE SMART GLOVE

John Neuman Gayan Samarasinghe

(188463H)

Degree of Master of Science in Electronics and Automation

Department of Electronic and Telecommunication Engineering

University of Moratuwa

Sri Lanka

December 2020

HAND ACTIVITY RECOGNITION USING A WEARABLE SMART GLOVE

John Neuman Gayan Samarasinghe

(188463H)

Thesis submitted in partial fulfillment of the requirements for the degree
Master of Science in Electronics and Automation

Department of Electronic and Telecommunication Engineering

University of Moratuwa
Sri Lanka

December 2020

DECLARATION OF THE CANDIDATE & SUPERVISOR

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: _____ Date: _____

The above candidate has carried out research for the Master's thesis under our supervision.

Name of the supervisor: Dr. Chamira Edussooriya

Signature of Supervisor: _____ Date: _____

Name of the supervisor: Dr. Ranga Rodrigo

Signature of Supervisor: _____ Date: _____

ABSTRACT

This project is aimed at designing, simulating and constructing a wearable device capable of performing activity recognition to track and monitor activities specific to the manufacturing industry.

This was done by designing data capturing glove to capture all necessary signals from the human body and provide necessary filtering to obtain low noise data. This is then passed through suitable pre-processing algorithms to create distinguishing features between activities. The best suited classification and post-processing algorithms were then designed and implemented to classify the captured data in to a specified set of activities.

The device was designed with an ESP8266 and a Raspberry Pi coded in C++ and Python respectively. Accelerometer & gyroscope sensors were used to collect data from the human body while a number of classical machine learning algorithms and convolutional neural networks were tested to classify the data.

For the activities pointing, wiping, tightening, loosening, picking, holding, pulling, pushing, hammering, walking, holding and walking and turning, the system was capable of classifying the test data with accuracies between 86% - 91%. The null set was classified with an accuracy of 100% with support vector machines with a linear kernel and the post processing algorithm. The same algorithm reached an accuracy of 91.3% for the activity classification while the support vector machine with RBF kernel and post processing algorithm reached an accuracy of 89.7%. The convolutional neural network trained on pre-processed 3D activity images and the post processing algorithm reached an accuracy of 86.2%.

The successfully created device will be used to obtain necessary analysis in the manufacturing space to optimize performance of the workers.

Key Words: Hand, Activity Recognition, Machine Learning, Convolutional Neural Network, Kalman Filter, Manufacturing, Industry

ACKNOWLEDGEMENT

My master's degree and research would not have been possible if it was not for the following people. I would like to thank;

My supervisors, Dr. Chamira Edussooriya and Dr. Ranga Rodrigo for their constant support, guidance and patience. Their constant back up and check and adjust in moving forward with this research and thesis has helped me immensely.

The academic and administrative staff of the Department of Electronic and Telecommunication Engineering, University of Moratuwa for their input throughout this MSc Program.

My fellow students in the same MSc Program who have worked together with me to get through many hurdles throughout the program

My wife, Gayathri Karunaratne, who has been a constant source of support & encouragement during the challenges of work, student work and life. I am truly thankful for having you in my life.

My parents, Dr. Merrill and Dr. Jayanthi Samarasinghe, who have always loved me unconditionally and whose good examples have taught me to work hard for the things that I aspire to achieve.

My friends and family who have offered their help in the data collection process

TABLE OF CONTENTS

<i>Declaration of the candidate & Supervisor</i>	<i>i</i>
<i>Abstract</i>	<i>ii</i>
<i>Acknowledgement</i>	<i>iii</i>
<i>Table of Contents</i>	<i>iv</i>
<i>List of Figures</i>	<i>vii</i>
<i>List of Tables</i>	<i>x</i>
<i>List of Abbreviations</i>	<i>xi</i>
<i>List of Appendices</i>	<i>xii</i>
<i>1. Introduction</i>	<i>1</i>
<i>1.1. Background Information</i>	<i>1</i>
<i>1.2. Project Description</i>	<i>2</i>
1.2.1. Project Scope	2
1.2.2. Initial Assumptions	3
1.2.3. Hypothesis / Project Contribution	3
<i>1.3. Activity detection scope</i>	<i>3</i>
<i>1.4. Project Architecture</i>	<i>5</i>
<i>2. Literature Review</i>	<i>6</i>
<i>2.1. Types of Activity Recognition</i>	<i>6</i>
<i>2.2. Similar Research</i>	<i>8</i>
<i>3. Smart Glove Design</i>	<i>12</i>
<i>3.1. Sensors</i>	<i>12</i>
3.1.1. Sensor Selection	12
3.1.2. Sensor Placement	14
3.1.3. Inertial Measurement Unit (IMU)	16

3.2. Processors	17
3.2.1. Glove Processor	18
3.2.2. Classification processor	18
3.3. Glove	19
3.4. Circuit Design	19
3.5. Algorithm	21
3.5.1. Glove Algorithm	21
3.5.2. Classification Processor Algorithm	22
3.6. Data Filtering	24
3.6.1. Kalman Filter	25
3.6.2. Verification of angles	28
3.7. Wi-Fi Communication	31
3.8. Data Pre-processing	32
3.8.1. Data Preparation	32
3.8.2. Pre-processing: Stage 1	35
3.8.3. Pre-processing: Stage 2	38
3.8.4. Feature Scaling	45
4. Learning Model Development and Results	46
4.1. Data Collection	46
4.2. Classical Machine Learning	47
4.2.1. Error Analysis	48
4.2.2. Base Performance	50
4.2.3. Window Size	52
4.2.4. Manual Dimensionality Reduction	53
4.2.5. PCA Dimensionality Reduction	55
4.3. Null Set Classification	56

4.4.	Convolutional Neural Networks	57
4.4.1.	Base Performance	57
4.4.2.	Window Size	60
4.4.3.	2 Channel Input Network	61
4.4.4.	Error analysis	62
4.5.	Post Classification Processing	69
5.	Overall Performance Results	73
5.1.	Hardware Performance	73
5.2.	Power Performance	74
5.3.	Machine Learning Performance	75
6.	Conclusion	78
7.	References	79
8.	Appendices	88

LIST OF FIGURES

Figure 1-1 - Project architecture.....	5
Figure 3-1 –Components of the smart glove design.....	12
Figure 3-2 – Illustration of finger joints.....	15
Figure 3-3 - GY521 board & custom-built breakout board size comparison [original in color].....	17
Figure 3-4 – Prototype 1 design [original in color].....	19
Figure 3-5 – Porotype 2 intermediate design [original in color].....	19
Figure 3-6 – Glove circuit diagram.....	20
Figure 3-7 –Glove algorithm.....	21
Figure 3-8 –Processor state diagram.....	23
Figure 3-9 - Glove dashboard v2 [Original in color].....	23
Figure 3-10 – Accelerometer & Gyroscope on a flat table (knock at t=0).....	25
Figure 3-11 – Kalman filter performance.....	28
Figure 3-12 – Drift of signals while stationary on flat surface. Yaw axes drift with time.....	28
Figure 3-13 - IMU testing setup [original in color].....	29
Figure 3-14 – Verification of roll angles against fixed reference.....	29
Figure 3-15 – Verification of pitch angles against fixed reference.....	29
Figure 3-16 – Roll stability over a period of 25 seconds.....	30
Figure 3-17 – Verification of roll angle accuracy in rapid motion.....	30
Figure 3-18 – Stages of Data Pre-processing.....	34
Figure 3-19 – Finger angle determination.....	35
Figure 3-20 – Finger angle determination.....	37
Figure 4-1 - Classical ML F1 Score (Base Case) 30 Window, Level 1 Pre-process.....	51
Figure 4-2 – k-Nearest Neighbor performance for k values 3,5,10 and 20 for all participants.....	52
Figure 4-3 - Classical ML F1 Score 20 Window, Level 1 Pre-process.....	52
Figure 4-4 - Classical ML F1 Score Comparison for Window Lengths.....	53
Figure 4-5 – Manual dimensionality reduction level comparison.....	54

<i>Figure 4-6 – Manual dimensionality reduction level to level improvement.</i>	55
<i>Figure 4-7 – 10 Highest contributing dimensions with PCA</i>	56
<i>Figure 4-8 – Null set performance for each of the selected algorithms</i>	57
<i>Figure 4-9 – Base CNN structure</i>	58
<i>Figure 4-10 – Base CNN performance</i>	59
<i>Figure 4-11 – Base CNN F1 scores</i>	59
<i>Figure 4-12 – Base CNN performance</i>	60
<i>Figure 4-13 – Window 20 vs. 30 CNN F1 scores</i>	60
<i>Figure 4-14 – 2 Channel input CNN performance.</i>	61
<i>Figure 4-15 – 2-channel vs. 3-channel input F1 scores.</i>	61
<i>Figure 4-16 – 1-CNN structure.</i>	62
<i>Figure 4-17 – 1-CNN vs base case scenario F1 scores</i>	62
<i>Figure 4-18 – 1- CNN performance.</i>	64
<i>Figure 4-19 – 1-CNN performance over 500 epochs.</i>	64
<i>Figure 4-20 – 1-CNN performance with sigmoid activation</i>	64
<i>Figure 4-21 – 1-CNN performance with slower learning rate (0.0001)</i>	65
<i>Figure 4-22 – 1-CNN performance with Adadelat optimizer</i>	65
<i>Figure 4-23 – 1-CNN performance with Adagrad optimizer.</i>	65
<i>Figure 4-24 – 1-CNN performance with input layer dropout regularization</i>	66
<i>Figure 4-25 – 1-CNN performance with hidden layer dropout regularization.</i>	66
<i>Figure 4-26 – 1-CNN performance with optimizer tuning and DRR=0.01</i>	67
<i>Figure 4-27 – 1-CNN comparison with Adagrad optimizer and DRR=0.01</i>	67
<i>Figure 4-28 – Combined 1-CNN performance with optimizer tuning and DRR=0.01</i>	68
<i>Figure 4-29 – Visual representation of Post Processing algorithm.</i>	70
<i>Figure 4-30 – Post processing results from 15 tests done with classical ML and NN classifiers.</i>	71
<i>Figure 5-1 - Glove current consumption</i>	75
<i>Figure 8-1 - Prototype 1 Holding Screwdriver [Original in color].</i>	88

<i>Figure 8-2 - Testing Data Accuracy for SVM classification with RBF kernel</i>	89
<i>Figure 8-3 - Testing Data Accuracy for RNN classification</i>	89
<i>Figure 8-4 - Testing Data Accuracy for K-Nearest Neighbor Classifier</i>	90
<i>Figure 8-5 - Testing Data Accuracy for Random Forrest Classifier</i>	90
<i>Figure 8-6 - Testing Data Accuracy for Decision Tree Classifier</i>	90
<i>Figure 8-7 - Testing Data Accuracy for Gaussian Naïve Bayes Classifier</i>	90
<i>Figure 8-8 – 1-CNN (Sigmoid activation on all layers)</i>	98
<i>Figure 8-9 – 2-CNN (5 FC Layers – Early Stopping)</i>	98
<i>Figure 8-10 – 2-CNN (5 FC Layers)</i>	98
<i>Figure 8-11 – 4-CNN (no Max-pool layer)</i>	98
<i>Figure 8-12 – 1-CNN (DRR = 0.4)</i>	99
<i>Figure 8-13 – 1-CNN (Input DRR = 0.2)</i>	99
<i>Figure 8-14 – 2-CNN (Input & Hidden DRR = 0.05)</i>	99
<i>Figure 8-15 – 2-CNN (Input DRR = 0.01)</i>	99
<i>Figure 8-16 – 1-CNN (Adagrad optimizer and DRR = 0.2)</i>	100
<i>Figure 8-17 – 1-CNN performance (Adagrad optimizer & DRR = 0.1)</i>	100
<i>Figure 8-18 – 1-CNN (Adagrad optimizer tuning, DRR = 0.05 & 6 FC Layers)</i>	100
<i>Figure 8-19 – 2-CNN (Adagrad optimizer tuning, DRR = 0.05 & 6 FC Layers)</i>	101
<i>Figure 8-20 – 2-CNN (Adagrad optimizer tuning, DRR = 0.05 & 6 FC Layers)</i>	101
<i>Figure 8-21 – 3-CNN (Adagrad optimizer tuning, DRR = 0.05 & 6 FC Layers)</i>	101
<i>Figure 8-22 – 1-CNN (5x5 kernel, Adagrad optimizer tuning & DRR = 0.05)</i>	101
<i>Figure 8-23 – 3-CNN (3x3 kernel, 1 Max pool layer & 2 FC CNN - Adagrad optimizer & DRR = 0.2)</i> ..	102
<i>Figure 8-24 – 2-CNN (5x5 first layer kernel - Adagrad optimizer & DRR = 0.01)</i>	102
<i>Figure 8-25 – 3-CNN (5x5 first layer kernel - Adagrad optimizer & DRR = 0.1)</i>	102

LIST OF TABLES

<i>Table 1-1 – Selected hand gestures common to industry.....</i>	<i>4</i>
<i>Table 2-1 - Summary of further similar research.....</i>	<i>9</i>
<i>Table 3-1 - Summary of Sensor Types explored for HAR</i>	<i>14</i>
<i>Table 3-2 – Implementing filtering on glove processor/Classification processor</i>	<i>24</i>
<i>Table 3-3 - Raw glove signals selected for Pre-processing (dark shade)</i>	<i>33</i>
<i>Table 3-4 – Statistical formula for pre-processing</i>	<i>39</i>
<i>Table 3-5 – 3-channel activity images derived from raw data and FFT values of data</i>	<i>43</i>
<i>Table 4-1 – Test data points collected by participants.....</i>	<i>46</i>
<i>Table 4-2 – CM - SVM classifier with the RBF kernel</i>	<i>47</i>
<i>Table 4-3 – Initial Error analysis for CM for SVM classifier</i>	<i>48</i>
<i>Table 4-4 – Updated CM - SVM classifier with the RBF kernel</i>	<i>49</i>
<i>Table 4-5 – Combined CM - common categories combined</i>	<i>50</i>
<i>Table 4-6 – Manual dimensionality reduction level comparison.....</i>	<i>54</i>
<i>Table 4-7 – Classical Machine Learning performance comparison</i>	<i>55</i>
<i>Table 4-8 – CM – 1-CNN with Adagrad optimizer & DRR=0.01</i>	<i>68</i>
<i>Table 4-9 – Combined CM - activities for 1-CNN with Adagrad optimizer & DRR=0.01</i>	<i>69</i>
<i>Table 4-10 – Example CM prior to passing through post processing algorithm.....</i>	<i>71</i>
<i>Table 4-11 – Example CM after to passing through Post processing algorithm</i>	<i>72</i>
<i>Table 5-1 - Design specification summary of the smart glove.....</i>	<i>73</i>
<i>Table 5-2 – Final Performance Values</i>	<i>77</i>
<i>Table 8-1 - Prototype 1 Design Specification Summary.....</i>	<i>88</i>
<i>Table 8-2 – Selection of features for stage 2 pre-processing</i>	<i>92</i>
<i>Table 8-3 - Examples of Convolutional Neural Net Images for each activity.....</i>	<i>96</i>

LIST OF ABBREVIATIONS

	<i>Definition</i>
1-CNN	Described in section 4.4.4
ANN	Artificial neural network
AR	Augmented reality
CM	Confusion matrix
CNN	Convolutional neural network
dim	Dimension
DoF	Degree of freedom
DFT	Discrete Fourier transform
DRR	Dropout regularization ratio
DT	Decision tree
EMG	Electromyography
FC	Fully connected
FFT	Fast Fourier transform
GMM	Gaussian mixture model
HAR	Hand activity recognition
HOB	Histogram of bends
HOG	Histogram of gradients
IC	Integrated circuit
IMU	Inertial measurement unit
IoT	Internet of things
k-NN	K- nearest neighbors
LDA	Linear discriminant analysis
LSTM	Long short-term memory
MCU	Micro controller unit
MEMS	Micro-electromechanical system
ML	Machine learning
MQTT	Message query
x-NN	x-nearest neighbors
sEMG	Surface electro-Myograph
RBF	Radial-basis function
ReLU	Rectified linear unit
RNN	Recurrent neural network
RPi	Raspberry Pi rev.3 model B
SVD	Singular value decomposition
SVM	Support vector machine
VR	Virtual reality
WCS	Worst case scenario
WFS	Wearable flexible sensors
acc_x	Accelerometer X value
acc_y	Accelerometer Y value
acc_z	Accelerometer Z value
$gyro_x$	Gyroscope around X-axis
$gyro_y$	Gyroscope around Y-axis
$gyro_z$	Gyroscope around Z-axis
α	Roll value
β	Pitch value
γ	Yaw value

LIST OF APPENDICES

<i>Appendix 1: Prototype 1</i>	88
<i>Appendix 2: Glove Schematic & PCB Design</i>	91
<i>Appendix 3: Stage 2 Pre-processing features</i>	92
<i>Appendix 4: Stage 2 Pre-processing Images</i>	96
<i>Appendix 5: Add. CNN Performance Charts</i>	98
<i>Appendix 6: Project Cost</i>	103

1. INTRODUCTION

1.1. Background Information

Activity recognition or gesture recognition is the of task extracting meaningful data from series of biological body movements to analyze and identify tasks performed at a given time. These movements or expressions can be done as a form of communication [1], usage of tools, travel or general behavior that can be studied to identify different attributes such as intention, efficiency, fatigue level, repetitiveness and ergonomics. It is one of the most researched topics today in the field of health, lean manufacturing, medicine and sociology. Using activity recognition by methods of on-body sensors or visual inputs such as cameras and depth sensors, a number of different aspects about the human body can be identified for varying purposes.

Being in a data-focused world, tracking gestures via information from activity recognition is a very common trend in devices and services in the market today. One of the most popular industries in this field is the health and fitness industry. With the global wearable device revenue for 2018 at US\$ 26.43 billion [2]; the number is only predicted to increase in the future [3] by almost double in two years. A very similar trend is seen in most other industries where activity tracking is used as a tool to identify possible opportunities for improvement in human motion.

In the midst of the rapidly growing wearable technology market, the benefits of wearable technology are not often reflected within the industrial environment. As wearable technology is used to enhance the personal lives of the wearer, the same paired with Lean manufacturing methodologies should be used to enhance the work life of workers in industries. Lean manufacturing is used in many manufacturing plants around the world that is focused on reducing waste through methods such as Toyota Production System, production scheduling and Kanban [4]. This results in improvements in speed of operation, hand motion, ergonomics, productivity and so on. Manual tracking is currently used by the manufacturing industry for planning and optimizing by consensually tracking the movements of a worker performing a set of predefined tasks. A Lean manufacturing tool known as a standardized work

combination chart is used to measure the different time values in performing a series of tasks [5]. This shows the wasted or non-productive movements that can be optimized to make the observed process more efficient. The process of creating a work combination chart is simple, focused, heavily time consuming and repetitive. As such this process can be automated and will be the focus of this thesis.

1.2. Project Description

This project is aimed at designing and building a wearable device to recognize common activities performed by workers at industrial manufacturing lines using machine learning (ML) such that these activities can be monitored, tracked and used for planning and optimizing in the future.

This is approached by designing a wearable device with sensors that can be worn by, factory production line workers, technicians or mechanics. The device will be capable of capturing the data required and sending them to a central processing unit to be classified in real-time or in batches. The data is then stored with appropriate time stamps in order to be used for further analyzing and optimizing.

1.2.1. Project Scope

Scope: To design a low-cost solution that is capable of monitoring the hand activities made by a worker in an industrial manufacturing/assembling environment without interfering with the motion of the worker.

Given the nature of consensual data collection, simultaneous usage was limited at five gloves which allows recording from up to five points in the same assembly line.

Aim: To track and monitor the movements of the hand in order to estimate and record the activity that is being done from a list of predefined activities common to the manufacturing industry.

The project scope is achieved by a number of deliverables.

- Sensor selection – Select the most suitable sensors for the task.
- Data filtering – Collect and process data from sensors.
- Mechatronic design – Design wearable device optimized for worker.
- Data collection – Collect suitable data for training and testing.

- Modeling – Employ an ML algorithm for activity recognition.
- Testing – Test the glove with test data.

1.2.2. Initial Assumptions

A number of assumptions were made based on simple practical experiments and prior knowledge.

- Measuring the activity on the dominant hand for industrial activities can often be sufficient compared to measuring throughout the body or from both hands.
- Most basic hand gestures are performed within a window of 0.75 seconds.
- Movement of fingers for industrial applications is such that the metacarpophalangeal (MCP) joint provides the yaw movement of the finger and the MCP, proximal interphalangeal (PIP) and distal interphalangeal (DIP) joints act together to provide the roll movement (see section 3.1.2).
- Collecting data from the back of hand, thumb, index and middle fingers are sufficient to recognize activities in the scope of this thesis (see section 1.3).

1.2.3. Hypothesis / Project Contribution

1. All activities were monitored at an equal window of 0.75 seconds. To ensure sufficient data is available within the window 30 samples were selected.
2. As per the paper by Huang *et al.* [6] in section 2.2, sensors on the dominant hand of the wearer can be used to determine the activity up to 80% accuracy. This is also backed up to a certain extent by the paper by Bruno *et al.* [7] where Activities of daily life are monitored.
3. Determine the skill / identity of the worker based on performance

1.3. Activity detection scope

As the project is aimed at identifying activities made by workers in an industrial environment, a number of activities that will fall within this scope was identified. These selected activities will be trained to the classification algorithm and only these activities will be monitored by the device during operation.

A list of common activities that can be identified in industry are shown in Table 1-1. These activities were then further ranked based on the complexity of the motion. Table

1-1 also provides a reference at the end of the row to a research paper that has used the same activity.

1. Temporal data – Gestures have a range of movement through time.
2. Hand orientation – Gestures have a unique hand orientation.
3. Finger position – Gestures have a unique finger position/bend.
4. One-time movement – Gestures do not repeat itself over time.

Table 1-1 – Selected hand gestures common to industry

Activity Index		Temporal data	Hand orientation	Finger position	One-time movement	
0	Point	X	X	✓	X	[8]
1	Wipe	✓	X	✓	X	[9]
2	Tighten	✓	✓	✓	X	[10]
3	Loosen	✓	✓	✓	X	
4	Pick (tool)	✓	✓	✓	✓	
5	Hold (frame)	X	X	✓	X	[10]
6	Pull (drawer)	✓	✓	✓	X	
7	Push (cart)	✓	✓	✓	X	
8	Hammer	✓	✓	✓	✓	[10], [9]
9	Walk	✓	X	X	X	[6], [8]
10	Hold & walk	✓	✓	✓	X	[11]
11	Turn	✓	X	X	✓	

By observing the movement of each section of the hand when performing activities in Table 1-1, it can be observed that the movement of the ring and little fingers are the same as the middle finger. This justifies the assumption made in section 1.2.2. The

measurement of thumb, index and middle fingers in reference the back of the hand (see section 3.1.2) is sufficient to identify the activities in Table 1-1.

1.4. Project Architecture

Using the basic structure for human activity recognition devices used by the paper by Attal *et al.* [12] a similar data flow was designed for this project is shown in Figure 1-1. The figure shows the processes involved in both training the classification algorithm and running the same.

- Blue – tasks that need to be conducted by the processor on the glove.
- Green – process for transmitting data between glove & classification processor.
- Red – processes for capturing data for training, training the model.
- Grey – processes for classification of data.
- Orange – processes to correct abnormalities and evaluate the performance.

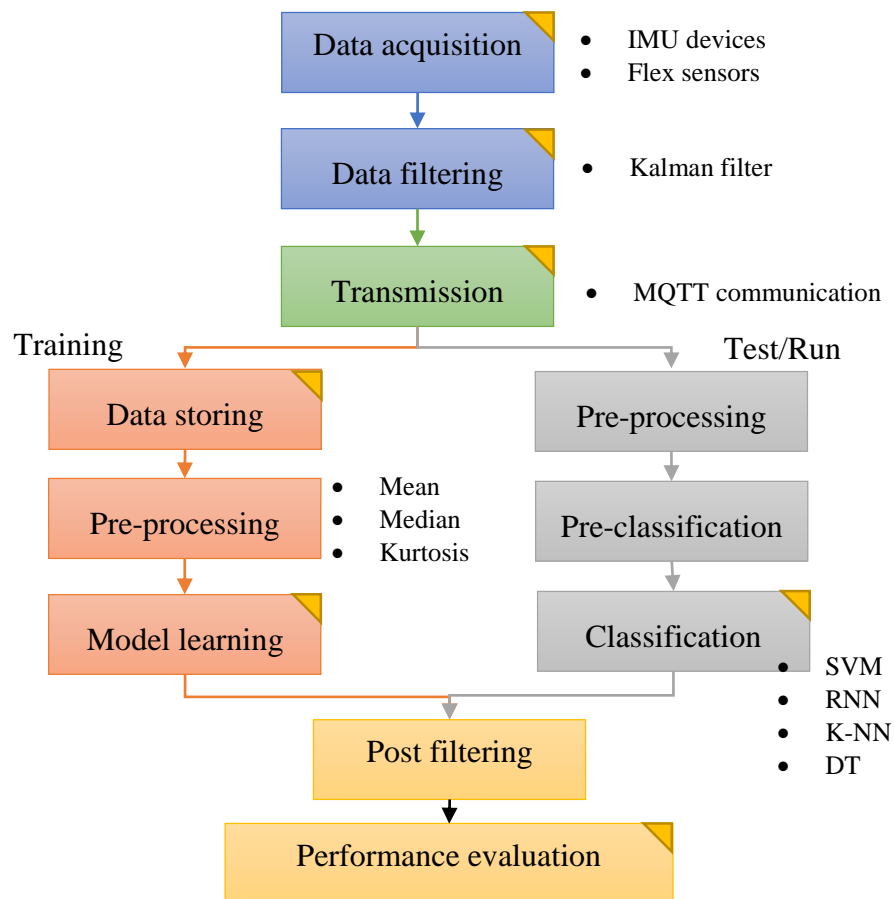


Figure 1-1 - Project architecture

2. LITERATURE REVIEW

This section briefly describes the current developments in human activity recognition and the recent research contributions similar to that of this thesis.

2.1. Types of Activity Recognition

The concept of Monitoring as a service (MaaS) is a service to keep track of activities such as step count, calorie intake, sleeping patterns and travel. Monitoring as a service uses GPS (global positioning system) and IMU (inertial measurement unit) sensors or built-in devices in phones, tablets and smart watches to identify basic human motion during day to day activities such as running or walking or repetitive workout patterns. These tracking applications includes iOS health [13] and Google fit [14] and third party such as ASICS Runkeeper [15].

Apart from the health-conscious general public, there is a special portion of the market that require health and fitness monitoring as a part of tracking the progress of rehabilitation [16]. In competitive sports such as swimming, underwater cameras such as those from Qualisys AB and motion analysis software are used to track body movements and optimize body movements for the most streamlined and faster strokes [17]. Moov [18] is a smart wearable device that can be worn either on the wrist or the ankles and provides the user with more general health and fitness data when performing day-to-activities. Moov devices can track activities such as running, cycling and gym workouts to provide detailed analysis on aspects such as cadence, power output, impact to knees and workout defects. Similar to Moov, Nexus [19] and Atlas [20] are gym specific wearable devices that keep track of the user's activities, power output and workout time. For professional athletes and rehabilitating athletes, the product Athos [21] allows the wearer or coach to monitor the individual muscle using electromyography (EMG) sensors.

Gest [22] is a wearable device on the hand, similar to the device in this project, with the objective of replacing the keyboard. The users finger movements and monitored as they type in air and the device estimates which key has been pressed by the user.

Many devices like anthropomorphic arms and services like 3-D CAD software are constantly made intuitive with the integration of control of inputs using natural human gestures. The uses span from mimicking robotic arms and systems, 3-D modeling and viewing, gaming/ virtual reality (VR)/ augmented reality (AR) control, computer mouse control and machinery control. Devices such as the Microsoft Kinect [23] have the capability of studying the body motions and using certain key motions to trigger specific control inputs. Gesture recognition is now used in the control of machinery, computer simulations and robotics. A paper by Mi *et al.* [24] describes the use of gesture recognition to tele-operate the motions of a robotic fish in water. The gestures are programmed in to natural motions that can be picked up by a sensor to instruct the robot to accelerate, decelerate, turn and stop¹.

Sense Glove [25], Xsens [26] and Cyberglove [27] are wearable devices on the hand which monitors the users hand movements in the domain of AR and VR. It provides motion tracking, force feedback and tactile feedback when the user is interacting with digital world. This glove is currently being used in preparation of animated movies that require natural movement of hands. The Myo armband by Thalmic Labs is a wearable gesture control arm band which spiked in popularity with researches and developers due to its easy use and functionality. The device uses surface EMG (sEMG) sensors to monitor the muscle movements from the forearm to control devices such as computers and drones.

Everyday there are new developments in activity recognition. Although activity recognition and health and fitness are becoming a saturated market, there are new research being conducted in activity recognition to achieve other objectives such as sign language detection [28], fall detection, health monitoring for medical analysis, arthritis rehabilitation [29], robot control and computer mouse control.

¹ More information on available devices: <https://johnsamarasinghe.blogspot.com/2019/04/hand-activity-recognition-literature.html>

2.2. Similar Research

The paper by Cornacchia *et al.* [30] provides a survey on activity recognition using wearable sensors. Another paper by Attal *et al.* [12] also looks at the different classification algorithms used in many researches to perform hand activity recognition (HAR). The findings of these two papers were summarized.

Pre-processing Algorithms - mean, median, variance, standard deviation, skewness, kurtosis, index of min/max, range, discrete Fourier transform (DFT), power spectral density (PSD), DC component for frequency domain features.

Classification Algorithms - threshold based reasoning, fuzzy rules, adjustable fuzzy clustering (AFC), K-nearest neighbor (k-NN), LogitBoost, support vector machine (SVM), random forests (RF), neural networks (NN), multinomial logistic discrimination (MLD), multilayer perceptron (MLP), Gaussian mixture models (GMM).

There are very few practical implementations of human activity recognition on industrial applications. One of the papers by Stiefmeler *et al.* [31] uses 27 sensors on the human body which include 7 IMU's, 8 Force Sensitive Resistors and 4 Ubisense tags. The tools used by the workers are also equipped with RFID (radio-frequency identification) tags and IMU's to provide additional data when the tool is being used. A new string matching-based segmentation and classification method was designed. This method encodes the signal properties in characters to create a string which can be compared against templates.

A dissertation by Hartmann [32] looks in a to an industrial scenario where a wrist mounted IMU sensor is used along with a top mounted camera which observes the movement of patterns on a block worn on the wrist. A similar research done by Tao *et al.* [10] on worker activity recognition and a paper by Benalcàzar *et al.* [33] uses sEMG sensors and the IMU sensors on the Myo armband. Tao uses this to form a stacked signal image. The image is then fed in to a CNN for feature extraction. The data classified using this method are simple operations such as hammering, tightening, grabbing and resting. Benalcàzar instead uses the kNN classifier to detect the hand activities; making a fist, wave in, wave out, open hand and pinching. The accuracy of

the data was found to be better than the Myo band proprietary algorithm showing the impact of sEMG sensors for human activity recognition. Another similar paper by Jiang *et al.* [34] uses both sEMG and IMU sensors are used to determine the hand gestures. The okay sign, peace sign, hand loose, finger snap, thumbs up, thumbs down, turn palm and walking fingers were air gesture activities recognized by this device. The surface gestures included the force levels exerted by index finger, all 5 fingers and the fist in different orientations.

