Department of Computer Science

BSCCS Final Year Project 2021-2022
Interim Report II

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
Date	:	January 13, 2022

For Official Use Only

1. Abstract

In recent years, wearable technologies have stimulated the development of various Internet of Things (IoT) applications that access the user status by pervasive physiological data tracking. However, existing touch-based wearables, especially smartwatches, suffer from the awkwardness of two hands occupation and finger occlusion problems, while the smartwatch interface realization is restricted by its tiny screen and interaction approach. Hence, this project presented a gesture-based touch-free interaction scheme that allows users to perform operational tasks without physical interactions on smartwatches. The proposed system employs a deep learning network to interpret arm, hand, and finger movements collected with the built-in accelerometer and gyroscope sensors for gesture recognition. The gesture-based interaction approach recognizes users' finger gestures and in-air writings as the intuitive input for the smartwatch applications without any external equipment.

This project proposed a multi-head one-dimensional convolutional neural network (1D-CNN) architecture with a novel gesture movement detection mechanism that are designed to operate locally on resource-constrained devices. This system distinguishes real-time gesture motion at any starting point to recognize 15 fine-grained gesture vocabularies and 5 in-air writing gestures. The proposed model with appropriate data augmentation achieves a user-independent gesture recognition accuracy of 98.81%, which gains the most accuracy improvement compared with different existing network designs. This project further demonstrates two application instances, including the music player and conversational application, with the proposed system to justify the convenience and feasibility of the gesture-based interacting approach.

2. Summary of Revisions & Additions

Since the interim report 1, most revisions and additions were made in section 5 (Proposed System Design). In section 5.3.1, some of the designed gestures were changed to become more fine-grained and simpler to reduce the physical effort and complexity. The data collection component in section 5.3.2 explains more about the detailed flow design on data handling by including the paired mobile application and desktop to assist the data collection. For the deep learning model employment in section 5.3.3, the novel multi-head CNN was proposed to replace the previous single-head CNN since it is desired to investigate the improvement of the novel structure compared with different existing network structures. For the interaction component in section 5.3.4, a simple gesture detection mechanism replaced the previous two-second iterative interval collection scheme. The system now performs motion detection and capturing at any starting point rather than waiting for each cycle's end to perform gestures. Moreover, the conversational application was implemented as another application instance to demonstrate the proposed system's feasibility.

Most of the descriptions and figures in section 5 were revised and changed to prevent overlapping with the content in the remaining section. The detail of the development environment section was revised and moved to section 6 as it is more related to the implementation rather than system design. Also, the testing procedures section was revised and put into section 7 as this section explains more about planning and procedures of the model result testing, which is more relevant to the result evaluation section than system design section.

Table of Contents

3. Introduction

3.1. Motivation & Background Information

Smartwatches have gained essential growth in popularity and become ubiquitous in daily life by providing high mobility and user status-awareness that smartphones cannot achieve (Chun et al., 2018). Currently, touch-based interaction is the dominant scheme of smartwatch controlling, which benefits from direct visual element interactions 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 wrist-mounted panel. Different from smartphones, smartwatches provide high mobility for recognizing physical activities with built-in inertial sensors. Wrist-mounted is the major characteristic that allows smartwatches to accurately collect physiological data through continual wrist connection (Rawassizadeh et al., 2014).

