Uppsala University

Department of Information Technology

Degree Project in Embedded Systems

Gauntlet-X1: Smart Glove System for ASL translation using Hand Activity Recognition

Authors:

Asif Mohamed, Vishnu Ullas, Paul Sujeet

Supervisor:

Lars Oestreicher

June 16, 2020

Abstract

The most common forms of Human Computer Interaction (HCI) devices these days like the keyboard, mouse and touch interfaces, are limited to working on a two-dimensional (2-D) surface, and thus don't provide complete freedom of accessibility using our hands. With the vast number of gestures a hand can perform, including the different combinations of motion of fingers, wrist and elbow, we can make accessibility and interaction with the digital environment much more simplified, without restrictions to the physical surface. Fortunately, this is possible due to the advancements of Microelectromechanical systems (MEMS) manufacturing of sensors, reducing the size of a sensor to the size of a fingernail.

In this thesis we document the design and development of Gauntlet-X1, a smart glove system comprising of Inertial Measurement Units (IMU) sensors that recognizes hand activity/gestures using combinations of neural networks and deep learning techniques such as Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN). The system captures IMU data and interfaces with the host server. In order to demonstrate this prototype as a proof of concept, we integrate to Android mobile applications based on 3-D interactivity like the American Sign Language (ASL), Augmented Reality (AR)/Virtual Reality (VR) applications and can be extended to further the use of HCI technology.

Acknowledgements

This thesis would not have been possible without the generous support of our supervisor, reviewer, thesis coordinator, professors and staff of the Department of Information Technology, and mainly the coordination among our team.

We are truly grateful to Lars Oestreicher for his encouragement, advice, and wisdom throughout this project. We greatly appreciate him for agreeing to be our Supervisor and showing great interest and enthusiasm from day one. Lars has helped us break down our overwhelming idea to this present proof of concept and holding regular meetings to ensure we reach our milestones during the thesis period. Despite the various hurdles, he helped us break a major deadlock by providing a new scope towards the realization of this thesis.

We would also like to thank Alexander Medvedev for agreeing to be our Reviewer for this immense project since the period we took a course "Embedded Control Systems" under him which developed into this idea. We also thank him for guiding us towards the appropriate methodologies for this thesis.

We would like to thank Mr. Justin Pearson, our thesis coordinator, for his willingness to hearing out and approving our ambitious idea and getting us in touch with our supervisor.

A special thanks to Fredrik Olsson who kindly offered his support in the areas of Inertial Navigational Systems. He answered our novice questions and provided various techniques for implementing the algorithms required for our research.

We truly appreciate Lars Oestreicher and Alexander Medvedev for their time and efforts in reviewing this document. Also, Lars assisted us in editing remnants of structure, grammar, and typos.

Our families have been greatly supportive throughout this period. We cannot thank them enough for their understanding, patience and the heavy financial support for us. They are the true inspirations of our lives.

Last but not the least is the team members who have gathered to make this arduous thesis idea into reality. It has been an incredible journey since our first project together here in this Department. We couldn't have asked for a better team.

We would like to finish by saying our prayers and thanks to God who has opened many doors and opportunities for us when none existed.

Contents

Chapter 1

Introduction

With recent advancements in Human Computer Interaction (HCI) technology, there have been vast research on minimising the gap between humans and computers. Humans interact with technology in their daily routine and HCI researchers study how to provide optimum comfort and performance to the user by integrating their daily routines with computers and various other technologies. The application of touch gestures is, for example, currently a big trend on our smart phones and tablets. This interactive technology provides the user with a natural feel to the interface, making it much more feasible to use and access. However, the biggest delimitation of this technology is that it still requires a physical 2-D surface to operate. Moreover, a touch interface is also to a high degree constrained by the specifications of the 2-D plane. This is where the interactive technology developed during this thesis project comes into play where we propose to provide an ability to interact with the real 3-D environment. The user is not constrained by

the specifics of a 2-D plane, since the real world presents itself as a three-dimensional canvas in front of the user. The user is given much more freedom in movement and more ways of interaction. We provide this technology to interact via multiple Internet-of-Things (IoT) devices in order to resolve real world applications and issues.

There are already various peripheral devices that provide the above mentioned functionality. What makes our technology unique and different is that it not only focuses on one single application area but a variety of multiple applications that can be used just through a single device. This product is currently in the state of "proof of concept", and thus we focus mainly on two applications: 1) an American Sign Language (ASL) interpreter and 2) interaction through Augmented Reality(AR). These two applications are only chosen as demonstration cases for the prototype, but the general system can be executed with relative ease to other application areas as well.

This thesis documents the design, development and implementation of a prototype of a wearable glove that has embedded Inertial Measurement Units (IMU) [1]. The IMU's are used to track the movements of each of the four fingers and the thumb on a single hand, as well as of the hand itself. We integrated five MPU6050 IMU's for the fingers and one MPU9250 IMU on the backside of the palm. The system is employed using an Arduino micro

[1] An IMU is an electronic devices that measures and reports linear acceleration, rotational rate and heading reference provided by the inbuilt accelerometer, gyroscope and magnetometer respectively.

controller placed along with the MPU9250 on the back of the palm, which collects data from the IMU's placed on the different points of interest on the glove. Each time a gesture is made while wearing the glove, the data from the IMU's for that particular gesture is recorded using a software created through MATLAB. Using this approach, we recorded the IMU data for 10 primary gestures that we wanted to implement for the prototype. We will go more into detailed description of the hardware in section 3.1.

The recorded data is then fed into a Machine Learning (ML) algorithm designed using PyCharm and Keras, with Tensorflow as back-end. A typical ML system is a set of algorithms and statistical models that computer systems use to perform specific tasks based on inference and pattern detection instead of pre-programmed, deterministic algorithms. The ML algorithm we developed is used to train the Gauntlet with various hand gestures from recorded data. The ML algorithm is part of a Neural Network, a framework of multiple ML algorithms that work together and process complex data units. Using this method, we trained the Gauntlet to recognize gestures with respect to the applications we mainly focused on. We have only focused on building two applications as this is a proof of concept for a prototype.

One of the main objectives of this thesis is to use this type of data processing unit to act as a general integrated peripheral device, exemplified as an implementation for two applications. The two applications we targeted are for American Sign Language (ASL) recognition, and interaction through Augmented Reality (AR). We decided to choose ASL as an example for the

demonstration of gesture recognition instead of the other available sign languages as it is more widely used internationally. With some more retraining, we will be able to use it for any available sign language, making the Gauntlet more internationally compatible and interesting. On the other hand, the reason we decided to target AR as our second application is so we can visualize the capabilities of the gesture training in an extent within a 3-D virtual world. It can be used for interacting with virtual objects and for learning purposes. Through these two applications, we show the full capacity of what the Gauntlet can offer.

1.1 Cognitive interaction

One of the biggest problems faced by hearing disabled people is to convey their message to a person who has no knowledge of sign language. Our sign language translation application may be a great aid in this case. It will support communication with others more efficiently. The product is easy to use and will easily integrate with their day to day habits. It can also be used to acquaint sign language efficiently. We want to implement a simple and user friendly HCI technology so that anyone can be able to use it to its full potential.

Another field of interest our product can target is the gaming industry. The Gauntlet can be used for various applications as the gaming field is vast and ever growing and our product has a potential to evolve over time. It can be easily integrated as a peripheral controller and can be used for interacting

with both AR and VR applications.

Although these two applications are very good examples of the applicability of our technology, the system is developed with the general aim to make it possible to catch the hand and finger positions in combination with the gestures. Once this is done, it is possible to transfer the result into any application where it could be suitable to capture this kind of movements.

1.2 Hypothesis and Research Questions

We started from the hypothesis that it would be possible to create an interface that would make it easier for a person with hearing deficiencies to use sign language to communicate with people that have no knowledge of sign language.

A person suffering from hearing deficiency communicates via sign language. Although, most of the time the other person has no knowledge of sign language which makes it harder to communicate. From this hypothesis, we decided to create an interface that allows to convert sign language into text or audio. This interface makes it easier for someone who is hearing impaired to communicate and also to provide a surface to use this interface as a means to learn sign language.

Our main Research Questions are:

1. Is an Arduino based solution powerful enough?

2. What ML algorithms are best suited?
3. How precise is it possible to design such a system?
4. How many hand activities can be recognized?
5. How many more applications can be integrated into the system (IOT)?
6. How to make the product less bulky to provide better freedom to the user?
7. How does training different aspects of the full data separately using individual neural networks affect the accuracy of the output?

1.3 Our Contribution

This thesis has three authors, and in this project we have contributed equally to the final result, but during some stages we have had separate areas of responsibility. In this section we will describe both the individual and the common, general efforts in the project and while writing the thesis.

1.3.1 Our Individual Contributions

Asif's contributions to this thesis project are as follows:

- Developed all the necessary MATLAB scripts to gather and record data from the Gauntlet and implementations of various filters for preprocessing the collected data.

- Developed the Python scripts for data gathering and analysis of the various NN models.
- Analysis of one of the NN models and picking the right configuration with respect to all the features of the Gauntlet's data.
- Built the optimized C++ code for the gauntlet in Arduino IDE.
- Implemented the server side of Gauntlet with serial connectivity and designed the Internet-of-Things (IoT) main hub for the Gauntlet.

Paul's contributions to this thesis project are as follows:

- Complete research and study on the relevant literature regarding the various areas within Machine Learning (ML) and Deep Learning that could benefit this paper.
- Analysis of two NN models and picking the right configuration with respect to all the features of the Gauntlet's data.
- Implementations of all the types of NN models with Keras and Tensorflow for the basis of our research paper.

Vishnu's contributions to this thesis project are as follows:

- Research on different types of AR and various implementations.
- Developed the Augmented Reality application for visualization of the recorded gestures within a 3-D Virtual world.
- Built the C# code for the AR based mobile applications.

- Analysis of one of the NN models and picking the right configuration with respect to all the features of the Gauntlet's data.
- Built the User Interface for the AR mobile applications.
- Implementation of the Internet-of-Things (IoT) network with integration with the Gauntlet.
- Implemented the client side of the Gauntlet using TCP/IP configuration.

1.3.2 The Prototype - Group Effort

Our group effort have been mostly contributed towards 3 magnanimous tasks. Initially, it was the physical design of the Gauntlet from scratch onto a heavy duty glove with minimalist digital circuitry and component placement for portability and compactness to tolerate free movement. Our biggest contribution is collecting the essential raw data from the Gauntlet for the given gestures for the functionality of neural network training. We have stored many versions of the data sets due to the variety of recording the movements. Last but not the least, this thesis paper was documented collectively with all our efforts together as well.

1.4 Delimitations

This section describes the various alterations in the main scope of this project.

- The Gauntlet was initially intended to be dual handed devices but due to resources and time, we have implemented only the right-hand glove for feasibility of the thesis.
- The hardware that was developed is still a prototype and cannot be used for commercial purposes.
- The gauntlet is at the moment not wireless as described in our specification as the factor of power supply integrated would slow down the development of the scope of research.
- Even though we mentioned using the SensorFusion App from external sources, it played a very small role in terms of implementation of filters as we required many more sensors to work with.
- The training data we recorded for the development of the neural networks are purely configured for the Gauntlet itself as the positioning of the sensors play a massive role on the raw data. Thus the comparison of our IMU database with pre-existing public IMU databases.
- The implemented neural network/ ML algorithms were developed from other sources and the HGR-net was not implemented as a base reference as mentioned in the specification.
- The number of applications designed for this Gauntlet have been limited to two yet we didn't implement google's speech-to-text.
- Probabilistic/Statistic based ML models proposed in the specification have not been implemented to compare the performance of the neural networks we have proposed in this paper.

1.5 Thesis structure

In Chapter 2 we summarize the various research papers we used as references for the different aspects involved in our thesis. In Chapter 3 we discuss about the hardware and software implementations used for the product development. In Chapter 4 we discuss the analysis of data and possible results of the experimental protocols. In Chapter 5 we summarize the conclusions we obtained during our evaluation. In Chapter 6 we discuss the future works that can be augmented on the prototype.

Chapter 2

Related Work

Hand gesture/activity recognition (HAR/HGR) has been studied through the use of different sensors such as Inertial Measurement Units (IMU), Surface Electromyography (sEMG) sensors, Flex sensors, Radio Frequency Identification (RFID) tags and also the conventional video capture recognition. Apart from the various studies using different sensors, there are many more studies to achieve HAR/HGR by the types of sensor fusion techniques, Machine and deep learning algorithms. In this chapter, we discuss the relevant literature related to Hand Activity/Gesture Recognition (HAR/HGR), Augmented Reality (AR) and American Sign Language (ASL), especially the papers investigating different deep learning architectures, wearable system for ASL and public data sets of sensor-based HAR/HGR.

2.1 Hand Activity/Gesture Recognition with various ML and NN algorithms

The recognition of human activities is a well studied field, including many daily activities such as fitness tracking and health monitoring or gestures for interaction with devices or sign language recognition. The authors of [1] have implemented a system with mobile sensors in smartphones. The sensor data would run through a CNN where they propose the use of a new weight sharing technique called *Partial Weight Sharing*. This technique is applied to accelerometer data for improved signals. They ran data of 3 public data sets, namely Skoda [2] (assembly line activities) , Opportunity [3] (activities in kitchen) and Actitracker [4] (jogging,walking etc.) and achieved accuracies of 88.19%, 76.83% 96.88% respectively. Even though their weight sharing technique gave good results they have left experimenting with larger data and broader sets of activities for future studies.

Similar to this, another paper [5] also ran their CNN model from input data of multiple on-body worn inertial sensors which are multi-channel time series signals. In this they propose a deep architecture of CNN to investigate the multi-channel time series data as given in Figure 2.1. This architecture uses temporal convolution and pooling to identify salient features from times series data. All these salient patterns are combined with multiple channels and then mapped onto the classification on human activities. They have used the Opportunity data set to acquire good results.