The research by Koskimaki *et al.* [35] describes the use of a single wrist mount accelerometer used in industrial applications to detect use of power drill, hammering, screwing and spanner use with an accuracy of almost 90% in a 1.5 second window. The standard statistical and frequency domain details were used as feature extraction while k-NN was used as the classification algorithm

A conference paper by Luzhnica *et al.* [8] uses a custom-built glove similar to the glove designed in this project to detect a number of everyday gestures. These include number one, two, three, four, five, thumbs up, thumbs down, point to self, shoot, scissor, cutthroat, continue, counting, knocking, waving, come here, go away, push away, never mind, talking, calling, walking, shoulder pat, point, swipe left, swipe right, swipe up, swipe down, turn, zoom and grasp. The paper uses linear discriminant analysis and logistic regression to achieve an accuracy of 98.5%

The Table 2-1 provides a brief summary of other similar research that has been found similar to the project in this thesis.

Table 2-1 - Summary of further similar research

Ref.	Sensors	Pre-processing	Classifier	Accuracy
[10]	Myo Armband (IMU and sEMG)	Discrete Fourier transform	CNN	98% half-half
[6]	Koala MEMS based 9-axis motion sensor	Mean, variance, skewness, kurtosis and root mean square	RF, decision tree & SVM	81% (DT), 73.2% (SVM non-dominant hand)

Ref.	Sensors	Pre-processing	Classifier	Accuracy
[33]	Myo armband (sEMG sensors and IMU)	Low-pass 4 th order Butterworth filter w/ cut off at 5Hz.	K-NN with DTW algorithm	89.5%
[9]	MS Kinect, InvenSense MPU9150	Moving average window	Hidden Markov model and DTW	93%
[34]	8 sEMG sensors (Tringo wireless EMG, Delsys)	Mean absolute value	Linear discriminant analysis	92.6% air & 88.8% surface
[8]	7 Motion, 13 bending, 5 pressure & magnetometer	Sliding window, FFT, complementary filter	Linear discriminant analysis	98.5%
[36]	Smart Watch	Autocorrelation	CRF, FR, HMM, SVM, DT	>90%
[37]	Beaglebone Accelerometer	Mean, RMS, standard deviation of accelerometer	RF	90-94%
[24]	Leap Motion Controller	Leap motion program	Thresholding	N/A
[38]	Accelerometer (ADXL202) on hip & wrist, GPS	Time domain: variance, median, skew, kurtosis, 25% and 7% percentile Frequency domain: peak, power	Custom decision tree, automatically generated decision tree, ANN, hybrid model	89%
[39]	Accelerometer	Mean, variance, skewness, kurtosis, RMS	RF, SVM, 3-NN	76.1% or 71.3%

A paper by Huang *et al.* [6] studies the impact of wearing a number of sensors, namely on the right wrist, left wrist and waist for the accuracy of activity classification. The study looks at machine learning (ML) algorithms in decision tree, random forest and support vector machines. In this study the actions; standing, lying, walking, sitting and dining are monitored. Although the study proves the general logic in stating that using multiple sensors on multiple parts of the body for activity recognition improves the accuracy compared to using only a sensor at one part of the body, a key takeaway is

also highlighted. It is shown that using the decision tree classifier only on the sensors for the dominant hand is only sacrificing 1% accuracy compared to using the same ML algorithm for all three locations of the human body.

Another interesting research by Maekawa *et al.* [40] looks at a different approach in measuring the cycle time of a factory worker without analyzing the individual activities. The procedure includes looking at the sequence of activities and identifying repeating patterns. This has resulted in an accuracy of about 96.5%.

The paper by Jain and Kanhangad [41] describes how a smart phone can be used to perform human activity recognition. The built-in sensors within the phone which include the accelerometer and gyroscope are used with histogram of gradient and centroid signature-based Fourier descriptor for pre-processing and multiclass SVM and k-NN classifiers to obtain a 97% accuracy. Using this method, activities such as walking, climbing stairs, sitting, standing and laying can be identified.

In conclusion, there are a large number of methods and algorithms used to identify hand and full body activity recognition both in and out of industrial environments. Table 2-1 shows the effectiveness of each of the algorithms used and provides a good indicator for this project and the algorithms that can be effectively used. In addition, a number of key learnings such as the ability to use the movement of the dominant hand to derive the movement of the body has been used to structure this project.

3. SMART GLOVE DESIGN

The design of the glove comprised of bringing together a number of concepts and components to ensure its function. This section describes each component shown in Figure 3-1.

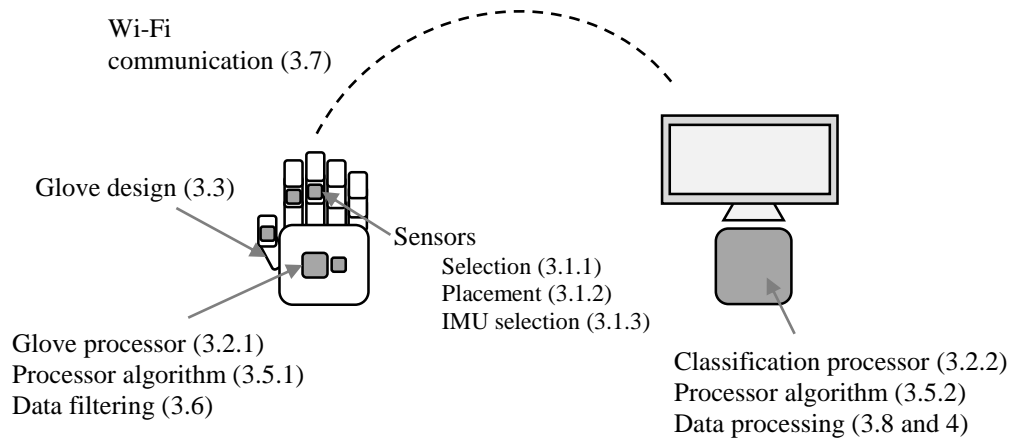


Figure 3-1 –Components of the smart glove design

The glove design is essentially two systems linked together via Wi-Fi communication (see section 3.7). The first system comprises of the wearable glove which contains the sensors (see section 3.1) and the processing unit (see section 3.2.1) running the algorithm explained in section 3.5.1. The second system contains the classification processor (see section 3.2.2) running the algorithm explained in section 3.5.2 which allow it to perform the classification (see sections 3.8 and 4)

3.1. Sensors

3.1.1. Sensor Selection

In order to classify activities listed in Table 1-1, a specific set of motions and poses from the hand must be obtained. These can be roughly separated to two criteria.

1. Static data
 - a. Pose of the hand in free space
 - b. Bending of each finger

- c. Adduction and abduction of each finger
 - d. Rotation of the thumb
2. Motion data
- a. Linear motion of the hand in free space
 - b. Rotational motion of the hand in free space

Vision based sensing is the most popular choice for activity recognition based on the number of research papers. However, there are some concerns using cameras for activity recognition. As the application for the device in this thesis is mainly based on activity recognition that is designed for industry, it is not ideal to use vision [6]. In industrial environments, the worker is always moving from one area to another as per their job requirement. As such, very good image processing is needed to isolate the correct worker from the moving and mostly noisy background. The camera needs to be positioned to obtain complete range of the worker in one frame and this may not be possible when the hands of the worker are within machinery or in any way out of direct line of sight of the camera.

Sensors placed on the body of the person being tracked is a suitable option given the application of the device. The commonly explored sensors were electromyography sensors, flex sensors and IMU sensors. Although other wearable flexible sensors could have provided a similar result, these sensors were found to be most direct and non-intrusive method of obtaining data without obstructing the movement of the worker or introducing additional tasks to the workers' job description.

Electromyography sensors are a very effective sensor that can be used to determine the exact muscle activity of hands by placing sensors away from the hand itself so as not to be intrusive to the wearer. This can be seen in the paper by Tao *et al.* [10] where a Myo armband placed close to the elbow is used for this purpose. However, due to the cost of these sensors, they had to be deprioritized.

Flex sensors are being used in many of the motion tracking gloves used in the market (see section 2.1). As the flex of a flex sensor is given by the deformation of the material which can be measured in terms of resistance, the response is close to the zeroth order response system making the usage straightforward. Flex Sensors were used in the first

prototype of this project (See Appendix 1). However, as these flex sensors are costly and are limited to 1 degree of freedom, a high cost is incurred when a flex sensor needs to be added for each degree of freedom of the fingers. This also makes the glove difficult to use. As such, these sensors were discontinued from this project.

IMU sensors are used for obtaining motion and orientation information. However, these sensors which typically consists of an accelerometer and gyroscope tend to be noisy and thus good filtration is needed to extract the actual signal (see section 3.6). For this project, the IMU sensor was chosen to provide the orientation and movement of the hand and also the orientation and movement of the fingers. This was due to the relatively low cost of the IMU and the ability to obtain information on more than one degree of freedom (DoF) of a finger from one sensor.

Table 3-1 - Summary of Sensor Types explored for HAR

Sensor type	Robustness	Cost	Accuracy	Selection
Vision	Low	High	High	No
EMG	High	High	High	No
Flex	Mod-High	High	High	No
IMU	High	Low	Mod-High	Yes

3.1.2. Sensor Placement

In order to obtain the most relevant data to classify the actions in Table 1-1, the correct sensors must be placed at the correct locations to extract the most relevant information from the human body.

The orientation of the hand is important for activity recognition and can be measured with a sensor placed at the back of the hand as it is the most static part of the hand. The sensor placed here will be able to act as a base measurement in order to determine the general motion through pose and also it will provide a reference position of the hand to extrapolate the position of the fingers (see section 3.8.2).

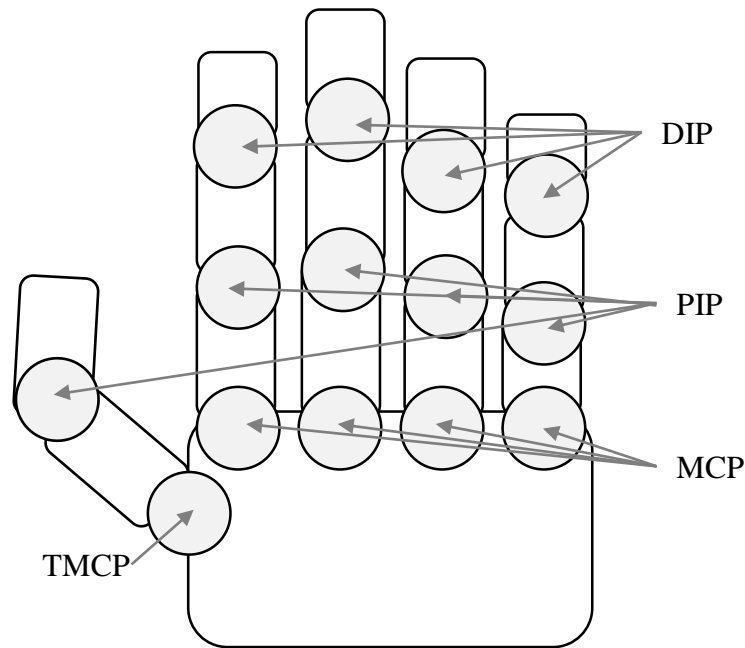


Figure 3-2 – Illustration of finger joints

The main DoF of the fingers are

- 1 DoF - Distal interphalangeal (DIP) joint of each finger (flexion/extension)
- 1 DoF - Proximal interphalangeal (PIP) joint of each finger (flexion/extension)
- 2 DoF - Metacarpophalangeal (MCP) joints (flexion/extension and abduction/adduction)
- 1 DoF - Trapeziometacarpal (TMCP) joint of the thumb to rotate [42]

For the application defined in this scope, two significant independently actuated joints can be identified. These are the joint at the knuckle (MCP and TMCP Joint) and the joint after the knuckle (PIP Joint). The DIP joint cannot be easily independently actuated, the motion of this joint relies on the PIP joint in many scenarios.

For the project the IMU sensor was used at the backhand a single IMU sensor for each finger was placed between the PIP and DIP joints. This can measure both the yaw of the finger created by the MCP joint and the roll of the all joints in that finger with reference to the IMU on the backhand. This can be seen in Figure 3-19 and Figure 3-20

The finger sensors were only measured with one sensor, although it is common to see that sensors are placed on each joint in other HAR gloves to measure all DoF [42]. In addition, the project only measures the thumb, index and middle fingers. This is

because an assumption was made that for this application, selected activities do not require the remaining two fingers to be actuated independently (see section 1.2.2).

3.1.3. Inertial Measurement Unit (IMU)

In section 3.1.1 the IMU sensor was selected for this project. The IMU model selection is important to ensure precise orientation and movement values are obtained for each measurement. The requirements needed by the IMU device are

- Suitable accelerometer with full scale range no more than 2g
- Suitable gyroscope with full scale range more than $\pm 500^\circ/\text{s}$
- Suitable magnetometer
- Precise data transfer more than 12 bits per reading.

Prioritizing the cost, the MPU6050 [43] was selected as the most suitable IMU device. This was selected based on the specifications and the popularity of the sensor in the electronics community. However, the MPU6050 does not have a built-in magnetometer, resulting in the inability to obtain a fixed reference parallel to the earth. This also results in the inability to calibrate the yaw of the device. Thus, there is a slow drift in the yaw axis due to the influence of external noise, as seen in Figure 3-12.

The specifications of the MPU6050 device were programmed based on the trial and error and limitations to human motion for the selected activities in Section 1.3 [43]

- Angular rate with full-scale range of $\pm 500^\circ/\text{s}$
- Accelerometer with a full-scale range of $\pm 2\text{g}$
- Sampling frequency of 10 kHz

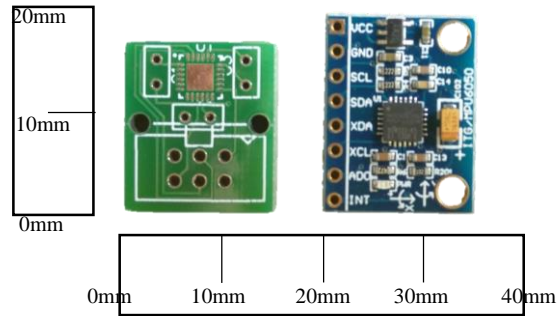


Figure 3-3 - GY521 board & custom-built breakout board size comparison [original in color]

Due to the criteria to reduce the size of the glove to improve comfort and range of motion, the GY521 board for the MPU6050 was redesigned to a smaller form factor (see Appendix 2). The board was designed such that the dimensions of the board are small enough to fit to the width at the mounting point of each finger and the mounting points are placed such that the critical components of the device do not move as much when the finger movements deflect the fabric of the glove. The backhand processor board was also designed with the IMU device built in to save space (see Appendix 2). Additional to designing the sensors, the sensors were to be wired with conductive yarn² that is sewn in to the fabric of the glove in order improve the usability of the glove and to act as an additional mounting point for the board.

3.2. Processors

The selection of the processors are completely dependents on the type of tasks required. Processor is selected to closely match the requirement while keeping the costs and form factor as low as possible. For the project two processors are used with 2 different applications.

- Glove processor
- Training & classification processor

² Conductive yarns were not added to the intermediate prototype.

3.2.1. **Glove Processor**

The main application of the glove processor is to collect data from all sensors on the glove, filter the data to obtain meaningful information and transmit the data to a centralized processor for further data analysis. The processor requirements for the glove processor are

- Required voltage for sensors: >3.3V
- Number of digital pins: 4
- I²C communication needed for IMU interfacing
- Wi-Fi communication needed for data transmission
- Small form factor

Based on these mandatory specifications, The WEMOS D1 Mini (ESP8266) was selected. The WEMOS D1 Mini is a moderately cheap, low power microcontroller and Wi-Fi board with 4MB flash based on ESP-8266EX [44]. The processor runs on 80MHz clock.

3.2.2. **Classification processor**

The application of this processor is to manage all the gloves processors as a central control platform, perform data collection and classification of the data collected. Collection of data, training and classifying data with a machine learning algorithm is a very computationally expensive task which require

- Ability to run Linux based operating system
- Ability to run Python-based applications
- Processor speed – More than 1GHz
- RAM – more than 512MB
- Wi-Fi enabled

In order to meet these requirements, the Raspberry Pi was selected for this application. The Raspberry Pi 4 model B (referred to as RPi) is a small, affordable computer with a much higher processing power than the glove processor.

3.3. **Glove**

The selection and design of the glove is important in order for the user to work with the device unhindered by it. The glove selected was a standard pair of cotton dotted gloves that is used in many industries to protect the hand from cuts, blisters and dirt. The glove is also stretchable and comes in a single size to fit all hands. This is important to match with the devices requirement to be universal.

Certain tasks in industry require delicate handling of equipment or the feel of the item in the operator's fingertips. For example, picking up paper or fabric require the tactile feedback to the user and delicate control to lift the item. In order to enable this, the fingertips of the cotton dotted gloves were carefully cut. This also enables the glove to be worn comfortably by any person despite the size of the hand or the length of the fingers.

The glove in Figure 3-5 was designed as an intermediate glove to enable quick data collection with existing components due to the delay in obtaining the final components due to the CoViD-19 lockdown in 2020.

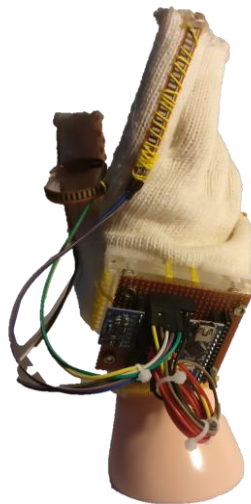


Figure 3-4 – Prototype 1 design
[original in color]

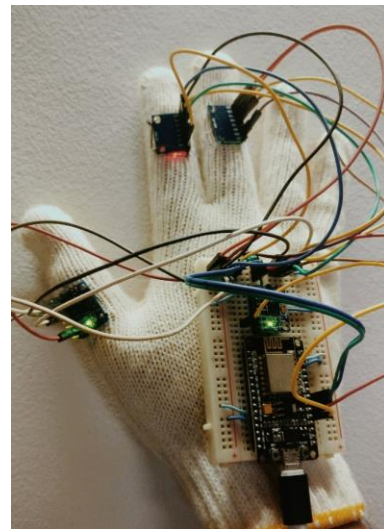


Figure 3-5 – Porotype 2 intermediate design
[original in color]

3.4. **Circuit Design**

The main components of the glove are connected together as per the diagram in Figure 3-6. Each of the IMU devices for the fingers are placed on their own PCB's and

attached to the finger. The IMU for the backhand and the glove processor is attached to a single PCB for easy connectivity and attached to the back hand of the glove.

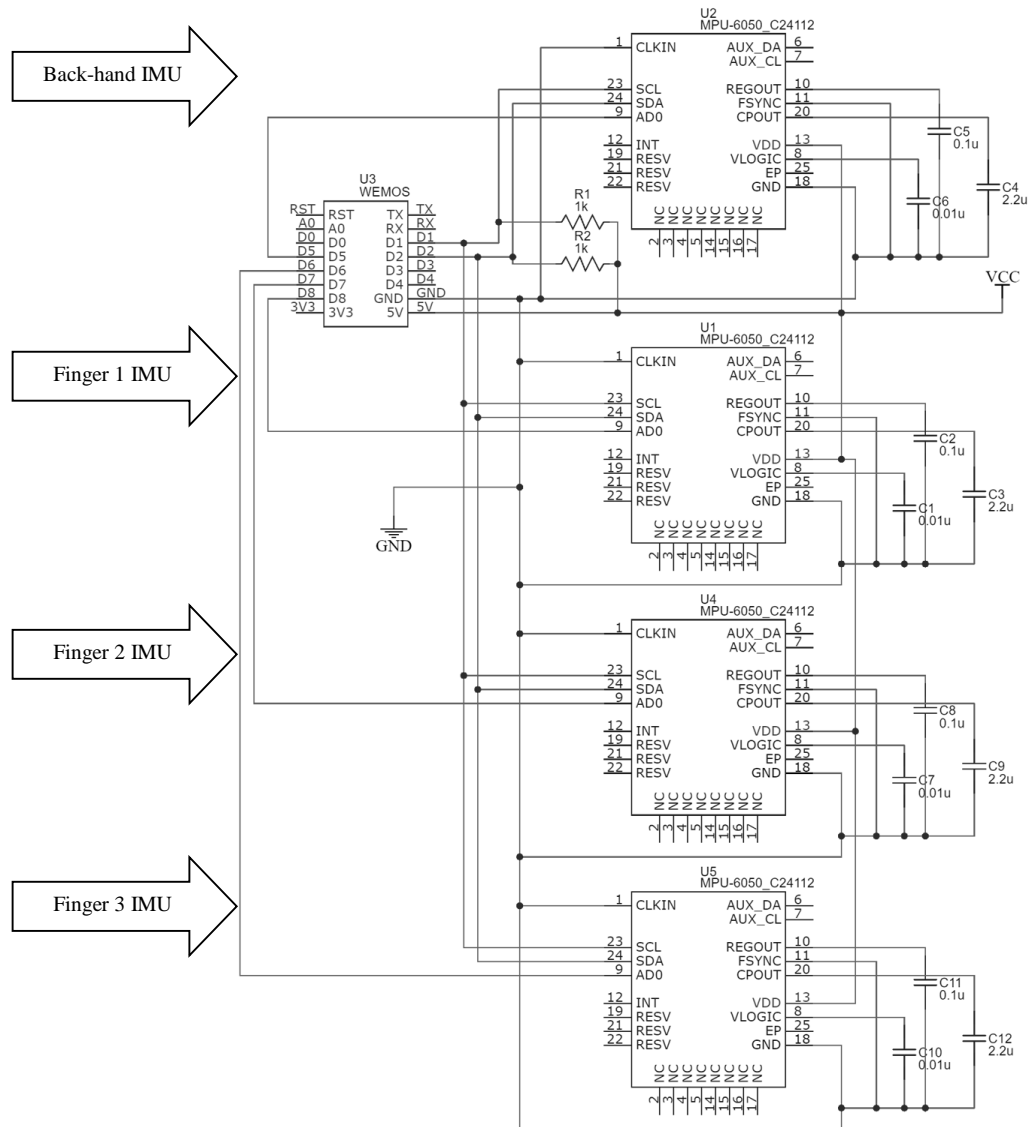


Figure 3-6 – Glove circuit diagram

3.5. Algorithm

3.5.1. Glove Algorithm

The software running on the processor on the glove is dedicated to do a number of specific tasks.

- Maintain connection to all 4 IMU devices
- Collect data from the IMU devices
- Perform data filtering
- Maintain MQTT (See section 3.7) connection through Wi-Fi to the MQTT server
- Send and receive status messages to and from the MQTT server.
- Send filtered data to the MQTT server when requested.

The basic process flow diagram to achieve these tasks can be seen in Figure 3-7.

When the device is powered on, the IMU devices are initiated and connived to the predefined MQTT server through the selected Wi-Fi network. The glove processor will then enter a continuous loop which

1. Refreshes the MQTT connection
2. Acquires raw IMU data
3. Processes roll, pitch and yaw
4. Passes data through data filtering
5. Publishes data to MQTT server

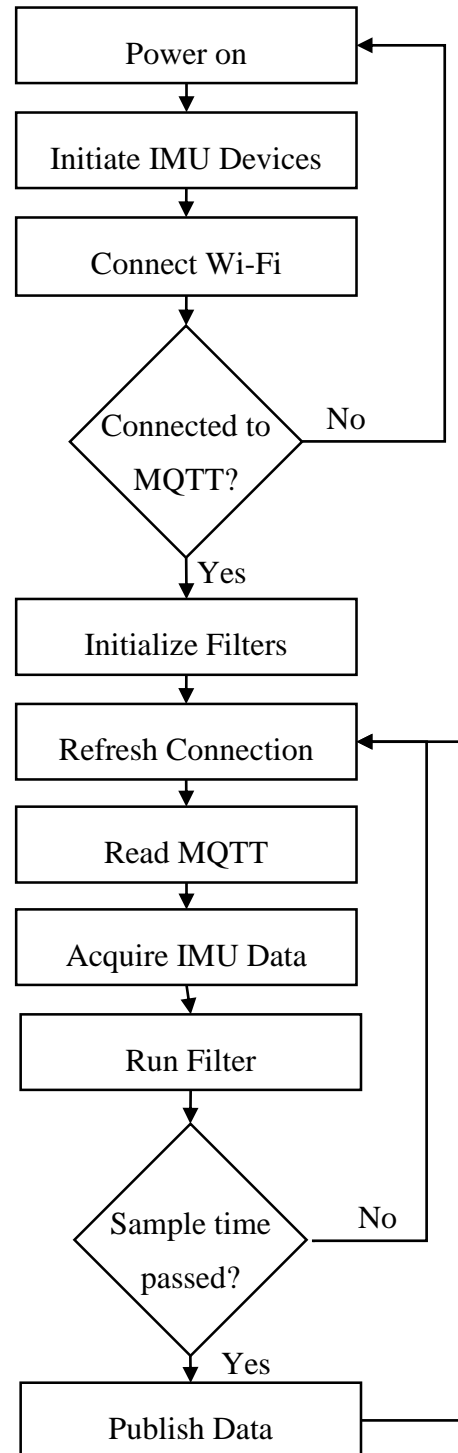


Figure 3-7 –Glove algorithm

3.5.2. Classification Processor Algorithm

The software for the algorithm run on the Raspberry Pi is dedicated to execute

- Connect to Wi-Fi and run Mosquitto as the MQTT server
- Manage connection to all gloves attached to the central processor
- Manage status messages to and from the gloves
- Request, collect and store training data from each glove
- Request, collect and classify live data from all gloves.

In order to achieve these tasks, an algorithm was developed as seen in Figure 3-8 where “<glove>/data”, “<glove>/status” and “<glove>/command” are MQTT topics which are used to broadcast different data types. The states 0 to 4 maintained for each glove and are

Status 0: Glove is offline – Glove has not published that the glove is online.

Status 1: Glove is online (listening) – Glove has signaled online status.

Status 2: Receiving data – Data is being received from the glove. The data is saved in a log file or classified depending on training data or data classification is requested.

Status 3: Data time-out – Data is not being sent to glove at required time.

Status 4: Refresh connection – Intermediate state informing the user if an interruption had taken place in the operation.

The algorithm in Figure 3-8 was then implemented in Python with the use of the Kivy library to display a user-friendly graphical user interface. This is a dashboard that can be used to view the status of all connected gloves at once and control each of the gloves that are connected. Currently the dashboard can connect up to 5 gloves (see Figure 3-9).

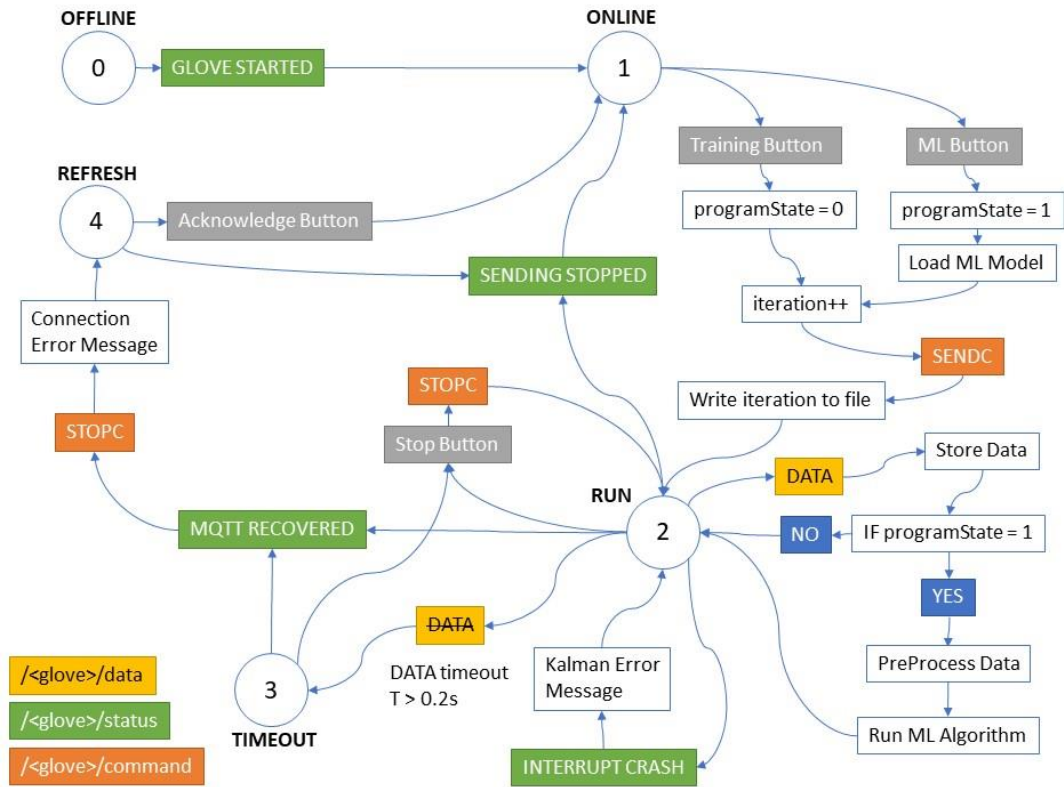


Figure 3-8 –Processor state diagram

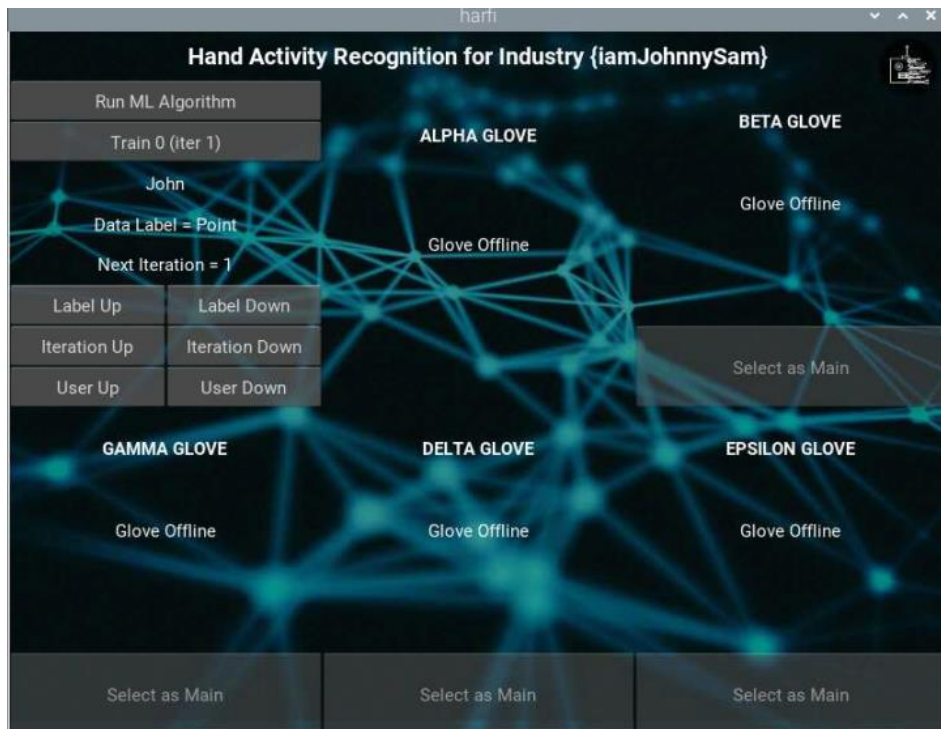


Figure 3-9 - Glove dashboard v2 [Original in color]

3.6. Data Filtering

The program for the processing unit on the glove is responsible for recording, filtering and transmitting data from the sensors on the glove. The captured data is filtered to eliminate the noise to receive a smooth signal. It should also be noted that the filtering was done on the glove processor rather than the central processor due a number of reasons summarized in Table 3-2.