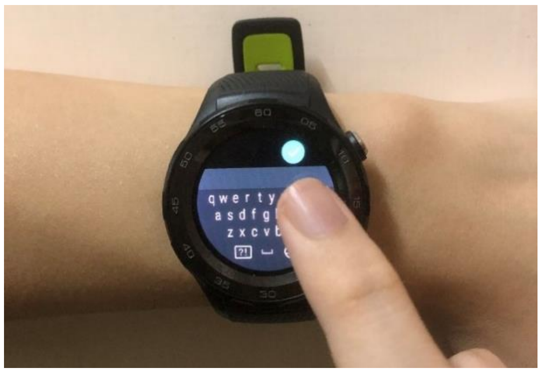
To explore the potential usage of smartwatches, various machine learning models are developed to classify the specific arms and hand movements pattern with wrist-mounted data (Mozaffari et al., 2020). Recently, there has been an increasing awareness on utilizing wearables-based recognition to investigate 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 learning models 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 convenience and potential of smartwatches, surveys and literature investigations suggested that existing faulty interaction limits their 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. Touch interaction depending on tiny soft keyboard and buttons 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 suffered from disabilities (Kurz et al., 2021). Additionally, smartwatches require users to operate with a tiny touch panel while the user's fingers can easily occlude over half of the watch face and overshoot the adjacent buttons (Oney et al., 2013). Users frequently bend their finger to minimize finger occlusion and repeating 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. Illustration of finger occlusion for using smartwatch touchscreen



Alternative touch-free inputs modalities are necessary to mitigate those touch interaction shortcomings. Though hand motion recognition has the potential to expand interaction area from the restricted screen, it is still unclear how accurate and precise the hand motion in different scales can be recognized and leveraged to improve existing touch interaction deficiencies.

3.3. Project Aims & Objective

This project aims to employ deep learning models to recognize subtle hand movements for tasks operating on off-the-shelf smartwatches to address the above problem. The primary objective is to validate smartwatch hand motion recognitions performance in different preciseness, including arms, wrist, and finger motions. Rather than heavily relying on touch interactions, the suitable and feasible one-handedness interactions will be investigated to expand usage scenarios by leveraging different scales of single-hand gestures, including finger motions and finger writing with deep learning. The gesture-driven interaction is expected to enhance the user experience by reducing physical effort and optimizing convenience for commodity-level smartwatches.

3.4. Project Scope & Deliverables

This project's scope consists of developing a Wear OS application to collect the real-time motion data representing the designed gestures and hand motions from the embedded sensors with signal processing. The sensor-based deep learning model will be developed and trained with the collected dataset to recognize various in-air fine-grained gestures and index finger writing. The resulting model will be integrated with the smartwatch application to achieve a touch-free gesture-based interaction that requires less attention to perform and have symbolic meanings to map with ordinary operational functions and text entry.

3.5. Report Organization

The report is divided into three main sections. The first section explores the existing recognition techniques and background for extending smartwatch interaction. Then, the second section presents the detailed interaction scheme design, involved components, and implementation of the proposed solution. Finally, experiment results and demonstrations will explain how this proposed interaction scheme achieves the project objective.

4. Literature Review

This section reviews different existing research of gesture recognition modalities. In particular, the potential problems and improvements will be evaluated. Hence, various existing recognition algorithms and deep learning models will be analyzed to determine the feasibility and limitation of implementing touch-free interaction on the off-the-shelf smartwatch.

4.1. Review of Smartwatch Gesture Recognition Modalities

4.1.1. Acoustic-based Gesture Recognition

An 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 track the acoustic data reflected by gestures. Wang et al. (2020) applied acoustic signals to recognize finger 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 of subtle hand motions propagating through the arm. Acoustic-based recognition can classify complicated gestures by differentiating the wave's nuance generated by various fine-grained finger 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 are not suitable for implementing passive gesture recognition under different environments.

4.1.2. Sensor-based Gesture Recognition

Smartwatch inertial sensors can track the slight wrist muscle 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 different 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, are often leveraged to record the subtle gesture movement by monitoring the device movement in various research.