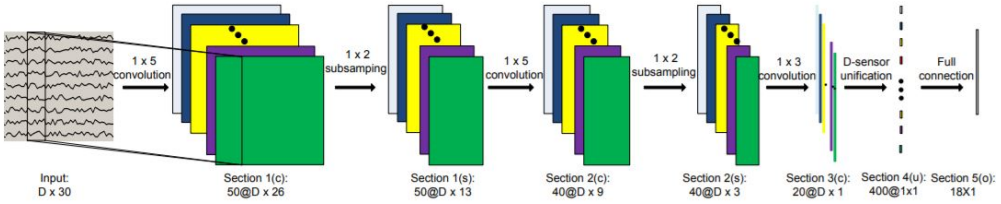


Figure 2.1: Illustration of the CNN architecture used for a multi-sensor based human activity recognition problems [5]

Another study [6] with just a single IMU sensor on the wrist used 2 deep learning algorithms 1-D CNN and RNN for 5 activities (standing, walking, running, walking upstairs and downstairs) from the Physical Activity Monitoring for Aging People (PAMAP2) [7] database. They have achieved accuracies of 95.43% and 96.95% for CNN and RNN respectively . These are really good results considering the use of only one IMU sensor, However, it is only feasible for this kind of macro human activities.

Compared to another study [8] which also uses a single wrist IMU sensor but for a different activities - data set (Opportunity [3] : kitchen activities) on a RNN model they achieved an average accuracy of 80.09%. In this paper [9] they developed a 2-D kernel CNN model given in Figure 2.2 to capture spatial dependencies of sensors and their axes and local dependency over time. Sensors in different position were grouped by the type of data to capture spatial dependency. They achieved an accuracy of 97.92% for the Skoda data set with 2-D CNN compared to 1-D's 97.40%. The reason they have used 2-D is the usage of multiple sensors (in both smart phones and

smart watches).

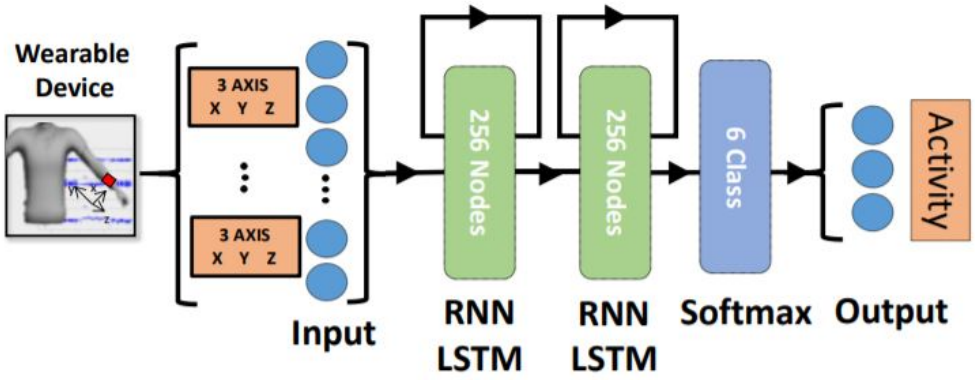


Figure 2.2: The proposed single IMU- and RNN-based hand activity recognition system [8]

A very interesting study [10] where they focus on every aspect of HAR i.e. the position of sensors, the sampling rate of raw data from sensors and sensor fusion techniques on different sensors differently for different activities using a 2 level ensemble technique which assigns custom weights and modalities after analysis from data of different sensors. They have experimented on 9 different activity labels. They also mentioned that they aim to study this further by adding environmental(GPS) and physiological sensors to get a better understanding of patterns and recognition of changes in these patterns over time.

2.2 American Sign Language (ASL)

This paper [11] builds a wearable system consisting of IMU and sEMG sensors for recognizing around 80 commonly used signs in the American Sign Language. They used four well known classification algorithms NaiveBayes, NeareastNeighbor, Decision Tree [12] and LibSVM [13] and they achieved an accuracy of 96.16 for 40 selected features. This paper shows the significance of sEMG sensors in increasing the accuracy of ASL recognition.

2.3 Finger and hand tracking

With the large advancements in Microelectromechanical systems(MEMS) technology we now have IMU's as small as our fingernails or even smaller. This lets us explore a large number of possible hand and finger movements in different permutations and combinations for interaction with computers or other devices. A glove was developed [14] called 'GyrGlove' with data captured by Python software where they used inertial based motion capture technique on a host computer. This captured data goes through a set of IMU calculations on MATLAB. The 3-D visualization of this glove was developed on PANDA3D. This paper laid a good foundation related to gestures and pattern recognition.

2.4 Augmented Reality

When it comes to the field of AR, we did not dwell deep into research and did not refer to much research papers more than online tutorials as our focus on AR was just as an application and not as research. But few of the research papers we looked into for AR definitely helped us pave the way for understanding what we wanted our AR applications to achieve.

One of the research papers that intrigued our interest relevantly is [15]. By combining low cost Microelectromechanical system (MEMS) IMU's and a wireless body sensor network (BSN), the authors were able to provide a highly accurate human body motion capture system which is visualized on a virtual environment using Unity3D. Another research paper that caught our attention and was more inclined into AR is [16]. This paper gives a really detailed and more in-depth understanding about Augmented Reality and it is applications. By combining compact light-weight IMU's, GPS and Computer vision, the author was able to create an accurate head pose tracking system forAR.

We tried to visualize the orientation and position of the glove in real time on Unity3D. Even though we could map the orientation of the glove, we were not able to achieve the position mapping in a 3-D space as we did not have the proper hardware to achieve it. It would require us to use a Global Positioning System (GPS) and a computer vision based system but our goal is to build a product which is feasible and compact. Hence, we decided to

visualize the gesture interactions with virtual objects rather than mapping the glove itself on a virtual environment.

Chapter 3

Implementation

We will separate the description of the implementation of the Gauntlet into three separate parts such as firstly describing the physical design of the prototype and the data acquisition process, and secondly describing the analysis through the use of Machine Learning and signal processing, and finally the applications designed for the Gauntlet.

3.1 Glove Hardware Design and Data Acquisition

We designed a single wearable glove prototype based on IMU sensors that collects time-series inertial data sampled at 25Hz as can be seen in Figure 3.1. There are two types of IMU sensors that are deployed onto the glove: One MPU9250 and five MPU6050's. The MPU6050 is positioned at every finger-

nail including the thumb whereas the MPU9250 is positioned at the centre of the back of the palm. All these senors are interfaced into the MCU i.e. Arduino Nano. Due to its features of compactness, low power consumption and cycle speed, we chose this microcontroller to integrate all the sensors. Lets discuss how we communicate and integrate these sensors.

(a) (b)

Figure 3.1: Gauntlet-X1 Prototype

We know two serial communication protocols that work with these sensors i.e. the Serial Peripheral Interface (SPI) and the Inter-Integrated Circuit (I2C) protocols. The SPI would be the ideal choice as multiple sensors can be communicated with ease using the chip select pins. But the wiring will be more complex and unnecessary making the Gauntlet bulkier with wires. With I2C, we cannot have more than two of the same sensors due to the competency of the alternate addresses. According to the [17] documentations, using the ADDR pin in MPU6050, we can change the address from 0x68 to 0x69. That limits the number of MPU6050 units we can use but isn't a problem for the MPU9250, as only one is used and it has a unique address of its own. But considering the fact that the speed of the clock in Arduino is fast enough

to change this address, we chose to implement a polling solution to fix the multiple interfaces of these sensors using I2C itself. We know that the data collected by the MCU will be from one sensor after the other i.e. the palm, then each finger respectively from the thumb to the pinkie. While we are shifting to each sensor we can change the address before reading the registers and switch back the original address of the sensor then carry on with others as well. This is a concept similar to the one implemented in the SPI where a CS pin is used to switch between slave modules respectively. By optimizing the library in this way, using minimal setups and no calibration, raw data can be collected by the Arduino at the maximum optimized sampling speed of 25Hz from all sensors semi-simultaneously.

The MCU is placed on a mini breadboard along with the palm sensor and all the sensors, using robust wiring, are connected directly to the MCU. All the modules are fixed on the glove using simple Velcro which is sewn onto the glove. Initially the system was proposed to use two gloves simultaneously (one for each hand), both wireless and battery driven but for the sake of debugging and ease of development as a prototype, we designed the prototype to be single-gloved right-handed and a wired connection over a USB port provided by the Nano. However, now that we have calibrated the prototype for one hand, the addition of a second glove becomes more or less a technical detail.

Once the Gauntlet was ready for use, our next objective was to establish what all the incoming data can be used for and how we can process and

analyze it for the most effective way to track hand movements and relay gestures. To proceed with this, we need to understand the data we gather. The next section will explain how to deal with this scenario.

3.2 Data Preprocessing with Pose estimation

In order to increase the efficiency of the calculations, the data collected needed to be preprocessed, using the various techniques described below.

We studied many research papers to understand encountered problems and issues that need to be addressed, such as treating the inertial data with various filtering methods, noise cancelling and sensor fusion techniques as mentioned earlier in Chapter 2. Most of these algorithms were implemented via MATLAB scripts so lets discuss these concepts in detail and evaluate how our data could be prepared in order to make more sense.

3.2.1 Madgwick/Mahony Filter

A Madgwick/Mahony Filter gives us orientation information from a combination of 9DoF/6DoF inertial data. According to the Figure 3.2, the filter uses a quaternion representation, allowing accelerometer and magnetometer data to be used in an analytically derived and optimized gradient-descent algorithm to compute the direction of the gyroscope measurement error as a quaternion derivative. Considering tracking of the motion of the hand, gaining orientation information might be sufficient to understand and visualize

the movements easily. But this is yet to be worked on. Using MATLAB scripting, we created classes for respective filters and set up serial communication with the Gauntlet to gather data continuously so that we can process the filter at a particular sampling rate. Each sensor will instantiate its own filters.

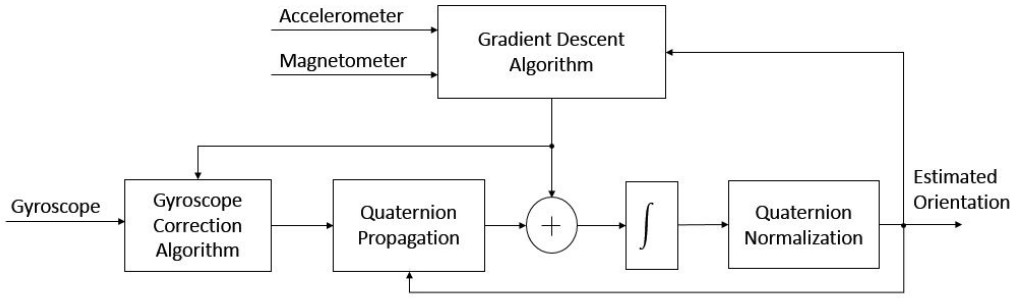


Figure 3.2: Block Diagram of the Operations in a Madgwick/ Mahony filter for 9DoF IMU

3.2.2 Extended Kalman Filter (EKF)

EKF is an extended version of the Kalman filter algorithm that estimates unknown variables for a non-linear based model. The EKF works by transforming the nonlinear models at each time step into linearized systems of equations. Generally in a single-variable model, we would do this using the current model value and its derivative. But in our case, we should safely assume that our system is a multi-variable non-linear system and the generalization of the multiple variables and equations is a Jacobian matrix computation. Then the linearized equations can be used in a similar manner

to the standard Kalman filter. So we used EKF in a calibration step to check if it could improve the noise and estimate the relative position and orientation.

Raw data gave better accuracy while training and testing data with our NN. Even though filters increased efficiency, the computation time and memory space that the application requires, seems unreasonable for the intentions of this thesis paper. We shall explain in detail about data processing based on it's reduced predictive accuracy and the excessive processing requirement in section 4.

3.3 Machine Learning (ML) Architectures

Human activity recognition is the research of classifying sequences of inertial sensory data recorded by specialized harness test-beds or smart phones into known well-defined movements. Classical approaches to the research involves hand crafting features from the time series data based on fixed-sized windows and training Machine Learning (ML) models, such as ensembles of decision trees. The difficulty is that this feature of engineering requires strong expertise in the field.

The amount of data that can be collected with the Gauntlet glove is very large and may show small, but important variations, and therefore the data needs to be analysed using methods developed essentially for this purpose. Obvious candidates for this type of data are the various methods used for

Machine Learning (ML). However, selecting the right kind of ML system and model is not easy. In this section we will discuss the various methods considered for the project.

Recently, deep learning methods such as Recurring Neural Network (RNN) like Long Short-Term Memory (LSTM) networks and variations that make use of one-dimensional Convolutional Neural Networks (CNNs) have been shown to provide state-of-the-art results on challenging activity recognition tasks with little to no data feature engineering, instead using feature learning on raw data. In this part, we are going to describe the different algorithms implemented using *supervised* learning.

3.3.1 Long Short-Term Memory (LSTM) Model

LSTM network models are a type of Recurrent NN that will be able to learn and remember over long sequences of input data as given in Figure 3.3 [18]. They are intended for use mostly with data that is comprised of long sequences of data, up to 200 or 400 time steps. Considering our activity recognition, it would be great fit to use a LSTM network. The model itself can support multiple parallel sequences of input data, including every axis of the inertial data. The model learns to extract features from sequences of observations and how to map the internal features to different HAR/HGR classes. The advantage of using LSTMs for sequence classification is that they can learn from the raw time series data directly, and in turn does not require domain expertise to manually engineer input features but we try different set

of features. The model can learn an internal representation of the time series data and ideally even achieve comparable performance to models fitted onto a version of the data set with engineered features.

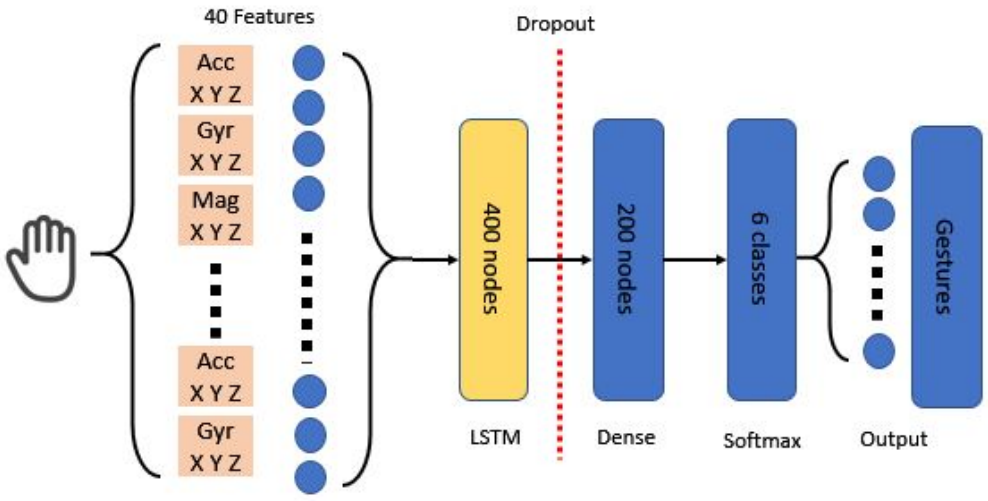


Figure 3.3: Description of the LSTM Architecture

We are going to define the model as having a single LSTM hidden layer followed by a dropout layer to reduce over-fitting the model to the training data. Additionally, a dense fully connected layer is used to interpret the features extracted from the LSTM layer in order to make the predictions. The efficient Adam version of stochastic gradient descent is used to optimize, and the categorical cross entropy loss function will be used as we are trying to solve a multi-class classification problem.

