



Department of Computer Science

**BSCCS Final Year Project Report
2021-2022**

21CS055

Gestures-based Touch-Free smartwatch development with deep learning

(Volume __ of __)

Student Name : HON, Hing Ting

Student No. : 55776782

Programme Code : BSCEGU3

Supervisor : Dr XU, Weitao

1st Reader : Dr ZHU Kening

2nd Reader : Prof LIANG Weifa

For Official Use Only

Student Final Year Project Declaration

I have read the project guidelines and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I hereby declare that the work I submitted for my final year project, entitled:

Gestures-based Touch-Free smartwatch development with deep learning

does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Project Guidelines.

Student Name: HON, Hing Ting

Signature: 

Student ID: 55776782

Date: March 31, 2022

1. Abstract

In recent years, wearable technologies have stimulated the development of various Internet of Things (IoT) applications to access the user status with pervasive physiological data processing. However, existing touch-based smartwatches still suffer from two hands occupation and finger occluding problems, while the usability of smartwatch interfaces is restricted by its tiny screen and interaction approach that is inconvenient and requires excessive physical effort to interact. Hence, this project presented a gesture-based touch-free interaction system that leverages the wrist-mounted characteristic of smartwatches to capture arm, hand, and finger motion data collected with the built-in accelerometer and gyroscope units for gesture recognition. The proposed system employs a deep learning model to classify motion data and recognize the single-handed gestures and in-air writings from users to perform different inherent operations in a small interface without external equipment.

This project proposed a lightweight multi-head one-dimensional convolutional neural network architecture with a gesture motion detection mechanism to interpret different sensor data sources and operate robust gesture recognition locally on a standalone smartwatch. This system captures real-time hand motion at any starting point to recognize 15 gesture vocabularies and 5 in-air writing gestures. This project also constructed different gestural models using different network configurations to compare and evaluate the performance of the proposed network architecture. The proposed model with appropriate data augmentation achieves a user-independent gesture recognition accuracy of 98.46%, which gains the most accuracy improvement compared with other network designs and achieves a satisfactory generalization result in recognizing the designed gestures from different users.

This project further demonstrates two working application instances, including the music player and message application, which are supported by the gestural interaction system to justify the convenience and feasibility under different layout settings. Users can directly activate specific functions or perform precise selection with single-handed gestures to achieve intuitive touch-free interaction on a tiny smartwatch interface. The system maintains complete interface visibility and accurate layout elements selection by eliminating the precise physical interaction when controlling smartwatch applications.

2. Acknowledgments

I would like to express my sincere appreciation and gratitude to my project supervisor, Dr. XU, Weitao, for providing the project idea and an opportunity to learn and work on this interesting project with unconditional hardware support. My project supervisor always patiently offers valuable guidance and feedback to point me in the right direction to achieve the project's goals. His willingness to give his time so generously has been very much appreciated. Also, I would like to extend my thanks to the participants for their unconditional help in participating in the experiments in this project. This project could not be completed without their cooperation in the experiments.

Table of Contents

1. Abstract.....	3
2. Acknowledgments	4
3. Introduction	7
3.1. Motivation & Background Information	7
3.2. Problem Statement	7
3.3. Project Aims & Objective	8
3.4. Project Scope & Deliverables	9
3.5. Report Organization	9
4. Literature Review	10
4.1. Review of Wearable-based Gesture Recognition Modalities	10
4.1.1. Acoustic Gesture Recognition	10
4.1.2. Sensor-based Gesture Recognition	10
4.2. Review of Existing Gesture Recognition Algorithms.....	11
4.2.1. Dynamic Time Warping	11
4.2.2. Support Vector Machines	12
4.3. Review of Deep Learning Classification Model.....	12
4.3.1. Convolution Neural Network.....	13
4.3.2. Long Short-Term Memory.....	13
4.4. Comparison & Discussion.....	14
5. Proposed System Design	15
5.1. Overview	15
5.2. System Diagram	15
5.3. System Components.....	17
5.3.1. Gesture Vocabulary	17
5.3.2. Data Collection Component.....	19
5.3.3. Classification Model Component	20
5.3.4. Application Component	22
6. Methodology & Implementation	23
6.1. Development Environment	23
6.1.1. Application Development Environment	23
6.1.2. Model Development Environment.....	23
6.2. Data Collection.....	23
6.2.1. Experiment Setup.....	23
6.2.2. Data Collection System.....	24
6.3. Data Preprocessing.....	26
6.3.1. Data Resampling.....	26

6.3.2.	Data Augmentation	26
6.4.	Gesture Model Training	28
6.4.1.	Network Architecture.....	28
6.4.2.	Model Training	29
6.5.	Gestural Smartwatch Application	30
6.5.1.	Gestural Interaction System.....	30
6.5.2.	Gestural Application Instances	31
7.	System Testing & Result	33
7.1.	Performance Testing Procedures.....	33
7.2.	Proposed Model Evaluation	34
7.2.1.	Data Augmentation Performance Evaluation	34
7.2.2.	Network Performance Comparison.....	35
7.2.3.	Recognition Confusion Evaluation	36
8.	Conclusion	39
8.1.	Summary of Achievements & Review.....	39
8.2.	Future Improvement.....	40
8.2.1.	Leverage Additional Data Modalities	40
8.2.2.	Increase Gestures Categories & Data Samples.....	40
8.2.3.	Employ Few-shot learning Approaches.....	40
9.	Plan & Schedule	41
9.1.	Project Schedule.....	41
9.2.	Gantt Chart	42
10.	References.....	43
11.	Appendix.....	46
11.1.	Appendix A. Monthly Logs.....	46
11.2.	Appendix B. Supplementary Diagrams	48

3. Introduction

3.1. Motivation & Background Information

Smartwatches have gained essential growth in popularity and become ubiquitous in daily life with the advantages of high mobility and user status-awareness that smartphones cannot achieve (Chun et al., 2018). Currently, touch-based interaction is the dominant scheme of smartwatch interface controlling, which benefits from direct visual element interaction and inherits the existing usage from other touchscreen devices (Sun et al., 2017). Thus, smartwatches support efficient information access and response, including instant messaging and social networking, through the enriched graphical interface on a small panel. Different from smartphones, wrist-mounted is the major characteristic that allows smartwatches to accurately collect physiological data through continual wrist connection (Rawassizadeh et al., 2014). It provides a unique opportunity of recognizing physical activities with built-in sensors.

To explore this potential usage of smartwatches, various machine learning models are developed to classify the specific arms and hand movements pattern with wrist-mounted data (Mozaffari et al., 2020). Recently, there has been an increasing awareness of utilizing wearables-based recognition to investigate the feasibility of providing an immersive user experience with precise body language recognition (Poongodi et al., 2020; Kurz et al., 2021). Since the development of motion recognition is still in infancy, smartwatches with superior motion sensing and machine learning capability are the critical component in achieving the evolution of human-computer interaction (HCI) methods with explicit body language cognition.

3.2. Problem Statement

Despite the high mobility and potential of smartwatches, surveys and literature investigations suggest that existing touch interaction limits its usability by requiring precise target acquisition (Chun et al., 2018; Rawassizadeh et al., 2014). The small touch panel maintains smartwatch mobility but restricts the input and output capabilities which make tasks onerous and inefficient with two restrictions.

First, as shown in *Figure 1*, smartwatches cannot provide one-handedness reliably and restrict the usage scenario since the touchscreen is not accessible for the users whose non-wearing hand is not available or suffering from disabilities (Kurz et al., 2021). Additionally, users' fingers easily occlude half of the tiny touch panel and reduce its visibility during interaction while it is difficult

and error-prone to select the correct small button (Oney et al., 2013). Users frequently bend their fingers to minimize finger occluding and repeat the finger tapping to select the target button, but this usage habit imposes an additional burden on their hands (Hara et al., 2015).

Figure 1. Picture of finger occluding when using smartwatch touchscreen



Alternative input modalities are necessary to mitigate those touch-based interaction shortcomings. Though smartwatches have the potential to expand interaction modalities with hand motion awareness, it is still unclear how accurate the hand motions in different scales can be recognized and leveraged to assist diverse inherent smartwatch operations.

3.3. Project Aims & Objective

This project aims to employ deep learning to recognize hand movements for operating the off-the-shelf smartwatches to address the above problem. The primary objective is to validate smartwatch hand motion recognition performance in different preciseness, including arms, wrist, and finger motions. Rather than heavily relying on touch interaction, the gestural interaction system will be developed to expand usage scenarios by leveraging different scales of single-handed gestures and finger writing to control the smartwatch applications in different layout settings. This system is expected to enhance the user experience of smartwatch interfaces by reducing physical effort and optimizing convenience with a touch-free scheme.

3.4. Project Scope & Deliverables

This project's scope involves developing a smartwatch application to collect real-time hand motion data representing the designed gestures from the embedded sensors for gesture recognition. The deep learning model will be developed and trained with the collected dataset to recognize various in-air gestures and index finger writing. The resulting system will be integrated with different smartwatch applications to achieve and support a touch-free gestural interaction with accurate gesture detection and recognition performance for executing general operations and text entry.

3.5. Report Organization

The report is divided into three main sections. The first section explores the existing smartwatch-based gesture recognition techniques and limitations. Then, the second section presents the detailed interaction system design, involved components, and implementation of the proposed solution. Finally, the experiment results and application instances demonstration will explain how this proposed system achieves the project objective.

4. Literature Review

This section reviews different existing research on gesture recognition modalities. In particular, the potential problems and improvements will be evaluated. Hence, various existing recognition algorithms and deep learning approaches will be analyzed to determine their feasibilities and limitations for implementing gesture recognition with off-the-shelf smartwatches.

4.1. Review of Wearable-based Gesture Recognition Modalities

4.1.1. Acoustic Gesture Recognition

The acoustic signal is the sound waves or vibrations produced in response to specific activities. Recent acoustic sensing systems utilize embedded microphones and sensors from wearables to capture the acoustic data reflected by gestures. Wang et al. (2020) applied acoustic signals to recognize gestures by classifying the measured wave's nuance with the microphones. Laput et al. (2016) developed an accelerated accelerometer data sampling to identify micro-vibrations. Acoustic recognition can classify the wave's nuance generated by various fine-grained hand motions. Nonetheless, acoustic signal transmission may experience dramatic signal distortion according to activity noise, so the maximum ambient noise level must be restricted during data capturing (Moreira et al., 2020). Therefore, acoustic sensing systems might not be suitable for implementing passive gesture recognition in different environments.

4.1.2. Sensor-based Gesture Recognition

Smartwatch inertial sensors can track the slight wrist movements indirectly induced by hand gestures. When the user extends or bends a finger, the hand tendons will produce a subtle wrist muscle movement to change the smartwatch motion velocity, which is sufficient for classifying different hand gestures. Kurz et al. (2021) employed inertial sensor data to classify hand sliding gestures with smartwatches and smart rings. Xu et al. (2015) also used the wrist- and finger-mounted inertial sensors to recognize various arm, hand, and finger gestures. The Inertial Measurement Unit (IMU), especially accelerometer and gyroscope units, are often leveraged to record subtle gesture movements by monitoring the device displacement in various research.

- Accelerometer data

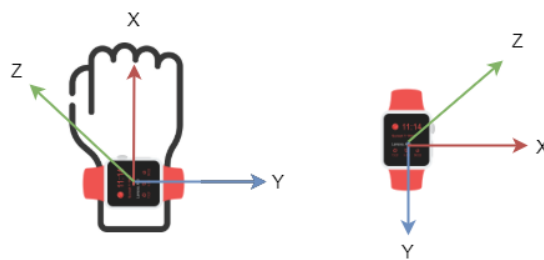
An accelerometer is the standard sensor on modern smart devices to measure gravitational acceleration forces representing the device displacement with the corresponding intensity and speed. This sensor streams the acceleration data in x, y, and z-axes to represent three-dimensional

linear movement trajectories corresponding to the various wrist and arm activities by continuously contacting the users.

- Gyroscope data

A gyroscope is another embedded sensor to calculate a rotation describing the change of angular velocity over specific periods. This sensor captures the subtler tilt angle changes and speed of the device in three dimensions by spinning and aligning different orientations and positions. Rotational and angular velocity data sequences can represent orientation and tilt changes corresponding to the various wrist and arms motions.