Table 3-2 – Implementing filtering on glove processor/Classification processor

	<i>Filtering on Glove</i>	<i>Filtering on Classification Processor</i>
Impact to software	High	Low
Chance of errors	Low	High
Access to raw data	Fast	Slow
Dependencies	None	Stable MQTT Connection

The sensors were initialized at the specifications detailed in section 3.1.3. The data capturing on all sensors is done at its maximum speed although the data is not transmitted at the same rate. This is done primarily to ensure the filtering algorithms described in section 3.6 can continue to run and maintain better prediction when data is needed.

The raw data obtained from the IMU is shown in Figure 3-10. Digital filtering will be required to remove noises embedded in the signal and obtain reliable values.

Accelerometers are very high sensitivity devices due to their structure. It produces a lot of noise in the very short-term results. However, the noise can be assumed to be random and Gaussian distributed. This means that an accurate value of the accelerometer can be obtained by using a low pass filter. As gyroscopes are reliable in the short-term and accelerometers are reliable in the long-term, it is needed to combine both accelerometer and gyroscope to obtain the ground truth of the orientation of the sensor.

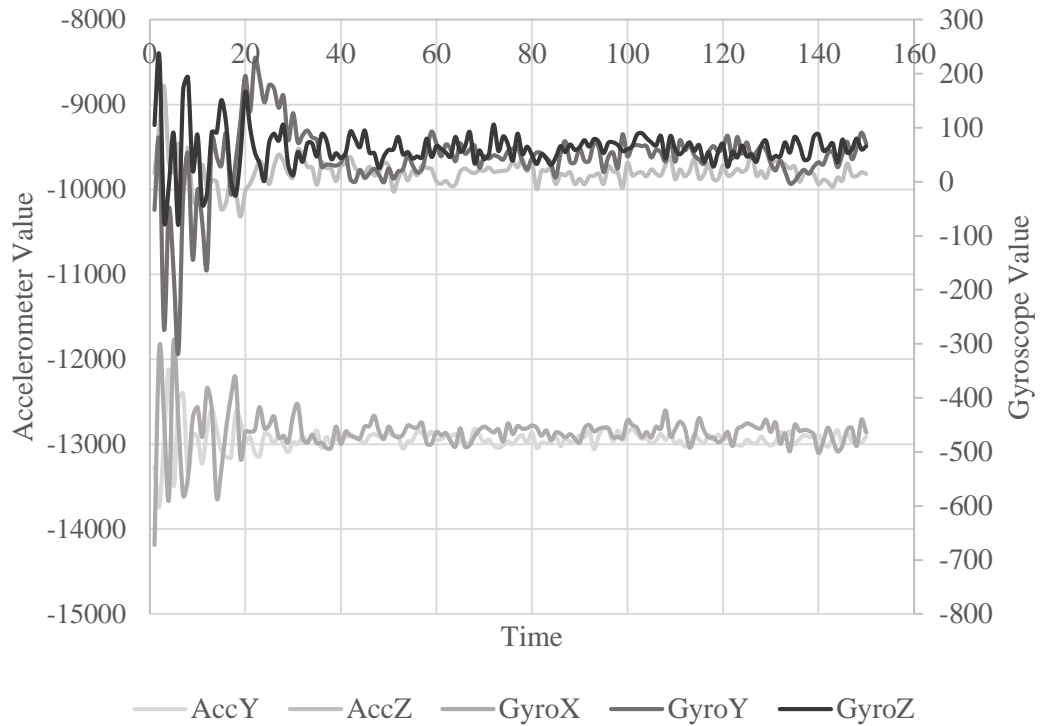


Figure 3-10 – Accelerometer & Gyroscope on a flat table (knock at t=0)

3.6.1. Kalman Filter

The Kalman filter is a method of linear quadratic estimation which observe a series of measurements that are engulfed in statistical noise and other inaccuracies over a period to provide an estimate of the underlying unknown variable. Kalman filters are capable of combining the readings from the accelerometer and gyroscope from their probabilities and obtaining a more accurate reading. Compared to digital low/high pass filtering, Kalman filters do not have time delay in providing the measurement as the filter will always predict future values based on the past.

The MPU6050 IMU device, with the accelerometer and the gyroscope provide 6 independent data values (excluding temperature) to measure the pose of the unit. These can be considered as our inputs, x and measurement, y .

Inputs	Measurements
acc_x : Acceleration in the x-axis	$gyro_x$: Angular velocity about the x-axis
acc_y : Acceleration in the y-axis	$gyro_y$: Angular velocity about the y-axis
acc_z : Acceleration in the z-axis	$gyro_z$: Angular velocity about the z-axis

To derive the mathematical model for the yaw, pitch and roll of the IMU device, equation (1) and (2) [45] were used. The roll and pitch of the that is obtained from the mathematical equation are predicted state estimates, \hat{x}_{t-1} (Priori estimates).

$$\hat{\alpha}_{t-1} = \tan^{-1} \left[\frac{\text{acc}_y}{\text{acc}_z} \right] \quad (1)$$

$$\hat{\beta}_{t-1} = \tan^{-1} \left[\frac{-\text{acc}_x}{\sqrt{\text{acc}_y^2 + \text{acc}_z^2}} \right] \quad (2)$$

The accelerometer only has one signal to determine the orientation, which is gravity. Hence it is not straightforward to determine the yaw of the device as yaw will be the rotation about the axis of gravity and reading on the accelerometer for this axis will not change for this motion.

The measurement of the system is also used to derive the measured output of the system. Equation (3) and (4) result in roll and pitch are the measurement from the system.

$$\alpha_t = \alpha_{t-1} + (\text{gyro}_x \times \Delta\text{time}) \quad (3)$$

$$\beta_t = \beta_{t-1} + (\text{gyro}_y \times \Delta\text{time}) \quad (4)$$

$$\gamma_t = \gamma_{t-1} + (\text{gyro}_z \times \Delta\text{time}) \quad (5)$$

The yaw can be approximated with only the gyroscope reading along with roll and pitch. This reading is prone to drift over time due to the nature of the gyroscope and the accumulation of errors as correcting this reading with the accelerometer is not possible. This can be seen clearly in Figure 3-11 and Figure 3-12.

A Kalman filter is designed with equations (6) to (10) with initial estimates $\hat{x}_{t-1/t}$ & $P_{t-1/t}$.

Stage 1.1: Predict: Projecting the state ahead

$$\hat{x}_{t/t-1} = A\hat{x}_{t-1/t+1} + Bu_t \quad (6)$$

Stage 1.2: Predict: Projecting the error covariance ahead

$$P_{t/t-1} = AP_{t-1/t}A^T + Q \quad (7)$$

Stage 2.1: Correction: Computing the Kalman gain

$$K_t = P_{t/t-1}H^T(HP_{t/t-1}H^T + R)^{-1} \quad (8)$$

Stage 2.2: Correction: Updating the estimate with the measurement

$$\hat{x}_{t/t} = \hat{x}_{t/t-1} + K_t(z_t - H\hat{x}_{t/t-1}) \quad (9)$$

Stage 2.3: Correction: Updating the error covariance

$$P_{t/t} = (1 - K_tH)P_{t/t-1} \quad (10)$$

The code in software was written with the help of code by Kristian Lauszus [46], [47].

- \hat{x}_{t-1} – Priori Estimate (Angle) is taken from the Accelerometer reading
- x_t – Measurement (Angular Velocity) is taken from the Gyroscope reading
- A – State transition model = $\begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}$
- B – Control-input model = $\begin{bmatrix} \Delta t \\ 0 \end{bmatrix}$
- H – Observation model = $[1 \ 0]$
- P – Initial error Covariance = $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
- Q – Process noise variance - accelerometer & gyroscope = $\begin{bmatrix} 0.001 & 0 \\ 0 & 0.003 \end{bmatrix}^3$
- R – Measurement noise variance = $\begin{bmatrix} 0.03 & 0 \\ 0 & 0 \end{bmatrix}^4$

Using the Kalman filter, the noise values in the roll and pitch calculation can be greatly filtered and a real time, accurate signal can be obtain as shown in Figure 3-11⁵. A further calibration routine was programmed to initialize the values of the IMU at zeros at the start in order to remove any static bias in the device [48].

³ Tested through trial and error and supported with work by Lauszus [51]

⁴ Tested through trial and error and supported with work by Lauszus [51]

⁵ Video demonstrating the performance of the Kalman filter: <https://youtu.be/idYYBqs3buA>.

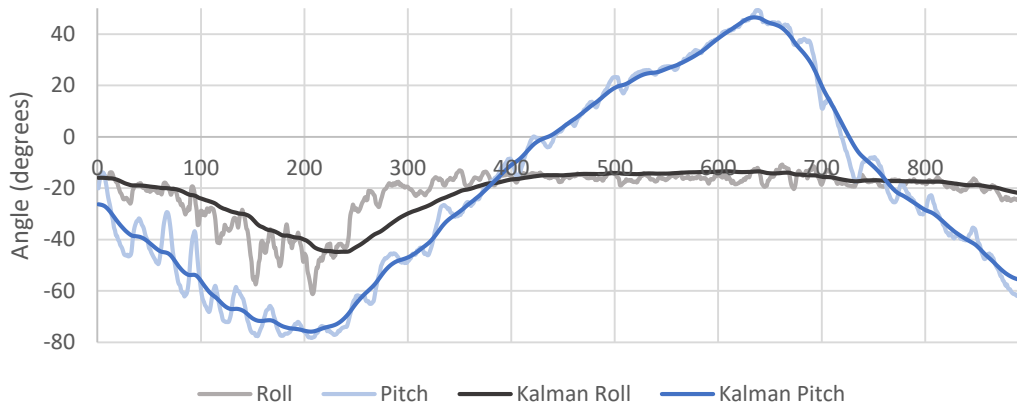


Figure 3-11 – Kalman filter performance

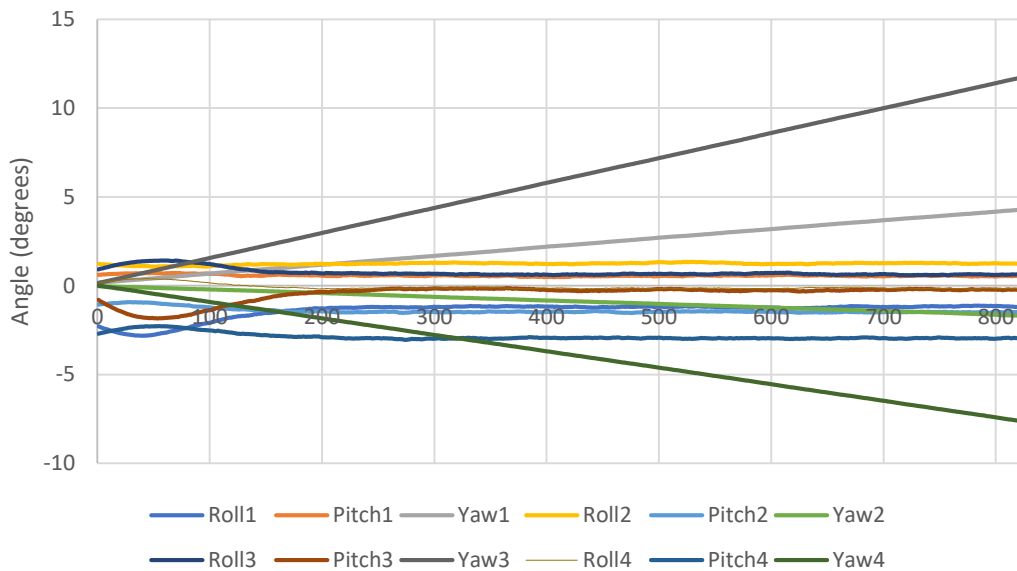


Figure 3-12 – Drift of signals while stationary on flat surface. Yaw axes drift with time. The cycle time with 4 IMU sensors running real-time Kalman filter was found to be 12.5ms (80Hz). However, the sampling time was set to 25ms (40Hz) while eight Kalman filters were operating at max capacity in the background

3.6.2. Verification of angles

A simple test rig was made to verify the angles output by the Kalman filter designed. The roll and pitch figures of all 4 Kalman filters was measured at the same time by fixing it on to the same breadboard as shown.

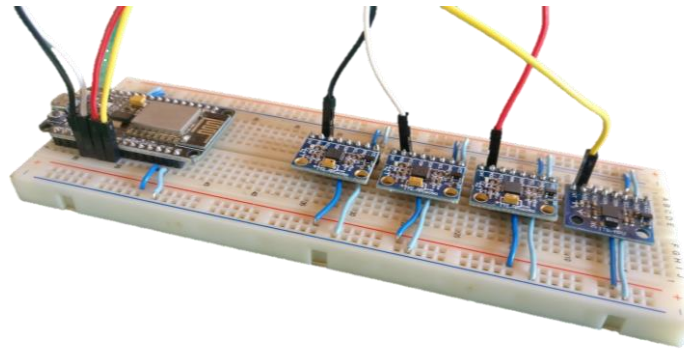


Figure 3-13 - IMU testing setup [original in color]

The test rig was fixed at different angles and the readings on the Kalman filter was recorded. Figure 3-14 and Figure 3-15 show the readings with their reference signals. It can be seen that all 4 IMU devices provide an accurate reading with relatively low error.

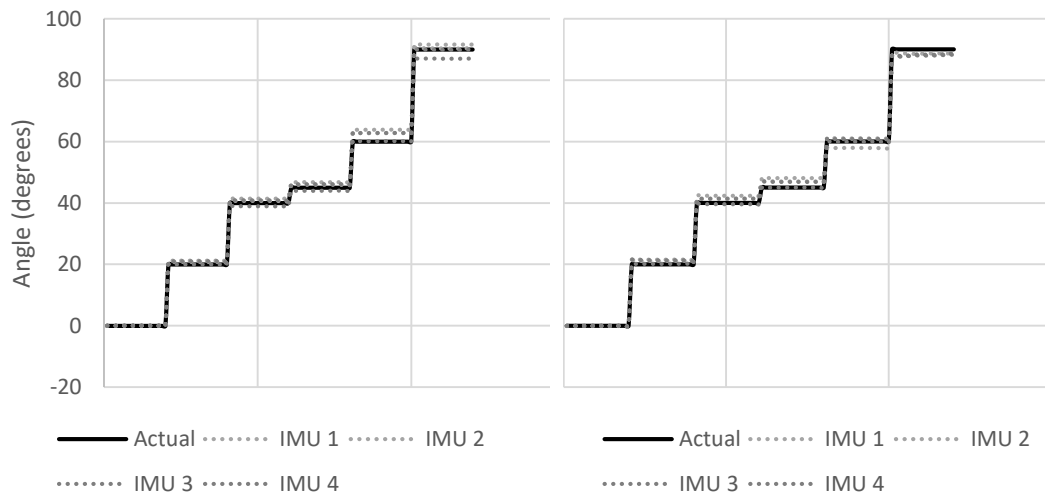


Figure 3-14 – Verification of roll angles against fixed reference

Figure 3-15 – Verification of pitch angles against fixed reference

Although over a short period of time the angles appear to be accurate, the drift of each sensor while maintaining the same angle must be closely measured. This can be seen in Figure 3-16 over a period of 25 seconds. It can be seen that the drift is very small and can be considered negligible. The range between the minimum and maximum values recorded were 0.25, 0.2, 0.14 0.14 degrees respectively for each of the sensors.

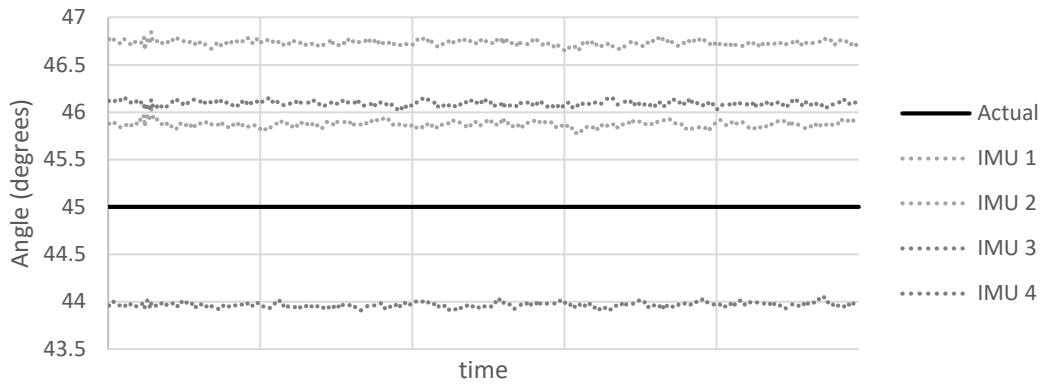


Figure 3-16 – Roll stability over a period of 25 seconds

The accuracy of the sensors should be measured while they are in motion. It can be seen from Figure 3-17 that the sensors closely match to the value of each other. The reference in this graph is estimated by manual observation through a slow-motion video. All the IMU values are very close to the reference value and there seems to be a second order response from each device. There is also a very important phenomenon that is seen by the graph in Figure 3-17. There is a phase delay between the each IMU where IMU 1 has the largest phase delay and IMU 4 has almost no delay. This occurs due the execution of the Kalman filter. As the program is written sequentially, IMU1 is calculated before IMU2 readings are monitored and so on. As a result, when the IMU data is requested, IMU 1 will have a reading that is in the past compared to the reading in IMU 2 and so on. However, for this test, the readings are within tolerance.

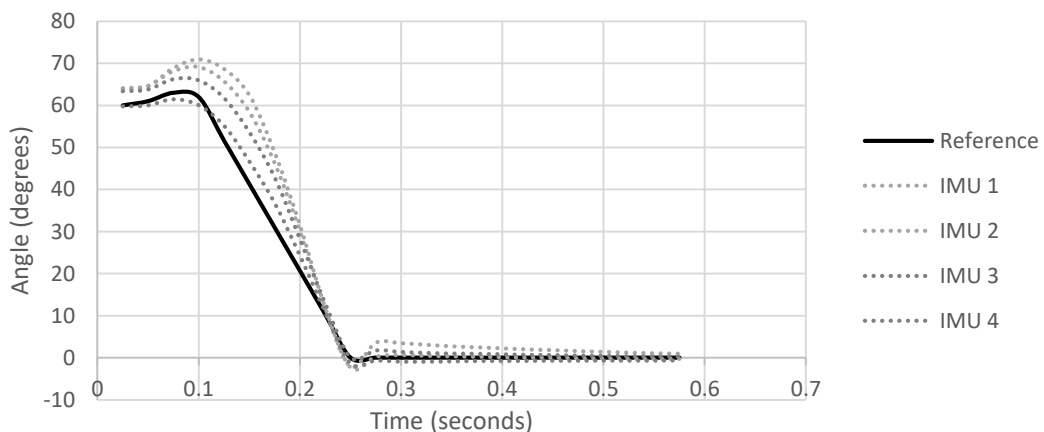


Figure 3-17 – Verification of roll angle accuracy in rapid motion

3.7. Wi-Fi Communication

Wireless communication is important in this project in order to enable the connectivity between the glove and the processing unit driving the classification algorithm. The glove comprises of its own processor which conducts the filtering of the data in real-time. The data is then transferred in packets to the RPi.

For this project multiple gloves require connectivity to a single centralized processor. This is enabled by transmitting data over Wi-Fi. To achieve the same connectivity, the paper by Mukhopadhyay *et al.* [49] summarizes 3 other networks which include ZigBee, Bluetooth and WiMAX. With reference to this list it can be decided that given the application, data rate and the general availability of the technology, Wi-Fi is the best suited technology to be used. Out of the protocols used to communicate over Wi-Fi, the MQTT (Message queuing telemetry transport) communication protocol is one of the most used methods which is also majorly used in the field of IoT devices. MQTT is an extremely light weight machine to machine connectivity protocol which works on the principle of publishing and subscribing to messages [50].

In order to prevent the complete bandwidth of the communication protocol getting used up during transmission, burst communication and continuous communication was used based on bandwidth usage. The transmission method is completely controlled by the user or the RPi that is receiving the data.

- Burst mode (20 - 30 Samples sent at 40Hz) - Used when multiple gloves are used at once or collecting specific data samples.
- Continuous mode (Continuous at 40Hz) - Used for continuous classification when only one gloves is operational.

The worst case scenario (WCS) for one transmitting device over MQTT is calculated.

Number of bits in a single reading = $(7 \times 8) = 56$ bits

Number of bits in a single sample WCS = $(56 \times 16) + (8 \times 15) = 1,016$ bits

Number of bits in a sample with header = $1016 + (25 \times 8) = 1,216$ bits

Number of samples per second at 40Hz = $(1216 \times 40) = 48,640$ bits/sec

The access point hardware used to enable the communication has a transfer speed up to 1,200,000 bits/second. As a result, 48,640 bits/second published by 1 glove over MQTT or 243,200 bits/second published by all 5 gloves over MQTT is not a strain to the Wi-Fi network (Header size estimated to be 25 bytes for this example [51]). However, the limitation lies within the speed of the RPi and the data transferring efficiency of the MQTT protocol itself. As per the experiments carried out by the team Flespi platform [52] it can be seen that the RPi with MQTT over Wi-Fi with SSL has a data transfer speed of approximately 48,003 bits/s. This means that only one glove can be used in its full capacity at time without saturating the network. The control over which glove can transmit at a time will be given to the RPi to ensure collision avoidance.

The glove will continuously publish the status messages over different topics of the MQTT network in order to provide necessary information. These signals include

- “GLOVE STARTED” to indicate the glove is online. It is also used to determine if the glove processor has forcibly undergone an automatic reboot to an error in the program or hardware.
- “MQTT RECOVERED” to indicate when the glove connects to the MQTT server. This also signifies if and when the signal dropped during transmission.
- “INTERRUPT CRASH” to indicate the Kalman filters on the glove have not had sufficient time to execute which results in duplication of previous values.

3.8. Data Pre-processing

Data pre-processing is the step of formatting and adjusting the data such that it is understandable to the learning algorithm and that differentiable features are highlighted. The raw data that is filtered can be passed through a number of different algorithms in order to obtain values that are able to give a better overview of the data.

3.8.1. Data Preparation

The data sent from the glove for pre-processing are the raw data and filtered data (see section 3.6) from the IMU devices on the gloves. The four IMU devices output 36 raw signals monitored on the glove and are listed in Table 3-3. Out of the 36 data monitored on the glove, there are 15 values that are sent from the glove. The data prefixed include

the value of the data collection iteration and the serial number of the person collecting the data. The data suffixed includes the type of activity being collected.

Table 3-3 - Raw glove signals selected for Pre-processing (dark shade)

Middle finger (IMU1)		Index finger (IMU2)		Thumb (IMU3)		Backhand (IMU4)	
acc_x	X	acc_x	X	acc_x	X	acc_x	✓
acc_y	X	acc_y	X	acc_y	X	acc_y	✓
acc_z	X	acc_z	X	acc_z	X	acc_z	✓
$gyro_x$	X	$gyro_x$	X	$gyro_x$	X	$gyro_x$	X
$gyro_y$	X	$gyro_y$	X	$gyro_y$	X	$gyro_y$	X
$gyro_z$	X	$gyro_z$	X	$gyro_z$	X	$gyro_z$	X
α	✓	α	✓	α	✓	α	✓
β	✓	β	✓	β	✓	β	✓
γ	✓	γ	✓	γ	✓	γ	✓

The data collected will have a dimension of $(\mu, 18)$ (15 signals + 1 user data + 1 activity), where μ is the number of data points. This dataset is then fed in to the training algorithm where it is split to 2 segments, Input data denoted by X_{raw} and output data denoted by y_{raw} . y_{raw} contains the output or the reference number of the activity.

$$\dim(X_{raw}) = (\mu, 17)$$

$$\dim(y_{raw}) = (\mu, 1)$$

As the data collected for this project are from a practical scenario using physical sensors, there are instances where the sensors, hardware, network connectivity or software have failed and there are missing data in the observations. The SciKit-learn library set provides an imputer class which provides 4 simple methods to handle missing data [53]. These include replacing the missing data with the mean, median, modal value or a constant value. For the designed glove, when there is an error in the sensors, hardware or the network connection, there is a time-out triggered by the central processor as seen in section 3.5.2. As a result of this, any missing data in the observations will be from the last section of collected data. Therefore, it was decided to ignore the final row of the observation if it contains any missing values. Missing data can also be identified in the form of lack of data points. There is an activity time

window defined for detecting a gesture in section 1.2.2. If the collected data in an iteration has less data points than the required data points in the window of observation, the entire iteration will be ignored.

Categorical data such as the type of activity in output cannot be computed by machines unless they are converted to numerical values. The output here is given as one of the selected gestures in Table 1-1. These values were transformed to numerical values by the data collection process by encoding each value in this category to the number seen in the first column in Table 1-1.

$$\dim(y_{ML}) = (\mu, 1)$$

This method is suitable for most classical machine learning algorithms. However, assigning a column for each output category is beneficial for neural networks as each output category can be assigned to an individual output node on the neural network. This converts the output array of dimension $(\mu, 1)$ to an output array of $(\mu, 12)$, where x is the number of data points.

$$\dim(y_{NN}) = (\mu, 12)$$

Once the data is prepared, the input data, X is preprocessed to obtain meaningful information from the data. The feature extraction/pre-processing is done in two stages in order to make maximum use of the processing capabilities and processing time.

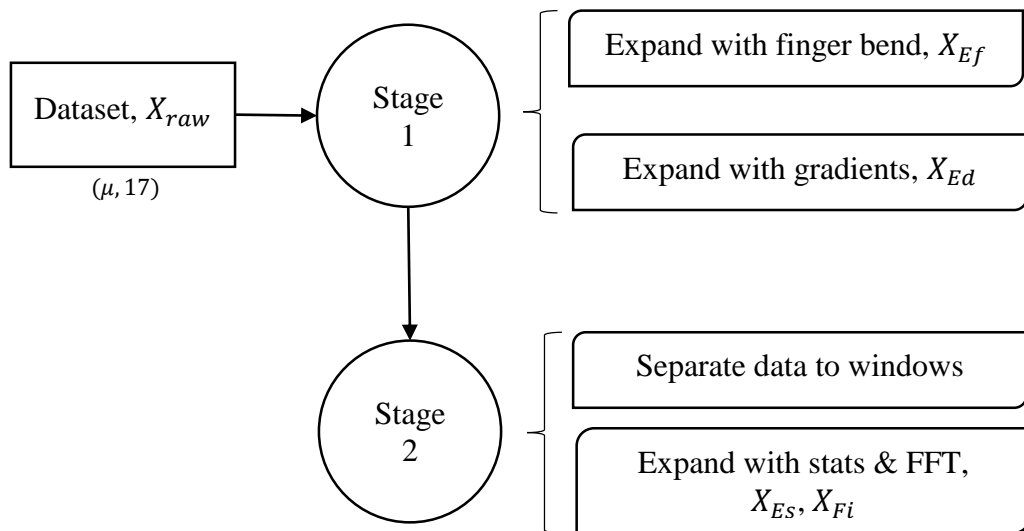


Figure 3-18 – Stages of Data Pre-processing

- **Stage 1:** Includes functions that transform the entire array of data without any regard for the output value or collection iteration of the data.
- **Stage 2:** Includes functions that dissect the data in to individual iterations and further dissect the data to windows to ensure that window contains exactly the number predefined data points needed.

3.8.2. Pre-processing: Stage 1

This stage of pre-processing includes functions that apply to the entire dataset irrespective of the output value of the iteration number. For example: The raw, pitch and yaw data received from the IMU processed to determine finger pose.