3.3.2 CNN-LSTM model

We are going to use a CNN model along with the LSTM model as given in Figure 3.4 except there is one complication. [18] The entire CNN model can be wrapped in a Time-Distributed layer to allow the same CNN model to read in each of the given sub-sequences in the window. This wrapper applies a layer to every temporal slice of an input. We define the CNN models with 1-D convolution layers that convolve with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs. The parameters we use for this layer is as follows: filters=128, kernel size=4 and activation as rectified Linear unit, following with a dropout layer and 1-D pooling layer. The extracted features are then flattened and provided to the LSTM model to read, extracting its own features before a final mapping to an activity is made.

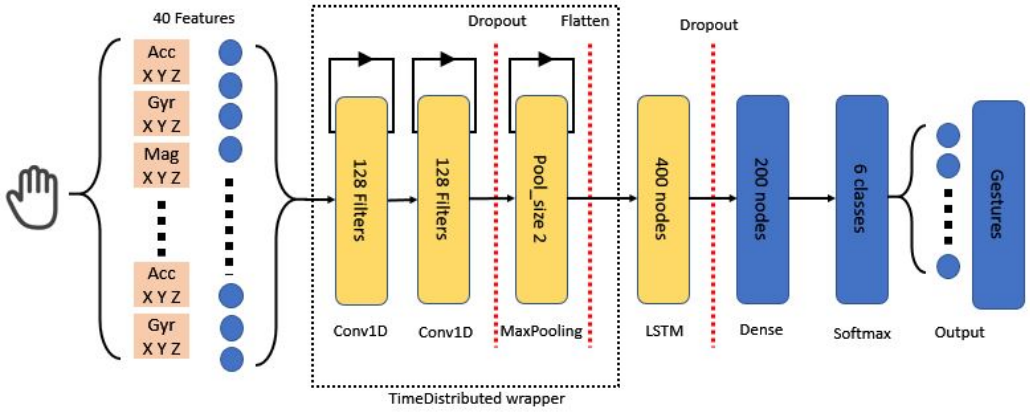


Figure 3.4: Description of the CNN-LSTM Architecture

3.4 American Sign Language (ASL)

One of the typical applications that can be elaborated from hand gestures is the recognition of sign language. For the example, we chose to detect ASL, since it can be said to serve as one of the predominant sign language among the hearing impaired community. For practical reasons our main focus is on the alphabetical signing that augments the vocabulary of ASL. Considering the rhythm, speed and movement of every letter and digit, the gauntlet will be a good fit to test if a real-time application for ASL is possible. Most of the letters have stationary gesture while a few have movable gestures which made it apt for us to test. Out of the 26 characters in the alphabet, we are developing HAR/HGR with 5 alphabet characters that are stationary and different orientations. Figure 3.5 shows the five different signs being signed in real-time. Note that there are a few signs that use both position and motion together, which complicates the detection process. We will discuss how we trained the ML models for these gestures in a later part of the thesis.

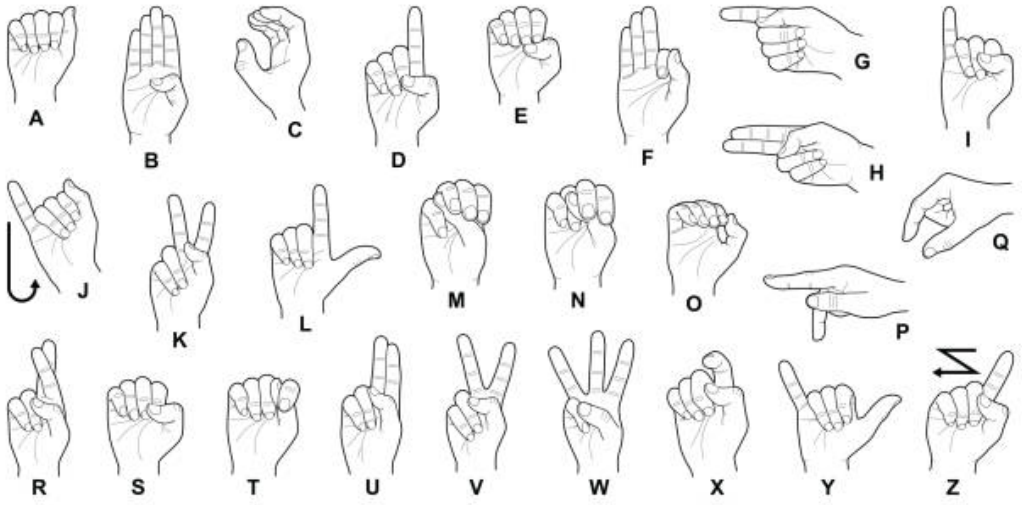


Figure 3.5: All the alphabet signs in ASL

3.5 Augmented Reality

Augmented Reality (AR) is the presentation of and the interaction with a virtual object in the real world. In simple terms, AR brings a virtual object into the real world, which the user is then able to interact with as if it was really placed in the environment, using different peripheral devices. AR technology is transforming the way companies design, manufacture, operate, and service products at a fast rate. It is one of the best approaches for visualizing virtual concepts in the 3-D world. We built two AR applications to visualize what the Gauntlet is capable of achieving with gestures in the virtual world. The applications are built using the *Unity* engine and *Vuforia*.

Unity [19] is one of the most popular development platforms for creating games and applications in 2-D, 3-D, VR andAR. The Unity engine provides users with a chance to experiment with a lot of features that are simple and easy to learn. The User Interface is comprehensive, and they provide various online tutorials.

Vuforia [20] is an AR integration platform created by the global software company PTC, who is leading in the industrial innovation platform. Vuforia is globally known as one of the major software systems used within IndustrialAR. It is the foremost technology used for integrating AR with today's industrial enterprise. It is one of the most widely used software for digital eye-wear and hand-held devices.

Unity has partnered with Vuforia to create one of the best AR creation platforms available. The User Interface makes it both appealing and simple to learn. Unity also provides the user with the choice to work on any platform such as Windows, Linux, Android, IOS etc. This way we were able to create the AR applications also for an Android platform, making it easier to present our AR applications through our hand-held devices.

During the implementation, we tried to visualize the orientation and position of the glove in real time on Unity3D. Even though we could map the orientation of the glove, we were not able to achieve the position mapping in a 3-D space as we did not acquire the proper hardware to achieve it. It would require us to use a complementary (Global Positioning System (GPS))

or a computer vision based system but our goal is to build a product which is feasible and compact. Hence, we decided to visualize the gesture interactions with virtual objects rather than mapping the glove itself on a virtual environment.

3.5.1 Different types of AR

There are various methods in AR to incorporate a virtual object into the real world. Each method has their own outcomes and advantages depending on the scale of the AR application. Below, we have explained these methods in brief.

Marker based AR – This method is also known as Recognition basedAR. The desired virtual object is "placed" on a recognized physical target or a "marker". The virtual object appears into the real world with the help of this existing physical marker. These markers can be of different types. The more distinctive and unique the marker, the better it is for the device camera to recognize. Image targets are one of the easiest ways to place a virtual object. Image targets include flat objects such as paper, magazines, photographs etc. When the device camera recognizes the marker, the virtual object will be tracked to its position on the marker.

Marker-less based AR – This method does not involve the use of markers. The virtual object is placed wherever the user wants to. The object usually appears to look like it is floating in mid-air, but it is possible to place the object on a flat surface. The placement of the virtual object is implemented by the device used to view the virtual object. With the help of Vuforia and

ARCore by Google, the device will scan the room for flat surfaces where the user can place the virtual object, in the case of touchscreen mobiles with a simple touch on the screen. This type of AR eliminates the need of a physical marker as any flat surface can be used as a ground.

Location based AR – This method is similar to the marker-less AR, it does not require a specific marker to position the virtual object. The main difference is that it is only based on information from the position sensors and Geo location devices, i.e. mostly GPS. This type of AR is mainly used over a wide area or various locations or even globally, like in the mobile applications Ingress and PokemonGo. Due to its use of the device's GPS, the positioning of the virtual object is more accurate compared to the other types.

As our application is small and just focuses on a small area of interest, we decided to work with Marker-less AR. Using this method, we visualized the five movable gestures in a 3-D virtual world through a AR mobile application.

3.6 Experiment protocols

As an initial experiment for development purposes, we collected gesture data from ourselves, 3 males from the age 24-26, heights ranging from 166 -178 cm and all right-handed. Lets discuss the approach in which data is recorded.

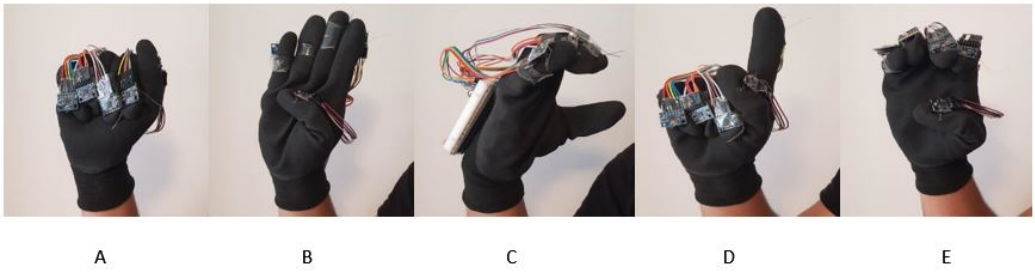


Figure 3.6: List of all the Stationary gestures for training

Figure 3.7: List of all the Movable gestures for training

Initially the first approach was to record the data based on the actual movement of signing a gesture i.e. from hands normal position to doing the gesture and finally back to the hands normal position. The data set has no constraints on the window sizes so each sample had a varying size from 30-50 time steps, which corresponds to 120-200ms respectively.

However, for the implemented neural networks above, it is not possible to train the neural network (NN) without a fixed data structure for the data samples due to the constraint of a fixed input shape of the networks. Thus

the data sets will have to be recorded and trained individually for stationary and movable gestures in separate networks also considering the fact that each type of gesture has a varied timing of completing the gesture i.e. lasting for several more milliseconds. This is why we recorded the two different types of gestures with window size of 20 and 40 time steps while having two different sets of experimental protocols for each NN data analysis. According to the training data for fitting, we can represent our data as (samples, time steps, features). As each time step is every 40ms, every data set sample is recorded for 1.6s and 0.8s respectively. Table 3.1 is the list of gestures we have collected data for.

No.	**Activity**	**Type**	**Duration (ms)**
1	Aph. A	Stationary	3,980
2	Aph. B		4,000
3	Aph. C		3,980
4	Aph. D		3,980
5	Aph. E		4,000
6	NULL		4,020
7	Left to Right	Movable	8,120
8	Right to Left		8,120
9	Down to Up		8,120
10	Up to Down		8,120
11	Circle		8,120
12	NULL		8,200
Total			72,760

Table 3.1: Various Gestures recorded for training neural networks

The first 5 gestures are *stationary gestures* meaning there will be no movement involved when signing the gesture, while the next 5 consists of distinc-

tive movements thus called *movable gestures*. Additionally to this, we have added another activity to each of the two categories called the *'null' gesture*. We have introduced this neutral gesture, which is basically no movement in the sensor data, i.e., the general idle position of the hand when standing and sitting, to make the relevant gestures more accurate to predict.

Time step	Full Hand												
	Palm			Fingers									
				Thumb		Index		Middle		Ring		Pinkie	
	Acc	Gyr	Mag	Acc	Gyr	Acc	Gyr	Acc	Gyr	Acc	Gyr	Acc	Gyr
1	2-4	5-7	8-10	11-13	14-16	17-19	20-22	23-25	26-28	29-31	32-34	35-37	38-40

Table 3.2: Data format of the Gauntlet

The Table 3.2 describes the Data format of the information received from the Gauntlet and how we classify its channels for experimental protocols.

Exp	Type of gestures	Type of data
1	Stationary	Palm
2		Fingers
3		Pure Acc
4		Pure Gyr
5		AccMag
6		GyrMag
7		AccGyr
8		All
9	Movable	Palm
10		Fingers
11		Pure Acc
12		Pure Gyr
13		AccMag
14		GyrMag
15		AccGyr
16		All

Table 3.3: Classification of Experiments - LSTM / CNN-LSTM

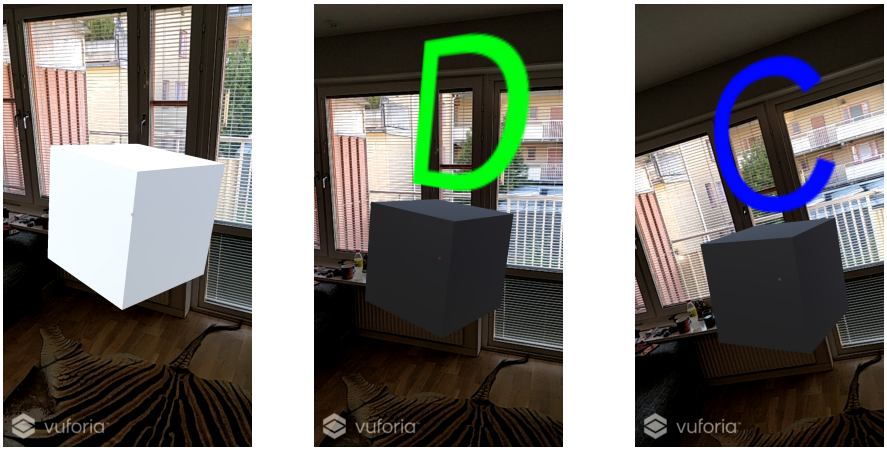
Using the Tables 3.1 and 3.2 describing our data-format, type of gestures and classification of the experiments, we implement Table 3.3 describing protocols to analyze these models based on accuracy, losses and prediction data to find the best working model that can describe our HAR/HGR effectively. Each of these experimental tables are implemented for each model we have described above in a server which is discussed later in the chapter. The server will be adopted on a PC equipped with Intel Core i7-3537U @ 2.5 Ghz CPU.