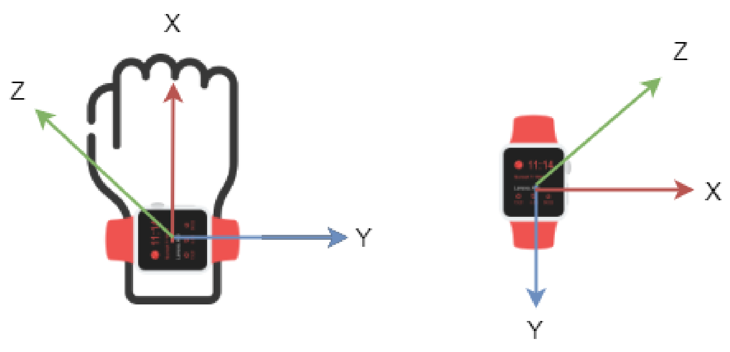
- Accelerometer data

An accelerometer is the standard sensor on modern smart devices to measure acceleration forces that represent the changes in the device displacement, orientation, and tilt. This sensor streams the acceleration data in x, y, and z-axes to represent linear movement trajectory 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 angle change of tilt in three dimensions by spinning and aligning different orientations and positions to determine the movement difference. Rotational and angular velocity data sequences can represent a movement speed corresponding to the various small-scale motions pattern.

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 arm shaking might also produce a similar record, so most of the existing research relied on arms position fixation to prevent the arm-oriented noise. Hence, a suitable learning algorithm is desired to achieve a case-sensitive gesture recognition experience by improving accelerometer-gyroscope data analysis and distinguish the data pattern between the designed gestures.

4.2. Review of Existing Gesture Recognition Algorithms

Recent research applied various pattern recognition and machine learning algorithms to process the accelerometer and gyroscope data combination to exhibit a superior accuracy in recognizing the gestures from off-the-shelf smartwatches.

4.2.1. Dynamic Time Warping (DTW)

Dynamic time warping (DTW) is the time analysis algorithms 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 training and the input sensor data sequence, which vary in performing speed for classifying 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 combination of DTW and K Nearest Neighbors (KNN) to perform a user-dependent writing gesture recognition application by comparing new samples to the reference samples of a specific user.

However, DTW only has superior gesture recognition accuracy for the dataset related to the specific user by matching the sensor data sequences with high computation costs. This user-dependent recognition is not representative for all general users and cannot generalize for 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 (SVM)

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

Nonetheless, most existing SVM-based method requires prior features extraction from the raw data, while this gesture classification accuracy often depends on the feature vector. It is required to convert the original data into representative data that the system can interpret with expert manual feature engineering and noise filtering. Data loss may occur during the feature engineering to reduce the recognition accuracy and increase the complexity of the algorithms (Muralidharan et al., 2021).

4.3. Review of Deep Learning Classification Model

Traditional pattern recognition and machine learning algorithms are restricted by their raw data processing ability, as converting the original data into representative data with manual data processing are crucial for those algorithms. In contrast, deep learning algorithms can automatically extract raw data features from data types, including images, audio, and video, for detection and classification without manual intervention.

4.3.1. Convolution Neural Network (CNN)

Convolution Neural Network (CNN) is a feed-forward neural network that is proved to have an outstanding recognition accuracy on analyzing spatial data with the concept of convolutions and pooling (Muralidharan et al., 2021). The convolutional layers detect and filter the features from the inputted spatial data, while the pooling layers repeatedly reduce the data dimensionality to distill the essential elements and reduce overfitting. It is further passed to fully connected layers for classification. Compared with SVM, the CNN model can directly interpret the original data without prior features extractions to learn the features automatically through the network and prevent data loss in the manual feature extraction process (Kwon et al., 2018).

Finger sliding gestures, which is also considered one type of HAR, can be classified using convolution to process the signal data. Kwon et al. (2018) and Yanay and Shmueli (2020) reported that the CNN model could recognize hand gestures without being restricted by user dependency. Different from the DTW algorithm, CNN can perform a more accurate and refined gesture motion analysis for the small segments with large kernel sizes and network depth (Chu et al., 2020). Though 1D CNN can only process the sensor data in a fixed time length, it is relatively straightforward to train the model by learning high-level sensor data features of specific gesture motion under increased data.

4.3.2. Long Short-Term Memory (LSTM)

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 and influences 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 sensor data samples for activities recognition (Zebin et al., 2018). Different from CNN, LSTM can classify the variable-length or infinite-length time-series data with an unknown delay period between critical activities. It reclusively extracts the sequential data according to specific sliding window size and maps it to a specific movement.