Finger bend

The placement of the sensors for the glove described in section 3.1.2 was such that the bend of the thumb, index and middle fingers can be determined by equation (11), where α is roll.

$$\text{finger_bend}_t = \text{backhand } \alpha_t - \text{finger } \alpha_t \quad (11)$$

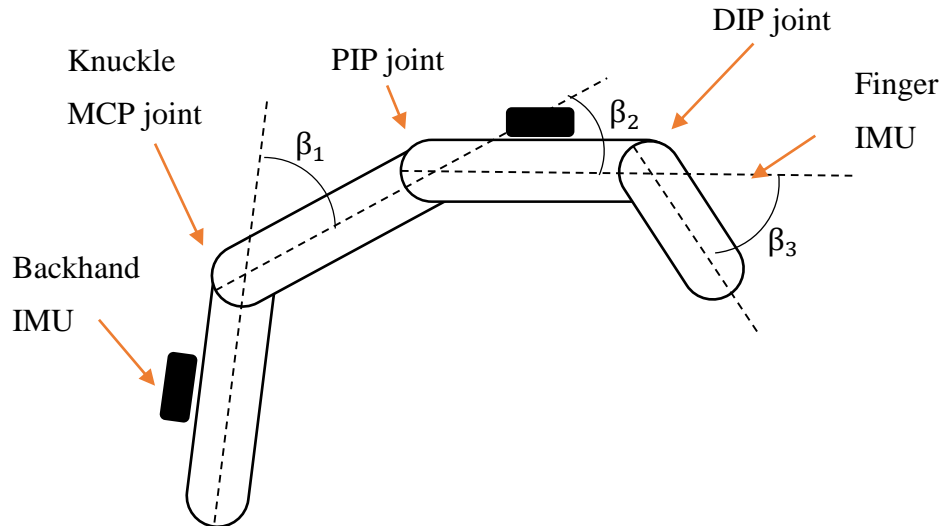


Figure 3-19 – Finger angle determination

Finger Rotation

The thumb has 3 DOF (see section 3.1.2) which allows the thumb to move around to the front of the hand. This motion is measured similar to equation (11) where β is pitch.

$$\text{thumb_rotation}_t = \text{backhand } \beta_t - \text{thumb } \beta_t \quad (12)$$

The 3 finger bend and 1 thumb rotation calculation output values are added to the input array, X_{raw} , which expands this array by 4 columns.

$$\dim(X_{Ef}) = (\mu, 21)$$

Gradients of raw values, accelerometer magnitude and averaging

The gradient or the differential of values is an important to calculate for certain data columns. As seen in section 3.6, the yaw axis of the IMU does not provide a steady signal respect to a fixed reference. The signal is arbitrary and is prone to drift over time. Therefore, equation (13) was used to obtain the change in yaw between two data points than the yaw provided by the glove, where γ is yaw.

$$\Delta\gamma_t = \gamma_t - \gamma_{t-1} \quad (13)$$

Calculating the same for all roll and pitch angles shows the amount of absolute movement in a unit measured time. This is also done for each of the finger bends and the thumb rotation

$$\Delta\text{finger_bend}_t = \text{finger_bend}_t - \text{finger_bend}_{t-1} \quad (14)$$

$$\Delta\text{thumb_rotation}_t = \text{thumb_rotation}_t - \text{thumb_rotation}_{t-1} \quad (15)$$

3-point averaging

The accelerometer data obtained from the glove is unfiltered data. In order to remove certain amount of noise from the accelerometer data the averaging method was found to be reasonably effective. The accelerometer data is sent through a 3-point window average to obtain the filtered accelerometer data.

$$\text{acc}_t = \frac{\text{acc}_{t-1} + \text{acc}_t + \text{acc}_{t+1}}{3} \quad (16)$$

The accelerometer raw data and filtered data are both used to calculate the magnitude of the accelerometer readings [36] in each data point.

$$|\text{acc}_t| = \sqrt{\text{acc}_x^2 + \text{acc}_y^2 + \text{acc}_z^2} \quad (17)$$

In order to execute the gradients and averaging functions a number of past data and future data is required. As a result, the algorithm cannot be run throughout the entire dataset as the data is not a continuous collection of data. As a result, the data is temporarily split to individual iterations as each iteration consists of 1 attempt to capture data. The data in each iteration is then sent through the algorithms. This results in the loss of 1 data point each from the beginning of each iteration and end of each iteration, resulting in the rows in the dimension of array, X_{Ef} being reduced to $\mu - 2i$, where i is the number of iterations of data collection

Finger Abduction and Adduction

The movement of the fingers closer and further apart can be measured as the motion in yaw axis. As the motion in the yaw axis, as described earlier is arbitrary in relative position, the difference in motion in yaw axis is used for this.

$$\text{finger_abduction} = \text{backhand } \Delta\gamma_t - \text{finger } \Delta\gamma_t \quad (18)$$

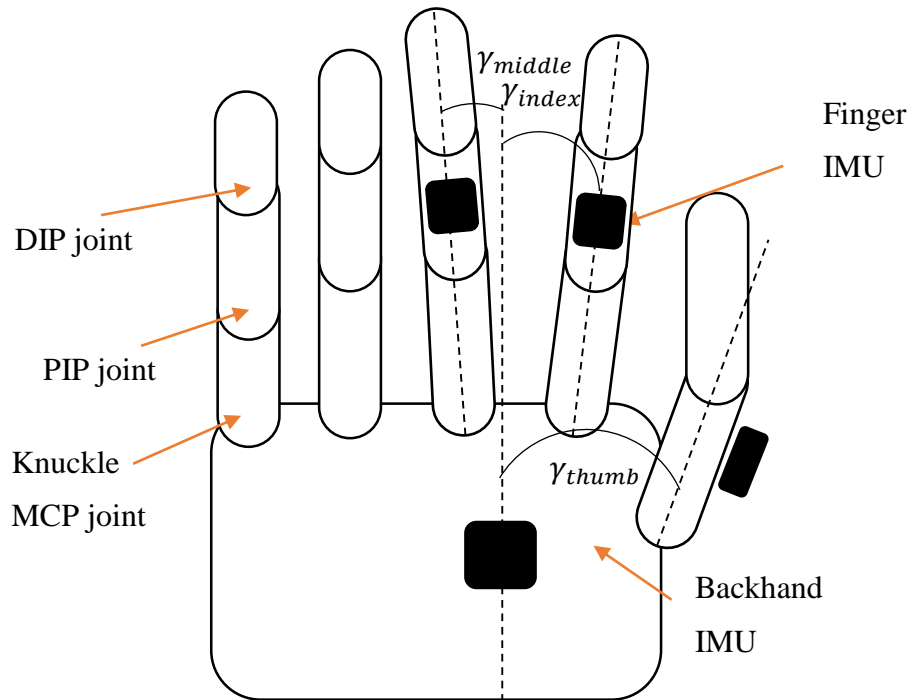


Figure 3-20 – Finger angle determination

With the addition of all the data calculated from equations (11) to (18), a total of 27 new data columns are added to the input array, X_{Ef} .

$$\dim(X_{Ed}) = (\mu - 2i, 48)$$

3.8.3. Pre-processing: Stage 2

In this stage important information or features are extracted from the sensor data by passing it through a number of algorithms. As most of the algorithms in this stage are designed to evaluate the data point, the input dataset, X_{Ed} is first split to individual iterations and then again split to individual windows of data.

As this is the final step in pre-processing, the initial columns of the input data which provides the serial number of the user and the iteration number are separated.

$$\dim(X_{Edw}) = (\mu - 2i, 46)$$

Pre-processing for Machine Learning

Some of the feature extraction methods used commonly in activity recognition are shown in section 2.2. These feature extraction methods along with some customized algorithms for feature extraction from the designed glove is mentioned in this section.

The algorithms in this section were implemented in Python using pre-defined libraries in order to ensure the calculation of the features were efficient. These pre-defined libraries were used from the built in libraries NumPy [54] and SciPy [55]. For algorithms that do not have pre-defined functions, a function was written.

Raw Values

One of the values that are used for the input dataset are the actual raw values obtained from the glove and stage 1 pre-processing. The list of values in the window are flattened to a 1-dimensional array of data.

Histograms

Histogram method, specifically histogram of gradient is generally an algorithm used in human detection in images and it has been used in a paper by Jain *et al.* [41]. In this algorithm, the data of each 1-dimensional array are binned to non-overlapping bins,

where ρ is the threshold. Use of histograms allows the ML algorithm to prioritize certain range of values over others.

$$\begin{aligned}
 &\text{High negative: } x_t - x_{t-1} < -\rho \\
 &\text{static motion: } -\rho \geq x_t - x_{t-1} \geq \rho \\
 &\text{High positive: } x_t - x_{t-1} > \rho
 \end{aligned}
 \tag{19}$$

- 3-bin Histogram of data – Histogram of data split to 3 bins with varying ρ
 - Accelerometer data (bin threshold at ± 1000)
 - Differential roll and pitch values (bin threshold at ± 5)
 - Differential yaw values (bin threshold at ± 10)
- 3-bin Histogram of gradients (HOG) - Histogram of gradient data split to 3 bins
- 12-bin Histogram of absolute bends (HOB) - Histogram of bends calculated for angles from $0^\circ - 180^\circ$ at bin sizes of 15° .

Statistical Values

Calculating various statistical values within the window of data is important to obtain different information about the datasets that will allow each output to be distinguished from each other. A number of algorithms were used as seen in Table 3-4, along with a number of algorithms unique to this project.

Table 3-4 – Statistical formula for pre-processing

	Equation	Ref
Median: of each column denotes midpoint of its frequency distribution.	$\text{med}(x)$	[36]
Mean: The mean provides the average value of each column in the dataset	Equation 20 $\bar{x} = \frac{\sum_{i=1}^N x_i}{N}$	[6], [39], [56]

Variance: The power of the given column of values with its mean removed	Equation 21 $\text{Var}_x = \frac{\sum_{i=1}^N (x_i - \bar{x})^2}{N}$	[6], [36], [56]
Standard Deviation: The deviation of the values from the mean value in a window.	Equation 22 $\text{SD}_x = \sqrt{\frac{\sum_{i=1}^N x_i - \bar{x} ^2}{N}}$	[37], [8], [56]
Minimum/Maximum: Provides the min and max values in a given window	$\min(x)$ $\max(x)$	[8], [35]
RMS: Square root of the average power of a dimension.	Equation 23 $\text{RMS}_x = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i)^2}$	[6], [36], [39]
Skewness: Indicates the lack of symmetry in a dimension	Equation 24 $\text{Skew}_x = \frac{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^3}{\left(\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2\right)^3}$	[6], [39]
Kurtosis: indicates the shape of distribution of the data in a given dimension	Equation 25 $\text{Kurt}_x = \frac{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^4}{\left(\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2\right)^2}$	[6], [36], [39]
Median Absolute Deviation: This provides the absolute deviation of the data from the central value.	Equation 26 $\text{MedAD}_x = \sqrt{\frac{\sum_{i=1}^N x_i - \text{med}(x) ^2}{N - 1}}$	
Mean Absolute Deviation: This provides the absolute deviation of the data from the average value.	Equation 27 $\text{MAD}_x = \sqrt{\frac{\sum_{i=1}^N x_i - \bar{x} ^2}{N - 1}}$	[36], [57], [56]

*custom algorithm

Zero Crossing rate: This is a count of times the data changes its' sign in a given window	Equation 28	[56],
	count if $x_t x_{t-1} < 0$	[34]

*custom algorithm

Slope sign change: This is a count of times the gradient of the data changes its sign in a given window	Equation 29	[34]
	count if $(x_t - x_{t-1})(x_{t-1} - x_t) < 0$	

*custom algorithm

Waveform Length: This calculates the area under the curve of the differential value graph.	Equation 30	[34]
	$l = \sum_{k=2}^L x_k - x_{k-1} $	

*custom algorithm

The statistical calculations are performed on each of the raw values and the finger bend values.

Linear Velocity

A number of activities such as “push” and “pull” have very similar hand spatial orientations and equal but opposite temporal movements. There are not many significant features that are identified to distinguish between similar activities of this nature. As such linear velocity algorithm was introduced which calculated the area under the curve for each of the accelerometer descriptors in order to obtain the velocity in a given window.

$$\dot{x} = \sum_{t=0}^N x_t \quad (31)$$

Fourier Descriptors

Fourier descriptors have also been referenced in [41], [10] and [8]. Using the variables in the frequency domain adds many advantages to the classification algorithm. As the

data collected here are invariant to translation, rotation and scaling in the time axis [41], the data collected in any orientation and hand size can be captured. To achieve the suitable FD's for this project;

1. The mean of the selected data window is obtained
2. The centroid distance is found using equation (32).

$$d(t) = \sqrt{(\Delta\alpha_t - \overline{\Delta\alpha})^2 + (\Delta\beta_t - \overline{\Delta\beta})^2 + (\Delta\gamma_t - \overline{\Delta\gamma})^2} \quad (32)$$

3. The Fourier coefficients are then calculated using the fast Fourier transform (FFT) which is a computationally equivalent to discrete frequency transformation equation in (33).

$$DFT_N = \frac{1}{N} \sum_{t=1}^N d(t) e^{-\frac{j2\pi nt}{N}} \quad (33)$$

Pre-processing for Convolutional Neural Network

The data for a convolutional neural network is prepared in images to match the standard convolution neural network frameworks. A 3-channel image is prepared for each data point.

- Channel 1: Raw values obtained from stage 1 pre-processing
- Channel 2: Magnitude of FFT values from values from stage 1 pre-processing
- Channel 3: Phase of FFT values from values from stage 1 pre-processing

The data pre-processed for machine learning at stage 2 cannot be used as the values obtained from this are up to 12-point values and cannot be used to fill an entire column on an image with meaningful patterns.

Standard applications for CNN use 2-dimensional FFT to pre-process data. However, as individual columns contain no related information to each other, any meaningful data may be lost if FFT is applied across the columns. Therefore, 1-dimensional FFT is applied across individual rows where rows denote the temporal space.

$$FFT = \begin{cases} n = 2r & \text{if even} \\ n = 2r + 1 & \text{if odd} \end{cases} \quad (34)$$

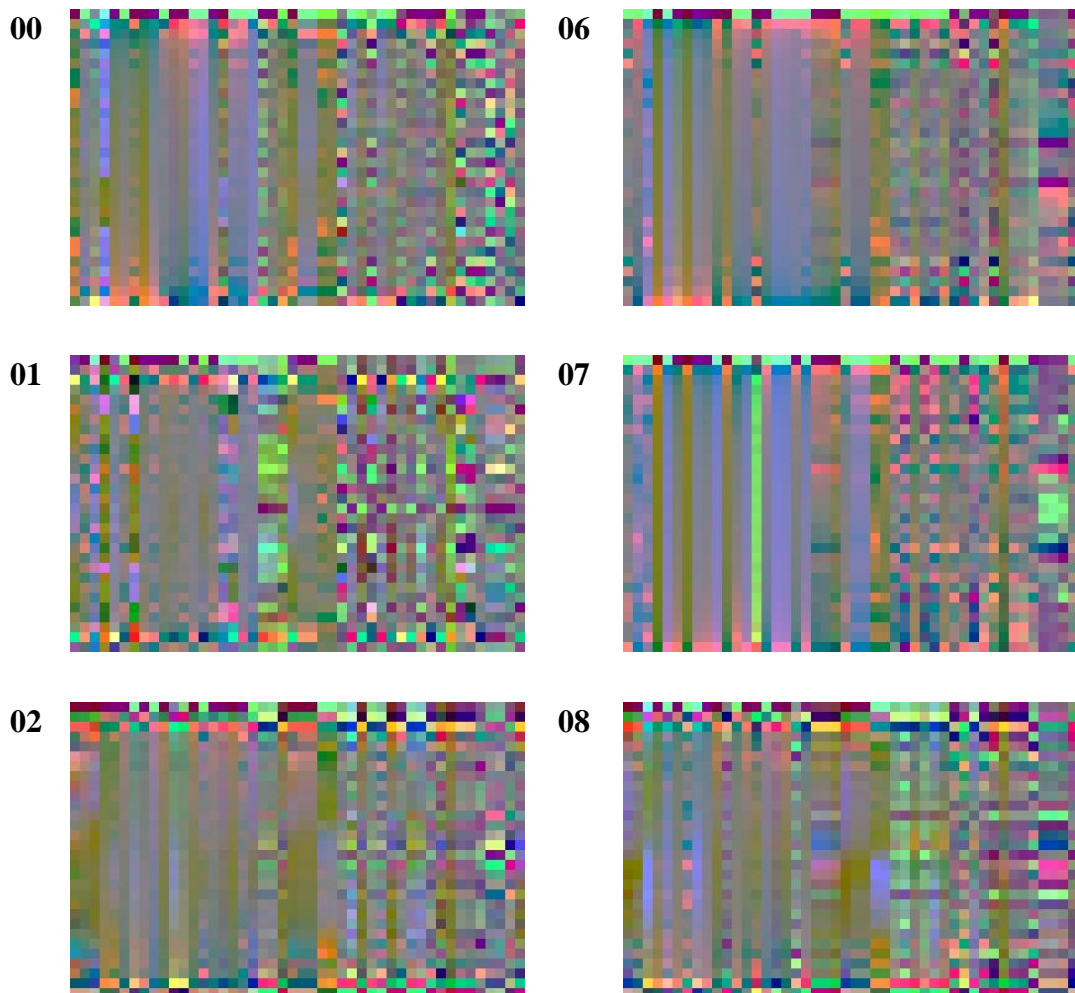
Where $r = 1, 2, \dots, \frac{N}{2} - 1$

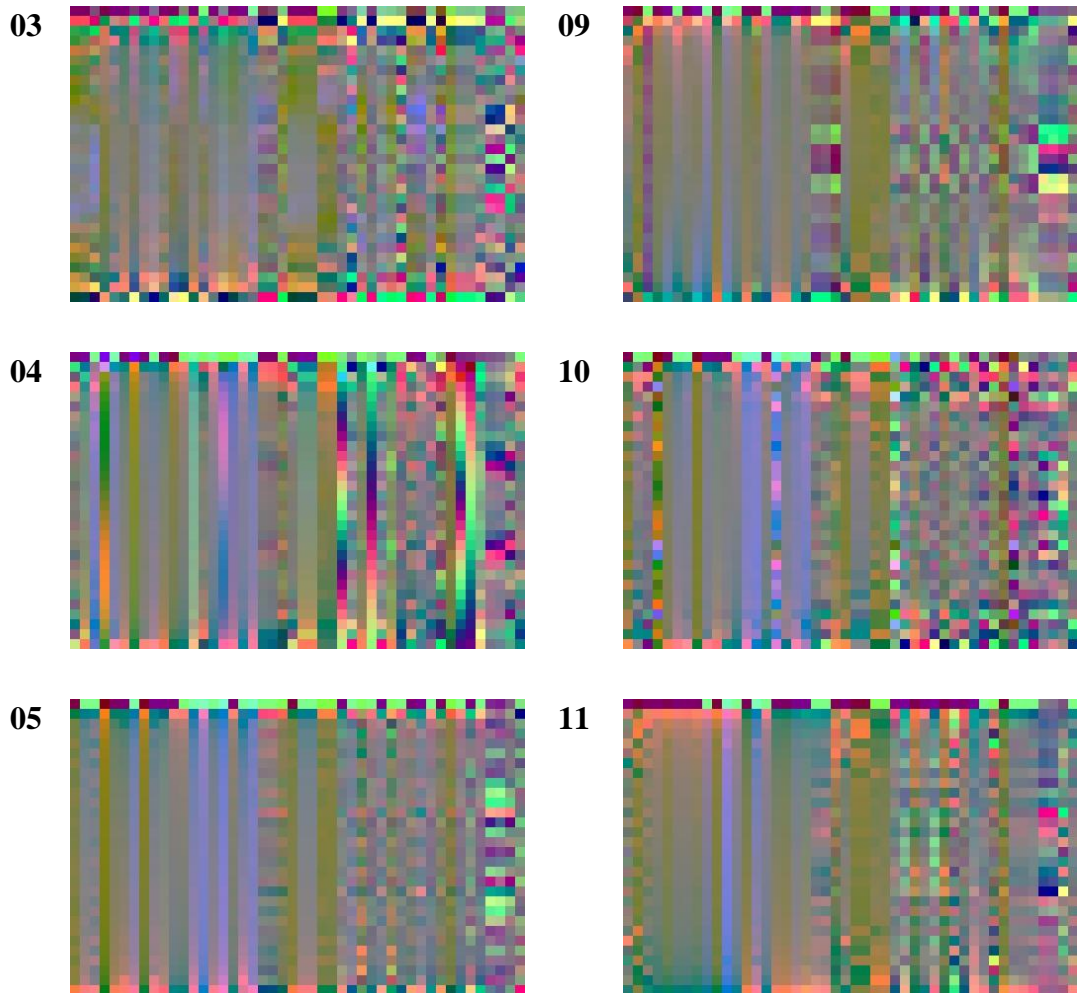
This results in a 4-dimensional array, X_{Fi} ;

- First Dimension: Data points
- Second Dimension: Window size (height of image) (20 or 30)
- Third Dimension: Width of pre-processed dataset (width of image)
- Fourth Dimension: Channel

$$\dim(X_{Fi}) = (\mu - 2i, \text{window}, 46, 3)$$

Table 3-5 – 3-channel activity images derived from raw data and FFT values of data





Examples of the 30 window activity images identified by these projected to the RGB color space is shown in Table 3-5 (also see Appendix 4). It can be seen that, although all the RGB images look in differentiable, the patterns on the each of the images shown in Appendix 4 have distinctly identifiable structures that have formed for each activity.

At the end of pre-processing the data is ready to be passed to the ML algorithms. The dimensions of the final data depend on the number of data points collected per iteration. A data point is considered as a row of input information containing the dimension (1, 17). Considering 35 data points in 1 iteration. There will be 33 data points once the pre-processing is complete. If the classification window size is 30, this results in 4 data windows for classification. This segmentation is done programmatically by looping through the data.

3.8.4. Feature Scaling

Feature scaling is a critical step in both machine learning and neural network algorithms. This involves scaling all the data in the input dataset to the same range of data in order for the learning algorithm to not have any bias.

Standard Scaler

The standard scaler algorithm is a built-in Gaussian scaling algorithm in the SciKit-learn preprocessing class which standardizes the features by removing the mean and scaling the values to unit variance [58].

$$\hat{x} = \frac{x - \bar{x}}{\text{std}(x)} \quad (35)$$

Majority of the values are varying close to the center of the range and the extremities are used very rarely. This resembles a standard Gaussian distribution (normal distribution). Passing these values through a Gaussian scaler which will apply a larger scale factor to the center of the distribution and a smaller scale factor to the extremities will result in a linear series of sensor data for easy classification.

Maximum Absolute Scaler

This scaler was designed for the pre-processing of activity images. This method scales the entire dataset from +1 to -1 in order to ensure that the images formed have the same maximum and minimum values.

$$\hat{x} = \frac{\hat{x}}{\max(|\hat{x}|)} \quad (36)$$

Using the data preprocessing explained in the section, the raw data from the sensors were cleaned to remove missing data and they were processed in to more meaningful data that can be used to discriminate activities from each other. New algorithms were introduced in order to reinforce the discrimination of data value between activities. Further all data fields were scaled between the fixed values in order to remove any bias in the classification process.

4. LEARNING MODEL DEVELOPMENT AND RESULTS

This section of the thesis is mainly a discussion of the results from the performance of machine learning algorithm for the Prototype 2 glove designed for this project.

4.1. Data Collection

The testing data was collected on a simulated industrial environment. The overall data collection summary can be seen in Table 4-1.

Table 4-1 – Test data points collected by participants

Participant 1	2,637	Participant 5	2,829	Participant 8	2,396
Participant 2	1,955	Participant 6	2,038	Participant 9	2,038
Participant 3	2,123	Participant 7	2,246	Participant 10	2,098
Participant 4	2,322				

A total of 54,303 training data points were collected from my hand in all orientations of each activity to ensure the machine learning algorithm can be properly trained. Each test participant has up-to two random and different orientations for each activity. As the training data is only collected from only one participant, there is a general bias in the data and may not be able to accurately capture the differences from one participant to another in a wider participant scope.

Based on prior research detailed in section 2.2, a number of machine learning algorithms have been selected to classify the preprocessed data

Classical machine learning

- Support vector machine (SVM)
- K-nearest neighbors (k-NN)
- Random forest (RF)
- Decision tree (DT)
- Linear discriminant analysis (LDA)

Deep learning

- Convolutional neural network (CNN)

The machine learning algorithms were implemented in Python with the help of SciKit libraries while the deep learning/neural network algorithms were implemented in Python with the help of SciKit, Tensor Flow and Keras libraries.

4.2. Classical Machine Learning

Once the training and the testing data have been collected, they were preprocessed with two-stage pre-processing (See sections 3.8.2 and 3.8.3) and feature scaling (See section 0)

When building the initial pre-processing algorithm list, a number of a trial and error iterations were done to identify the suitable pre-processing algorithms. This was done based on error analysis. Using the training set and data from participant 1 as the development set, the errors identified by the ML algorithms were analyzed.

Table 4-2 – CM - SVM classifier with the RBF kernel

CM		Actual											
		0	1	2	3	4	5	6	7	8	9	10	11
Prediction	0	452	132	1	9	107	0	42	230	42	0	1	185
	1	71	655	4	6	4	0	245	1	0	141	0	9
	2	0	0	236	112	0	0	1	21	0	0	0	0
	3	0	0	152	246	0	0	0	0	0	0	0	0
	4	1	25	0	0	35	0	0	5	0	0	0	4
	5	190	58	0	0	85	942	54	63	0	0	86	0
	6	0	0	0	0	0	0	14	74	0	0	0	0
	7	0	0	0	0	0	0	0	362	0	0	0	0
	8	0	0	0	0	0	0	0	0	508	2	0	0
	9	0	0	0	0	0	0	0	0	0	586	0	0
	10	365	0	0	0	0	423	0	0	0	0	1	0
	11	0	0	0	0	0	88	0	211	0	53	0	304

4.2.1. Error Analysis

The confusion matrix (CM) for the SVM classifier with radial-basis function (RBF) kernel for a prior training model that reached 75% accuracy is shown in Table 4-2. The data labels from 0 to 11 are the same labels detailed in Table 1-1.

It can be seen from the error analysis performed in Table 4-3, a large amount of errors may be resulting from the lack of a few algorithms that can be put in place for the pre-processing.

Table 4-3 – Initial Error analysis for CM for SVM classifier

Actual	Prediction	Suggested solution
Push	Point	Linear movement of the hand is not recorded in pre-processing. This was added with equation (31).
Point	Hold & walk	
Hold	Hold & walk	
Wipe	Point	A slope changing rate must be introduced to identify rapid back and forth movements. This is seen in section 3.8.3.
Walk	Wipe	
Pull	Wipe	
Push	Turn	Angular movement each sensor must be further deconstructed. This was done using a histogram of gradients in section 3.8.2
Tighten	Loosen	
Loosen	Tighten	
Point	Hold	Finger bend values must be explicitly defined. This was done in section 3.8.2.
Turn	Point	
Pick	Point	

The CM after adding the algorithms suggested in Table 4-3 can be seen in Table 4-4. The highlighted cells in Table 4-2 which signifies the highest misclassification numbers are the same in Table 4-4. The overall accuracy for the classifier with updated pre-processing is 88.0%, which is an improvement of 17.3%.

Table 4-4 – Updated CM - SVM classifier with the RBF kernel

CM		Actual											
		0	1	2	3	4	5	6	7	8	9	10	11
Prediction	0	221	0	0	0	0	16	0	0	0	0	0	0
	1	0	255	0	0	0	0	0	0	0	0	0	0
	2	0	0	79	2	0	0	0	0	0	0	0	0
	3	0	0	157	87	0	0	0	0	0	0	0	0
	4	0	0	0	0	53	0	0	0	0	0	0	0
	5	0	0	0	0	0	216	0	0	0	0	0	0
	6	0	0	0	0	0	0	10	0	0	0	0	0
	7	0	0	0	0	0	0	0	52	0	0	0	0
	8	0	0	0	0	0	0	0	0	118	0	0	0
	9	0	0	0	0	0	0	0	0	0	388	0	0
	10	0	0	0	0	0	0	0	0	0	0	82	0
	11	0	0	0	0	0	1	0	0	0	26	0	0

However, it can be seen that most activities continue to be misclassified, due to the actions in one activity being partly captured within the other activity. With analysis over many different iterations and test scenarios, activities that are persistent misclassifications were identified.

- Tighten – Commonly classified as loosen
- Loosen – Commonly classified as tighten
- Walk – Commonly classified as hold & walk
- Hold & Walk – Commonly classified as walk or hold
- Hold – Commonly classified as point or hold & walk
- Turn – Commonly classified as walk

By combining the common fields, where ‘A’ combines tighten and loosen and ‘B’ combines walk, hold and walk and turn, an accuracy of 99% is achieved (88% for

uncombined version). This was tested with the same classifier to obtain the CM in Table 4-5. It can be seen that the result obtained is as estimated. The final accuracy of this classifier is 98.8% with an F1 score of 98.9%.

Table 4-5 – Combined CM - common categories combined

CM		Actual								
		0	1	A	4	5	6	7	8	B
Prediction	0	221	0	0	0	16	0	0	0	0
	1	0	255	0	0	0	0	0	0	0
	A	0	0	325	0	0	0	0	0	0
	4	0	0	0	53	0	0	0	0	0
	5	0	0	0	0	216	0	0	0	0
	6	0	0	0	0	0	10	0	0	0
	7	0	0	0	0	0	0	52	0	0
	8	0	0	0	0	0	0	0	118	0
	B	0	0	0	0	4	0	0	0	493

The improvement in accuracy for the changes done through error analysis can be seen in this section. Although ~99% accuracy was obtained through the combination of similar activities, this is not used for the analysis in the remainder of the thesis. The improvements to pre-processing will be used while discarding the combination of activities, in order to ensure the scope of the project is achieved.

4.2.2. Base Performance

Initially, to identify a base performance, the accuracy of the each of the machine learning algorithms was tested with first testing set with the hypothesized values in section 1.2.3. This involves checking performance with a 30-sample window with all selected pre-processing algorithms.

Due to the accuracy-paradox [59] in comparing each of the machine learning algorithms, the recall and precision is used. The F1 score is used as a single measure to observe both recall and precision.

$$F1 \text{ Score} = \frac{2}{1/\text{Recall} + 1/\text{Precision}} \quad (37)$$

Using the F1 score method, the base case scenario is plotted with its classical machine learning performance in SVM (with RBF kernel), SVM (with linear kernel), k-nearest neighbors (with $k = 5$), random forest classifier (with $n = 20$), linear discriminant analysis (with singular value decomposition (SVD)) and Decision tree. The chart is shown in Figure 4-1.

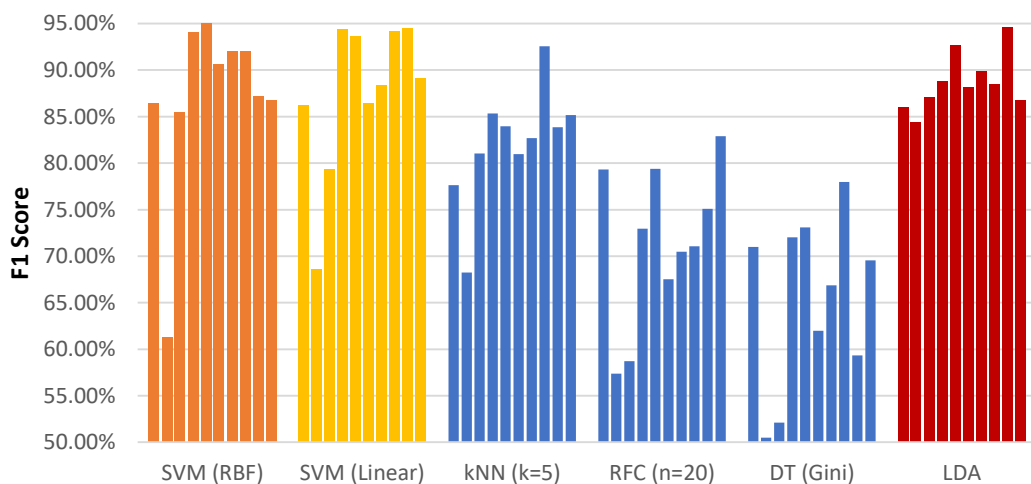


Figure 4-1 - Classical ML F1 Score | (Base Case) 30 Window, Level 1 Pre-process

It can be clearly seen that the performance of the LDA kernel with SVD solver has performed well with an average F1 score of 88.7% while SVM classifier with both RBF (average F1 score of 88.1%) and Linear kernel (average F1 score of 87.5%) have also performed well with the highest single accuracy of about 95% for Participant 9. k-NN classifiers have also performed moderately well with the highest F1 score 92.5% for Participant 8 while having an average F1 score of about 82.1%.