Figure 2. Illustration of smartwatch sensing movement directions



However, it is cautious that the sensor-based recognition is highly sensitive to arm orientation. Some unintentional subtle movement might produce data noise that affects the recognition performance. Hence, an intelligent learning algorithm is desired to learn the hidden patterns of accelerometer-gyroscope motion data representing different gestures and achieve a more case-sensitive recognition.

4.2. Review of Existing Gesture Recognition Algorithms

Recent research applied various pattern recognition and machine learning algorithms to process the motion data to exhibit a satisfactory accuracy in recognizing the gestures from smartwatches.

4.2.1. Dynamic Time Warping

Dynamic time warping (DTW) is the time analysis algorithm for measuring the similarity between two temporal data sequences with dynamic programming optimization. For gestures recognition, the DTW algorithm is often employed to evaluate the similarity between the training and input motion data to determine the correct gesture pattern with minimum distance. Hamdy et al. (2014) suggested that DTW achieves high gestures recognition accuracy on user-dependent learning

using accelerometer data template matching based on Euclidean distance. Yanay and Shmueli (2020) also proposed the DTW and K Nearest Neighbors (KNN) combination to perform a user-dependent writing gesture recognition by comparing new samples to the reference samples.

However, DTW only achieves an outstanding recognition accuracy for the dataset related to the specific user. This user-dependent recognition is not representative for all general users and cannot generalize to the user-independent paradigm. Thus, DTW is not suitable for implementing a robust gesture recognition that supports a more diverse range of unfamiliar users on smartwatches.

4.2.2. Support Vector Machines

Support Vector Machines (SVM) is a supervised learning model that decomposes and classifies data by discovering the most significant margin boundaries between different classes. It is widely employed for hand motion recognition by classifying the extracted features from motion data with lower computational complexity. Wen et al. (2016) introduced an SVM-based gesture recognition with the manually extracted features from the raw sensor data. The author suggested that the SVM model has outperformed accuracy over other machine learning algorithms, including K Nearest Neighbors (KNN), Naive Bayes (NB), and Logistic Regression (LR). Ameliasari et al. (2021) also employed SVM to implement gesture recognition with feature selection for sensor data using Pearson Correlation.

Nonetheless, SVM-based recognition requires prior manual features extraction and noise filtering to convert the original data into representative data, so data loss may occur during the feature engineering and reduce the classification accuracy (Muralidharan et al., 2021). Also, it is difficult to select a suitable kernel function and regularization for tuning the SVM according to the number of features and training samples.

4.3. Review of Deep Learning Classification Model

Traditional pattern recognition and machine learning algorithms are restricted by their ability to process raw and large datasets related to various users. In contrast, deep learning algorithms can automatically extract features for different raw data types, including image, audio, and video, to perform classification without manual intervention.

4.3.1. Convolution Neural Network

Convolution Neural Network (CNN) is a feed-forward neural network that achieves an outstanding classification for complex data types using the concepts of convolution and pooling (Muralidharan et al., 2021). The convolutional layers detect and filter the features from the inputted data, while the pooling layers repeatedly reduce the data dimensionality to distill the essential features. It is further passed to fully connected layers for classification. Compared with SVM, the CNN model can directly process the original data without prior manual feature extraction to learn the important features automatically through the network and prevent data loss during the feature extraction (Kwon et al., 2018).

Different scales of human activities can be classified using CNN to process the sensor data. Kwon et al. (2018) and Yanay and Shmueli (2020) reported that the CNN model could recognize hand motion data without being restricted by user dependency. Different from the DTW algorithm, CNN can perform a more refined motion data analysis for the small segments with large kernel size and network depth (Chu et al., 2020). The model can learn the high-level features of numerous gesture motion samples generated from different users.

4.3.2. Long Short-Term Memory

Long Short-Term Memory (LSTM) is an extended deep learning model of recurrent neural networks. It can exploit long-term dependencies with its complicated memory cell structure that feeds the results back to the network. LSTM retains critical data from the previous inputs to influence the current output. The memory cell leverages the gating concepts to memorize the previous data values over arbitrary time intervals and recognize the temporal correlations between data samples for activities classification (Zebin et al., 2018).

Due to its insensitivity to the delay period length, LSTM is ideal for classifying the long-term and variable-length data to solve the dynamic motion recognition with many-to-many inferencing. Since LSTM has less feature extraction compatibility with numerous weights and biases parameters, it is often integrated with convolution layers to perform high-level feature extraction for data input and capture the spatiotemporal correlation to support sequence prediction for human activities (Oluwalade et al., 2021).

4.4. Comparison & Discussion

Table 1. Comparison of existing research on wearable-based gesture recognition

Literature	Gesture Presentation	Modalities	Classification Model	Feature Engineer	Performance
(Wang et al., 2020)	15 in-air finger motion gestures	Acoustic signal with microphone	CNN-LSTM	✗	98.4%
(Laput et al., 2016)	17 in-air finger motion gestures	Bio-acoustic signal with accel.	SVM	✓	94.3%
(Hamdy et al., 2014)	8 in-air sliding gestures	Accel.	DTW, KNN, SVM, HMM	✓	~98%
(Kurz et al., 2021)	12 in-air sliding gestures	Accel., Gyro.	RF, SVM, KNN, NB	✓	98.8%
(Wen et al., 2016)	5 in-air finger motion gestures	Accel., Linear Accel., Gyro.	SVM	✓	98%
(Ameliasari et al., 2021)	8 in-air sliding gestures	Accel., Gyro.	SVM	✓	94%
(Yanay and Shmueli, 2020)	26 in-air writing gestures	Accel., Gyro.	DTW-KNN	✓	89.2%
			CNN	✗	83.2%
(Kwon et al., 2018)	10 hand gestures on surface	Accel.	CNN	✗	97.3%

The traditional recognition algorithms, including SVM and DTW, are investigated to recognize hand gestures with a relatively small dataset effectively. However, recent research employs deep learning models, including CNN and Conv-LSTM, to achieve satisfactory user-independent gesture recognition accuracy due to the capability of automatic feature extraction for the large dataset produced by multiple users under reduced loss with powerful computational engines.

However, no research had detailed implementation of applying gestural recognition to support and improve the interaction of smartwatch application interfaces under different layout arrangements. Also, it is challenging to apply deep learning for gesture recognition due to the high costs of collecting human-generated motion data. Therefore, this project aims to develop an interaction system that recognizes different scales of gestures to perform precise and touch-free control for layout elements and functions in ordinal smartwatch interfaces. It is also targeted to overcome the deficiency of dataset size and achieve a satisfactory gesture recognition performance for a robust gestural interaction.

5. Proposed System Design

5.1. Overview

This project focuses on recognizing 15 in-air gestures for distinct operational functions and 5 fingertip writing gestures for English characters entry to develop a touch-free smartwatch interaction system. The proposed system employs a deep learning model to implement the above recognition techniques with embedded inertial sensors from smartwatches. This system consists of three major technical components: the data collection component that captures accelerometer and gyroscope data related to hand motions, the deep learning model component that classifies and recognizes the designed gestures, and the application component that extends the existing applications to be interactable with designed gestures.

5.2. System Diagram

Figure 3. System diagram of the proposed system

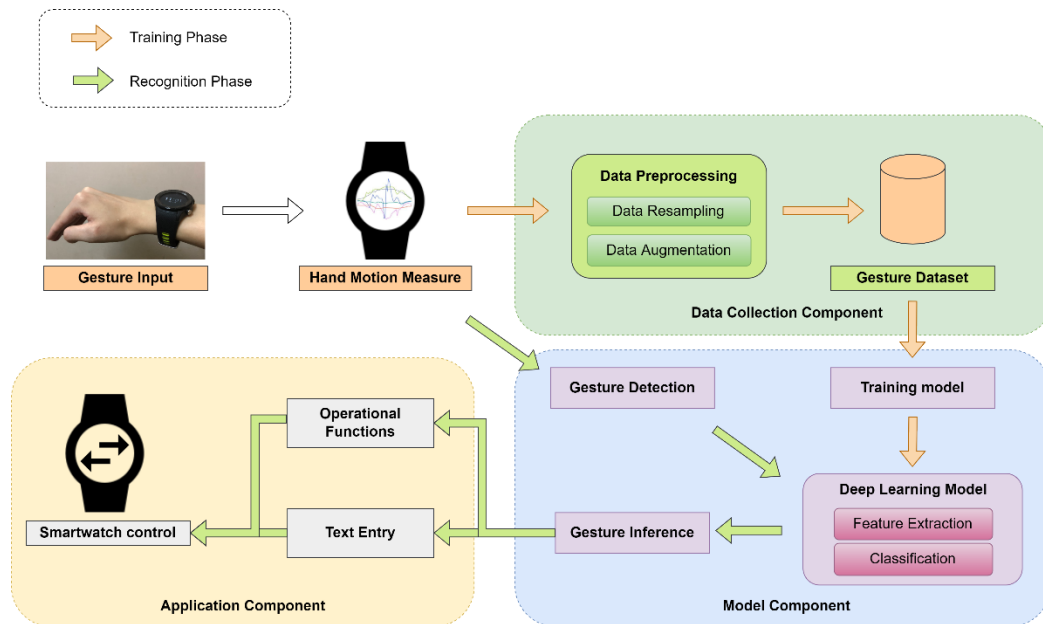


Figure 3 illustrates all technical components and the related system architecture. The proposed system is mainly divided into three mentioned components for the training and recognition phases. All components depend on the hand motion measuring system to transfer gesture inputs into sensor data sequences for gesture samples collection during the training phase and gestural interaction on smartwatch applications during the recognition phase. Hence, the hand motion measuring system will be implemented on the smartwatch to log the real-time sensor data that serve as a significant user input for other components.

- Data Collection Component

For the training phase, it is required to priorly collect a set of accelerometer and gyroscope data samples related to the designed gestures. After collecting sufficient data from multiple users, the data preprocessing techniques, including data resampling and data augmentation, will transform the collected data into a specific fixed range and increase the amount of data. The dataset with gesture class labeling will be leveraged to train the offline deep learning model for feature extraction and classification under the supervised learning approach.

- Model Component

The resulting model will be integrated as the core in the backend systems of the smartwatch interaction system during the gesture recognition phase. The system will receive the real-time sensor data from the hand motion measuring system to determine the occurrence of the gesture motion with the gesture detection mechanism and then identify the correct hand gesture category. It will compute the possible performing gesture by automatically extracting the features and performing gesture inference with the real-time captured hand motion data.

- Application Component





After the model computes the most possible performing gestures, the gesture result is treated as the primary input for the smartwatch applications. Each gesture class is mapped to the distinct operations for layout elements and functions on the related application interface. When the specific gesture is recognized, the system will execute the corresponding operational functions or character entry command for instant reaction. This application component is designed to support operations for different application instances with the same gesture vocabulary.





5.3. System Components





5.3.1. Gesture Vocabulary




There are different inherent operational functions required for smartwatch control. To implement the low-effort gestural smartwatch interaction, this project designs 15 simple single-handed gestures that are simple to perform and memorize, so it can avoid hand fatigue and require less attention to focus on the watch face for different operations. *Table 2* shows all the mapping of proposed gestures and supporting operations.

Table 2. Design of 15 gestures for different operations

Operations	Slide up	Slide down	Next Page	Previous Page
Gesture Vocabulary				
	Wrist Lifting	Wrist Dropping	Clockwise Circling	Counter-clockwise Circling


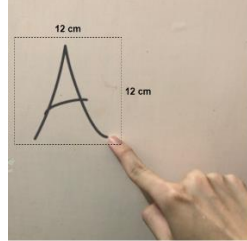
Operations	Select Previous	Select Next	Confirm selection	Back / Backspace
Gesture Vocabulary				
	Finger Flicking	Finger Waving	Finger Snapping	Finger Rubbing

Operations	Exit	Activate Functions	Copy	Paste
Gesture Vocabulary				
	Hand Rotation	Knocking	Hand Squeezing	Finger Pinching

Operations	Turn Up Volume	Turn Down Volume	Clear
Gesture Vocabulary			
	Right Flipping	Left Flipping	Hand Sweeping

Besides these operations, the in-air fingertip writing gestures are proposed as the most intuitive way to perform text entry since handwriting can immediately reflect the human thinking of the desired character input. This project will only focus on recognizing the 5 writing gestures and map to the first fifth upper-case English characters for the text entry demonstration. *Table 3* shows the proposed writing gestures representation for text entry.