Due to its insensitivity to the delay period length, LSTM is ideal for classifying the long-term and variable sensor data sequence to solve the dynamic motion recognition with many-to-many inferencing. Tai et al. (2018) and Chu et al. (2020) applied LSTM algorithms to achieve sensor-based multiple continuous gestures recognition that is highly correlated with time series. Since LSTM has less feature extraction compatibility with many weights and biases parameters, the overall classification accuracy is worse than CNN, which focuses on extracting higher-level features in the highest computational complexity for a relatively short data input that is not highly correlated with time-sequence (Oluwalade et al., 2021).

4.4. Comparison & Discussion

Table 1. Comparison of existing research on smartwatch interaction

Literature	Gesture Presentation	Modalities	Classification	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 hand sliding gestures	Accel.	DTW, KNN, SVM, HMM	✓	~98%
(Kurz et al., 2021)	12 in-air hand 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, NB, LR, KNN	✓	98% (SVM)
(Ameliasari et al., 2021)	8 in-air hand 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%
(Chu et al., 2020)	11 in-air hand sliding gestures in sequences	Accel., Gyro.	CNN	✗	92.3%
			LSTM	✗	86%
(Tai et al., 2018)	6 in-air hand sliding gestures in sequences	Accel., Gyro.	LTSM	✗	95%

Though the traditional pattern recognition algorithms, including SVM and DTW, are investigated to recognize hand gestures with relatively small dataset and few outliers effectively, the deep learning models, including CNN and LSTM, are expected to perform better classifications due to the capability of processing large dataset and automatic feature extraction under reduced loss with powerful computational engines. The deep learning model improves the gesture recognition accuracy in a user-independent manner for every scenario by dealing with complicated time series analysis of raw sensor data produced by multiple users.

Recent research employed the deep learning model to implement gesture recognition. However, most of them only focus on a limited number of gestures and have no detailed implementation to apply the gestures recognition for improving smartwatches interactions. Hence, this project will develop an offline deep learning model to investigate the accuracy and feasibility of identifying more gesture categories and achieve the high robustness of touch-free smartwatch interaction. The prototype design and methodology will focus on leveraging the gesture-driven interaction with the in-air hand gestures to associate with fundamental operational tasks in ordinal smartwatches.

5. Proposed System Design

5.1. Overview

This project focuses on recognizing 15 in-air gestures for operational functions and 5 fingertip writing gestures for English characters entry to develop a touch-free smartwatch interaction scheme. 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: data collection component that captures accelerometer and gyroscope data of propagated hand motions, a deep learning model that classifies 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 application