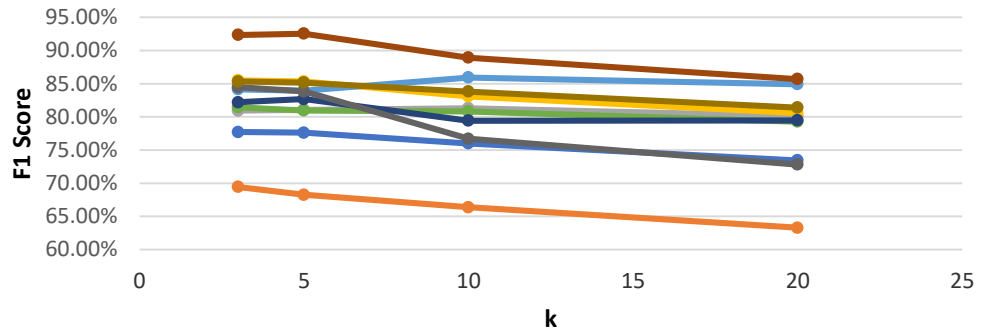


Figure 4-2 – k-Nearest Neighbor performance for k values 3,5,10 and 20 for all participants

The column for the k-NN in the Figure 4-1 only shows $k = 5$. This was chosen by experimenting a number of k values where $k = 5$ was selected as the best overall F1 score for all participants as seen in Figure 4-2. The same was done for random forest Classifier to obtain $n = 20$.

4.2.3. Window Size

The same data as the base case scenario can be plotted by adjusting the window length to 20 instead of 30 (see Figure 4-3). Very similar results can be seen compared to the base case where the LDA and SVM classifiers are outperforming the remaining classifiers.

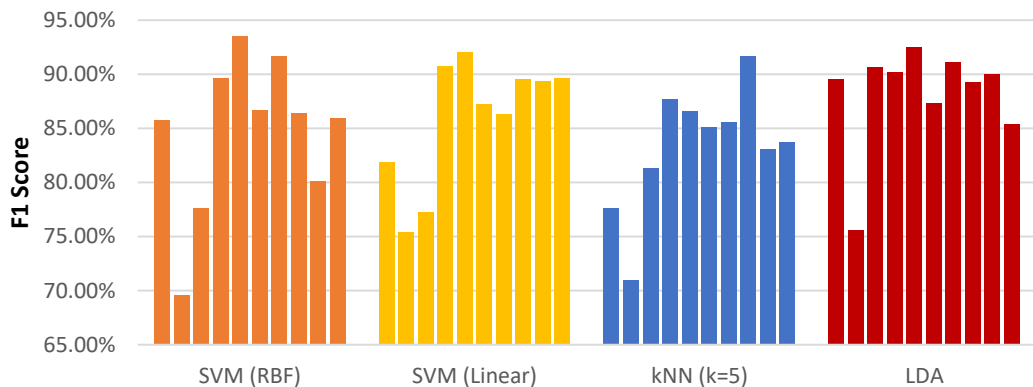


Figure 4-3 - Classical ML F1 Score | 20 Window, Level 1 Pre-process

By comparing the average F1 scores of all 10 participants against each of the window lengths for the top 3 performing classifiers, namely SVM with RBF kernel, SVM with

linear kernel and LDA with SVD, a clear picture can be seen on the overall performance. This chart is shown in Figure 4-4. From this chart it can be seen that

- There is a 0.42% and 1.45% increase in F1 score when using a 20-slide window for SVM with RBF kernel and k-nearest neighbors with $k = 5$, while there is a 2.67% and 0.55% increase in F1 score when using a 30-slide window for SVM with linear kernel and LDA.
- The LDA has the higher average F1 score overall in both window options.

As the increase in F1 score of 0.42% and 1.45% for using a slide window of 20 over 30, is far less than the improvement of 2.67% and 0.55% for using a slide window of 30 over 20, it can be concluded that the performance of the glove for classical machine learning is best performed with a window length greater than 20 or equivalently 0.5seconds.

Both LDA and Linear SVM work on very similar algorithms where it draws a plane between the data points in order to maximize the separability. In my opinion these algorithms perform better than the other algorithms are because they create a linear boundary in each parameter. A linear boundary is better than other clustering algorithms as all activities has spatial and temporal parameters which contains values that can be split with a clear boundary.

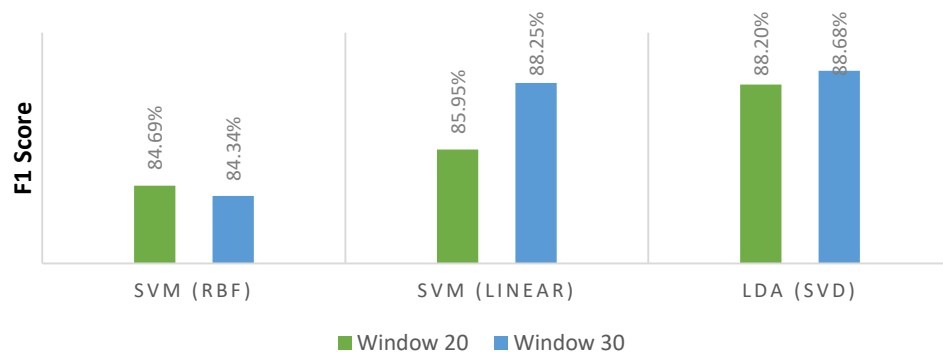


Figure 4-4 - Classical ML F1 Score Comparison for Window Lengths

4.2.4. Manual Dimensionality Reduction

By observing the results of the error analysis and by trial and error, a manual dimensionality reduction was done in 4 levels (see Table 4-6 and Appendix 3). This

was done mainly to eliminate the processing time while maintaining the highest possible F1 score.

Table 4-6 – Manual dimensionality reduction level comparison

Level	Selection	Dimensions for 30-window	Dimensions for 20-window
1	All features are selected	2,151	1,636
2	All features except raw values	755	705
3	All statistical values and Fourier Descriptors only	704	654
4	Statistical values only	322	322

It can be seen from the chart in Figure 4-5, each of the levels have varied improvements. These improvements compared to the base case is shown in Figure 4-6. From this figure it can be seen that the SVM with RBF kernel classify the data with an improvement of 2.5% when there are only statistical values, while the SVM with Linear kernel performs well with level 2, 3 and 4. The best improvement of 3.5% and 1% for the SVM with linear kernel and the LDA respectively is seen with all calculated features excluding raw values. This is justified as the raw values are often noisy due to the nature of the sensors used. This does not allow the classifier to fit the training data well.

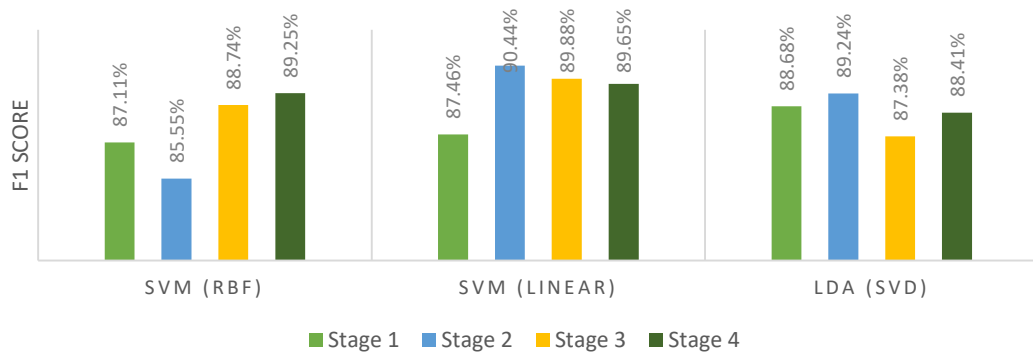


Figure 4-5 – Manual dimensionality reduction level comparison

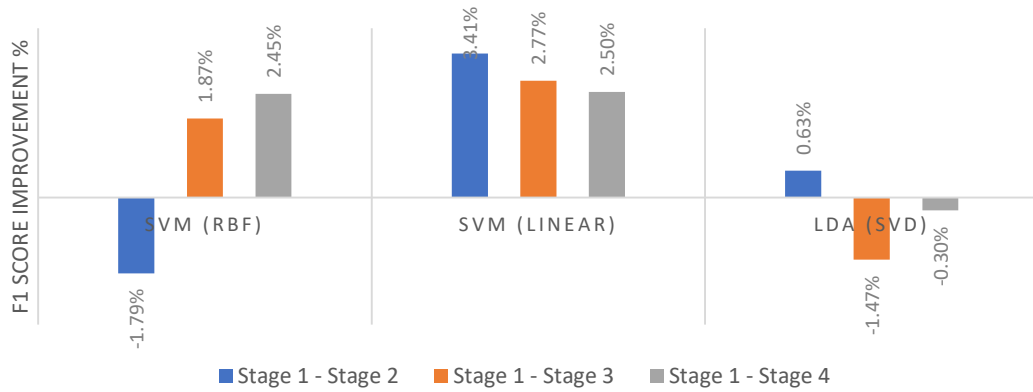


Figure 4-6 – Manual dimensionality reduction level to level improvement.

Using the averaged performance of each of the filters shown in Figure 4-5, the conclusion that can be made is shown in Table 4-7.

Table 4-7 – Classical Machine Learning performance comparison

	Pre-processing Level	F1 Score	Computation
SVM with linear kernel	2	90.44%	High
SVM with linear kernel	3	89.88%	Medium
SVM with linear kernel	4	89.65%	Low
SVM with RBF kernel	4	89.25%	Low
LDA with SVD	2	89.24%	High
SVM with RBF kernel	3	88.74%	Medium
LDA with SVD	1	88.68%	High

The performance of the SVM is also confirmed by the results from the paper by Chaturamali and Rodrigo [60], where the SVM showed a faster computational time compared to other classical machine learning implementations.

4.2.5. PCA Dimensionality Reduction

Given the machine learning application needs to run in real-time or close to real-time, it is important to reduce the load on the classification algorithm. Principal component analysis (PCA) is an algorithm used reduce the feature space of the input array, while

retaining as much information as possible. This helps by improving the interpretability of the data and reducing the time taken to train and run the algorithm [61]. PCA was performed on the training dataset to attempt to reduce the number of dimensions that are used.

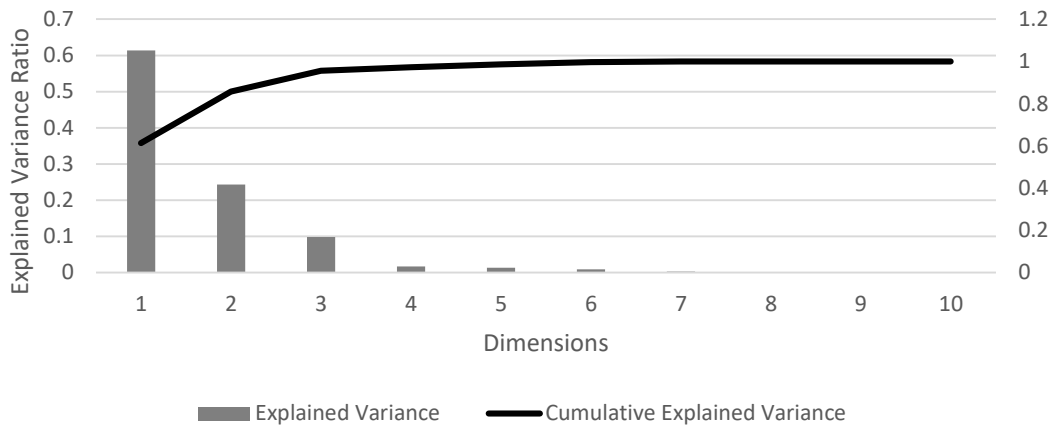


Figure 4-7 – 10 Highest contributing dimensions with PCA

Figure 4-7 shows the 10 highest contributing dimensions when PCA is applied to the training data. Out of the 2,152 dimensions initially identified, the top 6 dimensions can be used to achieve an explained variance ratio of 98.61%. Using the same dimensions for the final model running live on the RPi will have significantly better computation performance with a total dimensionality reduction of 99.72%.

4.3. Null Set Classification

Null set classification is the initial stage of classification of the real data when the activity recognition glove is deployed to the industry. A null set is defined in this project as the set of activities not defined in the scope of the project. Hand activities such as waiting, crossing hands and relaxing. are considered as null sets.

The data for null set identification is trained by clubbing all the training data collected for the project so far against a number of varied hand gestures that do not fall within the previously captured hand gestures. Data sets that are classified as null sets from this classifier are discarded and the remaining data is fed in to activity classification.

Since the null set classification needs to be done fast, it will be done through classical machine learning using the same data needed for the next stage of classification. As it

was seen in section 4.2.3 that data window of 30 pre-processed in stage 2, 3 or 4 is suitable, these are tested for null set classification. The averaged performance for the null set classification for each of the pre-processing stages is shown in Figure 4-8.

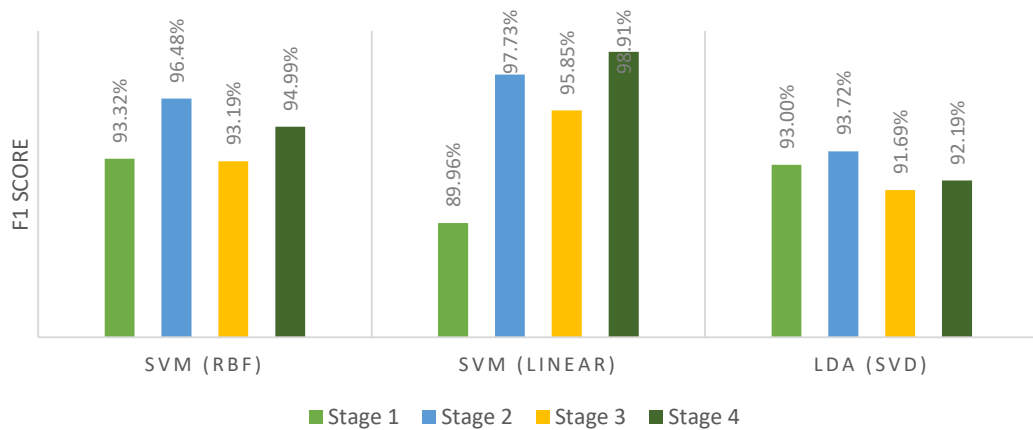


Figure 4-8 – Null set performance for each of the selected algorithms

It should be specially noted that using SVM with linear kernel for the data set passed through level 4 pre-processing obtains an F1 Score 98.91%. This value will immediately improve to 100% by passing it through the Post-processing algorithm described in section 4.5.

4.4. Convolutional Neural Networks

Convolutional neural network (CNN) is a type of deep neural network commonly associated with classifying images. The use of CNN is seen in some activity recognition algorithms where the activity values are Fourier transformed and arranged to form an activity image. A similar approach is used with the activities for this project through Stage 2 pre-processing in section 3.8.3.

Similar to the approach taken with the classical machine learning classification, the classification with convolutional neural networks with the activity images shown in the Appendix 4, is broken down in this section.

4.4.1. Base Performance

The base performance of the neural network is measured against the values in the hypothesis in Section 1.2.3 and the initial convolutional neural network with a number of trial and error attempts. This CNN has the structure in Figure 4-9.

Initial CNN structure:

1. Convolution Layer – 32 x 3x3 Kernel, ReLU Activation
2. Pooling Layer – 2x2 Kernel
3. Convolution Layer – 64 x 3x3 Kernel, ReLU Activation
4. Pooling Layer – 2x2 Kernel
5. Fully Connected (FC) – 128 Nodes, ReLU Activation
6. Fully Connected (FC) – 64 Nodes, ReLU Activation
7. Fully Connected (FC) – 32 Nodes, ReLU Activation
8. Fully Connected (FC) – 12 Nodes, Soft Max Activation

where ReLU is rectified linear unit activation.

CNN Parameters:

- Input Image Size: 30 x 46 x 3
Image height = 30
Image width = 46
Image channels = 3
- Batch Size = 32
- Optimizer: Adam
 - Learning rate: 0.001
 - $\beta_1 = 0.9$,
 - $\beta_2 = 0.999$
 - $\epsilon = 1 \times 10^{-7}$

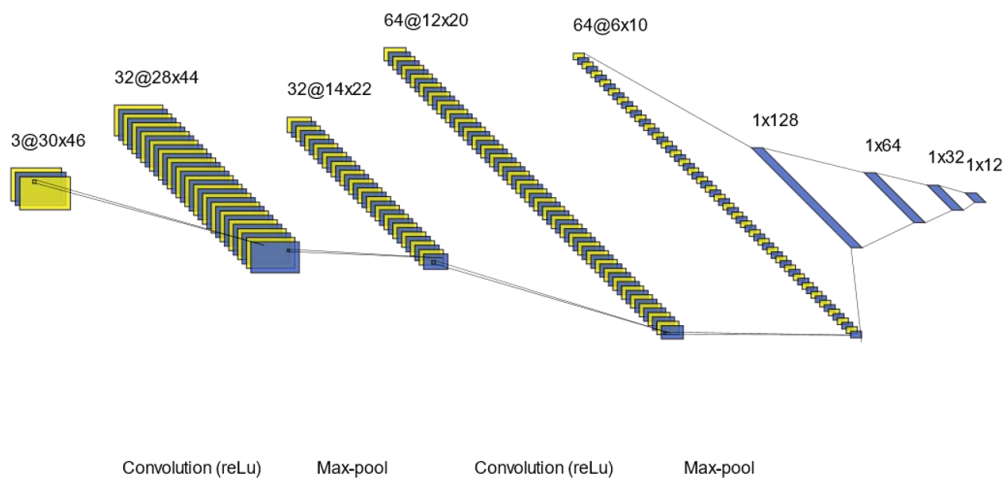


Figure 4-9 – Base CNN structure

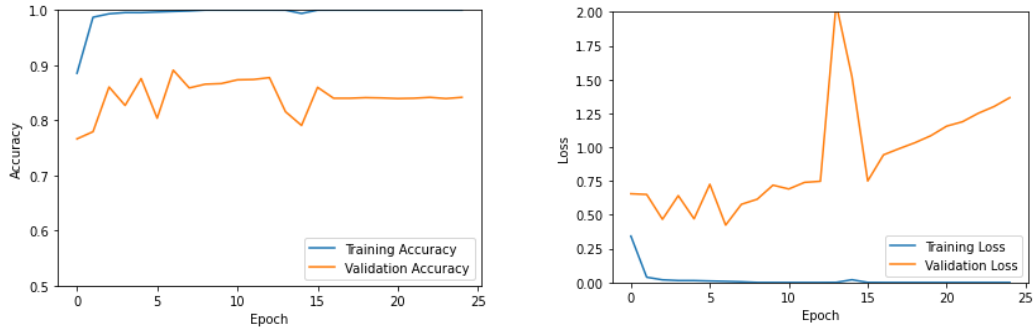


Figure 4-10 – Base CNN performance

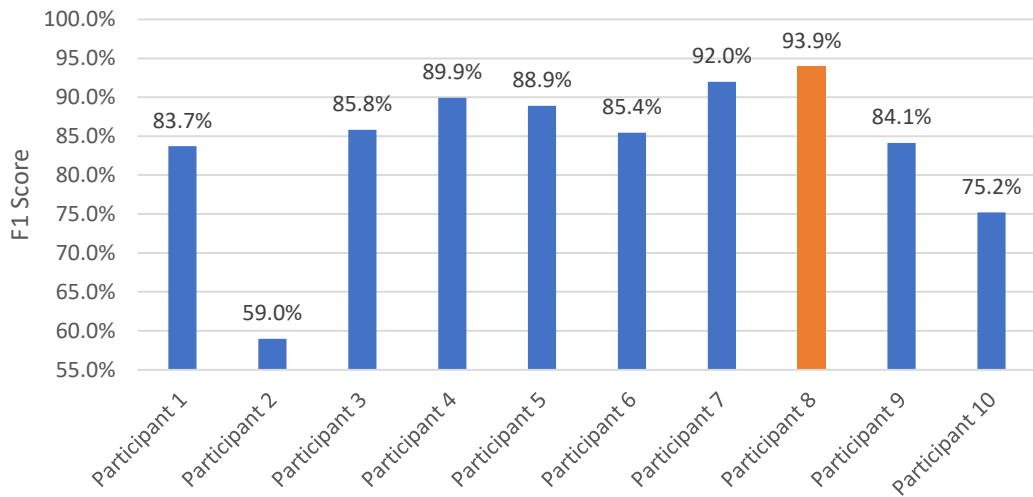


Figure 4-11 – Base CNN F1 scores

It can be seen that the training accuracy quickly rises to 100% while the loss drops to very small values within the initial 3 epochs which means that the neural network is overfitting the data. The validation data which is the test data of participant 1 is saturating at about 85%, close to 18 epochs. However, the validation loss continuously increases indicating overfitting of the neural network. Although the neural network has high accuracy, the network is not sure of the classifications (low probabilities).

The F1 scores of the test data from the trained network is shown in Figure 4-11. It can be seen that participant 8 has the highest F1 score of 93.9% which the entire test set has an average F1 score of 83.8%.

4.4.2. Window Size

The window size hypothesis was retested as in section 4.2.3. The same data was formatted with a window size of 20 and passed through a CNN of the same structure as the base case to obtain the results in Figure 4-12.

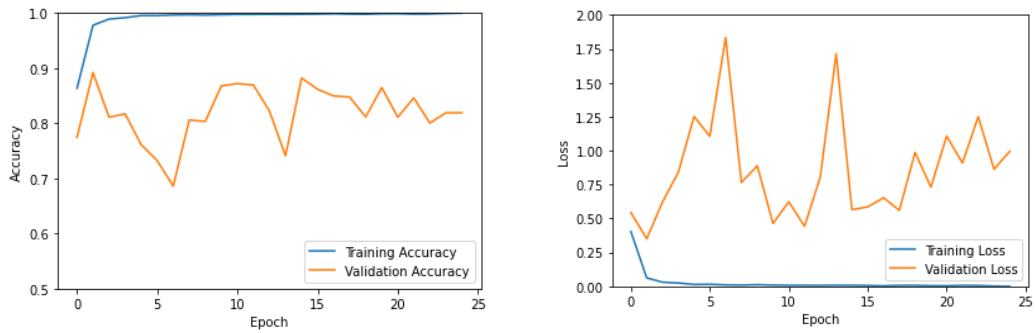


Figure 4-12 – Base CNN performance

It can be seen that the CNN performs poorly compared to the base case scenario. This can be also seen in the F1 score comparison in Figure 4-13. The highest F1 score is obtained at 90.7% for the same participant with an average F1 score dropping to 82.7% for the complete test set.

For CNN's the performance of a window size of 30 is better than the same with the window size of 20. As in classical machine learning, the remainder of the analysis will be continued with the images processed with a window size of 30.

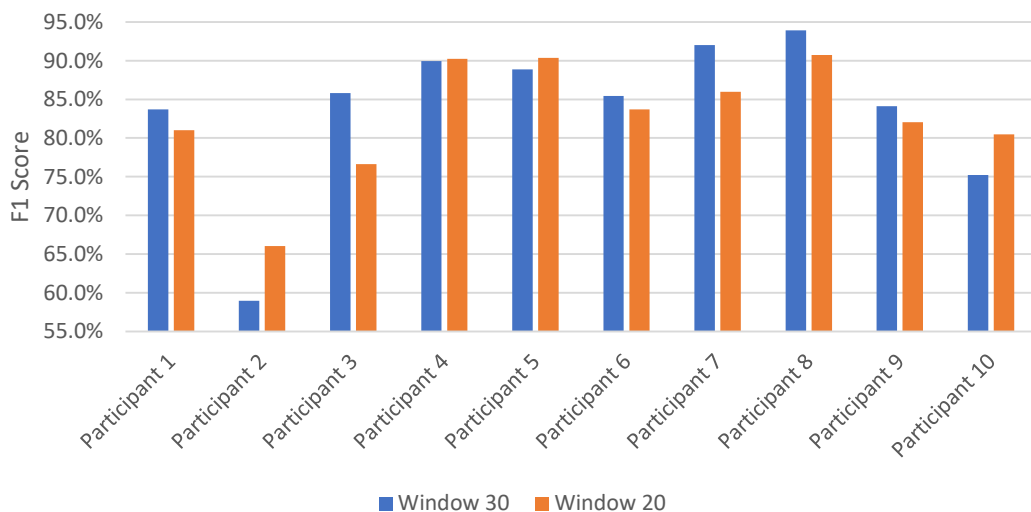


Figure 4-13 – Window 20 vs. 30 CNN F1 scores

4.4.3. 2 Channel Input Network

As mentioned in Section 3.8.3, the 2nd and 3rd channel of the 3-channel activity image is the Fourier magnitude and the phase. As these two channels also provide rich information regarding the movement in each of the sensors, the CNN was retrained only using these channels of the image. The performance of this CNN is shown in Figure 4-14. The figure shows the CNN starts overfitting instantly and does not perform well.

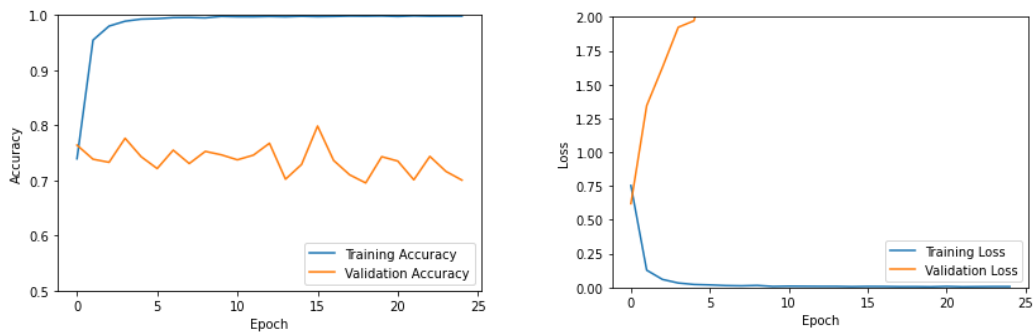


Figure 4-14 – 2 Channel input CNN performance

Again, it can be seen that the base scenario performs better than the 2-channel input CNN. This signifies that the raw data added in channel 1 provides some crucial topographies to the image that help the image to be classified better. The F1 score average for this CNN was obtained at 78.3% for the complete test set.

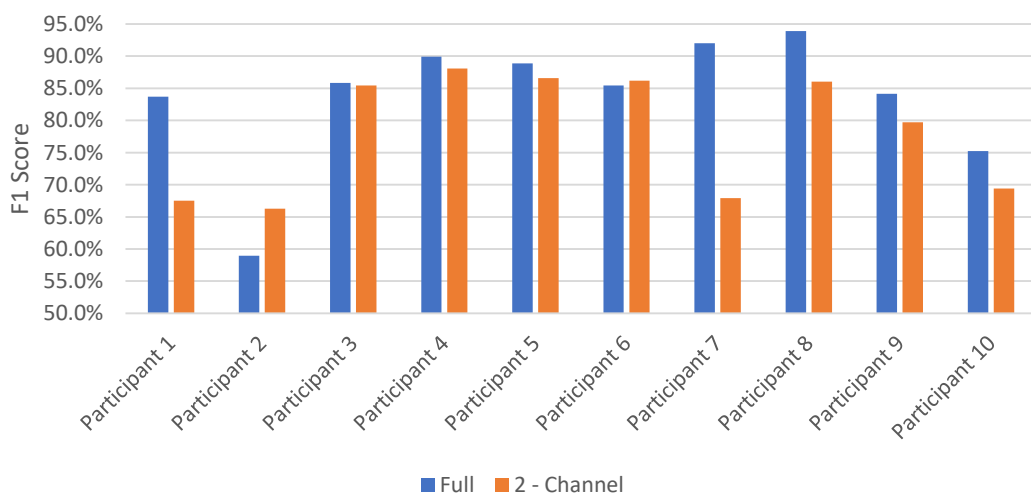


Figure 4-15 – 2-channel vs. 3-channel input F1 scores

4.4.4. Error analysis

In order to test the various effects of the hyper-parameters of the neural network on the performance, the network size was initially reduced to a 2 layer network (hereafter known as 1-CNN) comprising of 1 convolutional layer and 1 max pool payer as shown in Figure 4-16. The performance of the 1-CNN against the base case scenario is shown in Figure 4-17.

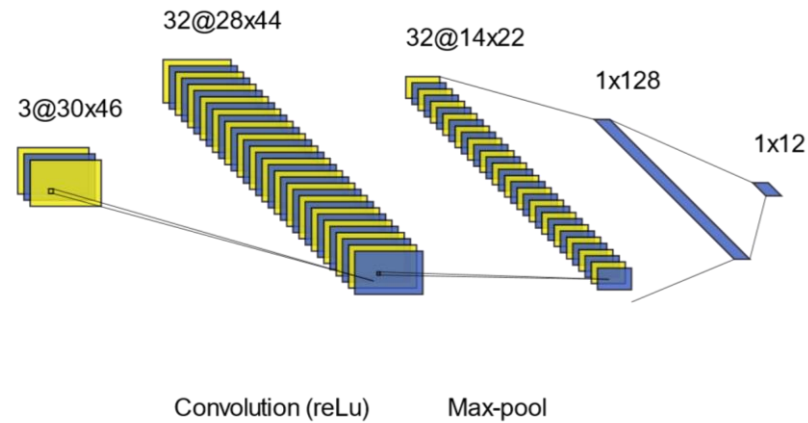


Figure 4-16 – 1-CNN structure

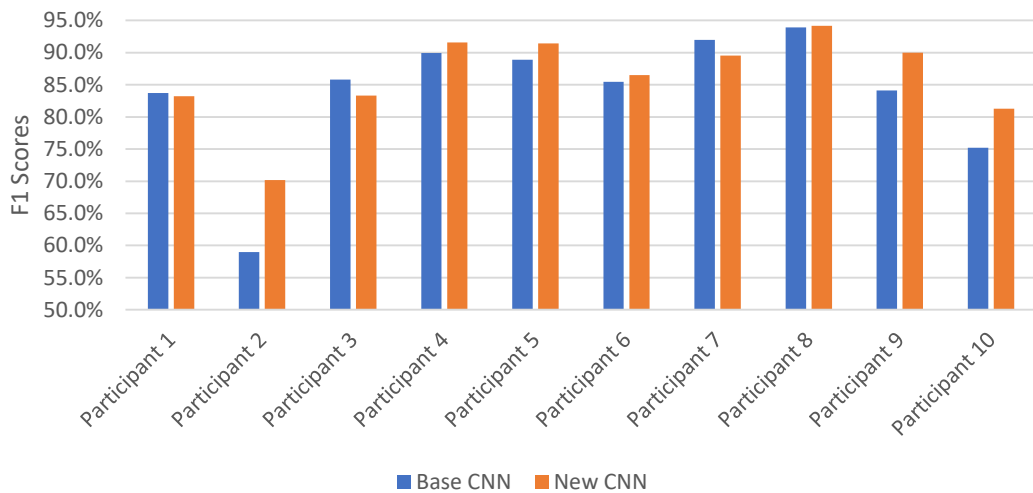


Figure 4-17 – 1-CNN vs base case scenario F1 scores

It can be seen that the 1-CNN performs better than the base case scenario with an average F1 score of 86.1% for the entire test set. It can be seen from the graphs in Figure 4-18, the validation loss over 25 epochs remains reasonably stable, however it

can be seen that when the number of epochs is increased to 500 from Figure 4-19, the validation loss quickly explodes.