Table 3. Design of 5 in-air writing gestures for English character entry

Operation	Touch Representation	Gesture Representation
5 Characters Text Entry (A to E)		
	Typing on the soft keyboard	In-air Writing

The above gestures will be performed in variable lengths since the performance speed for each gesture is different. Thus, the maximum time to complete a gesture is fixed to two seconds to align the window size of each sampling period, while the writing range for each writing gesture will be restricted inside an approximately 12cm * 12cm square. It standardizes the length of the sensor data for gesture motion classification and ensures that users can quickly complete the gestures without any burden.

Moreover, the motion data of static motion will also be collected to classify the null gesture class representing no gesture performing. If the null gesture class is detected, no operations will be executed on the application to prevent false-positive recognition. Hence, the proposed application will recognize 21 gestures, including the null gesture class.

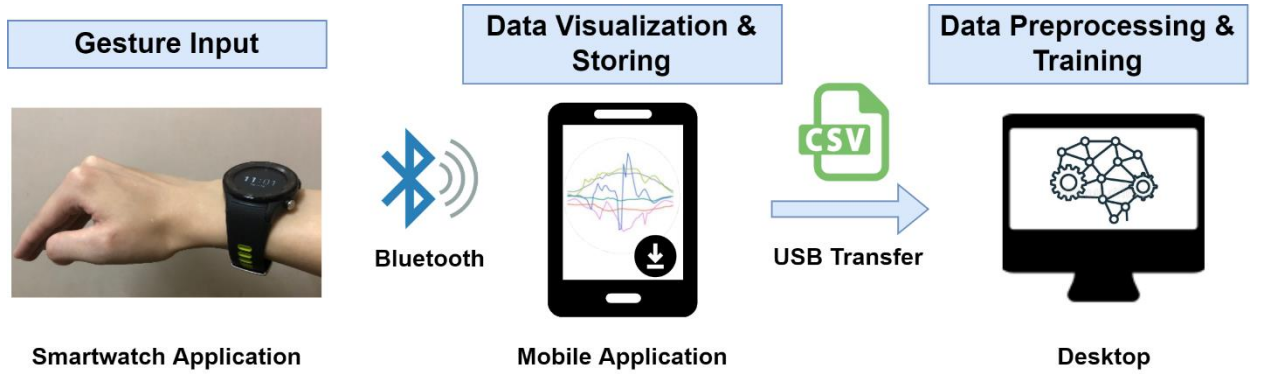
5.3.2. Data Collection Component

Since this project adopts sensor-based gesture recognition, the hand motion measuring system is developed as a smartwatch application and paired mobile application. It will collect and store the motion data sequences from the smartwatch accelerometer and the gyroscope unit on 3-axes at the maximum sampling rate of 100 Hz. The resulting data shape will be constructed by six time-series data channels (Ax, Ay, Az, Gx, Gy, Gz) that represent the values of x, y, z-axis related to the smartwatch's linear position and angular velocity from a fixed time step from 1 to T as shown below.

Accelerometer Data: $Ax = \{a_x^1, \dots, a_x^T\}, Ay = \{a_y^1, \dots, a_y^T\}, Az = \{a_z^1, \dots, a_z^T\}$

Gyroscope Data: $Gx = \{g_x^1, \dots, g_x^T\}, Gy = \{g_y^1, \dots, g_y^T\}, Gz = \{g_z^1, \dots, g_z^T\}$

Figure 4. Design of data collection flow



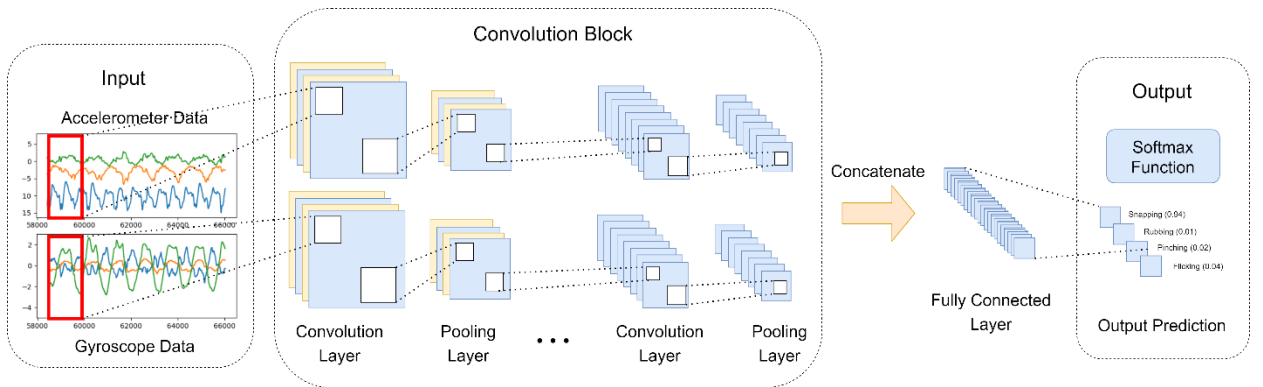
The smartwatch is used to priorly collect sufficient motion data samples of designed gestures for model training. Due to the concern of computation overhead and difficulty of data access with the Wear OS device, the motion data from the smartwatch will be transmitted to the paired smartphone at the end of each sample collection instead of real-time transmissions. As shown in *Figure 4*, after users complete each gesture sampling with the smartwatch, the collected data sequence will be transmitted to the paired mobile application in a data string format through Bluetooth for data visualization or storing purpose. The mobile application will save the received data string into CSV file format and then transfer it to the desktop through the USB. All collected data will be preprocessed and combined for offline model training on the desktop platform.

5.3.3. Classification Model Component

The deep learning algorithm is employed for advanced feature extraction and classification on the accelerometer and gyroscope data. Since all motion data are captured under a fixed length of two seconds, it will be more suitable to be processed by the CNN rather than LSTM that requires numerous parameters. Due to the advantages of essential feature detection, CNN can reduce the overall loss of the raw data to achieve high-level feature extraction under different dimensions. One-dimensional CNN (1D-CNN) architecture is superior at deriving hidden features in fixed-length data sequences, including sensor data under simple network configurations. Hence, this project will adopt a multi-head 1D-CNN model, which is the variant of the existing single-head 1D-CNN model architecture as inspired by Becker et al. (2019).

Different from the single-head CNN, the proposed model architecture consists of two separated convolution structures to process 2 motion data streams from different sensor sources. The reason for employing this architecture is to evaluate the classification impact of dedicated extracting features on separate data streams parallelly rather than sequentially. To satisfy the proposed architecture, the raw sensor data are split into two streams, including the accelerometer and gyroscope streams. The first convolutional head will receive the 3-axis data of the accelerometer, while the second convolutional head will receive the 3-axis data of the gyroscope. The multi-head 1D-CNN model is trained with a supervised approach to classify motion data based on training gesture labels. It captures the temporal characteristics of two separated inputted motion data and learns the automatically extracted features from sensor data by associating them with the related gesture category.

Figure 5. Semantic diagram of proposed multi-head 1D-CNN architecture



As shown in *Figure 5*, the designed multi-head CNN model will first separately process the accelerometer and gyroscope data for each gesture class with the same kernel configuration. Each

convolutional head processes the segmented data across the entire sensor data on 1D convolutional and max-pooling layers in parallel. The convolutional layers will continuously extract useful convolved feature maps from the data segments based on the filters used in the specific layer. The max-pooling layers perform signal maximizing for the sub-sampling region from the previous layer to obtain the abstract features that are represented compactly. It can also prevent overfitting and perform noise suppression by reducing the size of the feature maps.

After the final convolution and pooling in the convolution block, all the extracted feature maps will be concatenated to produce a resulting individual representation of accelerometer and gyroscope data. The fully connected layers will learn the non-linear combinations of the concatenated features. It will evaluate the correlation between the features and the corresponding gesture class. Finally, the softmax operator will compute the probability of the specific gesture occurrence with the output vector sequence for the recognition result. The resulting model will be launched into the smartwatch application system for real-time motion data processing and gesture recognition.

5.3.4. Application Component

For the real-time gesture interaction, it is also necessary to detect the occurrence of the performed gesture during the hand motion measurement. In this project, the system will monitor the hand motion in the background before gesture inference. If the change of the motion data exceeds a specific threshold, the smartwatch will consider that some hand movement from users is detected and further collect the remaining motion data for gesture recognition. This mechanism aims to receive users' gestures at any starting point and prevent unnecessary inference computation if no hand motion is detected. The proposed system will capture the hand motion for the classification model and obtain the gesture result to edit input text, browse and select layout elements, scroll, and switch pages in smartwatch application interfaces, as mentioned in *Table 2* and *Table 3*.

Most current Wear OS applications implement an extensive interface that provides a focused and scrollable view to display all layout elements in a list appearance. As shown in *Figure 6*, if users want to select a specific layout element on the interface precisely, the system will allow them to switch the focus between different items and pages by receiving a sequence of gestures. The layout elements will be highlighted for users to recognize the focusing item, while the frequently used functions will also be bounded with specific unique gestures. Thus, users can directly activate those functions from the applications to save the time for locating the button and save the space for displaying it on a tiny interface.

Figure 6. Implementation of gesture interaction for smartwatch application



6. Methodology & Implementation

This section introduces the detailed project implementation regarding the adopted methods to address the problem. The data collection procedures, data preprocessing algorithms, deep learning network architecture, and application instances are further discussed.

6.1. Development Environment

6.1.1. Application Development Environment

As this project aims to develop a touch-free interaction system on off-the-shelf smartwatches with sensor-based gesture recognition, Huawei Smartwatch 2 2018 is used as the major development platform for this project, while its embedded accelerometer and gyroscope are used to capture the force and angular rate of different hand movements accurately. Wear OS is the standard operating system for this commodity-level smartwatch to access software and hardware information. Therefore, Android Studio is the major development tool to develop the Wear OS applications and the paired mobile application for sensor data collection, gesture recognition, and gestural interaction with Java programming language. It allows the native applications to utilize Android APIs for sensor management, data gathering, and interface control.

6.1.2. Model Development Environment

For the deep learning model development and the data preprocessing, TensorFlow is used for training a deep learning model with an open-source deep-learning library Keras in Python. The offline neural network is implemented and trained with TensorFlow on the desktop platform and converted to the TensorFlow Lite (TFLite) model format that can be easily launched to a Wear OS or Android application with Android Studio. TFLite model expands the TensorFlow model's capability into a mobile environment. It is suitable for implementing and optimizing the deep learning framework for on-device and IoT inference and classification under the resource-restricted condition.

6.2. Data Collection

6.2.1. Experiment Setup

It is crucial to construct a sufficient and comprehensive dataset by collecting gesture samples from multiple users, which is also vital to determine the recognition and generalization ability of the classification models. Hence, this project invited 8 participants to provide motion data samples for the experiment. Each participant was asked to perform the 21 assigned gestures (20 designed

gesture classes and a null gesture class) for 40 times in a randomized order under different behaviors and orientations, as shown in *Figure 7*.

Figure 7. Data collection experiment under different orientations



Sitting & Raise Mounted Hand



Standing & Raise Mounted Hand

Since smartwatch users need to raise the mounted hand to focus on the watch face when interacting with the application interface, participants were asked to raise the mounted hand when performing the gesture during the data collection experiment. Also, each participant was required to tightly wear the same smartwatch on the wrist to closely capture the propagated hand and finger motion. It was strictly required that the smartwatch is equipped with the dominant hand to collect precise hand motions samples, so it was further assumed that the right hand was the dominant hand to ensure that all samples were related to right-hand gestures only for this project. The experiment was monitored by the experiment moderator to ensure the integrity of each data sample.

6.2.2. Data Collection System

The data collection system that consists of both smartwatch and mobile applications was leveraged to complete the whole experiment. Each participant was given instructions on the designed gestures technique and the collection procedure using the smartwatch and mobile applications. To prevent capturing unnecessary hand or body movement, participants were only required to press the start button on the smartwatch interface and perform gestures. The circular timer will count for two seconds to notify the participant about the current gesture sampling progress. When the timer reaches the end, the data sequence will be sent to the mobile application for subsequent procedures (*Top of Figure 8*).

The experiment moderator was responsible for handling the received data by saving or discarding each sample on the paired mobile application. The data visualizer displays the line charts of the received accelerometer and gyroscope data to ensure that the current trial captures all the

movement within a two-second window. The dropdown box allows the moderator to switch the gesture class and label the received data. The accumulative counter records the total samples saved for the current participant (*Bottom of Figure 8*).