While we are acquiring various types of data sets, in order to find the best model among these protocols, we still need to manually tune the configuration of the defined model based on the nodes of the various layers, the rate of dropout layer, training batch size and number of epochs to train the model without overfitting/underfitting the training data.

3.7 Applications

As we mentioned previously, we built two Android applications using the Unity engine based on AR technology. We have described the functions of both the applications in brief, below.

Alphabets - This application is used to visualize the stationary gestures made by the Gauntlet in the real world. This AR application, as in Figure 3.8, consists of a white box with 5 alphabets within (A to E). Depending on what sign language is made by the user wearing the Gauntlet, the corresponding alphabet pops out of the box in real time. This is a good way to visualize the ASL as alphabets in the virtual world and also provides a learning experience to the users.



(a) (b) (c)

Figure 3.8: Screenshots of the Android application "Alphabets" (a) start of the app, (b) when alphabet 'D' is signed, (c) when alphabet 'C' is signed.

The Cube - This application is used to visualize the moving gestures made by the Gauntlet in the real world. As the name suggests, the AR application, as in Figure 3.9, consists of just a white cube with some custom inbuilt effects and animations. Depending on the moving gesture made by the user wearing the Gauntlet, the cube performs an animation. The moving gestures used for this application are the motion of the hand upwards, downwards, to the left, to the right and a circle motion in front of the user. Each gesture has its own corresponding animation created and the application reacts to the gestures in real time.

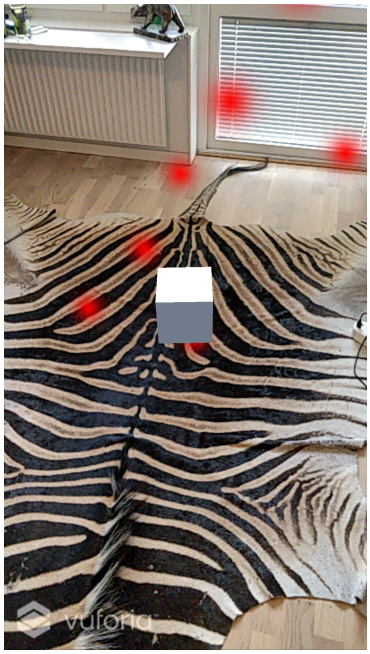


Figure 3.9: Screenshot of the Android application "Cube" shows animations based on the moving gestures.

3.8 Server and Internet-of-Things (IoT)

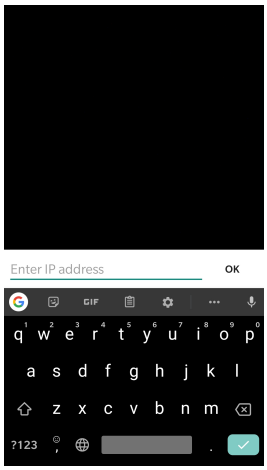


Figure 3.10: Screenshot of User Interface (UI) requesting for the server's IP address.

We developed a server module for data collection and preprocessing for the Gauntlet. Considering the sample rate and large amount of data accumulated at the Gauntlet side, we established a serial connection to the Gauntlet. Depending on performing test runs or deploying applications based on the models created, the module suffices. Using basic socket connectivity(TCP/IP), any mobile running theASL application, as in Figure 3.10, can connect to the server module via entering the IP address of the server into the applications interface.

The server can also be connected to multiple cellphones by switching automatically to the respective application as described in Figure 3.11. So to summarize the server module, it consists of three different sub-modules:

Gauntlet connectivity, Internet-of-Things (IoT) Application Manager, Data Acquisition/Prediction manager.

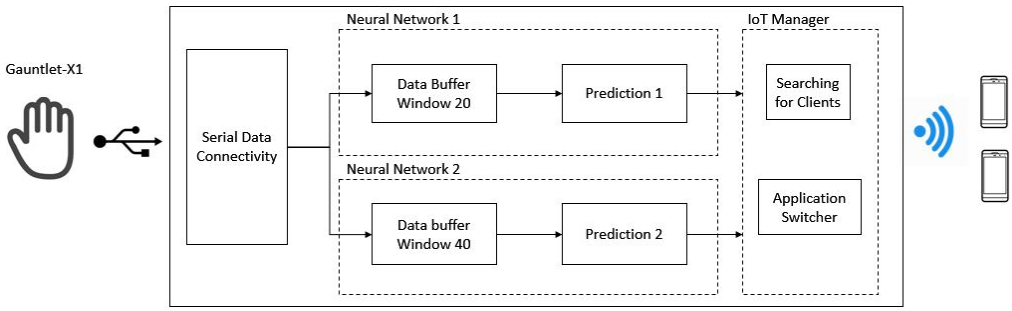


Figure 3.11: Block Diagram of the operations for Gauntlet's server

Chapter 4

Data Analysis

In this chapter we will go deeper into the matter of data analysis of the implemented algorithms and neural networks that have been described in chapter 3 and acquiring its results.

4.1 Impact of Preprocessed Data

Considering the algorithms we had discussed in the previous section, we employed these sensor fusion techniques with all the data i.e. accelerometer, gyroscope and magnetometer of each sensor module.

After we implemented these techniques in MATLAB and collected data from the Gauntlet at the given sampling rate, we came to the realization that some of the data is being lost from the Gauntlet while preprocessing all

the sensors during fusion. This is due to the heavy load of processing all the sensors in parallel given that we have six sensors to gather data from and to fuse. To dwell even deeper into the understanding of this problem would be a whole other area of research. Thus, we decided to see how these data would work offline with Neural Networks. Before we show the analysis in brief, we always split the data sets into 80% training data and 20% test data when we evaluate the Neural Network models. We then test these models using the Stationary data sets for now.

Initially we experimented by converting the raw data sets that were acquired by our fellow volunteers, into orientation or quaternion representations to feed into the Neural Networks. As we know, the current data set consists of 40 features including the timestamp. After converting using Madgwick filter, the preprocessed data outputs us 19 features including timestamp. These features are orientation representations i.e. roll, pitch and yaw for each sensor. With our base configuration of the Neural Network, we tried to train the LSTM model using the full data set (All) in this representation. It turns out the training accuracy does not increase to more than 33.6%, and thus our test accuracy dropped as well. Without hesitation, we realized that converting the data sets into other representations would not be a suitable method to proceed for the applications we have designed. So we tried even EKF on the data sets to give us quaternion representations. The preprocessed data outputs from 40 features to 25 features as each sensor gives us 4 vector quaternion. Again, with the rudimentary basic LSTM model, we fit these quaternion data sets accordingly to evaluate the performance of the

model. To our understanding, the training accuracy seems to overfit from the second epoch itself. Vaguely, it doesn't make sense.

From all these implementations, we assumed that using filters to fuse data and reduce noise would also affect the training process of the models. We also took into consideration of tuning the configuration of the model i.e. the number of nodes in each layer, the dropout rate and the batch size. But the model with the best training accuracy and test accuracy gave us not more than 73.4%. So far, this is the impact of sensor fusion over the data sets that can be fit into the Neural Network models.

4.2 Impact of non-processed data

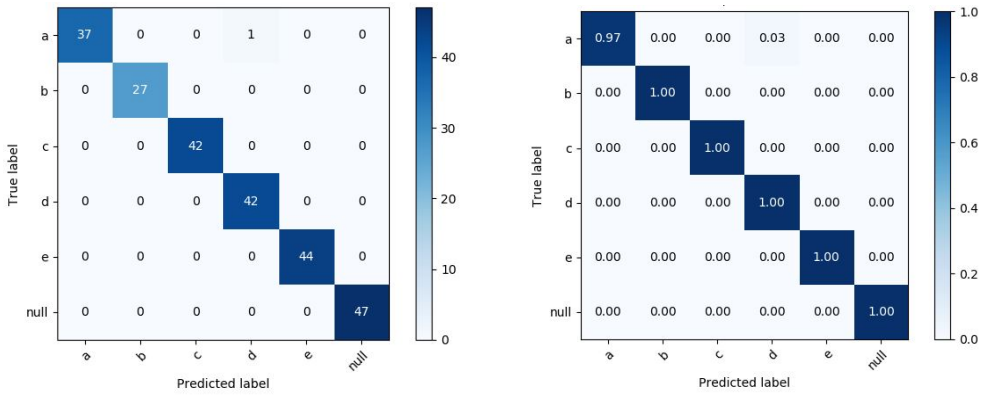
Since sensor fusion didn't work out so well, we decided to try training the models using the raw and non-calibrated data available to us. Initially, we set the basic configuration of the LSTM model and tried fitting all data sets which have window sizes of 20 time steps. To our surprise, we noticed that the training had a much better performance compared to the previous data sets. The training accuracy was shooting to 99.8% from epoch 0 to 10. While training had good results, the test accuracy also gave us 99% accuracy with a loss of 3-4% which is a good start to see how well the gauntlet could perform in real time. Lets examine the experiment protocols for each model we had defined in the previous section:

4.2.1 Experiments for Stationary data sets in LSTM

1st Layer nodes	Dropout rate	2nd Layer nodes	Types of Features	Training		Test	
				Accuracy (+/-)	Loss (+/-)	Accuracy (+/-)	Loss (+/-)
350	0.5	150	All	99.690% (+/-0.438)	1.054% (+/-0.719)	99.167% (+/-0.900)	3.311% (+/-3.302)
400	0.5	200	All	99.148% (+/-1.205)	2.447% (+/-2.752)	99.306% (+/-0.196)	2.523% (+/-1.087)
400	0.25	200	Fingers	99.977% (+/-0.046)	0.060% (+/-0.100)	98.583% (+/-0.204)	4.592% (+/-0.989)
400	0.25	200	PureGyr	99.977% (+/-0.046)	0.139% (+/-0.213)	98.667% (+/-0.408)	6.081% (+/-2.523)

Table 4.1: Optimal Experiments of LSTM model for Stationary Gestures

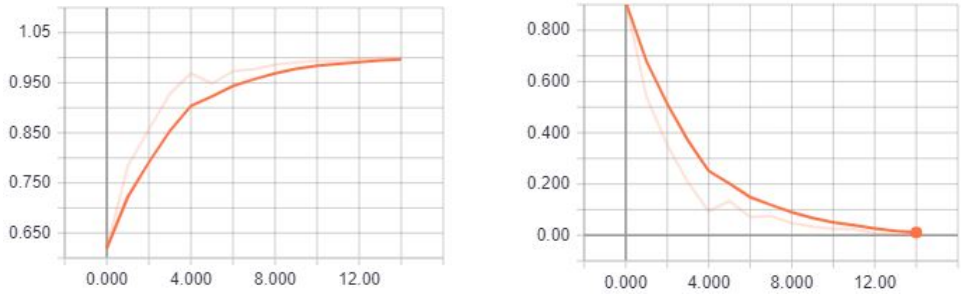
As we can see from Table 6.1 in the Appendix, 56 experiments with varied model configuration of the model and type of features have been implemented to find the closest optimal model of LSTM for Stationary gestures. When we sort the table based on the training accuracy and test accuracy, we pick optimal choices as given in Table 4.1 and chose the best of model which has the configuration of 400 neurons on the 1st LSTM input layer, a dropout rate of 0.5, 200 neurons in the 2nd hidden dense layer and training with all the features of the data set. The accuracy of training and test are close to 99.148% while the loss doesn't go more than 2.5%. Further, we fine tune the model based on epochs and batch size to see if there will be any difference. In order to validate if this model would be a good choice, we read more into its confusion matrix of the test samples and the see the performance in real-time predictions.



(a) (b)

Figure 4.1: Confusion Matrix of the LSTM model for Stationary Gestures (a) w/o normalizing (b) w/ normalizing

Looking at Figure 4.1, it portrays the confusion matrix of the predicted labels with the true labels using the test data sets which again are 20% of the main data set. As we can see, the predictions have been extraordinarily accurate in one of the trials which gave us exact classification of the stationary gestures. In Figure 4.2 this graph is created using Google's Tensorboard that is easily available to us to read more into the training analysis of the defined model. The training process has pretty much stopped at epoch 14 giving us the optimal training results without overfitting or underfitting the model in terms of accuracy as well as loss. Thus we conclude to use this model for our main application for the server for stationary gestures among the LSTM experiments.



(a) Epochs Vs. Training Accuracy (b) Epochs Vs. Training Loss

Figure 4.2: Training performance of the LSTM model for Stationary Gestures

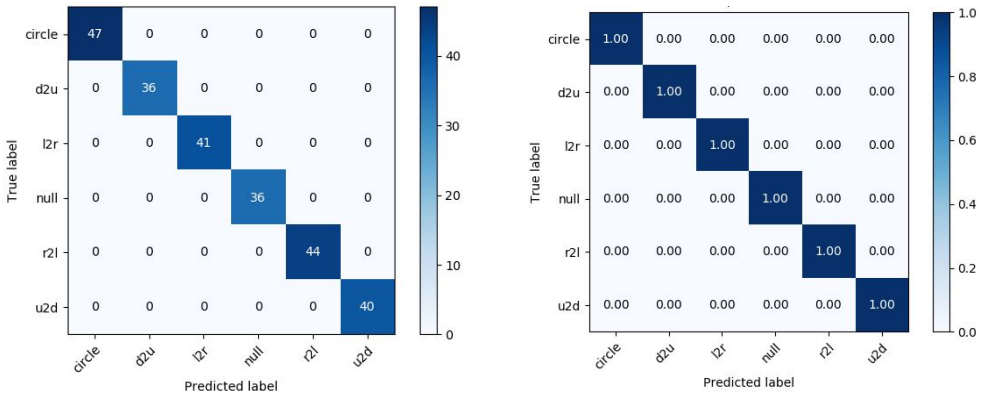
4.2.2 Experiments for Movable data sets in LSTM

1st Layer nodes	Dropout rate	2nd Layer nodes	Types of Features	Training		Test	
				Accuracy (+/-)	Loss (+/-)	Accuracy (+/-)	Loss (+/-)
400	0.25	200	All	100.000% (+/-0.000)	0.037% (+/-0.001)	100.000% (+/-0.000)	0.185% (+/-0.035)
400	0.25	150	All	100.000% (+/-0.000)	0.038% (+/-0.002)	99.863% (+/-0.193)	1.216% (+/-0.083)
400	0.25	200	All	100.000% (+/-0.000)	0.037% (+/-0.001)	100.000% (+/-0.000)	0.185% (+/-0.035)
400	0.5	200	All	100.000% (+/-0.000)	0.071% (+/-0.001)	100.000% (+/-0.000)	0.106% (+/-0.014)

Table 4.2: Optimal Experiments of LSTM model for Movable Gestures

As we can see from Table 6.2 in the Appendix, again 56 experiments with varied model configuration of the LSTM model and type of features have been implemented to find the closest optimal model of LSTM but for movable gestures. When we sort the table based on the training accuracy and test accuracy, using the optimal choices as given in Table 4.2, we find that the configuration of 400 neurons on the 1st LSTM input layer, a dropout rate of 0.25, 100 neurons in the 2nd hidden dense layer and training with all the features of the movable data sets. In this case, the accuracy of the training and test have given us surprisingly 100% while the loss is collectively

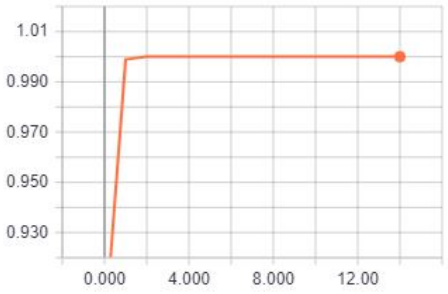
lesser than 1% which is almost negligible. Since the model gave us quite an adequate result on the training process, we avoid trying to fine tune the configuration at the moment and use this model to further examine its analysis.