As previously stated, it can be seen from graphs in Figure 4-18 and Figure 4-19, the training accuracy saturates at 100% early in the training while the validation accuracy remains close to 86%. Similarly, the training loss is continuously decreasing beyond 4.94×10^{-12} within 25 epochs, while the validation loss explodes and reaches about 5.72 before it saturates at this level at about 300 epochs. This is a clear indication of the network overfitting the training data. In order to address this issue, a number of regularization methods are used to balance the training and validation metrics.

From the loss graph in Figure 4-19, it can be seen that the validation loss is moving back and forth a number of times. Although it is not visible in the training data, the raw values of the training data showed that the values are noisy. This has been identified as the learning rate being too high, preventing the model on optimizing to the global optima. This can be addressed by adjusting the learning rate of the model.

Before making changes to the learning rate and regularization hyper-parameters, the model was also trained using a sigmoid activation function instead of the ReLU activation function to eliminate the possibility of the identified issues arising due to the activation function. This showed in Figure 4-20 that although the accuracy did not improve, the model is still overfitting and the graph is still noisy.

In order to adjust the learning rate to ensure that the model is able to optimize to the global optima of the parameter space, the learning rate was reduced through trial and error. Figure 4-21 shows the performance of the same neural network as Figure 4-18 with the learning rate reduced by a factor of 10. It can be clearly seen that the noise in the validation loss has significantly reduced, and the raw data of the training loss reflects the same.

The classifier used in the network so far is the Adam optimizer (adaptive moment estimation) which is a Stochastic gradient decent algorithm that maintains a single learning rate throughout [62]. By using a modification of the Adam optimizer such as adaptive gradient algorithm (Adagrad) or adaptive delta algorithm (Adadelata) the training can be improved. These optimizers maintain a per-parameter learning rate.

The Adagrad optimizer manipulates the learning rate during training such that the learning rate drops faster for frequent parameters while the learning rate drops slower for less frequent parameters [63], [64].

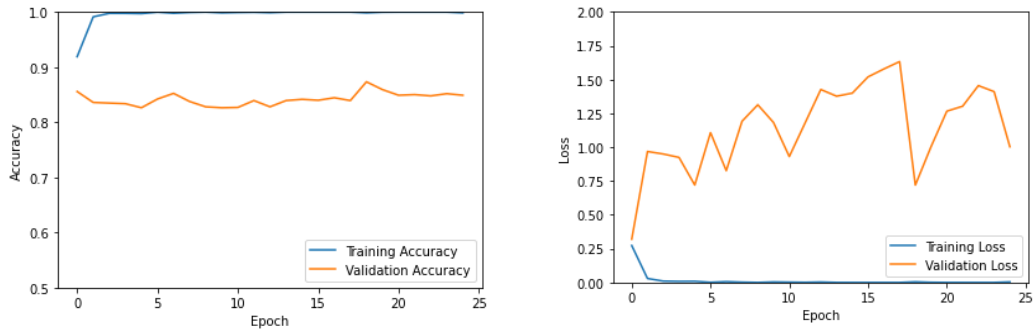


Figure 4-18 – 1- CNN performance

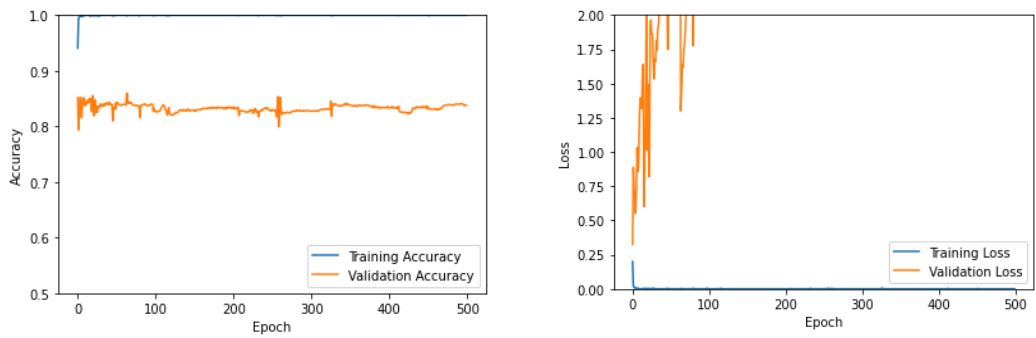


Figure 4-19 – 1-CNN performance over 500 epochs

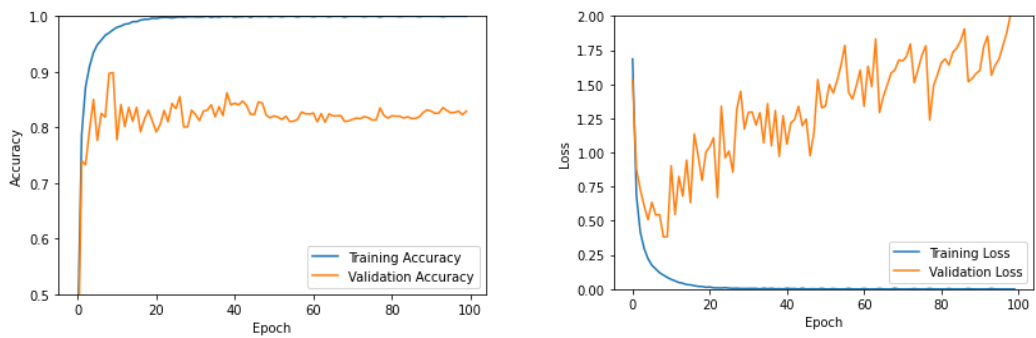


Figure 4-20 – 1-CNN performance with sigmoid activation

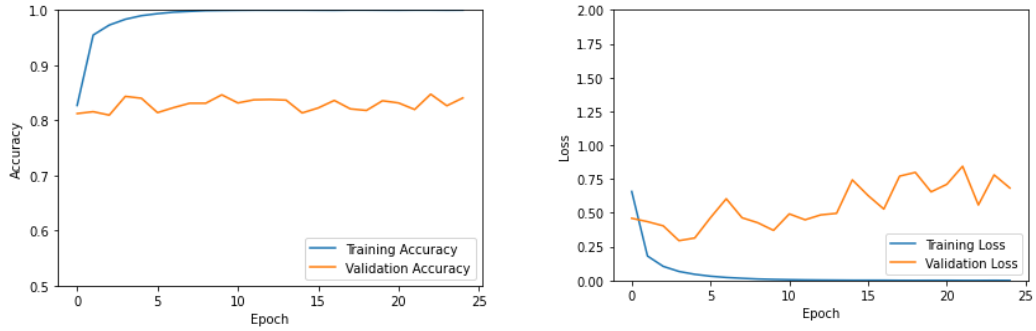


Figure 4-21 – 1-CNN performance with slower learning rate (0.0001)

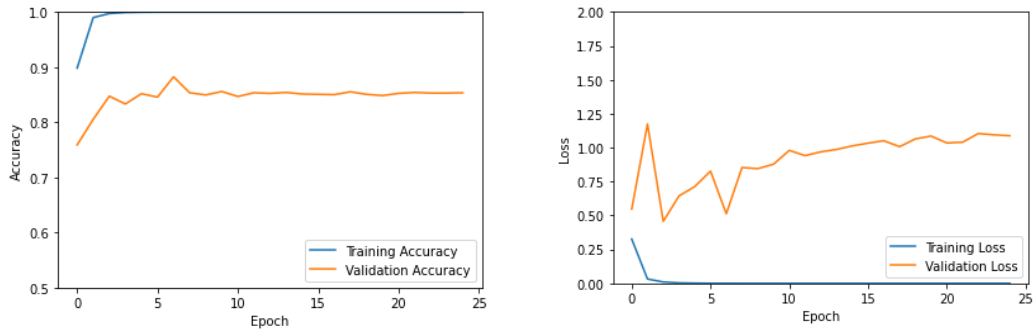


Figure 4-22 – 1-CNN performance with Adadelta optimizer

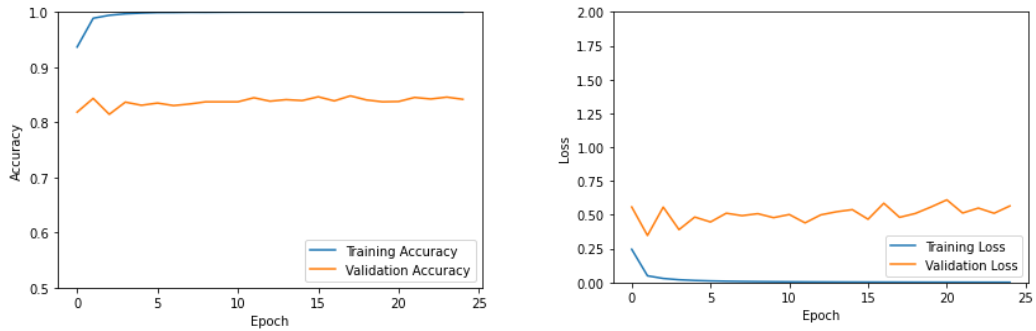


Figure 4-23 – 1-CNN performance with Adagrad optimizer

From the performance of the adaptive learning rate functions shown in Figure 4-22 and Figure 4-23, it can be seen that the Adagrad algorithms performs the best.

In order to address the overfitting of the training data, dropout regularization is introduced to the training model. Dropout regularization is the technique of randomly ignoring a number of units in a neural network in order to efficiently reduce over fitting by preventing complex co-adaptations on the training data [65]. Initially a dropout regularization ratio (DRR) of 0.2 was tested on the input layer and in the first hidden

layer. The performance of these can be seen in Figure 4-24 and Figure 4-25 respectively.

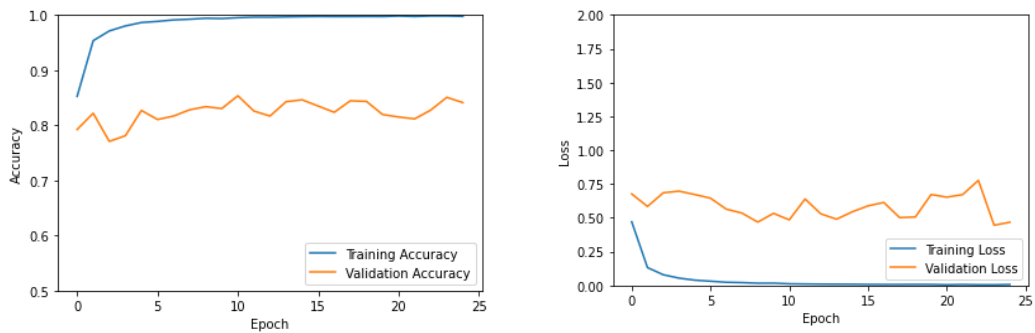


Figure 4-24 – 1-CNN performance with input layer dropout regularization

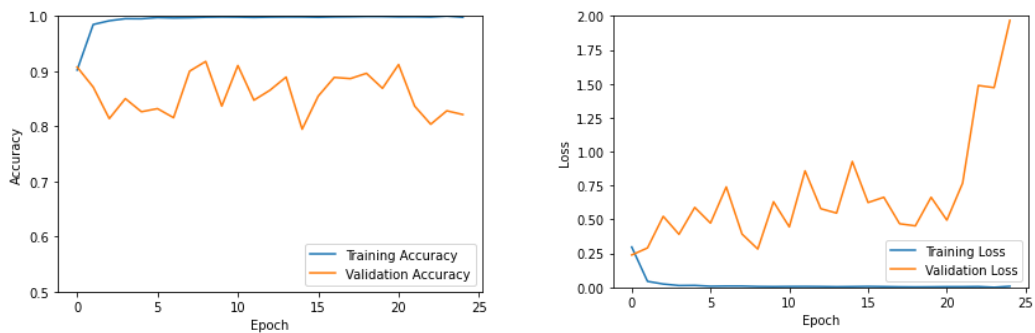


Figure 4-25 – 1-CNN performance with hidden layer dropout regularization

It can be seen that in both instances, the training data does not reach 100% accuracy and the training loss is moderately less for the input layer dropout regularization. The number was selected with trial and error as high dropout regularization ratios will significantly impact performance as seen in Figure 8-12.

By combining these two hyper-parameter changes we can obtain a suitable 1-CNN. This network will consist of the Adagrad optimizer with input dropout regularization ratio of 0.01. As seen in Figure 4-27 the new network performs better than the base case network with an overall F1 score of 85.1% for the entire test sets. It can be seen that the network does not perform as well as the neural network with the Adam optimizer and no dropout regularization. However, from Figure 4-26, it can be seen that the loss of the new neural network is much lower than the previous case shown in Figure 4-19. This means that the neural network has fit the training data quite well and the classifications have a higher confidence.

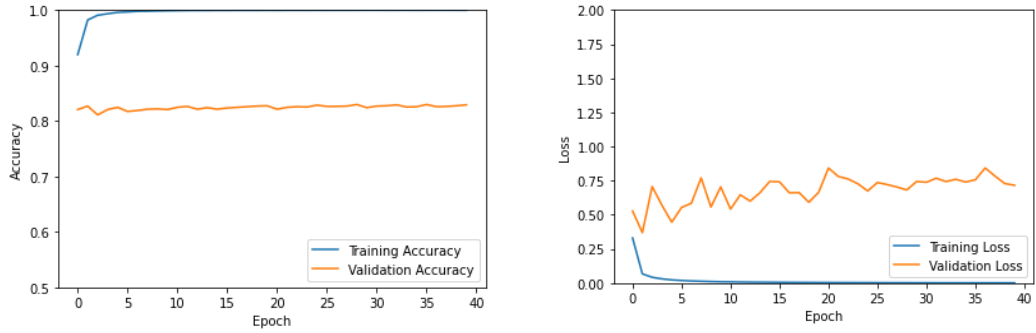


Figure 4-26 – 1-CNN performance with optimizer tuning and DRR=0.01

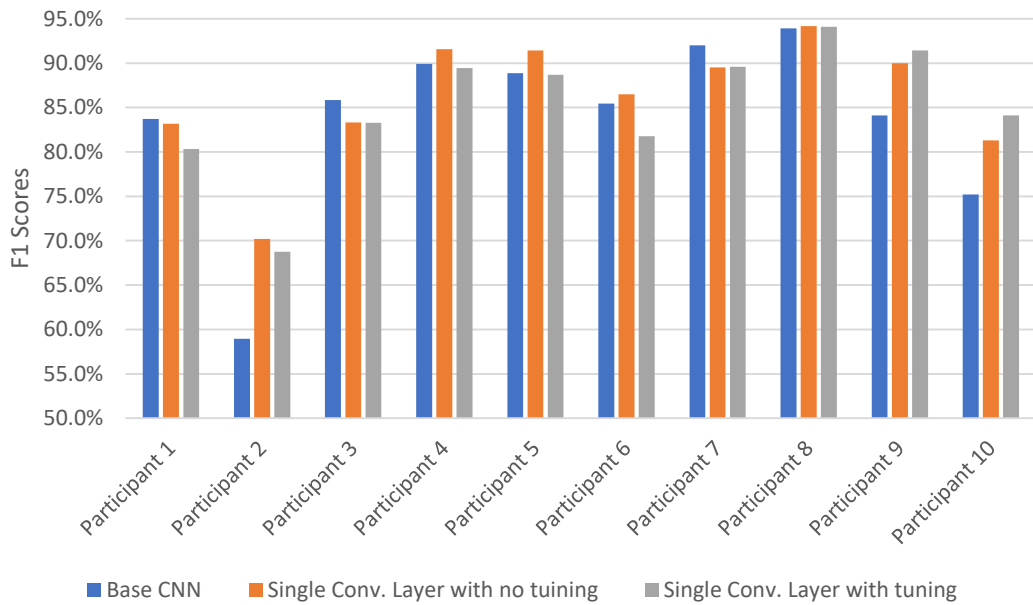


Figure 4-27 – 1-CNN comparison with Adagrad optimizer and DRR=0.01

Looking at the CM for the neural network in Table 4-8, the errors can be analyzed. It can be clearly seen that the errors occurred by the neural network is mostly due to the fact that it is not capable of differentiating between the same activities identified with the exception for the activity for “hold & walk”.

Compared to the classical machine learning algorithm, the neural network has been able to differentiate the activity hold & walk with its similar activities while failing to the differentiate the remaining activities also misclassified by the classical machine learning algorithm. By combining these activities, a better performance can be achieved by the same neural network. This is seen in Table 4-9 where ‘A’ is the

combination of “tightening” & “loosening” and ‘C’ is the combination of “walk” and “turn”.

Table 4-8 – CM – 1-CNN with Adagrad optimizer & DRR=0.01

CM		Actual											
		0	1	2	3	4	5	6	7	8	9	10	11
Prediction	0	208	0	1	0	0	25	0	0	0	0	3	0
	1	0	250	5	0	0	0	0	0	0	0	0	0
	2	0	0	80	1	0	0	0	0	0	0	0	0
	3	0	0	240	4	0	0	0	0	0	0	0	0
	4	0	0	0	0	3	0	0	0	0	0	0	0
	5	0	0	0	0	0	263	0	0	0	0	3	0
	6	0	0	0	0	0	0	10	0	0	0	0	0
	7	0	0	0	0	0	0	0	52	0	0	0	0
	8	0	2	2	0	0	0	0	0	114	0	0	0
	9	0	0	0	0	0	0	0	0	0	388	0	0
	10	0	0	0	0	0	0	0	0	0	0	82	0
	11	1	0	0	0	0	0	0	0	0	26	0	0

This improves the accuracy from 85% for the classifier without activity combination to 97.67% for the classifier with activity combination.

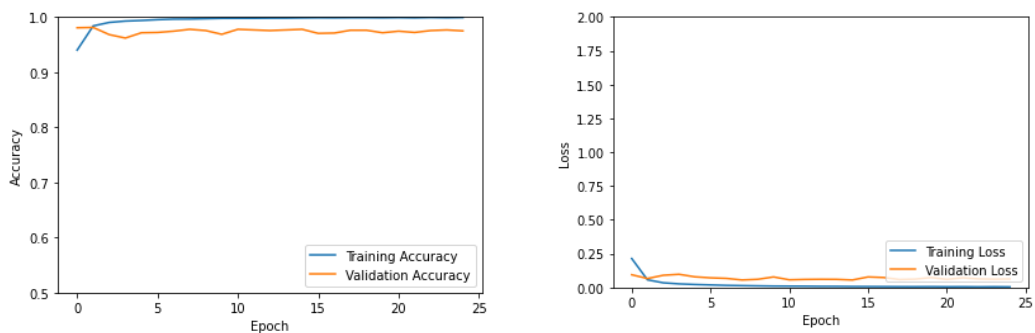


Figure 4-28 – Combined 1-CNN performance with optimizer tuning and DRR=0.01

Table 4-9 – Combined CM - activities for 1-CNN with Adagrad optimizer & DRR=0.01

CM		Actual									
		0	1	A	4	5	6	7	8	C	10
Prediction	0	208	0	1	0	25	0	0	0	0	3
	1	0	250	5	0	0	0	0	0	0	0
	A	0	0	325	0	0	0	0	0	0	0
	4	0	0	0	3	0	0	0	0	0	0
	5	0	0	0	0	263	0	0	0	0	3
	6	0	0	0	0	0	10	0	0	0	0
	7	0	0	0	0	0	0	52	0	0	0
	8	0	2	2	0	0	0	0	114	0	0
	C	0	0	0	0	0	0	0	0	414	0
	10	0	0	0	0	0	0	0	0	0	82

The important checkpoints of each of the testing iterations done for identifying the best-case scenario can be found in the Appendix 5.

4.5. Post Classification Processing

The post classification processing algorithm is a novel algorithm was written with the idea that classified activities within a short window cannot switch back and forth rapidly. In other terms, an activity recorded for $t = 1, 2, 3$ cannot switch to another activity at $t = 4, 5$ and revert back at $t = 6, 7 \dots$ These changes are identified in the classification and adjusted by passing through a categorical low pass filter. This is implemented in Python code.

```

def postProcess (y, preWindowSize, postWindowSize, threshold):

    for i in range (preWindowSize, (np.size(y)-postWindowSize)):
        window = y[i-preWindowSize:i+postWindowSize+1]
        pred = [0] * (np.amax(y)+1)

        for j in range (0,len(window)):
            val = int(window [j])
            pred[val] += 1

        maxVal = np.amax(pred)
        if maxVal >= threshold:
            maxIndex = pred.index(max(pred))
            y [i] = maxIndex

    return y

```

Using the function, the window length (before & after the current value) for the filter as well as the threshold can be set manually.

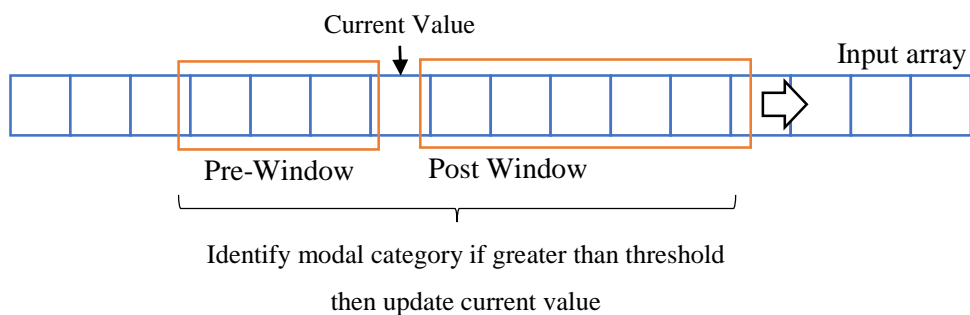


Figure 4-29 – Visual representation of Post Processing algorithm

For this project, through a number of trial and error attempts it was identified that the post processing algorithms needs to be implemented twice with the different parameters

- Pass 1 – pre-window = 1, post window = 1, threshold = 2
- Pass 2 – pre-window = 30, post window = 10, threshold = 31

Using these two passes in each of the classical machine learning algorithm output and convolutional neural network output, the chart in Figure 4-30 is seen.

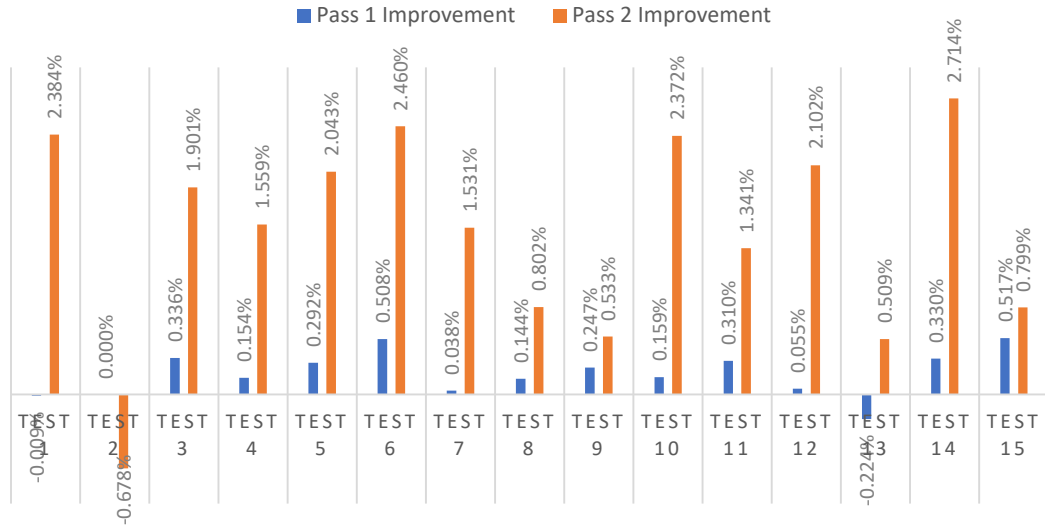


Figure 4-30 – Post processing results from 15 tests done with classical ML and NN classifiers

Table 4-10 – Example CM prior to passing through post processing algorithm

CM		Actual											
		0	1	2	3	4	5	6	7	8	9	10	11
Prediction	0	125	0	0	0	0	0	0	0	0	0	0	0
	1	0	107	0	0	0	0	0	0	0	0	0	0
	2	0	0	71	34	0	0	0	0	0	0	0	0
	3	0	0	6	115	0	0	0	0	0	0	0	0
	4	0	0	0	0	1	0	0	0	0	0	0	0
	5	0	0	0	0	0	71	0	0	0	0	0	0
	6	0	0	0	0	0	1	19	0	0	0	0	0
	7	0	0	0	0	0	0	0	10	0	0	0	0
	8	0	0	0	0	0	0	0	0	69	0	0	0
	9	0	0	0	0	0	0	0	0	0	233	0	0
	10	26	0	0	0	0	0	0	0	0	0	156	0
	11	0	0	0	0	0	0	0	0	0	0	0	137

It can be immediately seen that the post processing algorithm is capable of correcting misclassified data points to provide an F1 Score improvement varying from about 0% to 2.5%. There are also a few tests that have been negatively affected by this algorithm where correctly classified data are misclassified by this algorithm.

The CM shown in Table 4-10, is shown prior to passing this data through the post processing algorithm. This CM has an F1 score of 94.2%. The CM in Table 4-11 is the same output once passed through post-processing algorithm. This has improved the overall F1 score to 94.6%, however it can be seen that some of the correctly classified data has also been misclassified.

Table 4-11 – Example CM after to passing through Post processing algorithm

CM		Actual											
		0	1	2	3	4	5	6	7	8	9	10	11
Prediction	0	125	0	0	0	0	0	0	0	0	0	0	0
	1	0	107	0	0	0	0	0	0	0	0	0	0
	2	0	0	69	36	0	0	0	0	0	0	0	0
	3	0	0	5	116	0	0	0	0	0	0	0	0
	4	0	0	0	0	1	0	0	0	0	0	0	0
	5	0	0	0	0	0	71	0	0	0	0	0	0
	6	0	0	0	0	0	0	19	0	0	0	0	0
	7	0	0	0	0	0	0	0	10	0	0	0	0
	8	0	0	0	0	0	0	0	0	69	0	0	0
	9	0	0	0	0	0	0	0	0	0	233	0	0
	10	24	0	0	0	0	0	0	0	0	0	156	0
	11	0	0	0	0	0	0	0	0	0	0	0	137

Through testing, it was found that this algorithm works best for accuracies over 90% where a large number of data points are not misclassified. This classifier works best when there are only a very few numbers of outliers/misclassifications in the output.

5. OVERALL PERFORMANCE RESULTS

5.1. Hardware Performance

The design of the prototype 2 glove was done to an intermediate stage due to the limitations due to the CoViD-19 lockdown. The intermediate and final designs of this prototype consist of very similar design and can be summarized as in Table 5-1

Table 5-1 - Design specification summary of the smart glove

Design aspect	Value
Activities selected	12: All activities in Table 1-1
Sensors selected	MPU6050 for back hand and fingers
Measurements	Hand Movement – roll, pitch & differential yaw Finger bend, Finger abduction, Thumb rotation
Glove	Cotton dotted glove
Glove processor	WEMOS D1 Mini – ESP8266
Classification processor	Raspberry Pi 4
IMU filtering	Kalman filter on glove
Pre-processing	Statistical and Fourier
Classification	Machine learning, neural network
Post processing	Categorical low pass filter (see section 4.5)

The design performance of the prototype will be measured in both functional design aspects and ergonomic design aspects.

Overall, the functional design of the glove was very successful. The sensors and the transmission methods worked seamlessly with the selected processor to capture the required data and transmit as needed.

The sensors selected for the glove were able to capture all the required motions of the fingers from their selected positions. There were a number of errors registered by the sensor when manually inspecting the data and this was found mainly to be due to the

gimbal lock on the IMU. The PLL axis of the IMU was set to the X-axis to ensure the sensor will be in gimbal lock when the hand is facing vertically upwards. This was determined to be acceptable as it is very rarely, the hand and fingers will be facing straight upwards or straight downwards in an industrial working environment.

The processor was capable of consistently maintaining a cycle frequency close to 80Hz. And very rarely dipped below 60Hz. During these very rare occasions, a message will be displayed on the GUI (graphical user interface) to take action against this data. The filtered data values were consistently similar to the values obtained in section 3.6.2.

The transmission network was capable of transmitting all the required data from the gloves to the central processor. The data collected in batches were delivered with negligible delay. However, a very small delay was seen in the transmission when data is collected for a large amount of time. This delay will be negligible for short data collection times within 30 minutes.

With the support of this analysis, the functional requirement of the glove has been achieved for the scope of this project. Out of the two versions created, the intermediate glove design was created purely to achieve the functional requirement. As seen in Figure 3-5, the glove is not suitable for industrial use as it is bulky and contains a number of hanging wires that can come undone during normal use.

5.2. Power Performance

Power management of the data collection glove is necessary for the selection of a suitable battery size that can be coupled with the glove to ensure minimum obstruction to the worker.

The data collection glove uses a ESP8266 processor (see section 3.2.1) and four MPU6050 IMU devices (see section 3.1.1). During normal operation the glove is continuously collecting data from the sensors, running Kalman filter and publishing the data to the classification processor. Due to the use of Wi-Fi continuously the device is not able to be placed in any of its sleep modes [66].

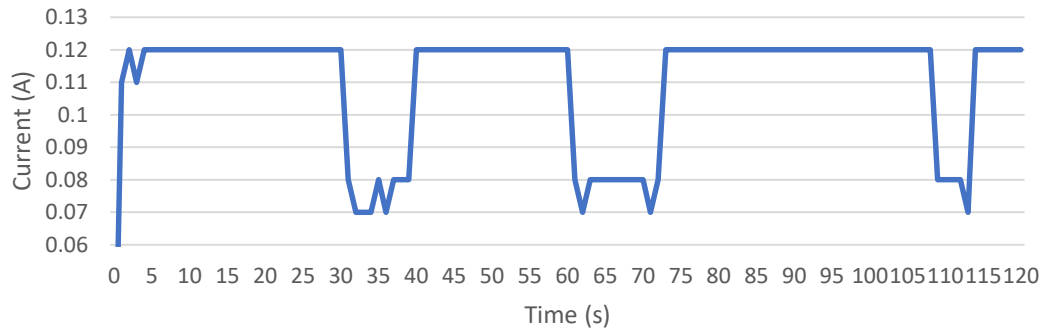


Figure 5-1 - Glove current consumption

By measuring the current draw of the glove during its operation while plugged in to a 5.02V source, the graph in Figure 5-1 can be observed for the first 120 seconds of operation. It can be seen that the device draws 0.12A at start-up and drops down to 0.07A-0.08A while waiting. During this state, the MQTT protocol is still communicating to maintain connection and all sensors and Kalman filters are running to maintain accurate values. At 40 seconds, the classification processor has requested data and the glove is actively transmitting data. At this stage, 0.12A current is consumed at 5.02V. As the glove does not heat up during long periods of use this current draw can be assumed to be steady during the full operation of the device.