Figure 8. Illustration of data collection procedure & interfaces



Finally, a total of 6,720 gesture motion samples were collected by requesting the 8 participants to provide 840 (21 gestures * 40 times) samples per each. The raw dataset was divided into training, validation, and test sets, while the training and validation sets underwent data augmentation.

6.3. Data Preprocessing

The data preprocessing procedure in this project is relatively simple compared with other similar projects. There was no calculation for specific features since it is expected that the neural network could automatically extract and learn the relevant features from the sensor data. Also, data normalization or smoothing was not applied because the collected dataset is already at comparable scales, and those approaches might confound the subsequent results. Hence, this project only adopted necessary preprocessing techniques, including data resampling and augmentation.

6.3.1. Data Resampling

Each collected gesture sample has inconsistent sampled sizes for the accelerometer and gyroscope data, which slightly exceeds the expected size ($100\text{Hz} * 2 \text{ seconds} = 200 \text{ rows}$) because the buffer difference of smartwatches sensors will cause an inconsistent sampling latency. Since the 1D-CNN can only process the fixed-length sequence data, all the collected data were resampled to a fixed shape of 200 rows * 3 columns (2 seconds * 100 Hz * 3 axes for accelerometer and gyroscope separately). The data resampling technique extracts a window of 200 rows that contains the movement for every sample rather than constructing a new data sequence to fit the expected size since it is desired to keep the input data close to its original form.

6.3.2. Data Augmentation

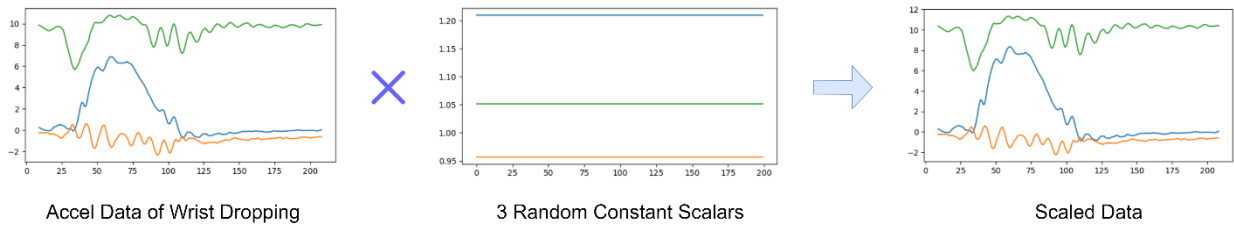
As the collected dataset is relatively small for training a deep learning model, it will lead to model overfitting, where the model closely corresponds to the small training data. Consequently, the model will have generalization errors when processing unseen data from unfamiliar users or data with slight motion differences. Hence, to overcome insufficient training data, several data augmentation approaches were applied to the original accelerometer and gyroscope data to create some slightly modified data copies for training dataset enrichment, which helps to avoid overfitting and improve generalization performance. The data scaling, time-warping, and magnitude-warping approaches were adopted to increase the variations and cover the undetected data input (Um et al., 2017).

- Data Scaling Approach

Data scaling approach will either increase or decrease the magnitude of the original motion data sample by multiplying the entire data with 3 different random constant scalars for each axis channel to form a modified data sequence. Each constant scaler is randomized in a range around one based on the Gaussian distribution and fitted to the sample shape. It can simulate the gesture

motion with a stronger magnitude when the scaler is larger than one or a weaker magnitude when the scaler is smaller than one on a specific axis (*Figure 9*).

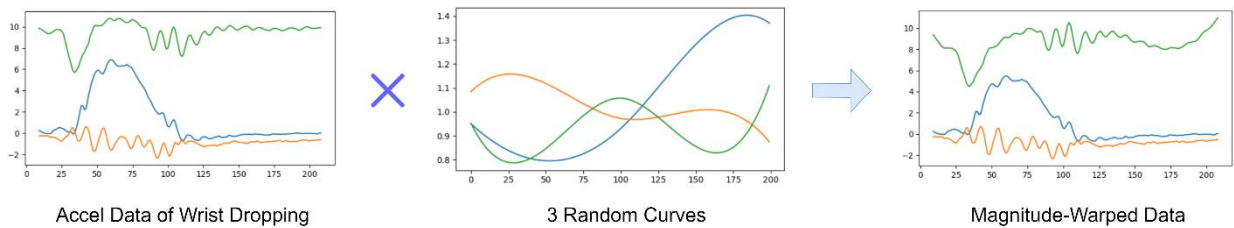
Figure 9. Illustration of Data Scaling process



- **Magnitude-Warping Approach**

Magnitude-warping has a similar principle to data scaling by applying noise to the entire sample for magnitude modification. However, the magnitude-warping approach will dynamically change the magnitude by convolving the data window with 3 different curves randomly around one rather than applying constant noise to the entire sample. Therefore, it is possible that the magnitude in the same axis channel is being strengthened and weakened at a different temporal position. It can simulate more strengths variety of fingers, hands, and arms movements (*Figure 10*).

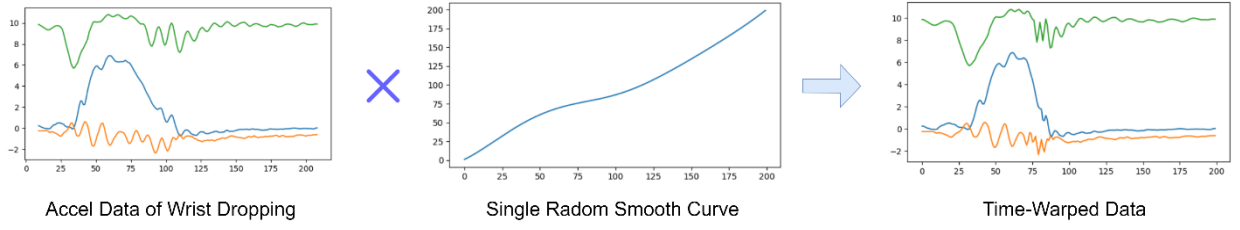
Figure 10. Illustration of Magnitude-Warping process



- **Time-Warping Approach**

Time-warping approach disturbs the temporal position of a specific sample. It smoothly distorts the time intervals by enlarging or compressing a specific range of motion data sequences. The data time steps are perturbed according to a random smooth curve with the same shape to determine the range, location, and magnitude of enlarging or compressing on all the axis channels in the motion data. The temporal position of the data is modified to simulate different users performing the same gesture at different speeds and paces (*Figure 11*).

Figure 11. Illustration of Time-Warping process



As the data might be over-distorted if the modification impact was too significant, the standard deviation for generating scalers and the complexity of the random curves were adjusted to be relatively low to prevent over-distortion. As a result, every individual original sample data was transformed with the above approaches respectively to construct three more synthetic data samples and grouped with the original one. After the complete data preprocessing, it formed a new dataset with a total of 23,520 gesture motion samples to enhance the generalization performance by preventing overfitting with more training sample variations.

6.4. Gesture Model Training

6.4.1. Network Architecture

The preprocessed data samples would be split into accelerometer and gyroscope streams and analyzed by the proposed multi-head 1D-CNN model parallelly. The detailed architecture and the learning process of the proposed network are described below (see *Figure 16* in Appendix B).

Table 4. Detailed network architecture

Layers	Filter Size	Filter No.	Activation	Param No.	Output size
2-head Input	-	-	-	-	2 * (200*3)
2-head 1D Convolution	10	16	ReLU	496	2 * (191*16)
2-head Max Pooling	2	-	-	-	2 * (95*16)
2-head 1D Convolution	10	32	ReLU	5,152	2 * (86*32)
2-head Max Pooling	2	-	-	-	2 * (43*32)
2-head 1D Convolution	10	64	ReLU	20,544	2 * (34*64)
2-head Max Pooling	2	-	-	-	2 * (17*64)
2-head Global Average Pooling	-	-	-	-	2 * (64)
Concatenate	-	-	-	-	(128)

Fully Connected	64	-	ReLU	8,256	(64)
Drop out (0.5)	-	-	-	-	-
Fully Connected	21	-	Softmax	1,365	(21)

As shown in *Table 4*, the architecture of the proposed network is composed of 3 one-dimensional convolutional layers that are interleaved by 3 max-pooling layers for each head of the network. All 1D convolutional layers consist of Rectified Linear Units (ReLU) activation function with the filter size of 10 to generate 16, 32, 64 filters from the first layers to the last layers. This sparse structure prevents the network overhead by interleaving max-pooling layers for feature reduction. The global average pooling is used to replace the flatten and fully connected layers by generating a single feature map for each classification category with the average of each previous feature map, so the output can be directly fed into the next layer with a more reasonable parameter number.

The generated feature maps from the two convolutional heads are concatenated into a single vector and fed into another fully connected layer (64 units) with ReLU to connect all features from the preceding layers. The dropout with a percentage of 0.5 is used after the fully connected layer. This regularization method randomly ignores some outgoing edges of hidden units (neurons) at each update with a probability of 0.5 during training time. The neurons dropping encourages the model to extract and learn more complex and robust features from the training data by reducing the reliance on the specific features. Finally, the gesture classification is performed by the softmax fully connected layer to associate the features from the previous layout with different gesture results. This network architecture is relatively shallow compared with other typical 2D CNN structures for image classification problems since the data size and classification categories in this project are relatively simpler. Therefore, it is not necessary to construct numerous complex layers.

6.4.2. Model Training

During the training procedure, 80% of the preliminary training data was split into the training set, and the remaining 20% formed into the validation set. Since the collected dataset is balanced for every gesture category and the model is expected to perform categorical classification, the categorical cross-entropy is adopted as the loss function. The accuracy is used as the metric function to measure the model performance. The proposed model employs a mini-batch training with 32 batch size, which fits the memory requirements of the CPU and GPU to increase the available computational parallelism and provide a generalization performance improvement.

Also, the model is trained with the Adam optimizer under 0.001 learning rate to achieve satisfactory results optimization by handling the stochastic gradient descent on the noisy problem to minimize the value of the loss function. The model is trained with a maximum of 10 epochs, and it only saves the model with the highest accuracy and the lowest loss on the validation set. Finally, the saved model would be further converted to a TFLite model format after the training procedure, so it could quickly connect with any Android and Wear OS application.

6.5. Gestural Smartwatch Application

6.5.1. Gestural Interaction System

After launching the resulting TFLite model, the system is implemented to support the gestural interaction scheme with gesture detection and inference components.

- Gesture Detection Algorithm

It is necessary for the system to recognize the occurrence of the gesture movement to capture the complete window of the correct motion data, so the gesture detection algorithm is significant to keep tracking the user's hand movements and identify the motion changes. The proposed system will perform gesture detection with a motion detection mechanism described in *Algorithm 1*. It first collects sensor data to construct a small window with 10 rows (100 milliseconds) and calculates the noise average for each axis data inside the window. If the average noise value of any axis channel is larger than (initial axis value + threshold) or smaller than (initial axis value - threshold), it considers that the hand motion is detected. The threshold here was set to 0.5, so the subtle hand motion can also be detected accurately. Once the hand motion is detected, the system will continue to collect the remaining 190 data rows to satisfy the required fixed length within 2 seconds for gesture inference. Otherwise, the window will be reset for the next motion detection to prevent unnecessary gesture inferencing.

Algorithm 1. Motion detection mechanism

Sensor Data Window in 100 ms: $D \leftarrow \{d_1, \dots, d_{10}\} \in \{Ax, Ay, Az, Gx, Gy, Gz\}$

Threshold: $t \leftarrow 0.5$

Average noise value of the Window: $Avg(D) \leftarrow \frac{d_1 + d_2 + \dots + d_{10}}{10}$

if $(Avg(D) \geq d_1 + t)$ **or** $(Avg(D) \leq d_1 - t)$

$MoveDetection \leftarrow True$

else

$MoveDetection \leftarrow False$

- Gesture Inference & Operations Mapping

After a complete motion collection cycle, the system passes the collected data to the deployed TFLite model for inferencing the matching gesture with the corresponding probability (confidence score). When the confidence score of the resulting matching gesture is higher than 0.5, the gesture result will be translated to the corresponding operation command to operate the application layout elements or execute the functions based on different application settings. Finally, the motion detection mechanism will be triggered again to start a new detection cycle for the next gesture input.