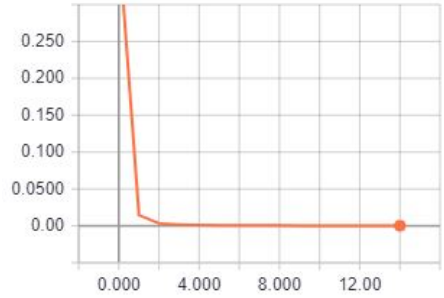
(a) (b)

Figure 4.3: Confusion Matrix of the LSTM model for Movable Gestures (a) w/o normalizing (b) w/ normalizing

Looking at Figure 4.3, it describes the confusion matrix of the predicted labels with the true labels using the test data set. As we can see, the predictions have been very accurate in one of the trials which gave us again the exact classification of the movable gestures. When we look at Figure 4.4, the training process has stopped at epoch 2 giving us the linear results without overfitting or underfitting the model even after epoch 10. Thus we decided to use this model for our main application for the server in real-time for movable gestures among the LSTM experiments.



(a) Epochs Vs. Training Accuracy



(b) Epochs Vs. Training Loss

Figure 4.4: Training performance of the LSTM model for Movable Gestures

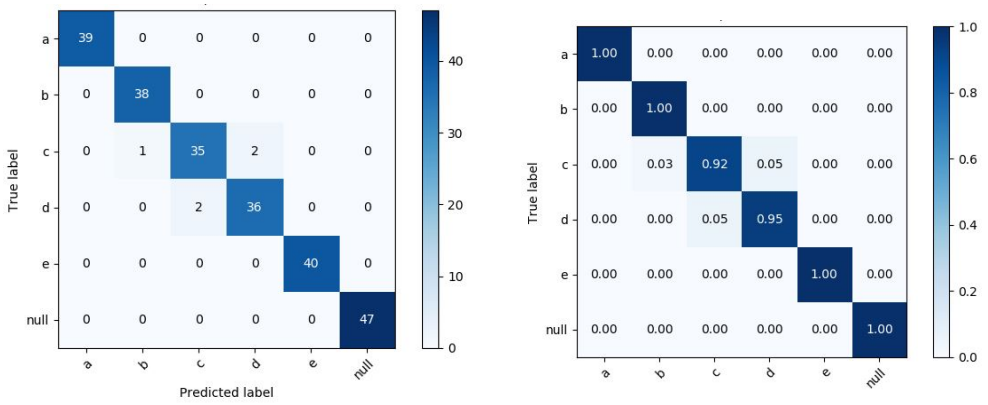
4.2.3 Experiments for Stationary data sets in CNN-LSTM

1st Layer nodes	Dropout rate	2nd Layer nodes	Types of Features	Training		Test	
				Accuracy (+/-)	Loss (+/-)	Accuracy (+/-)	Loss (+/-)
200	0.1	50	PureGyr	100.000% (+/-0.000)	0.431% (+/-0.033)	95.417% (+/-0.340)	13.945% (+/-0.443)
150	0.2	50	PureGyr	100.000% (+/-0.000)	0.561% (+/-0.071)	95.556% (+/-0.520)	14.302% (+/-1.833)
200	0.2	100	Fingers	99.845% (+/-0.219)	0.687% (+/-0.750)	99.167% (+/-0.340)	2.295% (+/-0.586)
200	0.1	50	AccMag	96.516% (+/-0.810)	8.761% (+/-2.562)	99.444% (+/-0.520)	2.586% (+/-0.789)

Table 4.3: Optimal Experiments of CNN-LSTM model for Stationary Gestures

As we can observe from Table 6.3 given in Appendix, 56 experiments with varied model configuration of CNN-LSTM model and type of features have been implemented to find the closest optimal model for Stationary gestures. Just as we did before, we have sorted the table as given in Table 4.3 based on the training accuracy and test accuracy to find the best of model even though overall the difference in each model gives us substantial appropriate

results. After consideration, we pick the configuration of 200 neurons on the 1st LSTM input layer, a dropout rate of 0.1 and 200 neurons on the second hidden dense layer while the time-distributed wrapper remains to be defined as the same basic configuration. This model was able to give us great results with all the features of the data set with training and test accuracy of collectively 100% and loss of again less than 1%. In this case, we didn't have to fine tune the configuration like the LSTM model we had created previously for movable. Lets try to analyze more into this suitable model using the confusion matrix and training process.

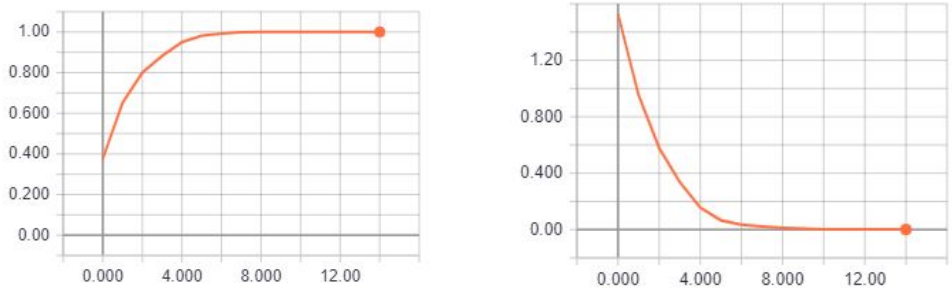


(a) (b)

Figure 4.5: Confusion Matrix of the CNN-LSTM model for Stationary Gestures (a) w/o normalizing (b) w/ normalizing

According to Figure 4.5, even though the accuracy and losses of the training and test process were great, the confusion matrix seems to still show us that the predictions were not absolute yet. The stationary gesture alphabet sign for 'c' and for 'd' still aren't possible to differentiate from each other

with the classification. Even though the alphabet sign for 'c' has a different orientation compared to the others, the model seems to be unable to classify the comparison with the alphabet sign for 'd' accurately. The Figure 4.6 shows us exponentially substantial results giving us what we need. Thus, we decided to not go along with this model for stationary gestures as the LSTM model gave us better results overall.



(a) Epochs Vs. Training Accuracy

(b) Epochs Vs. Training Loss

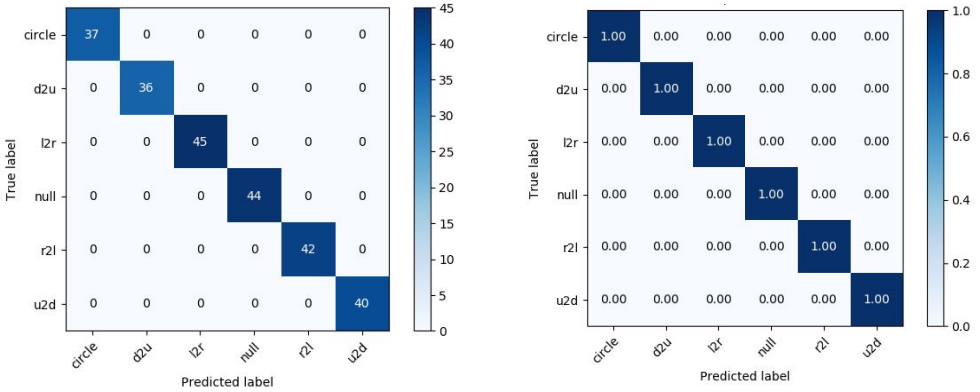
Figure 4.6: Training performance of the CNN-LSTM model for Stationary Gestures

4.2.4 Experiments for Movable data sets in CNN-LSTM

1st Layer nodes	Dropout rate	2nd Layer nodes	Types of Features	Training		Test	
				Accuracy (+/-)	Loss (+/-)	Accuracy (+/-)	Loss (+/-)
200	0.1	100	All	100.000% (+/-0.000)	0.035% (+/-0.003)	100.000% (+/-0.000)	0.038% (+/-0.012)
200	0.1	50	All	100.000% (+/-0.000)	0.049% (+/-0.012)	100.000% (+/-0.000)	0.039% (+/-0.013)
200	0.1	100	All	100.000% (+/-0.000)	0.035% (+/-0.003)	100.000% (+/-0.000)	0.038% (+/-0.012)
200	0.1	50	All	100.000% (+/-0.000)	0.049% (+/-0.012)	100.000% (+/-0.000)	0.039% (+/-0.013)

Table 4.4: Optimal Experiments of CNN-LSTM model for Movable Gestures

In Table 6.4 given in Appendix, we have 56 experiments with varied model configuration of the CNN-LSTM model and all categories of features have been implemented to find the optimal model for movable gestures. As given in Table 4.4, we sort the table based on the training accuracy and test accuracy, the best model has the configuration of 200 neurons on the 1st LSTM input layer, a dropout rate of 0.1, 50 neurons in the second hidden dense layer and training purely with only the Gyroscope readings of the data-format. Unlike the other experiments observed before, this particular model gave us great results similar to the LSTM model. Lets look more into detail about the training process and confusion matrix.

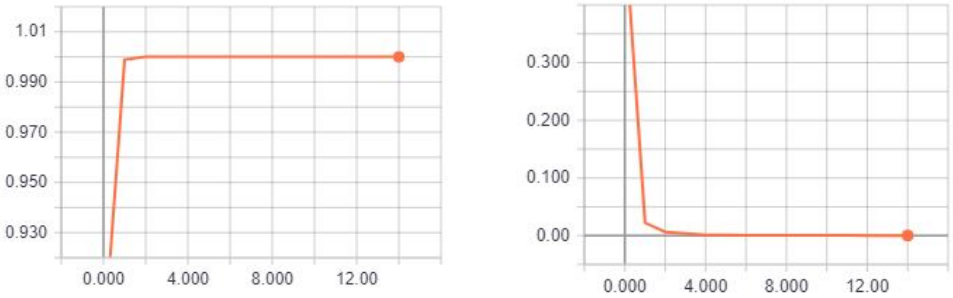


(a) (b)

Figure 4.7: Confusion Matrix of the CNN-LSTM model for Movable Gestures (a) w/o normalizing (b) w/ normalizing

According to Figure 4.8, we get linear accuracy and loss after epoch 4 which doesn't get affected even at epoch 15. Figure 4.7 gives us confusion matrix of the described model for the predicted labels and true labels of the

test data set. Judging from the results, we can conclude that CNN-LSTM also has favorable results with movable gestures. Thus, we decide to test both the LSTM and CNN-LSTM models with our real-time predictions using our application developed for the server.



(a) Epochs Vs. Training Accuracy

(b) Epochs Vs. Training Loss

Figure 4.8: Training performance of the CNN-LSTM model for Movable Gestures

Concluding our analysis, we have seen that sensor fusion or noise filters affected the performance with and without neural networks drastically that we decided to ignore any pre-processing of the data sets and it was more favorable to train the neural networks with completely raw data sets of different features. For the stationary gestures, there was only one option to pick from i.e. the LSTM model. Regardless of training with various categories of the features, we found that using all the features was the optimal solution to predicting our stationary hand-activity gestures. While for movable gestures, we have two options to choose from i.e. LSTM model with all features or the CNN-LSTM model with only Gyroscope features. Clearly, there has been

an impact of categorizing our features and training with fewer features still gave us same substantial results in terms of CNN-LSTM model thus picking it for predicting our movable hand-activity gestures. Lets conclude in the next chapter how the whole system is evaluated.

Chapter 5

Conclusion

In this paper, we implemented an empirical study about the hands-on inertial sensory modules and data acquisition for our implemented HAR/HGR system as known as the Gauntlet-X1. The block diagram in Figure 3.11 shows us the description of the whole smart glove system with its server.

The glove can be connected via USB to any system that can run the server which is written in Python scripts. Currently the server is running on the system with CPU Intel Core i7-3537U @ 2.5 Ghz. As the glove is sampled at 25Hz, the server has two data buffers that collects the gloves data continuously for buffer sizes of 20 and 40 time steps while the chosen LSTM and CNN-LSTM models have been uploaded to predict the respective data buffers which are full. Since our mobile applications are connected to our server via WLAN network, the server searches for incoming clients that have acquired the mobile apps mentioned in Chapter 3. But for the clients

to access the server, the apps are developed with a User Interface(UI) that would request the user to enter the IP address of the server in order to avoid interference of unwanted clients. Once any app gets connected, the predicted gesture of the respective neural network is sent to the client via TCP/IP packets and server ensures acknowledgement.

The conclusion of this thesis can certainly develop a solid foundation for future works in terms of neural networks, pattern/gesture recognition, sensor fusion algorithms, hardware development and firmware development. The next chapter describes the various possibilities where this project could be a starting point of such field of work.

Chapter 6

Future Work

In this chapter we will explain how the product can be further developed in various directions and why it should, as this is just a prototype.

Robust hardware - During the hardware design of the Gauntlet's prototype, we insisted to build it with feasible and easily-available modular components. But in the future by fabricating our own hardware including PCB design and flexible wiring, it is possible to accomplish much more in terms of performance, robustness of the sensors, portability and speed of the device.

GPU - Considering the burden of computation load of the Gauntlet, we may even require a GPU or a robust CPU that can accelerate these computations for further effective data mining and accurate training while saving time and resources. But again derives the question of integrating all these

peripherals into a comfortable glove based device.

Cognitive Language - In this thesis paper, We have only targeted ASL means of communication with the Gauntlet. However, humans have developed various forms of communication despite hearing or visual impairments. This prototype can be used to transform all these means into an intuitive HCI for improved ease of living.