Assuming the worst-case-scenario where a glove is to be used throughout the work-day for six hours and charged at the end of the day with 25% of the battery capacity remaining, a 14.5Wh battery will be needed. This requirement can be provided using two AA sized Li-ion batteries where each can provide up to 3000mAh at 3.6V [67]. Using a battery of this size will enable to glove to be used without any tethers, while batteries are attached to the wrist.

5.3. Machine Learning Performance

Compared to the findings in similar papers mentioned in section 2.2, the performance from both classical ML algorithms and convolutional neural networks have performed at par. The hypothesis for this thesis looked at 3 different aspects in section 1.2.3.

The first hypothesis that for an activity that is assumed to last at least 0.75 seconds a moving window of 0.75 seconds was chosen. This was backed up by the results from sections 4.2.3 and 4.4.2.

The second hypothesis that using sensors from the dominant hand can determine daily activities of the wearer such as walking was not verified. The results showed that classical machine learning algorithms were not able to distinguish between these activities, while CNNs were able to distinguish the ‘holding & walking’ activity. Although this result was seen in sections 4.2.1 and 4.4.4, this could be mainly due to the lack of appropriate pre-processing algorithms to differentiate these activities.

The final hypothesis that the skill level or the identity of the worker can be determined by the performance of the worker was not entirely verified. In the list of participants, participant 2, 3 and 9 were of relatively low skill, and this was seen in the classification of the machine learning algorithm. For example, the Figure 4-1 shows the machine learning algorithms have misclassified a larger number of data points for the same participants compared to the remaining participants.

However, to approach this hypothesis, a large number of skilled and unskilled members will need taken to form a participant pool of about 10,000 participants of varying skill level. A multi-stage classification will be required to first classify the action of the participants followed by the skill level of the same participant for the classified action. As a result of this, the final hypothesis is not achievable.

It can be seen that the performance of the CNN is lower than the performance most of the classical ML algorithms. The discrepancy is expected as all classical ML algorithms look at data as numerical values and fit them in to numerical classifications, whereas the CNN looks at the data points in terms of patterns and identifies groups of features instead of individual data.

The overall performance for the machine learning algorithms achieved F1-scores between 80% to 90%. The Table 5-2 outline a summary of all the tested F1 scores for the tested machine learning algorithms. However, it was seen that the training set was very often over fit by the machine learning algorithm. To overcome this issue a more training data is needed from a large number of participants of varying skill, body structure and technique in order to generalize the algorithm performance.

The performance of the machine learning algorithms can be further improved by combining similar activities. This approach of combining activities was not seen in other literature and may be due to similar activities not been considered in their scope.

- Combining “tightening” with “loosening” and “walking” with “turning” and “holding & walking” for classical ML an accuracy of 98.9% is obtained
- Combining “tightening” with “loosening” and “walking” with “turning” for convolutional neural networks an accuracy of 97.7% is obtained

Table 5-2 – Final Performance Values

	Parameters	F1 Score	F1 Score (Post Processed)
Null set			
Support vector machine (linear kernel)	Level 4 (pre-processing)	98.9%	100%
Classical Machine Learning			
Support vector machine (RBF kernel)	Level 4 (pre-processing)	89.25%	89.72%
Support vector machine (linear kernel)	Level 2 (pre-processing)	90.44%	91.28%
Linear discriminant analysis (SVD)	Level 2 (pre-processing)	89.24%	89.41%
Classical Machine Learning (Based on Computational Performance)			
Support vector machine (RBF kernel)	Level 4 (pre-processing)	89.25%	89.72%
Support vector machine (linear kernel)	Level 3 (pre-processing)	89.88%	90.19%
Linear discriminant analysis (SVD)	Level 4 (pre-processing)	88.4%	89.03%
Convolutional Neural Network			
1-CNN	3-D activity images	85.15%	86.2%

6. CONCLUSION

This thesis presented the use of a smart glove to perform activity recognition in industrial environments. A number of hypothesis was evaluated through the experiments carried out. The most suitable activity window size was selected between 0.5 – 0.75 seconds based on comparison data. The thesis also showed promise that measuring the hand movements, extrapolations can be made to certain full-body tasks. However, due to the lack of data, the possibility of determining the skill of the worker was not verified by this thesis.

The thesis also introduced a novel approach to post processing classified data using a categorical low pass filter which showed improvements in the classification up to 2.5%. The thesis also showed that classical machine learning algorithms that rely on linear discrimination of output performs better than clustering and other classical classifications.

Through these, the activities for industry were classified with F1 score of 91.3% through Support vector machine classifier with linear kernel and 86.2% with a 2-layer convolutional neural network with pre-processed activity images. By combining together similar activities that persistently get misclassified, F1 score of 98.9% and 97.7% was obtained for SVM and CNN respectively. This implies that the smart glove classification algorithm is suitable to be deployed for industrial hand activity recognition in both combined and uncombined states.

As of this thesis the smart glove is limited by its design. As future improvements, the glove will be designed to industrial standards and tested on large datasets to be able to extract further information such as skill level and identity of the worker.

7. REFERENCES

- [1] A. R. Sarkar, G. Sanyal and S. Majumder, "Hand Gesture Recognition: A Survey," *International Journal of Computer Applications*, vol. 71, no. 15, pp. 26-37, 2013.
- [2] Statista Research Department, "Fitness & activity tracker - Statistics & Facts," 22 05 2019. [Online]. Available: <https://www.statista.com/topics/4393/fitness-and-activity-tracker/>. [Accessed 18 06 2019].
- [3] Statista Research Department, "Wearable device sales revenue worldwide from 2016 to 2022 (in billion U.S. dollars)," 13 02 2019. [Online]. Available: <https://www.statista.com/statistics/610447/wearable-device-revenue-worldwide/>. [Accessed 18 06 2019].
- [4] Lean Manufacturing Japan, "Lean Manufacturing," [Online]. Available: <http://www.lean-manufacturing-japan.com/>. [Accessed 19 06 2019].
- [5] Shmula, "What Is A Standardized Work Combination Sheet?," 17 07 2017. [Online]. Available: <https://www.shmula.com/standardized-work-combination-sheet/23675/>. [Accessed 19 06 2019].
- [6] Y.-C. Huang, C.-W. Yi, W.-C. Ping, H.-C. Lin and C.-Y. Huang, "A study on multiple wearable sensors for activity recognition," *IEEE Conference on Dependable and Secure Computing*, pp. 449-452, 2017.
- [7] B. Bruno, F. Mastrogiovanni and A. Sgorbissa, "A Public Domain Dataset for ADL Recognition Using Wrist-placed Accelerometers," *IEEE International Symposium on Robot and Human Interactive Communication*, vol. 23, pp. 738-743, 2014.

- [8] G. Luzhnica, J. Simon, E. Lex and V. Pammer, "A Sliding Window Approach to Natural Hand Gesture Recognition using a Custom Data Glove," *IEEE Symposium on 3D User Interfaces (3DUI)*, 2016.
- [9] K. Liu, C. Chen, R. Jafari and N. Kehtarnavaz, "Fusion of Inertial and Depth Sensor Data for Robust Hand Gesture Recognition," *IEEE Sensors Journal*, vol. 14, no. 6, pp. 1898-1903, 2014.
- [10] W. Tao, Z.-H. Lai, M. C. Leu and Z. Yin, "Worker Activity Recognition in Smart Manufacturing Using IMU and sEMG Signals with Convolutional Neural Networks," *Procedia Manufacturing*, vol. 26, p. 1159–1166, 2018.
- [11] A. Malaise, P. Maurice, F. Colas, F. Charpillet and S. Ivaldi, "Activity Recognition With Multiple Wearable Sensors for Industrial Applications".
- [12] F. Attal, S. Mohammed, M. Dedabrishvili , F. Chamroukhi , L. Oukhellou and Y. Amirat, "Physical Human Activity Recognition Using Wearable Sensors," *MDPI Sensors*, vol. 15, no. 12, pp. 31314-31338, 2015.
- [13] Apple Inc, "iOS - Health," Apple Inc, [Online]. Available: <https://www.apple.com/lae/ios/health/>. [Accessed 18 06 2019].
- [14] Google, "Google Fit," Alphabet, [Online]. Available: <https://www.google.com/fit/>. [Accessed 18 06 2019].
- [15] Runkeeper, "Runkeeper," ASICS, [Online]. Available: <https://runkeeper.com/>. [Accessed 18 06 2019].
- [16] H. Zhou and H. Hu, "Human motion tracking for rehabilitation—A survey," *Biomedical Signal Processing and Control* , vol. 3, no. 1, pp. 1-18, 2008.

- [17] Qualisys, "Underwater Human Motion," [Online]. Available: <https://www.qualisys.com/applications/human-biomechanics/underwater-human-motion/>. [Accessed 18 06 2019].
- [18] Moov, "Moov," [Online]. Available: <https://welcome.moov.cc/>. [Accessed 19 06 2019].
- [19] Nexus, "Train with Nexus," [Online]. Available: <https://www.trainwithnexus.com/>. [Accessed 19 06 2019].
- [20] Atlas, "Atlas Wearable," [Online]. Available: <https://atlaswearables.com/>. [Accessed 19 06 2019].
- [21] Athos, "Live Athos," [Online]. Available: <https://www.liveathos.com/>. [Accessed 19 06 2019].
- [22] Gest, "Gest," [Online]. Available: <https://gest.co/>. [Accessed 20 06 2019].
- [23] Microsoft, "Kinect for Windows," Microsoft, [Online]. Available: <https://developer.microsoft.com/en-us/windows/kinect>. [Accessed 25 06 2019].
- [24] J. Mi, Y. Sun, Y. Wang, Z. Deng, L. Li, J. Zang and G. Xie, "Gesture Recognition based Teleoperation Framework of Robotic Fish," *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 137-142, 2016.
- [25] SenseGlove B.V., "Sense Glove," [Online]. Available: <https://www.senseglove.com/>. [Accessed 20 06 2019].
- [26] Xsens, "Xsens," Xsens, [Online]. Available: <https://www.xsens.com/>. [Accessed 30 03 2020].

- [27] CyberGlove Systems Inc., "CyberGlove," [Online]. Available: <http://www.cyberglovesystems.com/>. [Accessed 20 06 2019].
- [28] L. Labios, "Low-Cost Smart Glove Translates American Sign Language Alphabet and Controls Virtual Objects," UC San Diego, 12 July 2017. [Online]. Available: https://ucsdnews.ucsd.edu/pressrelease/low_cost_smart_glove_translates_american_sign_language_alphabet_and_control. [Accessed 10 July 2019].
- [29] A. M. Mohd Ali, M. Y. Ismail and A. A. Abdul Jamil, "Development of Artificial Hand Gripper for Rehabilitation Process," 2011.
- [30] M. Cornacchia, K. Ozcan, Y. Zheng and S. Velipasalar, "A Survey on Activity Detection and Classification Using Wearable Sensors," *IEEE Sensors Journal*, vol. 17, no. 2, pp. 386-403, 2017.
- [31] T. Stiefmeier, D. Roggen, G. Ogris, P. Lukowicz and G. Tröster, "Wearable Activity Tracking in Car Manufacturing," *Activity Based Computing*, pp. 42-50, 2008.
- [32] B. Hartmann, "Human Worker Activity Recognition in Industrial Environments," Scientific Publishing, Germany, 2011.
- [33] M. E. Benalcàzar, J. A. Zea, C. Motoche, A. G. Jaramillo, C. E. Anchundia, P. Zambrano, M. Segura, F. B. Palacios and M. Perez, "Real-Time Hand Gesture Recognition Using the Myo Armband and Muscle Activity Detection," *IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, 2017.
- [34] S. Jiang, B. Lv, W. Guo, C. Zhang, H. Wang, X. Sheng and P. B. Shull, "Feasibility of Wrist-Worn, Real-Time Hand, and Surface Gesture Recognition via sEMG and IMU Sensing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3376-3385, 2018.

- [35] H. Koskimaki, V. Huikari, P. Siirtola, P. Laurinen and J. Roning, "Activity recognition using a wrist-worn inertial measurement unit: A case study for industrial assembly lines," *2009 17th Mediterranean Conference on Control and Automation*, 2009.
- [36] C. Shen, B.-J. Ho and M. Srivastava, "MiLift: Efficient Smartwatch-Based Workout Tracking Using Automatic Segmentation," *IEEE Transactions on Mobile Computing*, vol. 17, no. 7, pp. 1609 - 1622, 2018.
- [37] P. Casale, O. Pujol and P. Radeva, "Human Activity Recognition from Accelerometer Data using a Wearable Device," Springer, Barcelona, 2011.
- [38] M. Ermed, J. Parkka, J. Mantyjarvi and I. Kohonen, "Detection of Daily Activities and Sports With Wearable Sensors in Controlled and Uncontrolled Conditions," *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, no. 1, pp. 20-26, 2008.
- [39] E. Thomaz, I. Essa and G. D. Abowd, "A Practical Approach for Recognizing Eating Moments with Wrist-Mounted Inertial Sensing," *UNICOMP*, 2015.
- [40] T. Maekawa, D. Nakai, K. Ohara and Y. Namioka, "Toward Practical Factory Activity Recognition: Unsupervised Understanding of Repetitive Assembly Work in a Factory," *UBICOMP*, 2016.
- [41] A. Jain and V. Kanhangad, "Human Activity Classification in Smartphones Using Accelerometer and Gyroscope Sensors," *IEEE Sensors Journal*, vol. 18, no. 3, pp. 1169-1177, 2018.
- [42] L. Dipietro, A. M. Sabatini and P. Dari, "A Survey of Glove-Based Systems and Their Applications," *IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews)*, vol. 38, no. 4, pp. 461 - 482, 2008.

- [43] InvenSense Inc, "MPU-6000 and MPU-6050 Product Specification Revision 3.4," 19 08 2013. [Online]. Available: https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf. [Accessed 12 May 2019].
- [44] WEMOS, "LOLIN D1 mini," WEMOS, 2019. [Online]. Available: https://docs.wemos.cc/en/latest/d1/d1_mini.html. [Accessed 15 03 2020].
- [45] NXP, "Tilt Sensing Using a Three-Axis Accelerometer," *Freescale Semiconductor, Application Note*, vol. AN3461, no. 6, 2013.
- [46] K. S. Lauszus, "Example-Sketch-for-IMU-including-Kalman-filter," 2012. [Online]. Available: <https://github.com/TKJElectronics/Example-Sketch-for-IMU-including-Kalman-filter>. [Accessed 15 03 2020].
- [47] Lauszus, "A practical approach to Kalman filter and how to implement it," 10 September 2012. [Online]. Available: <http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/>. [Accessed 17 May 2019].
- [48] Dejan, "Arduino and MPU6050 Accelerometer and Gyroscope Tutorial," How to Mechatronics, [Online]. Available: <https://howtomechatronics.com/tutorials/arduino/arduino-and-mpu6050-accelerometer-and-gyroscope-tutorial/>. [Accessed 17 03 2020].
- [49] S. C. Mukhopadhyay, "Wearable Sensor for Human Activity Monitoring: A Review," *IEEE Sensors Journal*, vol. 15, no. 3, pp. 1321 - 1330, 2015.
- [50] MQTT.org, "MQTT," [Online]. Available: <https://mqtt.org/>. [Accessed 16 03 2020].

- [51] S. Cope, "Understanding the MQTT Protocol Packet Structure," 11 1 2020. [Online]. Available: <http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/>. [Accessed 22 3 2020].
- [52] Flespi Platform, "HTTP vs MQTT performance tests," Medium.com, 23 1 2018. [Online]. Available: <https://medium.com/@flespi/http-vs-mqtt-performance-tests-f9adde693b5f>. [Accessed 22 3 2020].
- [53] scikit learn, "sklearn.impute.SimpleImputer," [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html#sklearn.impute.SimpleImputer>. [Accessed 16 04 2020].
- [54] NumPy.org, "NumPy," [Online]. Available: <https://numpy.org/>. [Accessed 22 04 2020].
- [55] SciPy.org, "SciPy.org," [Online]. Available: <https://www.scipy.org/>. [Accessed 22 04 2020].
- [56] U. Maurer, A. Smailgic, D. P. Siewiorek and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions," *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, pp. 4-16, 2006.
- [57] O. D. Lara and M. A. Labrador, "A Survey on Human Activity Recognition using Wearable Sensors," *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, vol. 15, no. 3, pp. 1192-1208, 2013.
- [58] scikit learn, "sklearn.preprocessing.StandardScaler¶," [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>. [Accessed 10 05 2020].

- [59] T. Afonja, "Accuracy Paradox," Medium, 8 December 2017. [Online]. Available: <https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b>. [Accessed 27 June 2020].
- [60] K. M. Chathuramali and R. Rodrigo, "Faster Human Activity Recognition with SVM," *The International Conference on Advances in ICT for Emerging Regions*, pp. 197-203, 2012.
- [61] I. T. Ian T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," The Royal Society, 13 April 2016. [Online]. Available: <https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202>. [Accessed 10 05 2020].
- [62] J. Brownlee, "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning," 03 07 2017. [Online]. Available: <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>. [Accessed 12 07 2020].
- [63] R. Gylberth, "An Introduction to AdaGrad," Medium, 03 05 2018. [Online]. Available: <https://medium.com/konvergen/an-introduction-to-adagrad-f130ae871827>. [Accessed 12 07 2020].
- [64] S. Ruder, "An overview of gradient descent optimization algorithms," 19 Jan 2016. [Online]. Available: <https://ruder.io/optimizing-gradient-descent/index.html#adagrad>. [Accessed 08 Jul 2020].
- [65] Kaggle, "Dropout Regularization in Deep Learning Models With Keras," Kaggle, 21 July 2019. [Online]. Available: <https://www.kaggle.com/pavansanagapati/what-is-dropout-regularization-find-out>. [Accessed 12 07 2020].

- [66] Espressif IOT Team, "ESP8266 Low Power Solutions," 4 2016. [Online]. Available: https://www.espressif.com/sites/default/files/9b-esp8266-low_power_solutions_en_0.pdf. [Accessed 10 12 2020].
- [67] NKON, "Sony / Murata US18650VTC6 3000mAh - 30A," [Online]. Available: <https://eu.nkon.nl/sony-us18650-vtc6.html>. [Accessed 10 12 2020].

8. APPENDICES

Appendix 1: Prototype 1

This section summarizes the work done to design and build prototype 1. The section will mostly focus on the sections where this prototype is different from the prototype designed for this project, explained in this thesis.

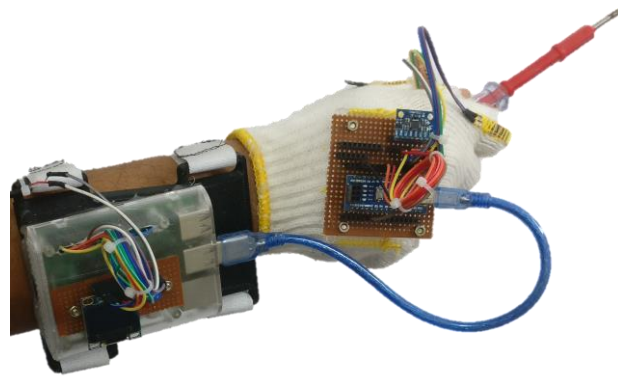


Figure 8-1 - Prototype 1 Holding Screwdriver [Original in color]

Table 8-1 - Prototype 1 Design Specification Summary

Design Aspect	Value
Activities Selected	4: Pointing, Wiping, Tightening, Picking
Sensors Selected	MPU6050 for back hand and flex sensors for fingers
Measurements	Hand Movement – Roll and Pitch Thumb Bend, Index Bend Thumb abduction relative to index finger
Glove	Cotton Dotted Glove
Glove Processor	Arduino Nano
Classification Processor	Raspberry Pi Model B Rev 3
IMU Filtering	Kalman Filter on glove
Classification	Classical Machine Learning, Neural Network

A video of the operation can be found here: <https://youtu.be/fiLtNbVrMOI>

The model trained for Prototype 1 was entirely based on data that was not pre-processed. Raw accelerometer data and filtered roll and pitch data was used. The model trained with the activities mentioned in 1.3 and satisfactory results were obtained where the device was able to classify the data approximately 90% of the time.

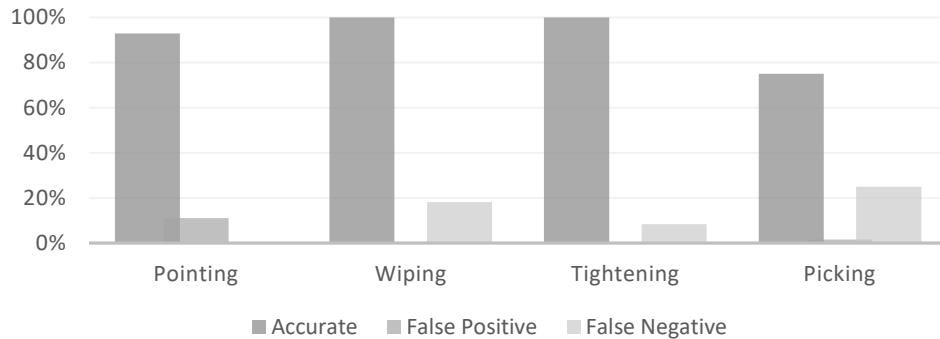


Figure 8-2 - Testing Data Accuracy for SVM classification with RBF kernel

When the RNN was tested the algorithm converged with accuracy of 97.4% and a loss of 16.18%. The same as the SVM was done with the Recurrent Neural Network that was trained and Figure 8-3 was observed. It can be seen that both the SVM and RNN performed very similar with the RNN performing slightly better than the SVM classifier.

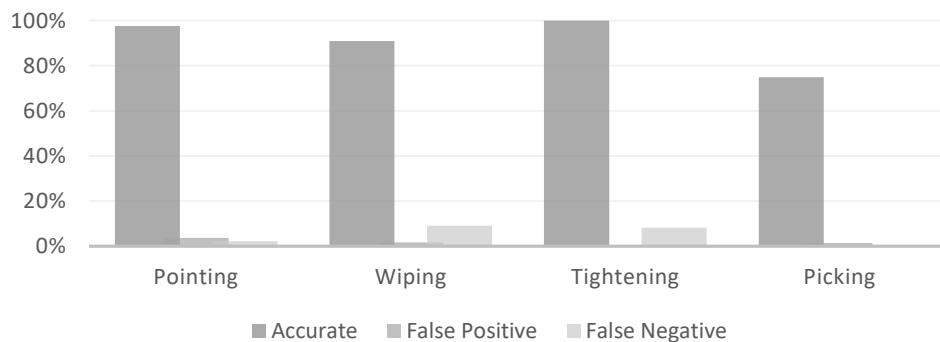


Figure 8-3 - Testing Data Accuracy for RNN classification

Figure 8-4 to Figure 8-7 show the performance of other classification algorithms that were also tested out using the same dataset. This will help point out any other classification algorithms that will provide much better results than the SVM or RNN classification algorithms.

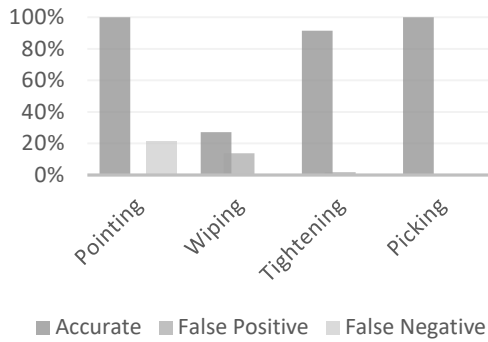


Figure 8-4 - Testing Data Accuracy for K-Nearest Neighbor Classifier

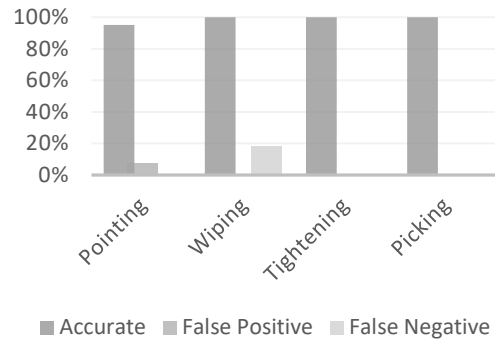


Figure 8-5 - Testing Data Accuracy for Random Forest Classifier

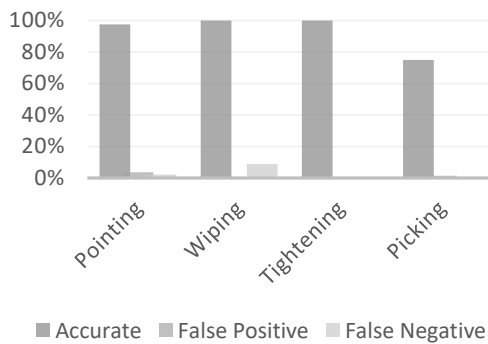


Figure 8-6 - Testing Data Accuracy for Decision Tree Classifier

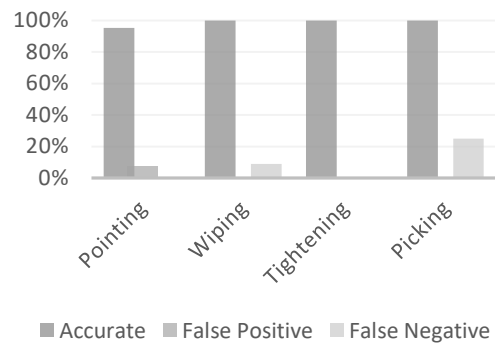
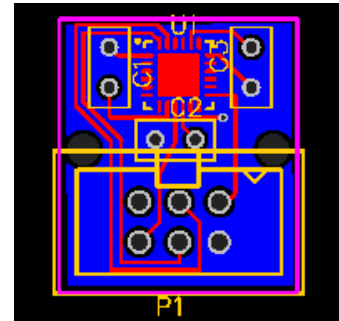
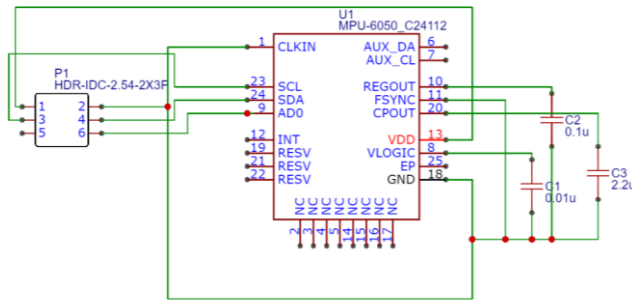


Figure 8-7 - Testing Data Accuracy for Gaussian Naïve Bayes Classifier

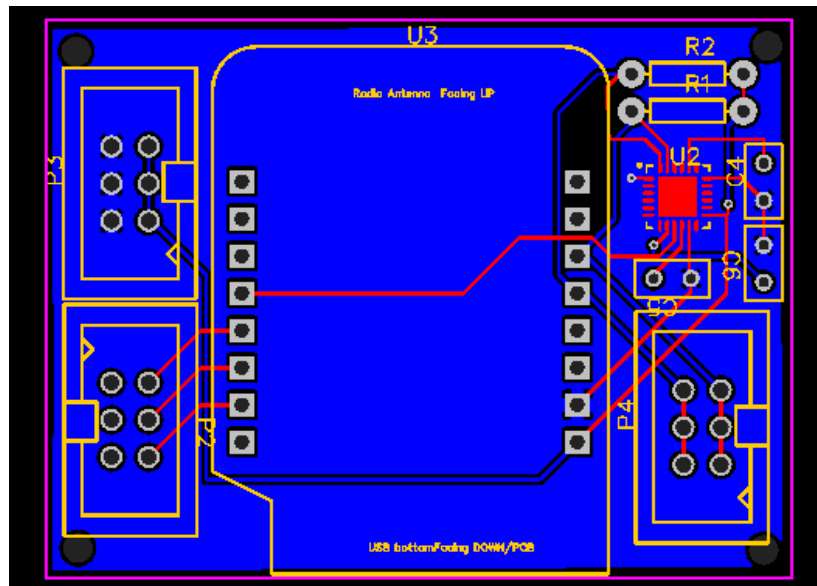
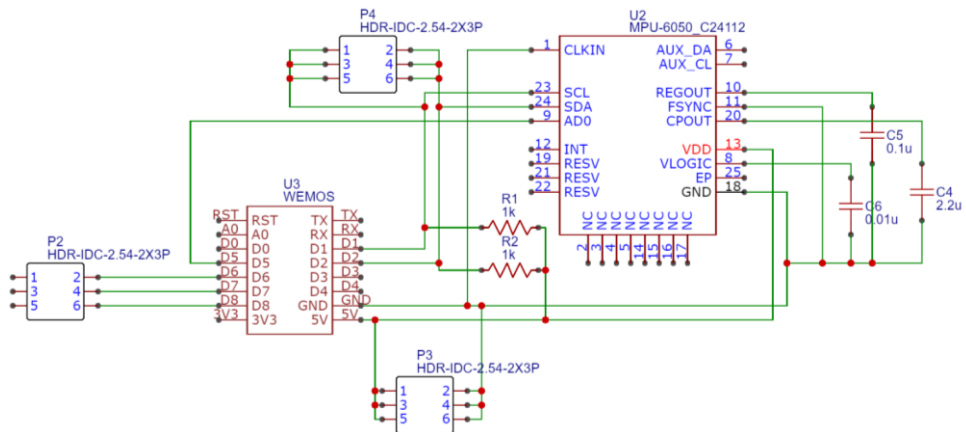
It can be seen that from Figure 8-4 to Figure 8-7, the Random Forrest classifier has been the most successful in classifying the data successfully for the given data set. Although more data is required to prove that Random Forest Classifier can outperform both SVM and the Recurrent Neural Networks, it is the best choice for the current data. This finding is backed up by the paper [30] where it states that Random Forests provide better classification than SVM and other techniques for human activity recognition. If the Random Forrest classifier does perform better the activity recognition device can be made very effective as ML algorithms have a faster training and prediction time compared to RNN's

Appendix 2: Glove Schematic & PCB Design

Finger Board



Back Hand Board



Appendix 3: Stage 2 Pre-processing features

Feature list

Table 8-2 – Selection of features for stage 2 pre-processing

Ind.	Field	Size	Level 1	Level 2	Level 3	Level 4
0	accX, accY, accZ	90	✓			
90	roll1, pitch1, yaw1	90	✓			
180	roll2, pitch2, yaw2	90	✓			
270	roll3, pitch3, yaw3	90	✓			
360	roll4, pitch4, yaw4	90	✓			
450	MBend, IBend, TBend	90	✓			
540	TRot	30	✓			
570	MYaw, IYaw, TYaw	90	✓			
660	Filtered accX, accY, accZ	90	✓			
750	Magnitude of Acc	30	✓			
780	Magnitude of Filtered Acc	30	✓			
810	dRoll1, dPitch1, dYaw1	90	✓			
900	dRoll2, dPitch2, dYaw2	90	✓			
990	dRoll3, dPitch3, dYaw3	90	✓			
1080	dRoll4, dPitch4, dYaw4	90	✓			
1170	dMBend, dIBend, dTBend	90	✓			
1260	dMYaw, dIYaw, dTYaw	90	✓			
1350	dTRot	30	✓			
1380	Histogram of accX, accY, accZ	9	✓	✓		