6.5.2. Gestural Application Instances

This gestural interaction system provides a quick and single-handed way to control smartwatches without physical touching. This project implemented two commonly used smartwatch application instances, including the music player and message application, to demonstrate the usability of the proposed system in different real-world layout settings. Those applications are controllable by the designed gestures to achieve touch-free interaction.

- Music Player

Figure 12. Functions of the music player supported by the proposed system

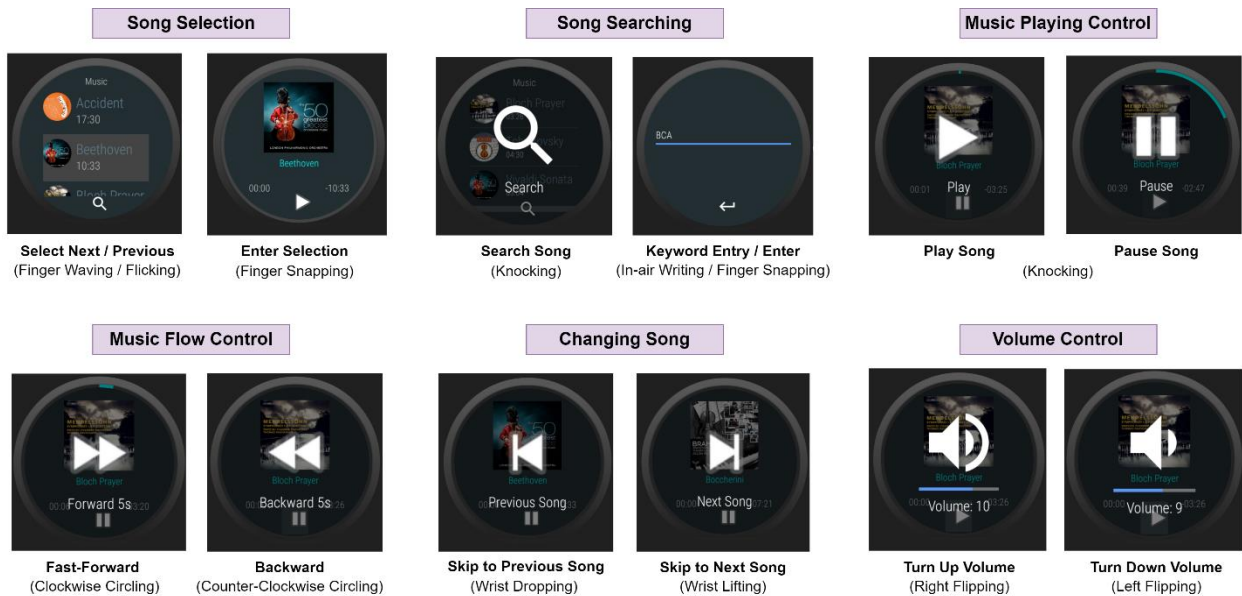
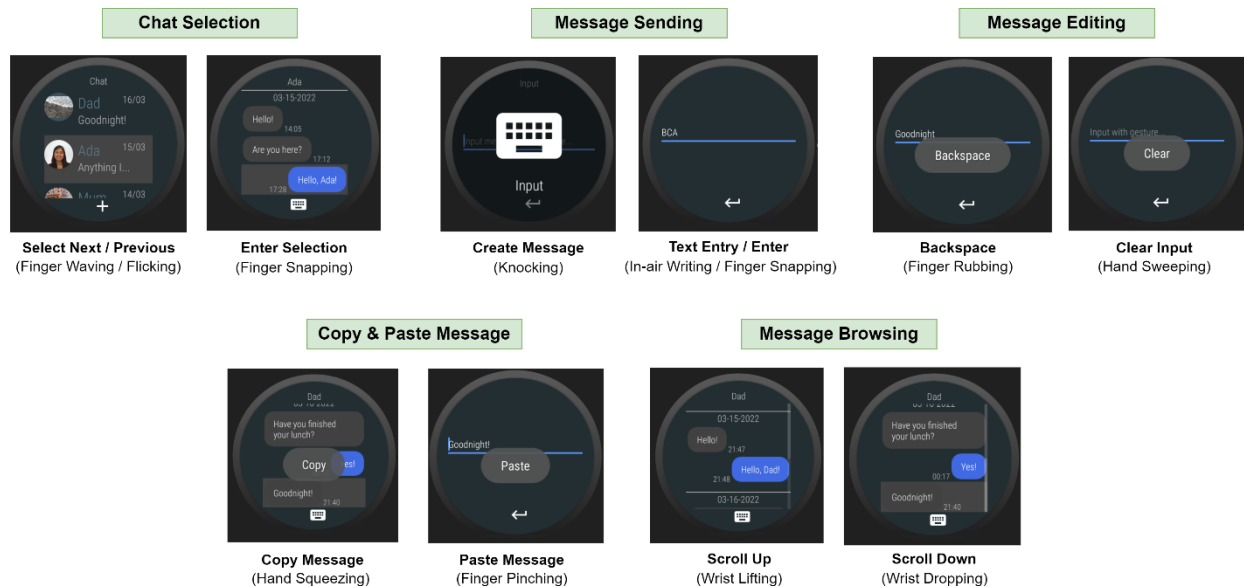


Figure 12 shows an overview of the ordinary function demonstrations for the gesture-based music player, including selecting a song from the list, searching a song with keywords, controlling music playing and flow, changing song, and controlling volume with the corresponding single-handed gestures only. Most frequently used functions can be triggered quickly with a single gesture

command, so the corresponding function buttons can be removed from the application interface to save more space for other necessary layout elements in the diminutive interface. After users perform the correct gesture, the corresponding icon and operation message are displayed to notify which operation is triggered.

- Message Application

Figure 13. Functions of the message application supported by the proposed system



Different from the music player, message applications focus on the message retrieving and text entry functions. In *Figure 13*, the gestural interaction system also allows users to perform common functions as other modern message tools, including selecting a chat, creating new messages, editing messages, and browsing message history, with the corresponding single-handed gesture only. The proposed application provides a more intuitive text entry with writing gestures for message input instead of a tiny soft keyboard to improve the input efficiency by expanding the input area and preventing pressing the adjacent key incorrectly.

It is believed that this gestural interaction scheme could also support most commercial smartwatch applications and inherent operational functions with abundant gesture vocabularies under different layout arrangements. The proposed gestural interactions system receives quick and simple user input without being limited by screen size and provides complete visibility of the interface for users by eliminating finger occluding. It also remains capable of precise selection and text entry with a sequence of gestures and quickly accessing the essential functions with a single gesture to reduce the physical effort and mistakes during diminutive interface interaction.

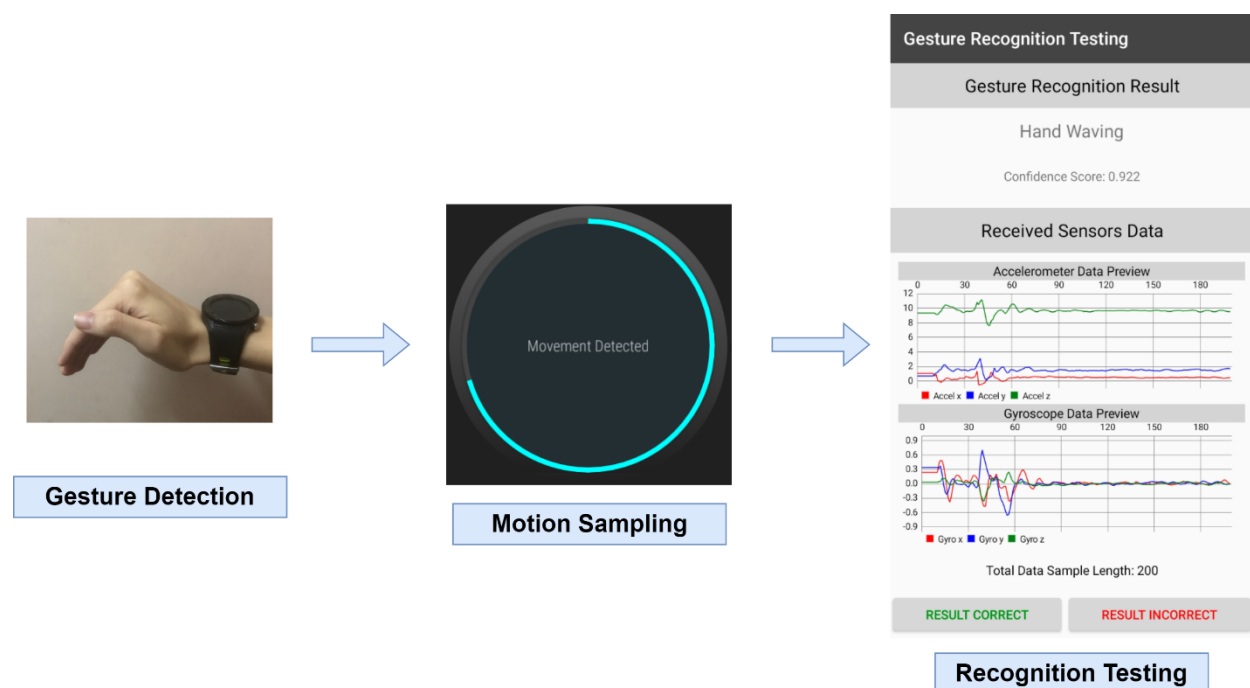
7. System Testing & Result

7.1. Performance Testing Procedures

After successfully training the deep learning model, it is significant to evaluate its generalization performance by measuring different metrics. The recognition performance was first verified by leave-one-person-out cross-validation (LOOCV). It split the gesture data from an individual participant into a test set, while the remaining data was used for training. The test set represents the motion data from unfamiliar users to measure the model performance on user-independently identifying all 21 designed gestures (including null gesture class).

Additionally, the TFLite model would be manually tested after being launched into the smartwatch. One of the experiment participants was requested to wear the watch and perform gestures to test the actual recognition accuracy for familiar users under the proposed gesture detection mechanism. Each designed gesture was performed 20 times to measure the rate of true-positive and false-positive recognition during the actual using situation. To manage the testing result easily, the paired mobile application assists the data receiving for data visualization and gesture classification when the gesture motion is detected. It displays the confidence score and the inference result of each data submission for result recording (*Figure 14*).

Figure 14. Illustration of testing procedure & interface



The following four indicators, including the accuracy (rate of correct recognition), precision (rate of true-positive recognition), recall value (sensitivity of recognition), and f1-score, were calculated to evaluate the model recognition performance. The following equations show how the mentioned indicators are derived.

Equation 1. Performance measurement calculation of the 4 indicators

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative}}$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{F1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

7.2. Proposed Model Evaluation

There were two batches of evaluations, including the generalization improvement validation for data augmentation by comparing the performance of the same proposed model with and without data augmentation. The second batch of evaluations compared the performance of the proposed multi-head network with other existing networks to justify the choice of the proposed network.

7.2.1. Data Augmentation Performance Evaluation

Table 5. Result summary of the proposed model with/without data augmentation

Validation	Accuracy	Precision	Recall	F1-score
No Data Augmentation				
Leave-one-person-out	92.86%	94.36%	92.86%	92.54%
Manual Testing	90.95%	89.32%	90.95%	89.41%
*With Data Augmentation				
Leave-one-person-out	98.29%	98.34%	98.29%	98.28%
Manual Testing	98.81%	98.88%	98.88%	98.81%

The first measurement for the proposed model without data augmentation achieve an average accuracy of 92.86% and an f1-score of 92.54% for the LOOCV. For the manual testing, the accuracy and f1-score are decreased to 90.95% and 89.41%, respectively. Those results show that although the network without adopting data augmentation can recognize the designed gesture most of the time, the recognition error rate from both familiar and unfamiliar users is relatively high. The confusion map in *Figure 17* in Appendix B shows that most validation gesture classes are confused since the small training dataset causes the data overfitting and restricts its generalization performance to recognize the new gesture samples.

The second measurement for the proposed model with data augmentation achieves a relatively high average accuracy and f1-score of 98.28% for the LOOCV and 98.81% for the manual testing. This model has an outperformed performance on recognizing and generalizing designed gestures than the previous model by increasing the accuracy by around 7%. Most of the confusion pairs from the prior model can be successfully classified. It is proved that the proposed data augmentation effectively prevents the overfitting problem and generalizes across most users. Hence, this resulting model allows the proposed gestural system to support different users. It is also concluded that a shallow 1D-CNN network structure with appropriate data augmentation is sufficient to perform the sensor-based gesture recognition with a relatively small dataset.