Wireless and Power - As mentioned earlier, the Gauntlet was intended to be wirelessly accessible, as an addition to the USB support. But this thesis aimed to accomplish a proof of concept of the idea; power was one of the major factors the Gauntlet needs to consider. Even using low-power consuming modules, the prototype would have become much bulkier and restrict the ease of movement for distinct gestures. However, if the hardware is designed from scratch, it is a great possibility to add the features of on-board power supply and wireless interface.

Internet-of-Things (IoT) - At the moment, we have targeted two types of mobile applications using AR technology. However, there is much more room for improvement in terms of application such as home automation, CAD designing and other PC applications. With the current design, it is feasible to add more applications based on more trained gestures or pre-processed visualization of the hand.

Intuitive AR Environment - This paper does not dwell further into the world of VR /AR environments as the applications were designed to prove the versatility of the Gauntlet by integrating ML-based gestures with the 3-D world. The possibilities for this kind of interactive device are endless and there is room for further development of applications to make it more realistic and integrated in the day to day life of the user.

User Interface - We want to improve the design of UI for the server and the application side to make it more user friendly. The users will be able to choose from various options which we provide depending on what all applications we have integrated with the product. The user will be able to easily switch between applications with ease.

Regardless, of whether all the features above can be implemented, the results so far still prove that the Gauntlet can be regarded as a proof of concept for the basic underlying ideas.

Bibliography

[1] Ming Zeng, Le T. Nguyen, Bo Yu, Ole J. Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *Proceedings of 6th International Conference on Mobile Computing,Applications and services(MobiCASE)*, pages 198–205, Texas, USA, 2014.

[2] P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Tröster. Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection. *Wireless Sensor Networks*, pages 17–33, 2008.

[3] Ricardo Chavarriaga, Hesam Sagha, Alberto Calatroni, Sundaratejaswi Digumarti, Gerhard Tröster, José del R. Millán, and Daniel Roggen. The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15):2033–2042, 2013.

[4] J. W. Lockhart, G. M. Weiss, J. C. Xue, S. T. Gallagher, A. B. Grosner, and T. T. Pulickal. Design considerations for the wisdm smart

phone-based sensor mining architecture. In *Proceedings of Fifth International Workshop on Knowledge Discovery from Sensor Data*, pages 25–33, 2011.

[5] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 3995–4001, Buenos Aires, Argentina, 2015.

[6] E.Valarezo, P.Rivera, J.M.Park, G.Gi, T.Y.Kim, M.A.Al-Antari, M.Al-Masni, and T.S. Kim. Human activity recognition using a single wrist imu sensor via deep learning convolutional and recurrent neural nets. *Journal of ICT, Design, Engineering and Technological Science*, 01(01):1–5, 2017.

[7] A. Reiss and D. Stricker. Introducing a new benchmarked dataset for activity monitoring. In *Proceedings of IEEE 16th International Symposium Wearable Computers (ISWC)*, pages 108–109, New York, USA, 2012.

[8] Patricio Rivera, Edwin Valarezo, Mun-Taek Choi, and Tae-Seong Kim. Recognition of human hand activities based on a single wrist imu using recurrent neural networks. *International Journal of Pharma Medicine and Biological Sciences*, 06(04):114–118, 2017.

[9] Sojeong Ha, Jeong-Min Yun, and Seungjin Choi. Multi-modal convolutional neural networks for activity recognition. In *Proceedings of IEEE*

International Conference on Systems, Man, and Cybernetics, pages 3017–3022, Hong Kong, China, 2015.

[10] Seungeun Chung, Jiyoun Lim, Kyoung Ju Noh, Gague Kim, and Hyuntae Jeong. Sensor data acquisition and multimodal sensor fusion for human activity recognition using deep learning. *MDPI Sensors Journal*, 19(1716):1–20, 2019.

[11] Jian Wu, Lu Sun, and Roozbeh Jafari. A wearable system for recognizing american sign language in real-time using imu and surface emg sensors. *IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS*, 20(05):1281–1290, 2016.

[12] J. R. Quinlan. *C4. 5: Programs for Machine Learning*. Amsterdam, Netherlands, 2014.

[13] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol*, 2:1–27, 2011.

[14] Edward Nelson Henderson. An inertial measurement system for hand and finger tracking. 2011.

[15] Yang Zhang, Yunfeng Fei, Lin Xu, and Guangyi Sun. Micro-imu-based motion tracking system for virtual training. July 2015.

[16] Stelian-Florin Persa. Sensor fusion in head pose tracking for augmented reality. Jan 2006.

[17] InvenSense Inc. *MPU-6000 and MPU-6050 Product Specification*.

[18] Jason Brownlee. How to develop rnn models for human activity recognition time series classification. *Deep Learning for Time Series*, November 2018.

[19] Unity Technologies. Unity for all.

[20] PTC. Vuforia engine developer portal.

APPENDIX

1st Layer nodes	Dropout rate	2nd Layer nodes	Types of Features	Training		Test	
				Accuracy (+/-)	Loss (+/-)	Accuracy (+/-)	Loss (+/-)
400	0.25	200	AccMag	97.933% (+/-0.659)	5.014% (+/-1.329)	98.167% (+/-0.624)	3.992% (+/-0.665)
400	0.5	200	AccMag	97.213% (+/-0.413)	7.796% (+/-1.396)	97.778% (+/-1.094)	5.980% (+/-4.267)
400	0.5	150	AccMag	94.619% (+/-1.697)	15.746% (+/-6.172)	92.500% (+/-7.273)	18.913% (+/-12.067)
400	0.25	150	AccMag	97.019% (+/-1.835)	10.357% (+/-8.344)	95.417% (+/-5.347)	10.788% (+/-11.362)
350	0.25	150	AccMag	97.135% (+/-1.616)	8.443% (+/-4.296)	97.639% (+/-1.874)	5.762% (+/-4.762)
350	0.25	200	AccMag	98.568% (+/-0.761)	4.158% (+/-1.364)	97.778% (+/-1.712)	4.384% (+/-2.791)
350	0.5	200	AccMag	96.283% (+/-0.435)	10.397% (+/-1.676)	98.194% (+/-0.196)	4.727% (+/-0.956)
350	0.5	150	AccMag	96.129% (+/-0.646)	12.015% (+/-0.932)	97.917% (+/-0.900)	6.557% (+/-2.114)
400	0.25	200	All	98.908% (+/-1.342)	3.407% (+/-3.931)	99.000% (+/-1.253)	3.290% (+/-4.399)
400	0.5	200	All	99.148% (+/-1.205)	2.447% (+/-2.752)	99.306% (+/-0.196)	2.523% (+/-1.087)
400	0.5	150	All	98.606% (+/-0.528)	4.493% (+/-1.572)	99.028% (+/-0.196)	2.374% (+/-0.446)
400	0.25	150	All	99.032% (+/-0.333)	2.898% (+/-0.959)	98.611% (+/-0.786)	4.264% (+/-2.379)
350	0.25	150	All	99.884% (+/-0.164)	0.919% (+/-0.483)	98.472% (+/-0.520)	3.607% (+/-0.956)
350	0.25	200	All	99.613% (+/-0.468)	1.443% (+/-1.611)	99.167% (+/-0.340)	2.038% (+/-0.478)
350	0.5	200	All	97.909% (+/-1.094)	5.949% (+/-3.143)	94.306% (+/-1.934)	17.348% (+/-8.217)
350	0.5	150	All	99.690% (+/-0.438)	1.054% (+/-0.719)	99.167% (+/-0.900)	3.311% (+/-3.302)
400	0.25	200	Fingers	99.977% (+/-0.046)	0.060% (+/-0.100)	98.583% (+/-0.204)	4.592% (+/-0.989)
400	0.5	200	Fingers	99.923% (+/-0.055)	0.354% (+/-0.178)	96.389% (+/-2.214)	11.807% (+/-8.406)
400	0.5	150	Fingers	99.071% (+/-0.843)	3.214% (+/-2.499)	96.667% (+/-1.701)	11.842% (+/-5.643)
400	0.25	150	Fingers	99.187% (+/-0.251)	2.686% (+/-1.299)	98.472% (+/-0.520)	4.320% (+/-0.717)
350	0.25	150	Fingers	99.729% (+/-0.383)	0.694% (+/-0.764)	99.028% (+/-0.196)	4.744% (+/-0.145)
350	0.25	200	Fingers	99.690% (+/-0.290)	1.199% (+/-0.836)	98.194% (+/-1.094)	4.297% (+/-2.520)
350	0.5	200	Fingers	99.458% (+/-0.383)	2.215% (+/-1.338)	99.167% (+/-0.340)	2.790% (+/-0.527)
350	0.5	150	Fingers	99.729% (+/-0.219)	1.740% (+/-0.792)	99.028% (+/-0.520)	3.425% (+/-2.131)
400	0.25	200	GyrMag	98.118% (+/-1.275)	5.902% (+/-4.678)	94.667% (+/-1.654)	18.886% (+/-2.922)
400	0.5	200	GyrMag	94.774% (+/-0.905)	15.671% (+/-2.663)	95.833% (+/-0.340)	11.144% (+/-0.507)
400	0.5	150	GyrMag	93.302% (+/-1.724)	19.180% (+/-4.556)	95.972% (+/-0.708)	12.019% (+/-2.598)
400	0.25	150	GyrMag	96.554% (+/-0.239)	11.073% (+/-1.509)	97.083% (+/-0.680)	8.312% (+/-1.506)
350	0.25	150	GyrMag	96.167% (+/-0.577)	12.653% (+/-1.875)	95.694% (+/-0.982)	11.118% (+/-1.536)
350	0.25	200	GyrMag	95.935% (+/-2.018)	11.309% (+/-5.079)	92.778% (+/-3.161)	20.504% (+/-6.068)
350	0.5	200	GyrMag	94.580% (+/-0.673)	16.096% (+/-2.676)	95.417% (+/-2.381)	15.496% (+/-6.042)
350	0.5	150	GyrMag	93.457% (+/-1.057)	17.986% (+/-2.637)	93.333% (+/-2.381)	19.379% (+/-6.994)
400	0.25	200	Palm	92.544% (+/-0.945)	19.923% (+/-2.224)	92.667% (+/-1.700)	18.222% (+/-2.212)
400	0.5	200	Palm	85.134% (+/-2.137)	35.236% (+/-2.999)	82.778% (+/-1.094)	34.383% (+/-1.444)
400	0.5	150	Palm	85.288% (+/-2.475)	36.274% (+/-4.834)	88.889% (+/-1.288)	29.540% (+/-2.986)
400	0.25	150	Palm	89.857% (+/-1.178)	26.426% (+/-2.034)	89.028% (+/-3.233)	27.442% (+/-9.100)
350	0.25	150	Palm	88.424% (+/-1.219)	30.119% (+/-2.580)	88.889% (+/-2.643)	30.379% (+/-6.865)
350	0.25	200	Palm	88.386% (+/-1.397)	28.473% (+/-3.768)	91.389% (+/-2.390)	21.663% (+/-3.692)
350	0.5	200	Palm	86.101% (+/-2.726)	33.057% (+/-6.004)	90.000% (+/-3.281)	25.679% (+/-2.685)
350	0.5	150	Palm	82.733% (+/-1.449)	39.304% (+/-2.303)	85.972% (+/-3.161)	36.441% (+/-0.990)
400	0.25	200	PureAcc	95.889% (+/-1.030)	10.060% (+/-2.620)	95.250% (+/-1.253)	9.887% (+/-2.228)
400	0.5	200	PureAcc	96.129% (+/-1.143)	9.615% (+/-1.465)	96.111% (+/-0.708)	10.934% (+/-1.095)
400	0.5	150	PureAcc	93.225% (+/-2.048)	16.187% (+/-5.467)	96.389% (+/-1.416)	7.285% (+/-1.909)
400	0.25	150	PureAcc	96.129% (+/-1.332)	10.479% (+/-3.583)	94.722% (+/-1.416)	9.480% (+/-3.488)
350	0.25	150	PureAcc	96.632% (+/-0.684)	8.868% (+/-1.842)	97.222% (+/-0.196)	6.937% (+/-0.840)
350	0.25	200	PureAcc	95.432% (+/-1.765)	12.963% (+/-5.648)	97.639% (+/-1.039)	7.335% (+/-2.591)
350	0.5	200	PureAcc	95.703% (+/-0.435)	9.223% (+/-0.619)	95.694% (+/-1.375)	11.259% (+/-5.279)
350	0.5	150	PureAcc	96.090% (+/-0.610)	9.831% (+/-0.796)	97.222% (+/-0.520)	7.286% (+/-0.889)
400	0.25	200	PureGyr	99.977% (+/-0.046)	0.139% (+/-0.213)	98.667% (+/-0.408)	6.081% (+/-2.523)
400	0.5	200	PureGyr	98.722% (+/-0.990)	4.239% (+/-3.170)	96.528% (+/-0.520)	11.591% (+/-2.227)
400	0.5	150	PureGyr	96.864% (+/-1.710)	10.581% (+/-5.575)	94.444% (+/-2.553)	15.804% (+/-6.510)
400	0.25	150	PureGyr	98.800% (+/-0.477)	3.765% (+/-1.220)	97.083% (+/-1.179)	7.920% (+/-1.655)
350	0.25	150	PureGyr	97.677% (+/-1.731)	8.224% (+/-6.521)	95.417% (+/-1.179)	17.868% (+/-5.072)
350	0.25	200	PureGyr	98.451% (+/-1.452)	6.172% (+/-6.014)	96.667% (+/-1.800)	10.601% (+/-4.651)
350	0.5	200	PureGyr	96.477% (+/-2.689)	11.188% (+/-9.045)	93.472% (+/-2.750)	17.929% (+/-7.143)
350	0.5	150	PureGyr	98.722% (+/-0.251)	4.665% (+/-1.126)	95.833% (+/-2.453)	12.446% (+/-5.728)