1389	Histogram of Filtered accX, accY, accZ	9	✓	✓	✓	
1398	HOG of accX, accY, accZ	9	✓	✓	✓	
1407	HOB of Roll1, Pitch1	24	✓	✓	✓	
1431	HOB of Roll2, Pitch2	24	✓	✓	✓	
1455	HOB of Roll3, Pitch3	24	✓	✓	✓	
1479	HOB of Roll4, Pitch4	24	✓	✓	✓	
1503	Histogram of dRoll1, dPitch1, dYaw1	9	✓	✓	✓	
1512	Histogram dRoll2, dPitch2, dYaw2	9	✓	✓	✓	
1521	Histogram dRoll3, dPitch3, dYaw3	9	✓	✓	✓	
1530	Histogram dRoll4, dPitch4, dYaw4	9	✓	✓	✓	
1539	Histogram magnitude of Acc	9	✓	✓	✓	
1548	HOG of magnitude of Acc	3	✓	✓	✓	
1551	Linear Speed of accX, accY, accZ	3	✓	✓	✓	
1554	Histogram of MBend, IBend, TBend	36	✓	✓	✓	
1590	Histogram of MYaw, IYaw, TYaw	9	✓	✓	✓	
1599	HOB of TRot	12	✓	✓	✓	
1611	Histogram of dMBend, dIBend, dTBend	9	✓	✓	✓	
1620	Statistics of accX, accY, accZ	42	✓	✓		

1662	Statistics of Filtered accX, accY, accZ	42	✓	✓	✓	✓
1704	Statistics of Roll1, Pitch1, Yaw1	42	✓	✓	✓	✓
1746	Statistics of Roll2, Pitch2, Yaw2	42	✓	✓	✓	✓
1788	Statistics of Roll3, Pitch3, Yaw3	42	✓	✓	✓	✓
1830	Statistics of Roll4, Pitch4, Yaw4	42	✓	✓	✓	✓
1872	Statistics of magnitude of Acc	14	✓	✓	✓	✓
1886	Statistics of MBend, IBend, TBend	42	✓	✓	✓	✓
1928	Statistics of MYaw, IYaw, TYaw	42	✓	✓	✓	✓
1970	Stats TRot	14	✓	✓	✓	✓
1984	FD RPY1, RPY2, RPY3, RPY4	60	✓	✓	✓	
2044	FD dRPY1, dRPY2, dRPY3, dRPY4	60	✓	✓	✓	
2104	FD Bends	15	✓	✓	✓	
2119	FD Filtered Acc	15	✓	✓	✓	
2134	SMA Filtered acc	1	✓	✓	✓	
2135	FD acc	15	✓			
2150	SMA acc	1	✓			

Statistical Elements

- Median – Statistical median
- Mean – Statistical mean
- Variance – Statistical variance
- Standard Deviation – Statistical standard deviation
- Minimum & Maximum – Statistical minimum and maximum number

- RMS – Statistical Root mean square
- Skew – Statistical data skew
- Kurtosis – Statistical kurtosis
- Median Absolute Deviation – Statistical median absolute deviation
- Mean Absolute Deviation – Statistical Mean Absolute Deviation
- Zero Crossing – Number of times the data in window cross zero point
- Slope Gradient Changes – Number of times the data in window change sign
- Waveform Length – Sum of all the difference in adjacent values in the window

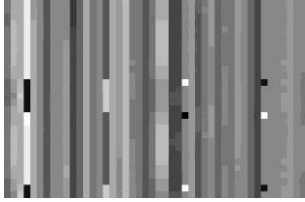


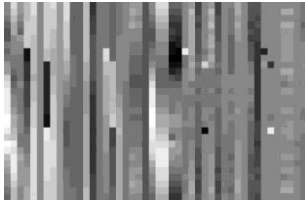
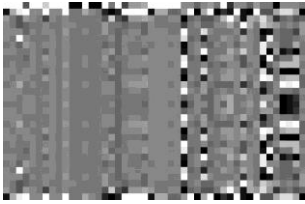
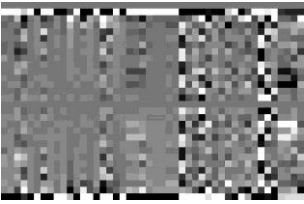
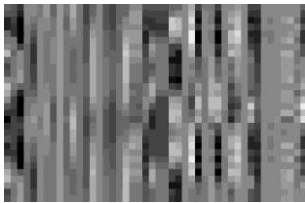

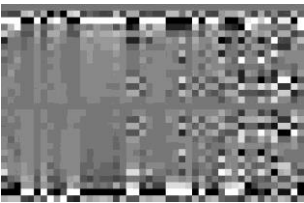
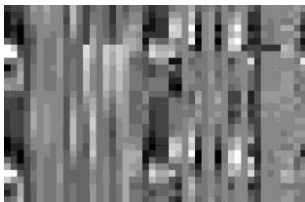

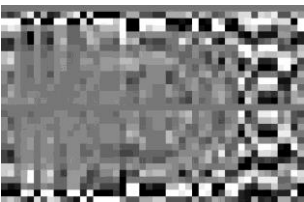
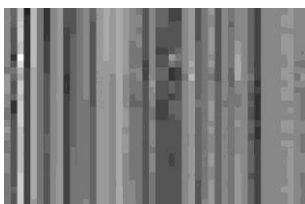

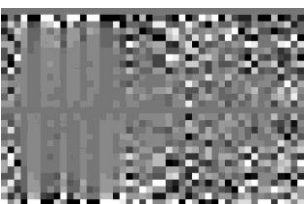



```

def fnc_stats (window):
    val.append(np.median(window))
    mean = np.mean(window)
    val.append(mean)
    val.append(np.var(window))
    val.append(np.std(window))
    val.append(np.amin(window))
    val.append(np.amax(window))
    val.append(np.mean([v**2 for v in window]))    #RMS
    val.append(stats.skew(window))
    val.append(stats.kurtosis(window))
    val.append(stats.median_absolute_deviation(window))
    for i in range(len(window)):
        mad += round(np.absolute(window[i] - mean),2)
    val.append(mad/len(window))
    for i in range(1, len(window)):
        if (window[i] * window[i-1]) < 0:
            zeroCrossing += 1
        if ((window[i]-window[i-1])*(window[i-1]-window[i])) < 0:
            slopeChanges += 1
        waveLength = waveLength + window[i] - window[i-1]
    val.append(zeroCrossing)
    val.append(slopeChanges)
    val.append(waveLength)
    return val

```

Appendix 4: Stage 2 Pre-processing Images

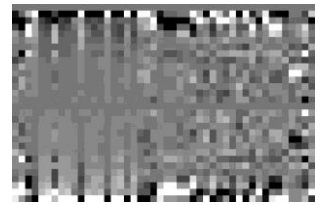
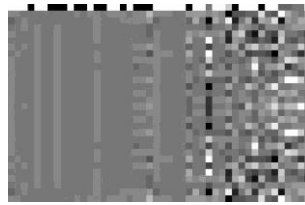
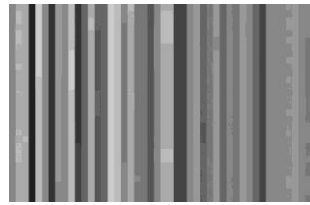
Table 8-3 - Examples of Convolutional Neural Net Images for each activity

	Raw	FFT Magnitude	FFT Phase
Pointing			
Wiping			
Tightening			
Loosening			
Pick			
Hold			

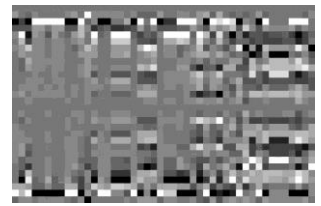
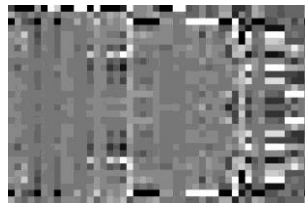
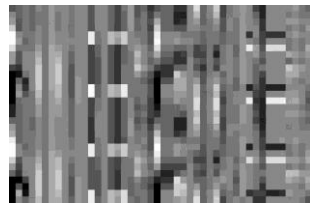
Pull



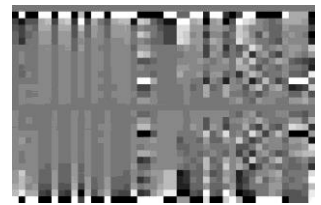
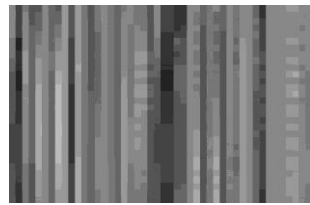
Push



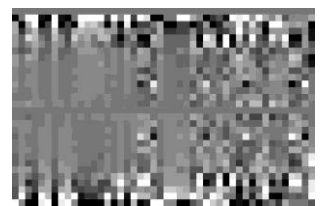
Hammer



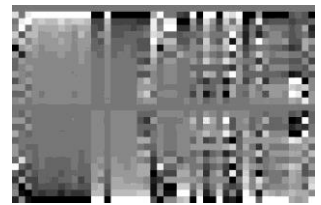
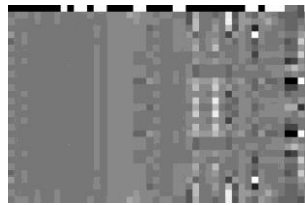
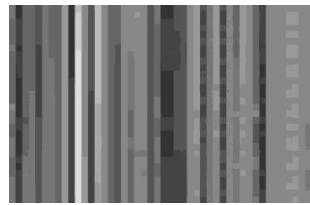
Walk



Hold & Walk



Turn



Appendix 5: Add. CNN Performance Charts

Sigmoid Activation

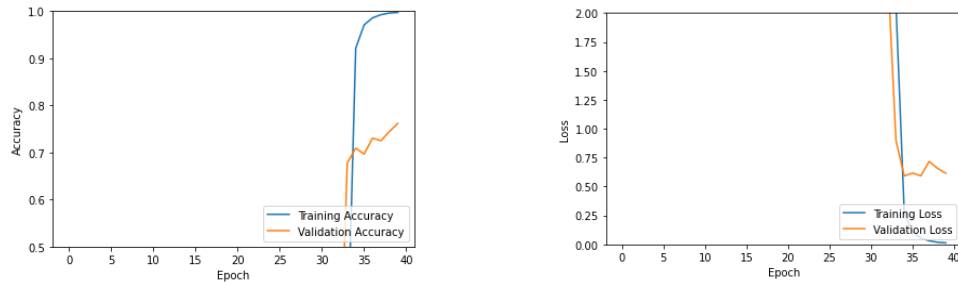


Figure 8-8 – 1-CNN (Sigmoid activation on all layers)

Early Stopping

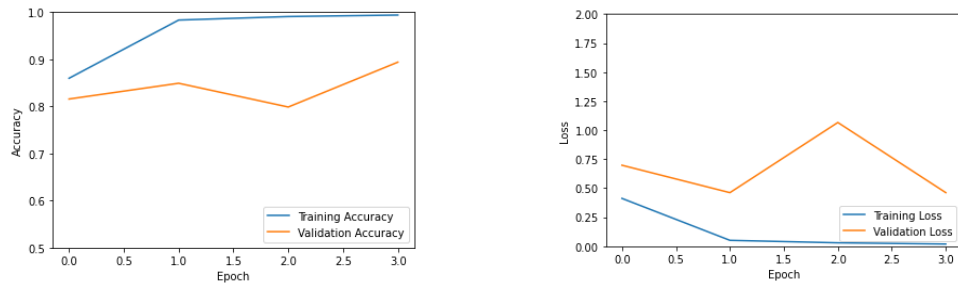


Figure 8-9 – 2-CNN (5 FC Layers – Early Stopping)

Layer Manipulation

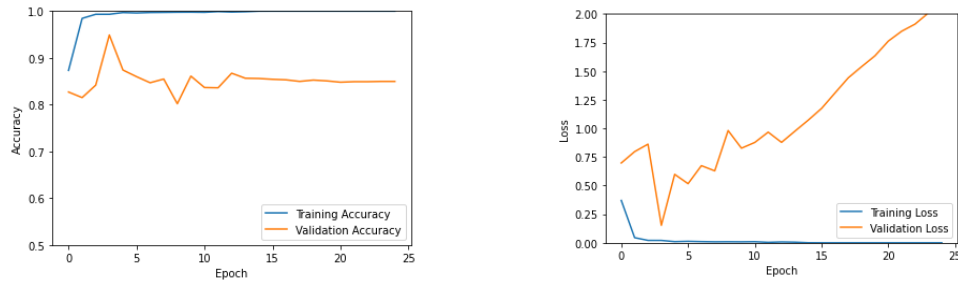


Figure 8-10 – 2-CNN (5 FC Layers)

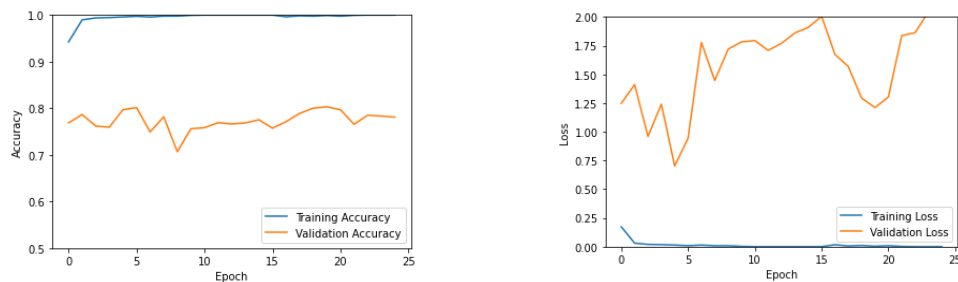


Figure 8-11 – 4-CNN (no Max-pool layer)

Dropout Regularization

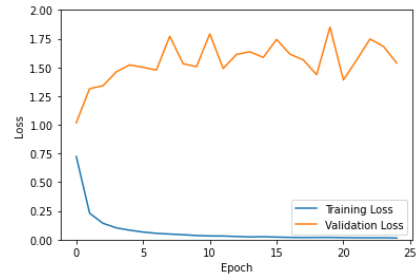
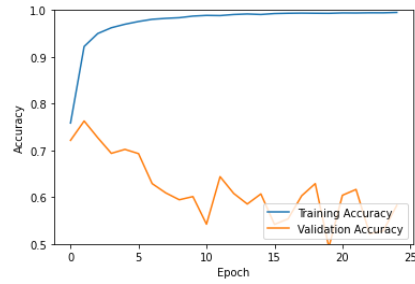


Figure 8-12 – 1-CNN (DRR = 0.4)

Reducing Dropout Regularization

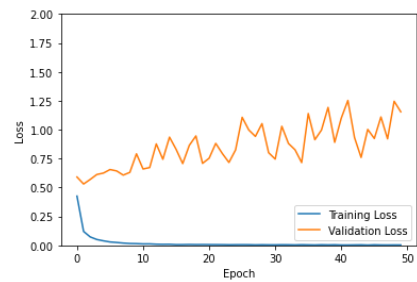
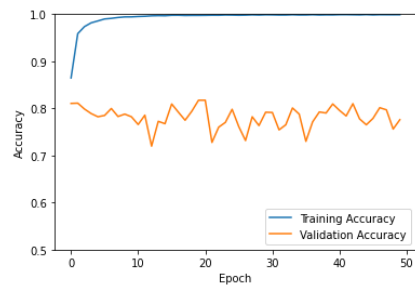


Figure 8-13 – 1-CNN (Input DRR = 0.2)

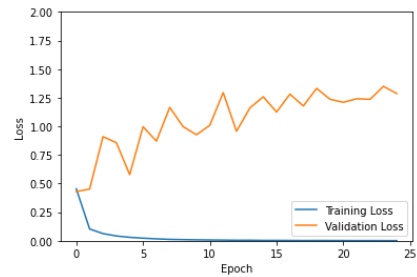
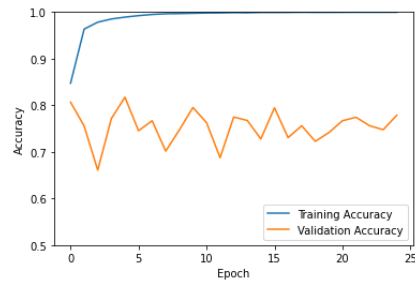


Figure 8-14 – 2-CNN (Input & Hidden DRR = 0.05)

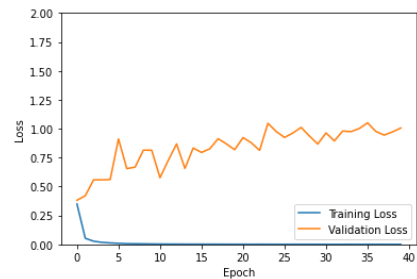
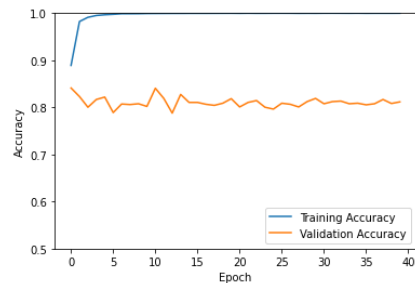


Figure 8-15 – 2-CNN (Input DRR = 0.01)

Adagrad with Dropout Regularization

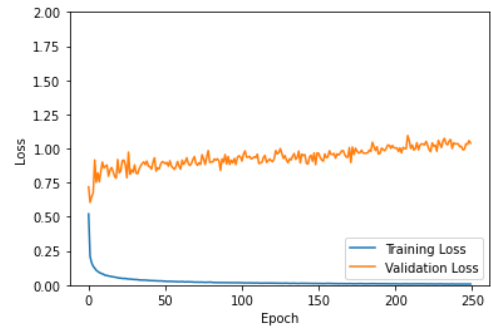
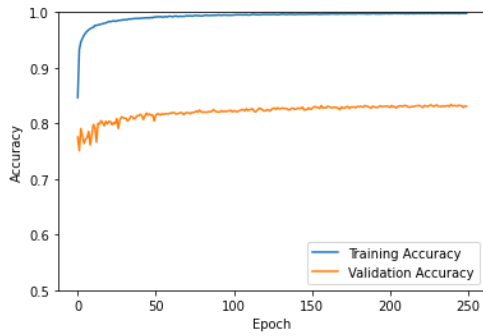


Figure 8-16 – 1-CNN (Adagrad optimizer and DRR = 0.2)

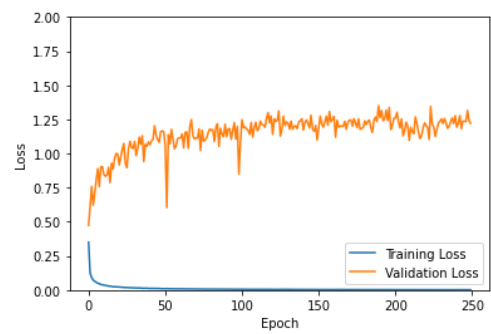
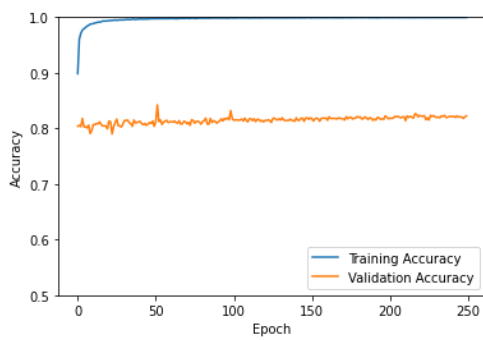


Figure 8-17 – 1-CNN performance (Adagrad optimizer & DRR = 0.1)

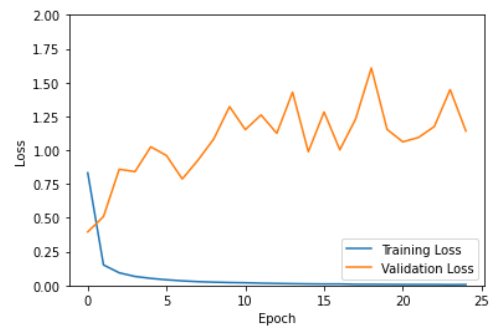
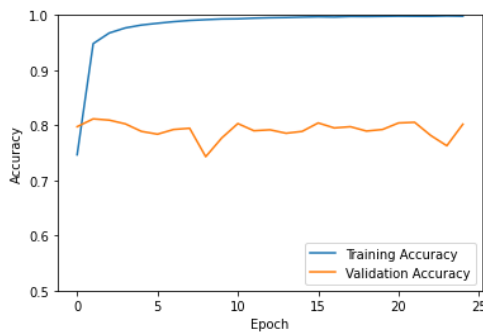


Figure 8-18 – 1-CNN (Adagrad optimizer tuning, DRR = 0.05 & 6 FC Layers)

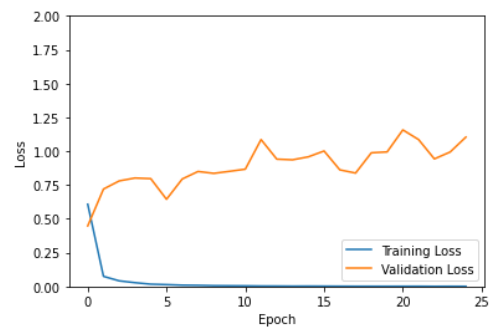
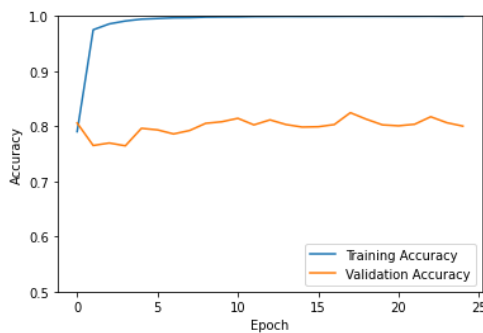


Figure 8-19 – 2-CNN (Adagrad optimizer tuning, DRR = 0.05 & 6 FC Layers)

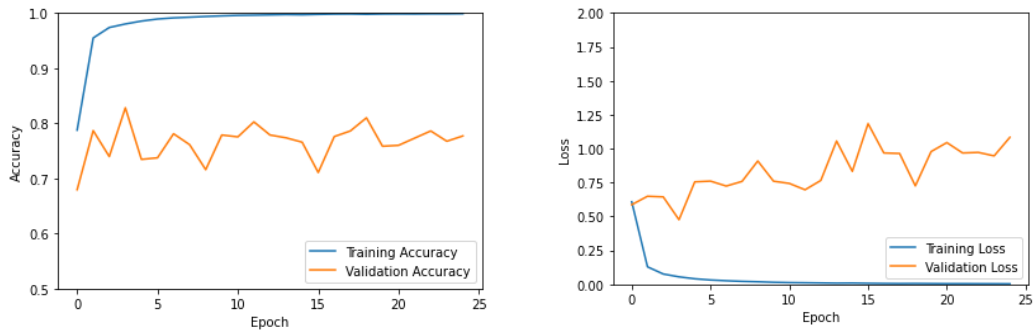


Figure 8-20 – 2-CNN (Adagrad optimizer tuning, DRR = 0.05 & 6 FC Layers)

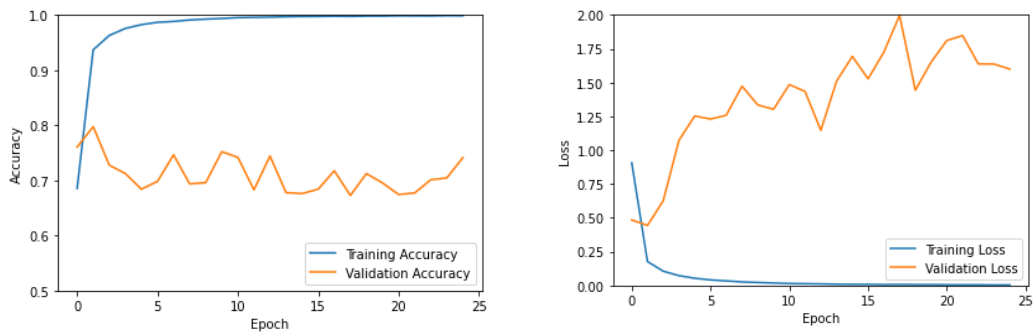


Figure 8-21 – 3-CNN (Adagrad optimizer tuning, DRR = 0.05 & 6 FC Layers)

Kernel Manipulations

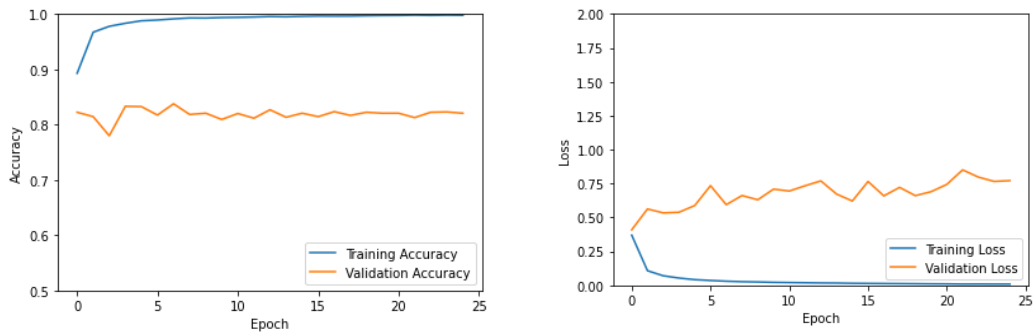


Figure 8-22 – 1-CNN (5x5 kernel, Adagrad optimizer tuning & DRR = 0.05)

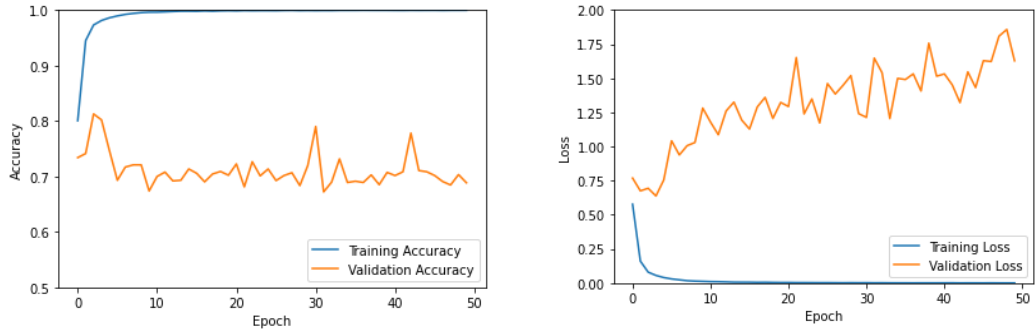


Figure 8-23 – 3-CNN (3x3 kernel, 1 Max pool layer & 2 FC CNN - Adagrad optimizer & DRR = 0.2)

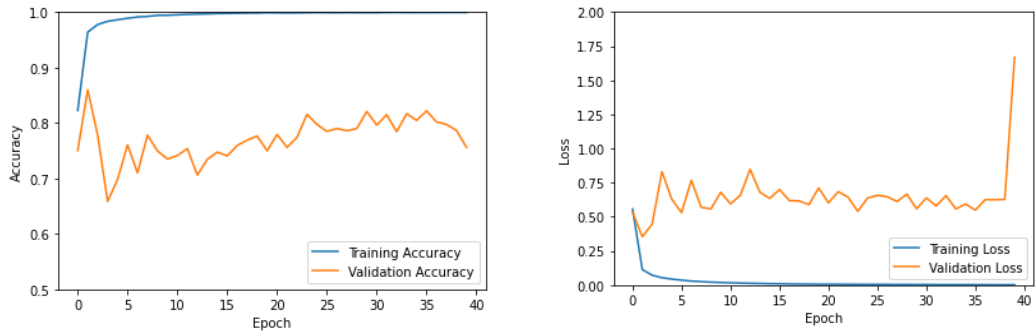


Figure 8-24 – 2-CNN (5x5 first layer kernel - Adagrad optimizer & DRR = 0.01)

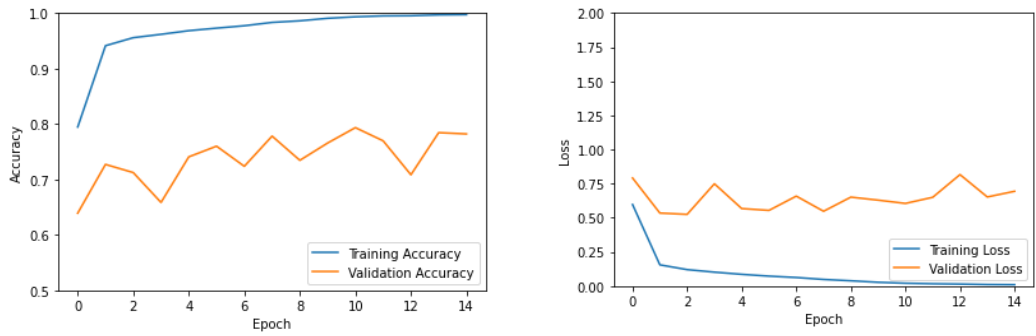


Figure 8-25 – 3-CNN (5x5 first layer kernel - Adagrad optimizer & DRR = 0.1)

Appendix 6: Project Cost

As the total cost of the project is borne by the project owner, the project was designed for the lowest cost. In order to keep the costs low

- Parts were sourced from Chinese online vendors through eBay and AliExpress
- Personal components that were already available were used.
- Manufacturing was done through local companies with low cost links in China.
- Assembling was done personally with the help of family members.

This resulted in the costs for each section of the project.

Approximate cost of Prototype 1	LKR	27,000.00
Approximate cost of Prototype 1 development	LKR	35,000.00
Approximate cost of Prototype 2: Glove only	LKR	3,765.62 ++ ⁶
Approximate cost of Prototype 2: Processing only	LKR	13,314.25
Approximate cost of Prototype 2: Average per user (5 users)	LKR	6,408.47 ++
Approximate cost of Prototype 2: Total Cost (5 users)	LKR	32,042.35 ++
Approximate cost of the complete project (to date)	LKR	46,844.05

The cost of a final prototype glove of LKR 7,500.00 (approximated adjustment cost added to LKR 6,408.47) is an ideal value for an industrial application. The justification for this is that in industry the measuring of time human movement is not a direct benefit to the bottom-line savings of the factory. Instead the device is used as an enabler to identify possible areas of improvement.

⁶ Due to the limitations in 2020, certain aspects of the project were not completed in full, and certain alternative approaches were taken. As a result, some of the costs are mentioned with a ‘++’ symbol to indicate the increase in value expected when the incomplete areas are completed in full. It should also be noted that tools used and man-day costs are not added to the complete project cost.