7.2.2. Network Performance Comparison

As the proposed network requires two separate sensor data as input, the state-of-art single-head 1D-CNN architecture that only receive a single stack of data was also implemented for comparison and evaluation. The single-handed 1D-CNN shares the same configuration with the proposed network but removes one parallel convolutional head, concatenation, and fully connected layer. The first experiment measured the performance of the single-head network with single sensors adopted to evaluate the necessity of each sensor for gesture classification. The second experiment measured the performance of a single-head CNN with both sensors employed to process the stacked accelerometer and gyroscope data sequentially. Those results were used to compare with the proposed network architecture for evaluations.

Table 6. Result summary of comparing different models

Adopted Sensor	Validation	Accuracy	Precision	Recall	F1-score
Single-head 1D-CNN					
Accel. only	LOOCV	94.88%	94.61%	94.88%	94.85%
	Manual Testing	94.29%	94.75%	94.29%	94.03%
Gyro. only	LOOCV	94.88%	95.36%	94.88%	94.69%
	Manual Testing	86.67%	85.96%	86.67%	86.67%
Accel. & Gyro.	LOOCV	96.55%	96.88%	96.55%	96.51%
	Manual Testing	96.67%	96.90%	96.67%	96.90%
*Proposed Multi-head 1D-CNN					
Accel. & Gyro.	LOOCV	98.29%	98.34%	98.29%	98.28%
	Manual Testing	98.81%	98.88%	98.88%	98.81%

Table 6 displays four indicator results of the measurements. Comparing the result of single-head 1D-CNN between adopting a single sensor and both sensors, it is evident that the average accuracy is decreased by 2% without adopting both motion sensors. According to the confusion maps (Figure 18 & Figure 19 in Appendix B), some fine-grained gesture classes are highly confused with around 25% to 50% chance. These results imply that the employment of both gyroscope and accelerometer is essential for fine-grained gestures recognition. For the comparison between single- and multi-head network architecture with both motion sensors, it is apparent that the proposed multi-head network architecture slightly improves the recognition accuracy by around 2% from the single-head network architecture. It indicates that separating convolutional structures for different sensor sources allows the network to have a more dedicated features extraction on a fully independent basis to deal with two sensor data streams, which is beneficial for classification that involves multiple heterogeneous data streams to improve the recognition performance.

7.2.3. Recognition Confusion Evaluation

Finally, it is also necessary to examine the category level performance of the proposed multi-head CNN model. The gesture classification results of the LOOCV and manual testing are summarized as a confusion matrix in Figure 15. It shows that the model can accurately recognize most of the gesture classes, mainly resulting in nearly 100% accuracy in recognizing “Wrist Dropping & Lifting,” “Hand Flipping,” “Hand Rotation,” “Hand Sweeping,” and “Knocking.” Those highly recognizable gestures propagate massive displacement and angular changes to the sensors and

have unique motion characteristics that can be classified easily. Therefore, it is proved that the proposed system has excellent classification performance in coarse-grained gestures.

Figure 15. Confusion matrix of all validation results for the proposed 2-head network

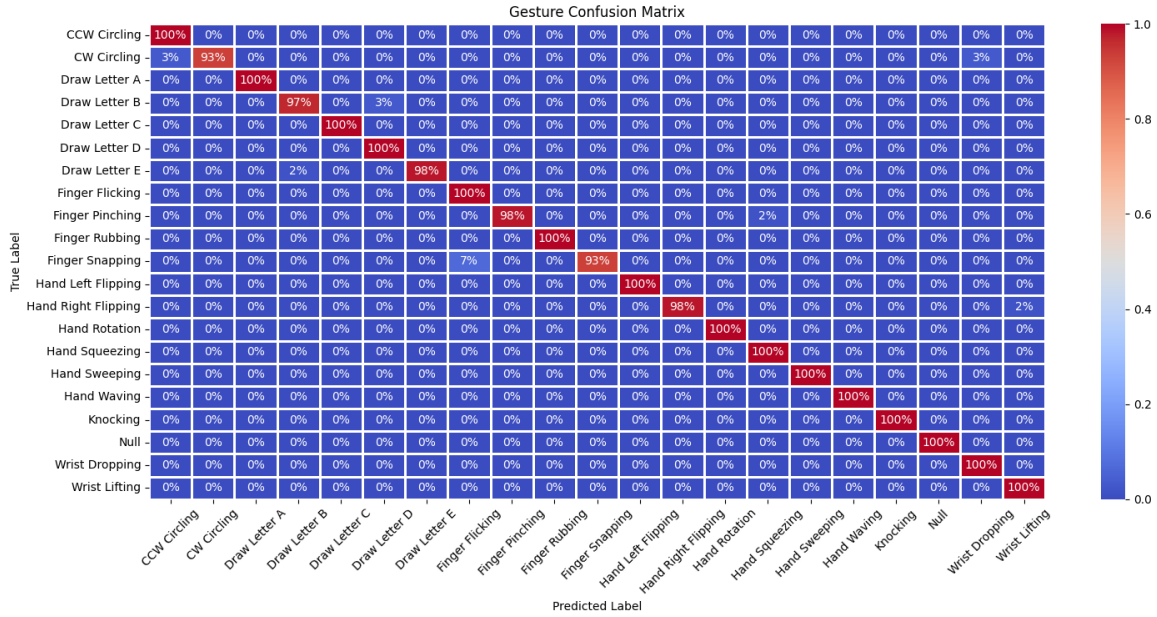


Table 7. Confusion pairs ranking

Rank	Gesture Confusion Pairs	Total error rate
1	Finger Snapping	7%
	Finger Flicking	
2	Clockwise circling	3%
	Counter-clockwise circling	
3	Draw Letter B	3%
	Draw Letter D	
4	Clockwise circling	3%
	Wrist Dropping	
5	Finger Pinching	2%
	Hand Squeezing	

However, the model recognition performance drops considerably for similar and fine-grained gestures. The confusion table given in Table 7 points out that “Finger Flicking” and “Finger Snapping” are the most confused pair. It fails to recognize “Finger Snapping” accurately, and it is always confused with “Finger Flicking” for a 7% chance (4 out of 60 times). The recognition error

also occurs by misidentifying the “Counter-clockwise circling” to “Clockwise circling” for a 3% chance (2 out of 60 times). It is observed that the movements of the top-ranked confusion gesture pairs are relatively small and similar, especially for “Finger Snapping” and “Finger Flicking.” These findings imply that the major cause of the classification confusion is the similar motion characteristics of the fine-grained gestures and the deficiency of only relying on the 3-dimensional inertial sensors to measure subtle hand movements for gesture classification.

The above evaluation results show that the proposed multi-head model with data augmentation achieves excellent accuracy in classifying 20 designed gestures performed by familiar and unfamiliar users compared with other models. Although the error rate of all confusion pairs is lower than 8%, it is still necessary to mitigate the impact of the classification error for the confused gesture pairs when using the proposed system. Thus, the system will only recognize those gestures when the corresponding confidence score is higher than 0.85. It aims to prevent obtaining the incorrect gesture result with an ambiguous confidence score that is slightly higher than 0.5 among the confused gestures. Although this adjustment increases the chance of repeating the same gesture more than once for a target operation, it can prevent triggering unexpected operations when the recognition error happens after performing those confused gestures.

8. Conclusion

8.1. Summary of Achievements & Review

This project presents a gesture-based interaction system to perform real-time detection and recognition for gestures using motion data from the embedded accelerometer and gyroscope units in an unmodified off-the-shelf smartwatch. It employs a new user input modality that consists of 15 simple hand gestures are designed to perform operational functions, and 5 in-air writing gestures are detected for text entry to achieve touch-free interaction on existing smartwatch interfaces. The multi-head 1D convolution neural network is designed to process hand motion data from different sensor sources in parallel to enhance feature extraction efficiency and gesture classification accuracy. This project also emphasizes the importance of data augmentation techniques to tackle the deficiency of a small dataset with training data enrichment and prevent model overfitting.

The proposed network achieves a high accuracy of 98.46% in a user-independent manner with the gesture motion dataset produced by 8 participants. This encouraging result suggested that a small dataset with appropriate augmentation can also construct a well-generalized gestural recognition model and minimize the required cost and amount of human-generated data. Also, compared with other state-of-art network architectures, the shallow sparsity network architecture with multiple convolutional heads according to the number of motion data sources is sufficient to achieve satisfactory recognition performance for different users. This project also demonstrates two real-life application instances enabling rapid gesture motion detection and simple gestural interaction to perform precise layout elements selection and activate diverse functions under different layout arrangements. This single-handed interaction system can be quickly launched to most commercial smartwatch applications to improve the efficiency and convenience of operating diminutive interfaces and overcome the finger occluding and fat-finger problems with a touch-free scheme. It is believed that this system can enhance human-computer interaction through an intuitive input modality with explicit body language recognition of off-the-shelf smartwatches.

8.2. Future Improvement

8.2.1. Leverage Additional Data Modalities

Currently, the proposed system only leverages the inertial sensors data to classify the designed gestures. However, some gestures with similar motion data confuse model recognition due to the sensor-based recognition deficiency. Thus, the hybrid use of additional data modalities, including audio data, can improve the recognition accuracy since each gesture produces unique finger or hand friction sounds. Most off-the-shelf smartwatches are equipped with a high-quality microphone that can capture those acoustic data to further differentiate the confused gesture patterns with hybrid sensor-acoustic recognition. It can also be used to correct the gesture detection accuracy by measuring the ambient sounds changing. Hence, the acoustic data extraction will be further studied and applied to the system in the future.

8.2.2. Increase Gestures Categories & Data Samples

Moreover, the proposed system only successfully classifies the in-air writing for 5 English characters due to cost limitations. The writing gesture samples of the remaining 21 characters and numbers are also necessary to enrich the model to completely replace the tiny soft keyboard. It is also interesting to extend the system to support the complicated writing gesture of Chinese characters for text entries. Moreover, since the current dataset and system only support right-hand gestures, it is desired to collect another dataset produced by left-handed users to improve the model for supporting ambidextrous users. The enriched labeled dataset also allows the learning model to learn more hidden features and improve the proposed system's performance.

8.2.3. Employ Few-shot learning Approaches

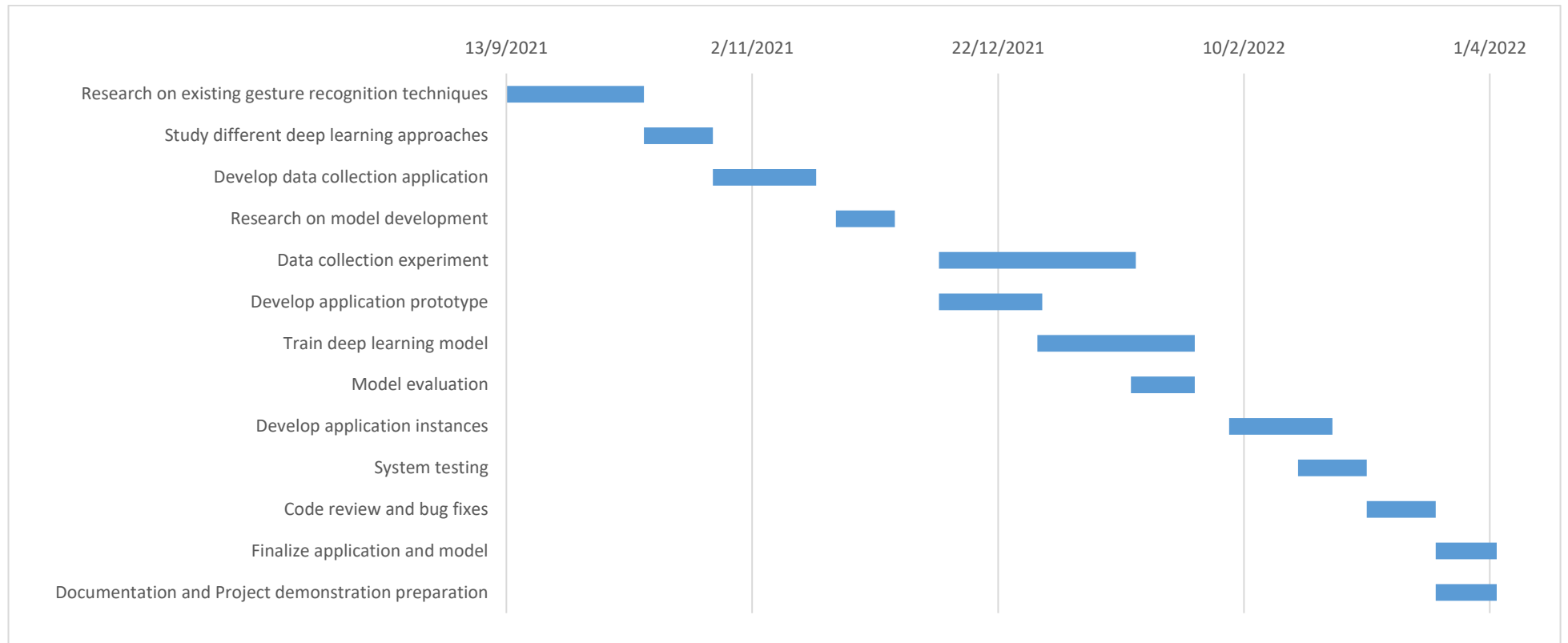
The resulting deep learning model has limited abilities to learn and generalize new gesture classes since it is required to re-train a new model with an updated dataset for supporting new gesture classes. The few-shot learning techniques with pre-training and fine-tuning can tackle this deficiency. It teaches the model to learn the learning algorithm itself rather than classifying training data. Convolutional Siamese Neural Network can be used for solving few-shot gesture learning by learning the relationship between input data and sample data pairs to determine their similarities with a small amount of supporting gestural samples. The new gestural recognition tasks can be supported by updating the support set to reduce the re-training cost. Thus, it is planned to visit the few-shot gestural learning to extend the scalability of the proposed system in the future.