Table 6.1: Experiments on LSTM model for Stationary Gestures

1st Layer nodes	Dropout rate	2nd Layer nodes	Types of Features	Training		Test	
				Accuracy (+/-)	Loss (+/-)	Accuracy (+/-)	Loss (+/-)
400	0.25	200	AccMag	97.836% (+/-0.827)	5.600% (+/-1.790)	98.770% (+/-0.885)	3.304% (+/-2.061)
400	0.25	150	AccMag	98.633% (+/-0.887)	3.547% (+/-1.988)	96.995% (+/-0.697)	7.920% (+/-0.683)
400	0.5	200	AccMag	97.722% (+/-0.492)	6.141% (+/-1.284)	96.995% (+/-1.961)	9.554% (+/-6.197)
400	0.5	150	AccMag	97.380% (+/-0.279)	7.326% (+/-0.562)	95.082% (+/-1.673)	10.931% (+/-2.533)
350	0.25	200	AccMag	98.595% (+/-0.685)	3.606% (+/-1.455)	96.858% (+/-1.843)	8.649% (+/-5.058)
350	0.25	150	AccMag	98.064% (+/-0.492)	4.860% (+/-0.733)	97.951% (+/-0.885)	6.139% (+/-2.477)
350	0.5	200	AccMag	98.140% (+/-0.376)	5.064% (+/-0.890)	97.951% (+/-0.580)	5.104% (+/-1.350)
350	0.5	150	AccMag	98.368% (+/-0.352)	5.073% (+/-1.107)	97.268% (+/-0.193)	7.077% (+/-1.662)
400	0.25	200	All	100.000% (+/-0.000)	0.037% (+/-0.001)	100.000% (+/-0.000)	0.185% (+/-0.035)
400	0.25	150	All	100.000% (+/-0.000)	0.038% (+/-0.002)	99.863% (+/-0.193)	1.216% (+/-0.083)
400	0.5	200	All	100.000% (+/-0.000)	0.071% (+/-0.001)	100.000% (+/-0.000)	0.106% (+/-0.014)
400	0.5	150	All	100.000% (+/-0.000)	0.093% (+/-0.002)	100.000% (+/-0.000)	0.079% (+/-0.024)
350	0.25	200	All	100.000% (+/-0.000)	0.047% (+/-0.003)	100.000% (+/-0.000)	0.271% (+/-0.120)
350	0.25	150	All	100.000% (+/-0.000)	0.061% (+/-0.001)	100.000% (+/-0.000)	0.306% (+/-0.062)
350	0.5	200	All	100.000% (+/-0.000)	0.101% (+/-0.011)	100.000% (+/-0.000)	0.117% (+/-0.014)
350	0.5	150	All	100.000% (+/-0.000)	0.113% (+/-0.002)	100.000% (+/-0.000)	0.656% (+/-0.075)
400	0.25	200	Fingers	100.000% (+/-0.000)	0.151% (+/-0.048)	99.454% (+/-0.193)	3.174% (+/-0.423)
400	0.25	150	Fingers	100.000% (+/-0.000)	0.120% (+/-0.048)	99.727% (+/-0.386)	0.879% (+/-0.472)
400	0.5	200	Fingers	99.696% (+/-0.107)	0.944% (+/-0.072)	99.180% (+/-0.000)	6.278% (+/-2.541)
400	0.5	150	Fingers	99.734% (+/-0.142)	0.853% (+/-0.186)	98.770% (+/-0.335)	2.900% (+/-0.872)
350	0.25	200	Fingers	100.000% (+/-0.000)	0.098% (+/-0.008)	98.497% (+/-0.386)	4.652% (+/-0.559)
350	0.25	150	Fingers	100.000% (+/-0.000)	0.103% (+/-0.011)	98.770% (+/-0.000)	6.538% (+/-0.889)
350	0.5	200	Fingers	99.848% (+/-0.142)	0.671% (+/-0.110)	97.951% (+/-1.533)	4.569% (+/-2.628)
350	0.5	150	Fingers	99.696% (+/-0.194)	1.014% (+/-0.419)	98.087% (+/-0.193)	15.866% (+/-3.211)
400	0.25	200	GyrMag	100.000% (+/-0.000)	0.054% (+/-0.009)	99.863% (+/-0.193)	0.386% (+/-0.214)
400	0.25	150	GyrMag	100.000% (+/-0.000)	0.060% (+/-0.010)	100.000% (+/-0.000)	0.471% (+/-0.144)
400	0.5	200	GyrMag	100.000% (+/-0.000)	0.089% (+/-0.010)	99.317% (+/-0.386)	1.666% (+/-0.777)
400	0.5	150	GyrMag	99.962% (+/-0.054)	0.192% (+/-0.082)	100.000% (+/-0.000)	0.140% (+/-0.051)
350	0.25	200	GyrMag	100.000% (+/-0.000)	0.069% (+/-0.008)	99.590% (+/-0.000)	0.761% (+/-0.176)
350	0.25	150	GyrMag	100.000% (+/-0.000)	0.077% (+/-0.008)	100.000% (+/-0.000)	0.299% (+/-0.197)
350	0.5	200	GyrMag	100.000% (+/-0.000)	0.154% (+/-0.041)	100.000% (+/-0.000)	0.140% (+/-0.033)
350	0.5	150	GyrMag	100.000% (+/-0.000)	0.119% (+/-0.016)	99.590% (+/-0.000)	1.084% (+/-0.515)
400	0.25	200	Palm	98.861% (+/-0.186)	3.171% (+/-0.529)	99.044% (+/-0.697)	2.212% (+/-1.378)
400	0.25	150	Palm	99.013% (+/-0.376)	3.036% (+/-0.833)	98.634% (+/-0.511)	2.906% (+/-0.498)
400	0.5	200	Palm	97.570% (+/-1.688)	5.759% (+/-2.960)	97.814% (+/-0.697)	4.460% (+/-1.324)
400	0.5	150	Palm	98.823% (+/-0.194)	3.146% (+/-0.315)	99.180% (+/-0.335)	2.160% (+/-1.008)
350	0.25	200	Palm	98.330% (+/-0.619)	4.355% (+/-1.478)	98.497% (+/-0.193)	3.953% (+/-0.716)
350	0.25	150	Palm	98.140% (+/-0.284)	3.903% (+/-0.830)	98.497% (+/-0.193)	3.569% (+/-1.054)
350	0.5	200	Palm	98.785% (+/-0.439)	4.095% (+/-1.657)	97.951% (+/-0.335)	5.993% (+/-1.981)
350	0.5	150	Palm	97.988% (+/-0.268)	5.661% (+/-1.231)	98.770% (+/-0.580)	4.077% (+/-1.960)
400	0.25	200	PureAcc	97.798% (+/-0.194)	5.191% (+/-0.415)	96.858% (+/-1.022)	6.173% (+/-2.567)
400	0.25	150	PureAcc	97.836% (+/-0.161)	6.371% (+/-0.595)	96.585% (+/-2.534)	9.890% (+/-6.612)
400	0.5	200	PureAcc	97.874% (+/-0.299)	5.936% (+/-0.137)	97.678% (+/-0.697)	5.664% (+/-1.934)
400	0.5	150	PureAcc	96.507% (+/-0.757)	8.546% (+/-1.979)	97.678% (+/-1.843)	6.337% (+/-2.737)
350	0.25	200	PureAcc	97.950% (+/-0.405)	4.976% (+/-0.961)	93.033% (+/-2.035)	20.456% (+/-7.335)
350	0.25	150	PureAcc	97.608% (+/-0.566)	6.007% (+/-0.584)	99.044% (+/-0.386)	3.075% (+/-0.415)
350	0.5	200	PureAcc	96.241% (+/-0.161)	8.424% (+/-0.214)	98.770% (+/-0.669)	3.781% (+/-0.919)
350	0.5	150	PureAcc	96.887% (+/-0.512)	7.376% (+/-1.152)	98.907% (+/-0.386)	3.755% (+/-0.277)
400	0.25	200	PureGyr	99.582% (+/-0.512)	2.208% (+/-2.667)	98.087% (+/-0.193)	8.485% (+/-1.496)
400	0.25	150	PureGyr	99.848% (+/-0.142)	0.667% (+/-0.492)	98.497% (+/-0.193)	7.148% (+/-0.795)
400	0.5	200	PureGyr	99.317% (+/-0.405)	2.180% (+/-0.989)	97.404% (+/-0.697)	12.379% (+/-5.643)
400	0.5	150	PureGyr	99.506% (+/-0.299)	1.769% (+/-0.673)	98.907% (+/-0.511)	6.075% (+/-3.000)
350	0.25	200	PureGyr	99.734% (+/-0.194)	0.740% (+/-0.335)	98.087% (+/-0.386)	6.444% (+/-0.740)
350	0.25	150	PureGyr	99.924% (+/-0.107)	0.483% (+/-0.306)	96.995% (+/-0.386)	8.669% (+/-0.725)
350	0.5	200	PureGyr	99.355% (+/-0.561)	2.857% (+/-2.200)	96.995% (+/-1.932)	14.517% (+/-5.794)
350	0.5	150	PureGyr	99.317% (+/-0.093)	2.006% (+/-0.197)	97.814% (+/-0.193)	7.445% (+/-0.921)

Table 6.2: Experiments on LSTM for Movable Gestures

1st Layer nodes	Dropout rate	2nd Layer nodes	Types of Features	Training		Test	
				Accuracy (+/-)	Loss (+/-)	Accuracy (+/-)	Loss (+/-)
200	0.1	100	AccMag	97.716% (+/-0.772)	6.119% (+/-1.722)	98.750% (+/-0.340)	3.977% (+/-0.321)
200	0.1	50	AccMag	96.516% (+/-0.810)	8.761% (+/-2.562)	99.444% (+/-0.520)	2.586% (+/-0.789)
200	0.2	100	AccMag	96.554% (+/-1.291)	7.639% (+/-2.864)	97.917% (+/-0.900)	5.718% (+/-2.092)
200	0.2	50	AccMag	97.561% (+/-0.962)	6.169% (+/-1.763)	98.056% (+/-1.195)	5.457% (+/-2.902)
150	0.1	100	AccMag	97.445% (+/-0.190)	7.450% (+/-0.926)	98.333% (+/-0.589)	4.600% (+/-1.313)
150	0.1	50	AccMag	96.748% (+/-0.664)	9.690% (+/-2.226)	95.833% (+/-4.139)	10.394% (+/-7.285)
150	0.2	100	AccMag	96.554% (+/-1.793)	9.163% (+/-4.457)	98.611% (+/-0.856)	5.006% (+/-1.723)
150	0.2	50	AccMag	97.600% (+/-0.631)	6.536% (+/-1.482)	97.639% (+/-2.161)	5.796% (+/-3.891)
200	0.1	100	All	99.032% (+/-0.522)	2.857% (+/-0.995)	97.778% (+/-1.375)	7.026% (+/-5.557)
200	0.1	50	All	99.187% (+/-0.379)	2.528% (+/-1.152)	99.167% (+/-0.340)	1.967% (+/-0.602)
200	0.2	100	All	98.684% (+/-0.646)	4.123% (+/-1.876)	98.611% (+/-0.520)	3.700% (+/-0.162)
200	0.2	50	All	97.909% (+/-1.643)	5.774% (+/-4.627)	96.111% (+/-1.094)	11.494% (+/-2.993)
150	0.1	100	All	98.026% (+/-0.435)	5.660% (+/-1.435)	97.778% (+/-0.520)	4.755% (+/-0.998)
150	0.1	50	All	98.645% (+/-0.772)	3.649% (+/-1.604)	98.472% (+/-0.196)	6.221% (+/-1.391)
150	0.2	100	All	97.948% (+/-1.424)	7.057% (+/-4.287)	97.917% (+/-0.589)	7.295% (+/-3.351)
150	0.2	50	All	98.064% (+/-0.860)	5.690% (+/-2.477)	98.056% (+/-0.196)	5.214% (+/-1.420)
200	0.1	100	Fingers	99.961% (+/-0.055)	0.201% (+/-0.060)	98.194% (+/-0.520)	6.174% (+/-0.982)
200	0.1	50	Fingers	99.961% (+/-0.055)	0.197% (+/-0.037)	99.028% (+/-0.196)	2.652% (+/-0.748)
200	0.2	100	Fingers	99.845% (+/-0.219)	0.687% (+/-0.750)	99.167% (+/-0.340)	2.295% (+/-0.586)
200	0.2	50	Fingers	99.961% (+/-0.055)	0.336% (+/-0.169)	97.778% (+/-0.520)	5.430% (+/-1.186)
150	0.1	100	Fingers	99.884% (+/-0.095)	0.440% (+/-0.236)	98.889% (+/-0.196)	3.593% (+/-1.399)
150	0.1	50	Fingers	99.884% (+/-0.000)	0.503% (+/-0.053)	99.167% (+/-0.000)	1.658% (+/-0.153)
150	0.2	100	Fingers	99.923% (+/-0.110)	0.323% (+/-0.189)	98.194% (+/-0.393)	4.289% (+/-1.196)
150	0.2	50	Fingers	99.961% (+/-0.055)	0.242% (+/-0.103)	97.917% (+/-0.680)	7.781% (+/-1.396)
200	0.1	100	GyrMag	79.017% (+/-3.416)	47.753% (+/-4.597)	77.917% (+/-3.062)	51.003% (+/-5.965)
200	0.1	50	GyrMag	80.062% (+/-0.602)	45.917% (+/-1.202)	79.861% (+/-4.281)	44.527% (+/-5.882)
200	0.2	100	GyrMag	76.500% (+/-3.047)	52.267% (+/-4.807)	75.000% (+/-3.923)	53.140% (+/-5.942)
200	0.2	50	GyrMag	76.345% (+/-2.520)	53.225% (+/-4.210)	78.056% (+/-5.208)	49.747% (+/-7.347)
150	0.1	100	GyrMag	79.133% (+/-0.477)	47.196% (+/-1.363)	75.833% (+/-5.475)	57.190% (+/-13.793)
150	0.1	50	GyrMag	78.591% (+/-0.855)	50.549% (+/-1.351)	77.500% (+/-2.657)	53.340% (+/-7.302)
150	0.2	100	GyrMag	73.597% (+/-3.177)	59.576% (+/-6.100)	77.083% (+/-4.181)	53.680% (+/-8.585)
150	0.2	50	GyrMag	72.125% (+/-2.894)	62.010% (+/-6.058)	75.972% (+/-3.408)	53.231% (+/-4.685)
200	0.1	100	Palm	85.947% (+/-3.212)	31.852% (+/-5.201)	86.667% (+/-3.552)	30.466% (+/-5.101)
200	0.1	50	Palm	85.676% (+/-2.247)	34.238% (+/-4.577)	85.972% (+/-1.712)	33.289% (+/-4.562)
200	0.2	100	Palm	86.450% (+/-3.630)	31.295% (+/-5.932)	85.000% (+/-2.381)	34.918% (+/-4.297)
200	0.2	50	Palm	84.243% (+/-2.121)	37.548% (+/-2.856)	85.000% (+/-1.701)	34.232% (+/-1.365)
150	0.1	100	Palm	86.334% (+/-1.170)	31.992% (+/-0.857)	87.917% (+/-1.227)	29.027% (+/-4.034)
150	0.1	50	Palm	85.211% (+/-2.350)	33.490% (+/-2.645)	80.972% (+/-1.416)	38.866% (+/-2.479)
150	0.2	100	Palm	82.733% (+/-0.881)	39.593% (+/-1.792)	85.833% (+/-0.900)	35.927% (+/-0.685)
150	0.2	50	Palm	85.366% (+/-3.169)	34.778% (+/-5.510)	88.056% (+/-4.388)	27.631% (+/-6.087)
200	0.1	100	PureAcc	95.819% (+/-1.077)	9.896% (+/-2.279)	94.444% (+/-3.087)	12.456% (+/-5.328)
200	0.1	50	PureAcc	96.748% (+/-0.622)	7.281% (+/-0.992)	97.361% (+/-1.678)	7.748% (+/-2.457)
200	0.2	100	PureAcc	97.174% (+/-0.712)	7.284% (+/-1.068)	95.833% (+/-0.340)	9.005% (+/-0.495)
200	0.2	50	PureAcc	95.548% (+/-0.438)	9.920% (+/-1.076)	98.611% (+/-0.196)	5.642% (+/-0.618)
150	0.1	100	PureAcc	96.516% (+/-0.095)	6.881% (+/-0.213)	97.500% (+/-0.340)	6.900% (+/-0.733)
150	0.1	50	PureAcc	96.283% (+/-1.150)	8.505% (+/-2.313)	96.806% (+/-0.196)	9.338% (+/-0.865)
150	0.2	100	PureAcc	96.748% (+/-0.342)	8.077% (+/-0.974)	96.250% (+/-1.559)	7.758% (+/-2.321)
150	0.2	50	PureAcc	95.974% (+/-0.428)	9.547% (+/-1.614)	95.833% (+/-0.589)	11.606% (+/-2.103)
200	0.1	100	PureGyr	99.884% (+/-0.095)	0.652% (+/-0.244)	95.417% (+/-0.680)	18.150% (+/-4.154)
200	0.1	50	PureGyr	100.000% (+/-0.000)	0.431% (+/-0.033)	95.417% (+/-0.340)	13.945% (+/-0.443)
200	0.2	100	PureGyr	99.497% (+/-0.290)	2.152% (+/-1.346)	95.833% (+/-1.483)	14.690% (+/-2.833)
200	0.2	50	PureGyr	99.961% (+/-0.055)	0.536% (+/-0.173)	93.611% (+/-0.520)	18.782% (+/-2.397)
150	0.1	100	PureGyr	99.845% (+/-0.219)	0.993% (+/-0.599)	93.611% (+/-0.520)	28.009% (+/-1.330)
150	0.1	50	PureGyr	99.768% (+/-0.164)	1.110% (+/-0.225)	94.444% (+/-0.520)	15.761% (+/-2.171)
150	0.2	100	PureGyr	99.226% (+/-1.014)	2.419% (+/-2.549)	93.750% (+/-2.357)	20.315% (+/-6.176)
150	0.2	50	PureGyr	100.000% (+/-0.000)	0.561% (+/-0.071)	95.556% (+/-0.520)	14.302% (+/-1.833)