9. Plan & Schedule

9.1. Project Schedule

No.	Item	Start Date	Completion Date	Duration
1	Research on existing gesture recognition techniques	13/9/2021	10/10/2021	28
2	Study different deep learning approaches	11/10/2021	24/10/2021	14
3	Develop data collection application	25/10/2021	14/11/2021	21
4	Research on model development	19/11/2021	30/11/2021	12
5	Data collection experiment	10/12/2021	18/1/2022	40
6	Develop application prototype	10/12/2021	30/12/2021	21
7	Train deep learning model	30/12/2021	30/1/2022	32
8	Model evaluation	18/1/2022	30/1/2022	13
9	Develop application instances	7/2/2022	27/2/2022	21
10	System testing	21/2/2022	6/3/2022	14
11	Code review and bug fixes	7/3/2022	20/3/2022	14
12	Finalize application & model	21/3/2022	3/4/2022	14
13	Documentation & Project demonstration preparation	21/3/2022	3/4/2022	14

9.2. Gantt Chart



10. References

- Ameliasari, M., Putrada, A. G., & Pahlevi, R. R. (2021). An Evaluation of SVM in Hand Gesture Detection Using IMU-Based Smartwatches for Smart Lighting Control. *JURNAL INFOTEL*, 13(2), 47-53. <https://doi.org/10.20895/infotel.v13i2.656>
- Becker, V., Fessler, L., & Sörös, G. (2019). GestEar: combining audio and motion sensing for gesture recognition on smartwatches. In *Proceedings of the 23rd International Symposium on Wearable Computers (ISWC '19)*. Association for Computing Machinery, New York, NY, USA, 10–19. <https://doi.org/10.1145/3341163.3347735>
- Chu, Y. C., Jhang, Y. J., Tai, T. M., & Hwang, W. J. (2020). Recognition of Hand Gesture Sequences by Accelerometers and Gyroscopes. *Applied Sciences*, 10(18), 6507. MDPI AG. <http://dx.doi.org/10.3390/app10186507>
- Chun, J., Dey, A., Lee, K., & Kim, S. (2018). A qualitative study of smartwatch usage and its usability. *Human Factors and Ergonomics in Manufacturing & Service Industries*, 28(4), 186-199. <https://doi.org/10.1002/hfm.20733>
- Hamdy Ali A., Atia A., Sami M. (2014). A comparative study of user dependent and independent accelerometer-based gesture recognition algorithms. In: Streitz N., Markopoulos P. (eds) *Distributed, Ambient, and Pervasive Interactions. DAPI 2014. Lecture Notes in Computer Science*, vol 8530. Springer, Cham. https://doi.org/10.1007/978-3-319-07788-8_12
- Hara K., Umezawa T., Osawa N. (2015) Effect of Button Size and Location When Pointing with Index Finger on Smartwatch. In: Kurosu M. (eds) *Human-Computer Interaction: Interaction Technologies. HCI 2015. Lecture Notes in Computer Science*, vol 9170. Springer, Cham. https://doi.org/10.1007/978-3-319-20916-6_16
- Kurz, M., Gstoettner, R., & Sonnleitner, E. (2021). Smart Rings vs. Smartwatches: Utilizing Motion Sensors for Gesture Recognition. *Applied Sciences*, 11(5), 2015. MDPI AG. <http://dx.doi.org/10.3390/app11052015>
- Kwon, M. C., Park, G., & Choi, S. (2018). Smartwatch User Interface Implementation Using CNN-Based Gesture Pattern Recognition. *Sensors*, 18(9), 2997. <https://doi.org/10.3390/s18092997>
- Laput, G., Xiao, R., & Harrison, C. (2016). Viband: High-fidelity bio-acoustic sensing using commodity smartwatch accelerometers. *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. <https://doi.org/10.1145/2984511.2984582>
- Moreira, B. S., Perkusich, A., & Luiz, S. O. D. (2020). An Acoustic Sensing Gesture Recognition System Design Based on a Hidden Markov Model. *Sensors*, 20(17), 4803. MDPI AG. <http://dx.doi.org/10.3390/s20174803>

- Mozaffari, N., Rezazadeh, J., Farahbakhsh, R., & Ayoade, J.O. (2020). IoT-based Activity Recognition with Machine Learning from Smartwatch. *International Journal of Wireless & Mobile Networks*, 12, 29-38. <http://dx.doi.org/10.5121/ijwmn.2020.12103>
- Muralidharan, K., Ramesh, A., Rithvik, G., Reghunaath, A. A., Prem, S., & Gopinath, M. P. (2021). 1D Convolution approach to human activity recognition using sensor data and comparison with machine learning algorithms. *International Journal of Cognitive Computing in Engineering*, 2, 130-143. <https://doi.org/10.1016/j.ijcce.2021.09.001>
- Oney, S., Harrison, C., Ogan, A., & Wiese, J. (2013). ZoomBoard: a diminutive qwerty soft keyboard using iterative zooming for ultra-small devices. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2799-2802). <https://doi.org/10.1145/2470654.2481387>
- Oluwalade, B., Neela, S., Gichoya, J.W., Adejumo, T., & Purkayastha, S. (2021). Human Activity Recognition using Deep Learning Models on Smartphones and Smartwatches Sensor Data. *HEALTHINF*. <https://doi.org/10.5220/0010325906450650>
- Poongodi T., Krishnamurthi R., Indrakumari R., Suresh P., Balusamy B. (2020) Wearable Devices and IoT. In: Balas V., Solanki V., Kumar R., Ahad M. (eds) A Handbook of Internet of Things in Biomedical and Cyber Physical System. Intelligent Systems Reference Library, vol 165. Springer, Cham. https://doi.org/10.1007/978-3-030-23983-1_10
- Rawassizadeh, R., Price, B. A., & Petre, M. (2014). Wearables: Has the age of smartwatches finally arrived?. *Communications of the ACM*, 58(1), 45-47. <https://doi.org/10.1145/2629633>
- Sun, K., Wang, Y., Yu, C., Yan, Y., Wen, H., & Shi, Y. (2017). Float: one-handed and touch-free target selection on smartwatches. *Proceedings of the 2017 chi conference on human factors in computing systems* (pp. 692-704). <https://doi.org/10.1145/3025453.3026027>
- Um, T.T., Pfister, F.M., Pichler, D., Endo, S., Lang, M., Hirche, S., Fietzek, U.M., & Kulić, D. (2017). Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. Wang, Y., Shen, J., & Zheng, Y. (2020). Push the Limit of Acoustic Gesture Recognition. *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 566-575. <https://doi.org/10.1109/TMC.2020.3032278>
- Wen, H., Ramos Rojas, J., & Dey, A. K. (2016). Serendipity: Finger gesture recognition using an off-the-shelf smartwatch. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 3847-3851). <https://doi.org/10.1145/2858036.2858466>
- Yanay, T., & Shmueli, E. (2020). Air-writing recognition using smart-bands. *Pervasive and Mobile Computing*, 66, 101183. <https://doi.org/10.1016/j.pmcj.2020.101183>

- Xu, C., Pathak, P. H., & Mohapatra, P. (2015). Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications* (pp. 9-14). <https://doi.org/10.1145/2699343.2699350>
- Zebin, T., Sperrin, M., Peek, N., & Casson, A. (2018). Human activity recognition from inertial sensor time-series using batch normalized deep LSTM recurrent networks. In *IEEE EMBC*. <https://doi.org/10.1109/embc.2018.8513115>

11. Appendix

11.1. Appendix A. Monthly Logs

October 2021

- Completed the submission of the project plan
- Completed the studies related to smartwatches sensors and the basic Wear OS application development guide
- Completed the literature review on the existing touch-free interaction scheme for smartwatches
- Focusing on the studies and research related to machine learning and deep learning algorithms for sensor-based gestures and human activities recognition

November 2021

- Completed studies and research related to machine learning and deep learning algorithms that related to sensor-based gestures and human activities recognition
- Completed the submission of the Interim report 1
- Completed Environment setup for the deep learning model training with CNN
- Developing the prototype of the Wear OS application

December 2021

- Processing the Gesture Data Collection Phase
- Adjusted the data collection component and associate it with the paired smartphone
- Completed the basic wear and smartphone application prototype
- Attempting to construct and apply the proposed CNN model on the smartwatch for real-time testing
- Refining the gesture data and model development

January 2022

- Completed data collection experiment with all the invited participants
- Completed the submission of Interim report 2
- Completed the gesture detection algorithm for the smartwatch
- Completed the preliminary development and evaluation of the proposed CNN model
- Fine-tuning the proposed CNN model

February 2022

- Completed the development and evaluation of the proposed CNN model
- Completed the basic functions development of gesture-based application instances (Message tools & Music Player) for the smartwatch application
- Launched the resulting model to the gesture-based Wear OS application by associating the gestures classes with the application functions for the testing stage
- Testing the model recognition performance and functions of the proposed gesture-based smartwatch application instances

March 2022

- Finalized and launched the design of gestures and operational functions mapping for different smartwatch application instances
- Completed the real-time testing of gestural model recognition performance and expected operational function effects for different application instances
- Finished the code review of the entire proposed systems with smartwatch, smartphone applications, and the model training platform
- Completed the bug fixes and some minor adjustments for the gesture-based application instances and the paired smartphone applications