Table 6.3: Experiments on CNN-LSTM for Stationary Gestures

1st Layer nodes	Dropout rate	2nd Layer nodes	Types of Features	Training		Test	
				Accuracy (+/-)	Loss (+/-)	Accuracy (+/-)	Loss (+/-)
200	0.1	100	AccMag	97.532% (+/-1.249)	7.024% (+/-2.558)	96.038% (+/-1.581)	10.522% (+/-2.706)
200	0.1	50	AccMag	98.747% (+/-0.335)	3.511% (+/-0.802)	98.634% (+/-0.193)	4.397% (+/-0.642)
200	0.2	100	AccMag	98.709% (+/-0.268)	4.056% (+/-0.784)	97.541% (+/-2.090)	6.132% (+/-3.105)
200	0.2	50	AccMag	98.557% (+/-0.352)	4.365% (+/-1.244)	97.678% (+/-0.697)	5.690% (+/-1.873)
150	0.1	100	AccMag	98.064% (+/-0.426)	4.964% (+/-1.273)	98.907% (+/-0.193)	3.666% (+/-1.112)
150	0.1	50	AccMag	97.722% (+/-0.581)	6.836% (+/-0.992)	98.634% (+/-0.193)	3.810% (+/-0.829)
150	0.2	100	AccMag	98.026% (+/-0.598)	5.623% (+/-1.060)	98.361% (+/-0.580)	4.219% (+/-0.785)
150	0.2	50	AccMag	97.722% (+/-0.518)	5.782% (+/-1.527)	97.678% (+/-0.842)	5.001% (+/-1.483)
200	0.1	100	All	100.000% (+/-0.000)	0.035% (+/-0.003)	100.000% (+/-0.000)	0.038% (+/-0.012)
200	0.1	50	All	100.000% (+/-0.000)	0.049% (+/-0.012)	100.000% (+/-0.000)	0.039% (+/-0.013)
200	0.2	100	All	100.000% (+/-0.000)	0.050% (+/-0.014)	100.000% (+/-0.000)	0.140% (+/-0.110)
200	0.2	50	All	100.000% (+/-0.000)	0.090% (+/-0.041)	100.000% (+/-0.000)	0.046% (+/-0.019)
150	0.1	100	All	100.000% (+/-0.000)	0.052% (+/-0.016)	100.000% (+/-0.000)	0.036% (+/-0.007)
150	0.1	50	All	100.000% (+/-0.000)	0.093% (+/-0.015)	100.000% (+/-0.000)	0.198% (+/-0.107)
150	0.2	100	All	100.000% (+/-0.000)	0.075% (+/-0.010)	100.000% (+/-0.000)	0.052% (+/-0.018)
150	0.2	50	All	100.000% (+/-0.000)	0.110% (+/-0.022)	100.000% (+/-0.000)	0.068% (+/-0.020)
200	0.1	100	Fingers	99.962% (+/-0.054)	0.446% (+/-0.036)	99.454% (+/-0.386)	2.027% (+/-1.168)
200	0.1	50	Fingers	99.924% (+/-0.054)	0.459% (+/-0.080)	99.590% (+/-0.000)	1.304% (+/-0.282)
200	0.2	100	Fingers	99.848% (+/-0.142)	0.827% (+/-0.153)	99.454% (+/-0.193)	1.550% (+/-0.547)
200	0.2	50	Fingers	99.468% (+/-0.598)	1.609% (+/-1.224)	99.727% (+/-0.193)	1.440% (+/-0.670)
150	0.1	100	Fingers	99.241% (+/-0.673)	2.355% (+/-1.119)	98.907% (+/-0.842)	2.640% (+/-2.071)
150	0.1	50	Fingers	99.734% (+/-0.234)	1.031% (+/-0.588)	99.863% (+/-0.193)	0.737% (+/-0.349)
150	0.2	100	Fingers	99.658% (+/-0.246)	1.300% (+/-0.302)	99.863% (+/-0.193)	0.760% (+/-0.060)
150	0.2	50	Fingers	99.165% (+/-0.234)	2.937% (+/-0.485)	99.863% (+/-0.193)	1.560% (+/-0.281)
200	0.1	100	GyrMag	100.000% (+/-0.000)	0.110% (+/-0.076)	100.000% (+/-0.000)	0.112% (+/-0.069)
200	0.1	50	GyrMag	100.000% (+/-0.000)	0.057% (+/-0.003)	99.863% (+/-0.193)	0.316% (+/-0.121)
200	0.2	100	GyrMag	100.000% (+/-0.000)	0.058% (+/-0.006)	100.000% (+/-0.000)	0.046% (+/-0.018)
200	0.2	50	GyrMag	100.000% (+/-0.000)	0.088% (+/-0.028)	100.000% (+/-0.000)	0.119% (+/-0.017)
150	0.1	100	GyrMag	100.000% (+/-0.000)	0.066% (+/-0.006)	100.000% (+/-0.000)	0.269% (+/-0.116)
150	0.1	50	GyrMag	100.000% (+/-0.000)	0.093% (+/-0.013)	99.727% (+/-0.193)	0.334% (+/-0.176)
150	0.2	100	GyrMag	100.000% (+/-0.000)	0.178% (+/-0.070)	99.590% (+/-0.335)	0.739% (+/-0.476)
150	0.2	50	GyrMag	100.000% (+/-0.000)	0.211% (+/-0.050)	100.000% (+/-0.000)	0.087% (+/-0.022)
200	0.1	100	Palm	98.747% (+/-0.246)	3.488% (+/-0.812)	97.404% (+/-1.022)	6.314% (+/-1.576)
200	0.1	50	Palm	99.355% (+/-0.194)	1.706% (+/-0.388)	98.361% (+/-1.207)	5.115% (+/-3.210)
200	0.2	100	Palm	99.317% (+/-0.372)	2.160% (+/-0.987)	98.497% (+/-0.193)	2.755% (+/-0.753)
200	0.2	50	Palm	98.595% (+/-0.268)	3.741% (+/-1.041)	98.224% (+/-0.193)	4.201% (+/-0.877)
150	0.1	100	Palm	98.937% (+/-0.215)	3.215% (+/-0.460)	99.044% (+/-0.193)	2.150% (+/-0.423)
150	0.1	50	Palm	98.140% (+/-0.352)	4.984% (+/-0.839)	98.634% (+/-0.511)	4.693% (+/-1.462)
150	0.2	100	Palm	98.519% (+/-0.738)	4.060% (+/-1.584)	97.404% (+/-0.966)	6.059% (+/-1.693)
150	0.2	50	Palm	98.899% (+/-0.376)	3.269% (+/-0.786)	98.087% (+/-0.193)	5.245% (+/-0.371)
200	0.1	100	PureAcc	97.532% (+/-0.545)	5.842% (+/-0.118)	96.858% (+/-1.022)	8.365% (+/-2.946)
200	0.1	50	PureAcc	96.849% (+/-0.468)	7.468% (+/-0.599)	95.765% (+/-1.022)	9.293% (+/-0.861)
200	0.2	100	PureAcc	96.317% (+/-0.879)	8.650% (+/-1.704)	97.131% (+/-0.580)	6.838% (+/-1.390)
200	0.2	50	PureAcc	97.570% (+/-0.512)	6.290% (+/-1.125)	97.814% (+/-0.386)	5.222% (+/-0.606)
150	0.1	100	PureAcc	96.393% (+/-0.954)	8.665% (+/-2.519)	98.361% (+/-0.885)	5.704% (+/-2.040)
150	0.1	50	PureAcc	97.760% (+/-0.459)	5.819% (+/-0.773)	98.224% (+/-1.352)	7.002% (+/-3.952)
150	0.2	100	PureAcc	96.811% (+/-0.246)	7.652% (+/-0.652)	96.448% (+/-0.193)	9.046% (+/-0.455)
150	0.2	50	PureAcc	97.342% (+/-0.268)	7.768% (+/-0.795)	95.902% (+/-1.739)	9.735% (+/-4.514)
200	0.1	100	PureGyr	98.519% (+/-0.335)	4.420% (+/-0.827)	96.858% (+/-1.175)	9.156% (+/-4.906)
200	0.1	50	PureGyr	98.747% (+/-0.161)	4.213% (+/-0.893)	98.224% (+/-0.842)	5.374% (+/-2.190)
200	0.2	100	PureGyr	98.368% (+/-0.545)	5.144% (+/-1.878)	99.044% (+/-0.193)	3.255% (+/-0.307)
200	0.2	50	PureGyr	98.064% (+/-0.335)	6.580% (+/-0.814)	97.268% (+/-1.175)	7.034% (+/-2.481)
150	0.1	100	PureGyr	98.026% (+/-0.234)	5.677% (+/-0.753)	98.634% (+/-0.193)	4.634% (+/-0.957)
150	0.1	50	PureGyr	98.102% (+/-0.299)	5.627% (+/-1.320)	96.585% (+/-1.022)	10.005% (+/-2.105)
150	0.2	100	PureGyr	97.874% (+/-0.568)	6.559% (+/-0.211)	97.131% (+/-0.885)	10.042% (+/-1.792)
150	0.2	50	PureGyr	96.393% (+/-1.263)	9.600% (+/-3.560)	96.038% (+/-0.842)	9.614% (+/-1.383)

Table 6.4: Experiments on CNN-LSTM for Movable Gestures

List of Figures

List of Tables

Acronyms

1-D One-Dimension. 29

2-D Two-Dimension. 4, 5, 16, 32

3-D Three-Dimension. 1, 4, 7, 10, 18, 19, 31, 32, 34, 61

AR Augmented Reality. 1, 6–8, 10, 11, 14, 19, 31–34, 39, 40, 60, 61

ASL American Sign Language. 1, 2, 5, 6, 14, 18, 30, 31, 39, 42, 60, 71

BSN Body Sensor Network. 19

CAD Computer-Aided Design. 60

CNN Convolutional Neural Network. 1, 15, 16, 27, 29, 52, 55, 71, 72

CPU Central Processing Unit. 38, 57, 59

CS Chip Select. 23

DoF Degree-of-Freedom. 24, 25, 71

EKF Extended Kalman Filter. 2, 25, 26, 45

GPS Global Positioning System. 17, 19, 32, 34

GPU Graphical Processing Unit. 59

HAR/HGR Hand Activity/Gesture Recognition. 1, 14, 15, 27, 30, 38, 57

HCI Human Computer Interaction. 1, 4, 7, 60

I2C Inter-Integrated Circuit. 22, 23

IMU Inertial Measurement Units. 1, 5, 6, 12, 14, 16–19, 21, 25, 71

IoT Internet-of-Things. 2, 5, 10, 11, 42, 43, 60

LSTM Long Short-Term Memory. 1, 2, 27–29, 38, 45–55, 57, 67, 68, 71–73

MCU Micro-controller. 22, 23

MEMS Microelectromechanical Systems. 1, 18, 19

ML Machine Learning. 1, 2, 6, 10, 12, 15, 21, 26, 27, 30, 61

NN Neural Network. 1, 10, 11, 15, 26, 27, 36

PCB Printed Circuit Board. 59

RFID Radio Frequency Identification. 14

RNN Recurring Neural Network. 16, 17, 27, 71

sEMG Surface Electromyography. 14

SPI Serial Peripheral Interface. 22, 23

UI User Interface. 11, 32, 42, 58, 61, 72

USB Universal Serial Bus. 23, 57, 60

VR Virtual Reality. 1, 8, 32, 61