11.2. Appendix B. Supplementary Diagrams

Figure 16. Visualization of the proposed deep learning network

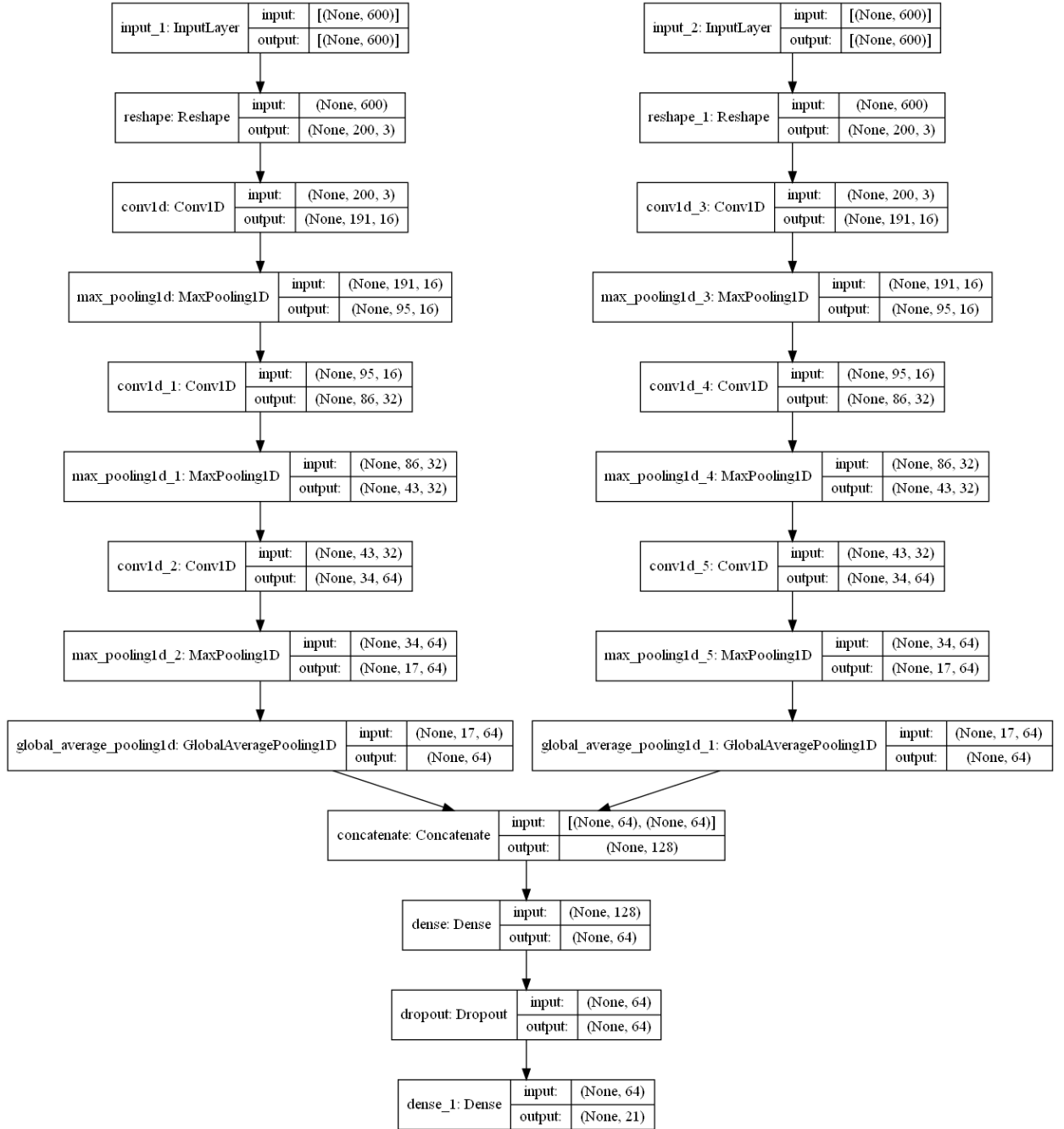
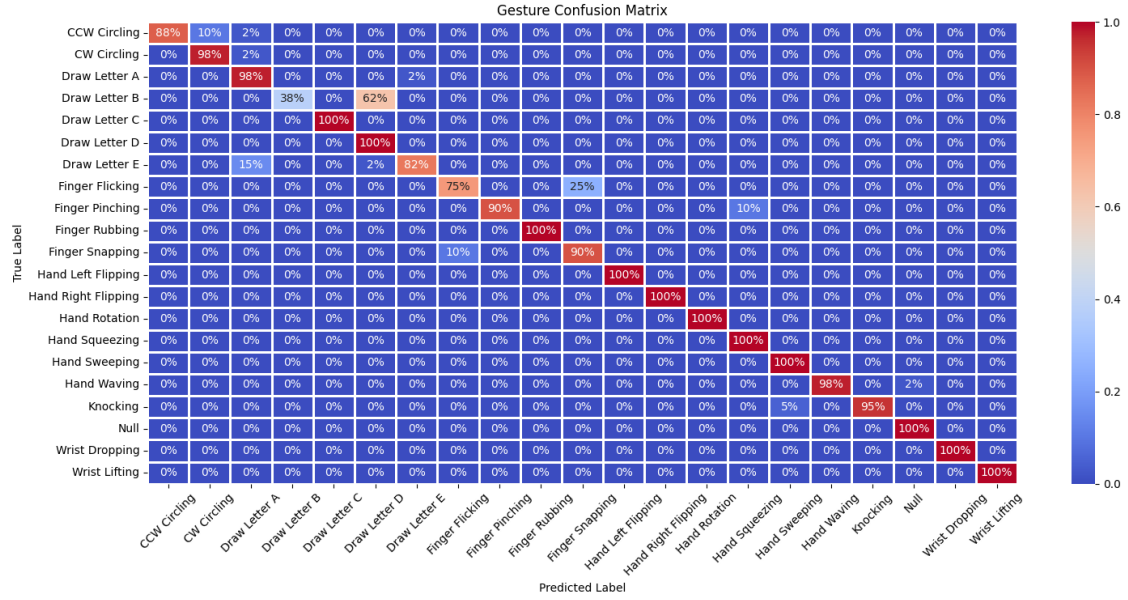
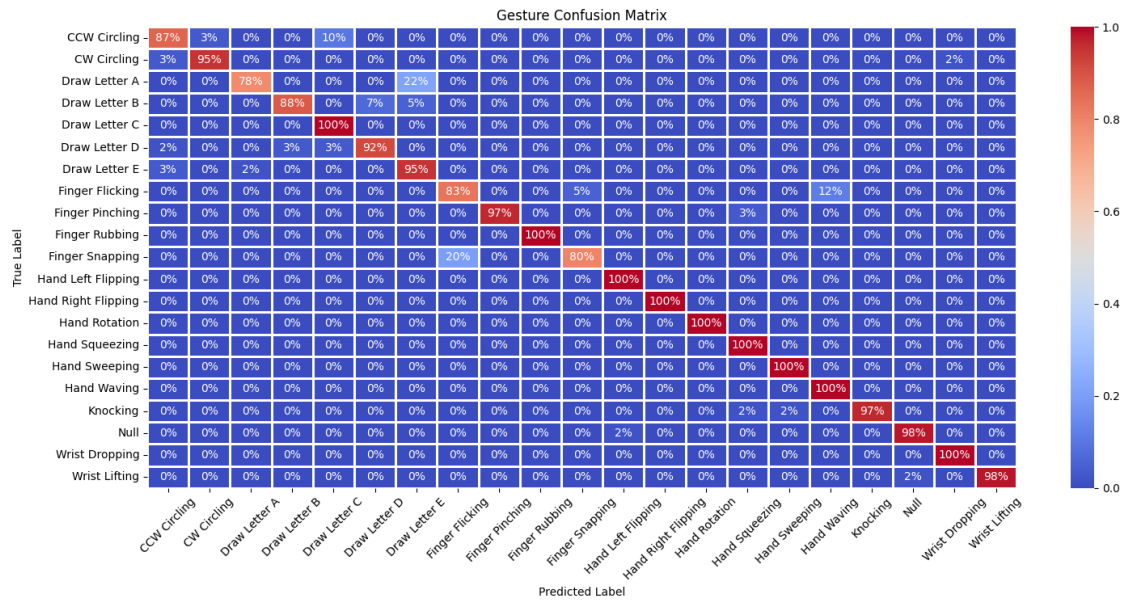


Figure 17. Confusion matrix of all validation results for the network without data augmentation



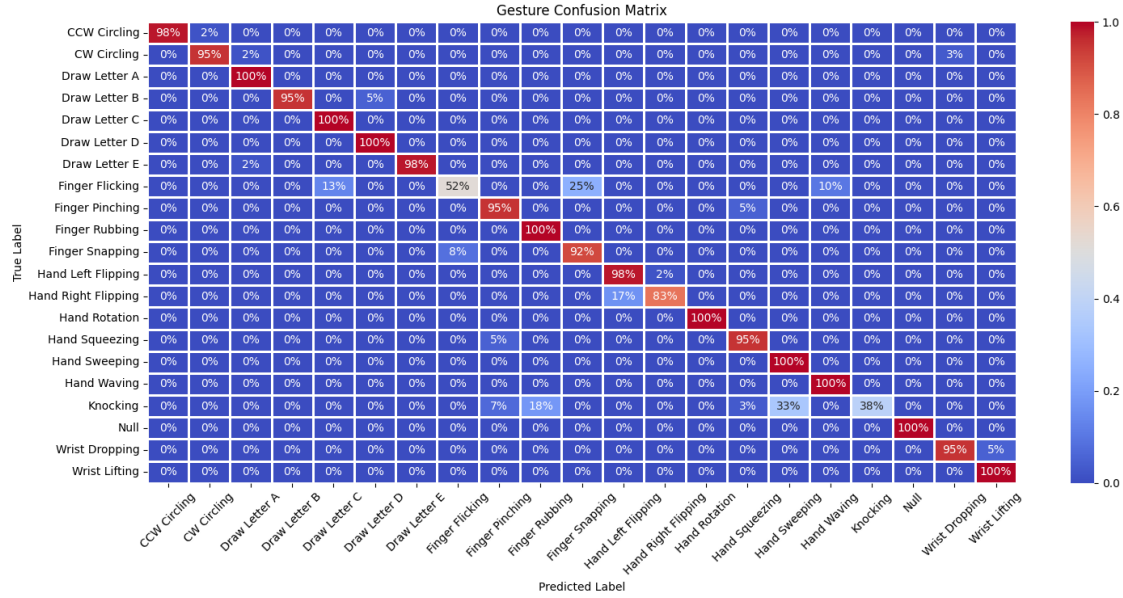
	Accuracy	Precision	Recall	F1-score
Leave-one-person-out	92.86%	94.36%	92.86%	92.54%
Manual Testing	90.95%	89.32%	90.95%	89.41%

Figure 18. Confusion matrix of all validation results for the network with accelerometer only



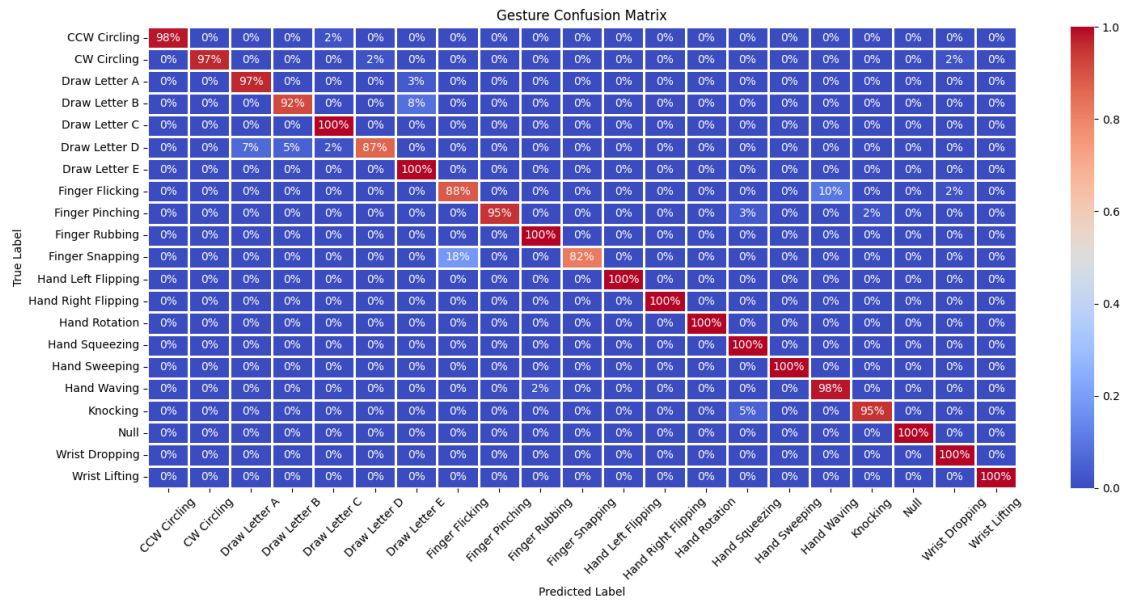
	Accuracy	Precision	Recall	F1-score
Leave-one-person-out	94.88%	94.61%	94.88%	94.85%
Manual Testing	94.29%	94.75%	94.29%	94.03%

Figure 19. Confusion matrix of all validation results for the network with gyroscope only



	Accuracy	Precision	Recall	F1-score
Leave-one-person-out	94.88%	95.36%	94.88%	94.69%
Manual Testing	86.67%	85.96%	86.67%	86.67%

Figure 20. Confusion matrix of all validation results for the 1-head network



	Accuracy	Precision	Recall	F1-score
Leave-one-person-out	96.55%	96.88%	96.55%	96.51%
Manual Testing	96.67%	96.90%	96.67%	96.90%