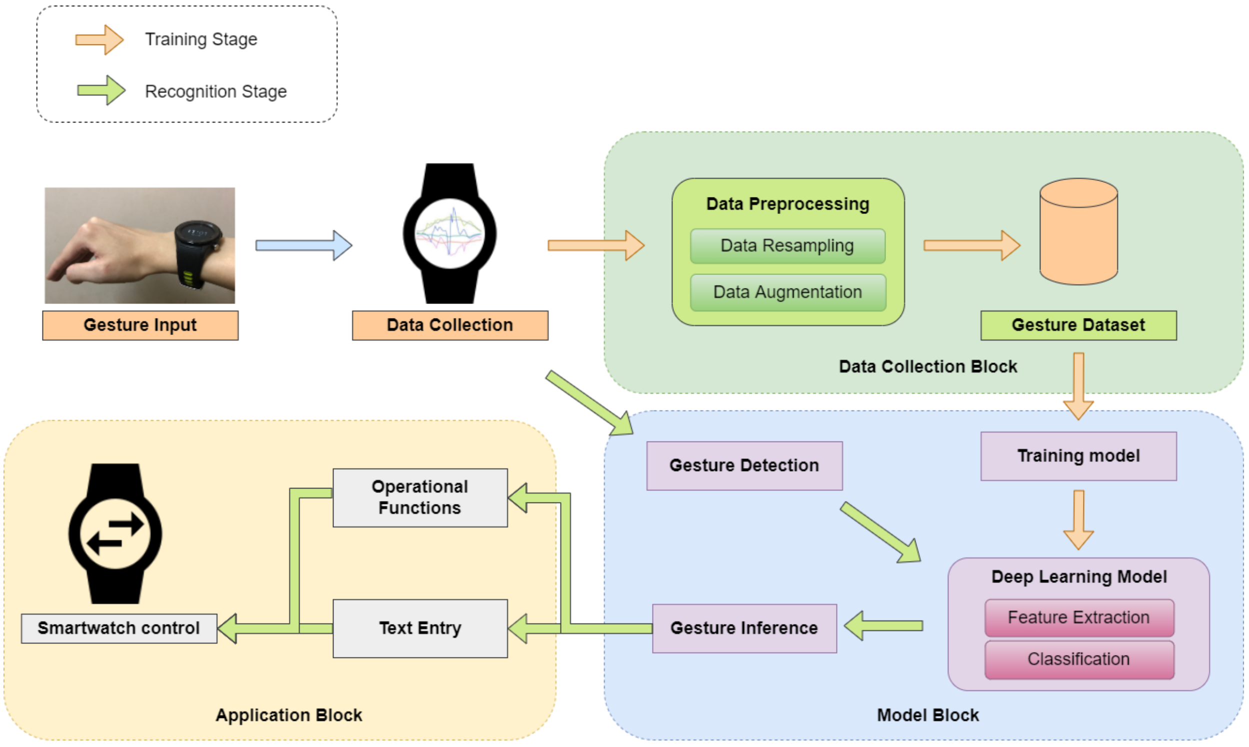


Figure 3 illustrates all technical components and the related system architecture. The proposed system is mainly divided into three blocks: Data Collection Block, Model Block, and Application Block for two stages. They depend on the data collection component to transfer gesture inputs into sensor data sequences to collect gesture samples during the training stage and control the applications during the recognition stage. Hence, the data collection component will be

implemented on the smartwatch to log the real-time sensor data that serve as a significant user input for other blocks.

- Data Collection Block

For the training stage, 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,” transform the collected data with 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 extractions and classification under the supervised learning approach.

- Model Block

The resulting model will be integrated as the core in the backend systems of the smartwatch interaction system during the gesture recognition stage. The system will receive the real-time sensor data to determine the occurrence of the gesture motion with gesture detection mechanism and then identify the designed in-air finger gesture classes. It will compute the possible performing gesture by automatically extracting the features and performing inference with the real-time captured user motion sample from the data collection component.

- Application Block

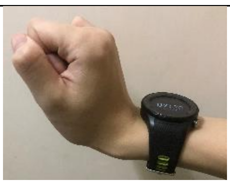
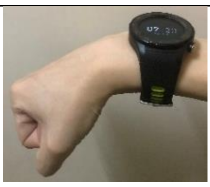
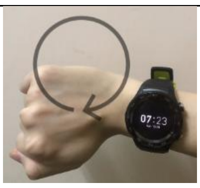
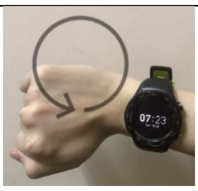
After the model computes the most possible performing gestures, the resulting gesture is treated as the primary input for the application. Each gesture class is mapped to the unique operations on the smartwatch application. When the specific gesture is recognized, the system will execute the corresponding operational functions or English character entry command. The layout elements or functions from the application will have an instant reaction according to the performed gesture.

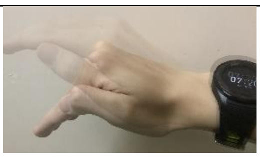
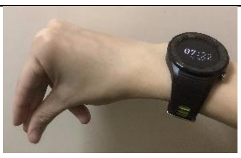
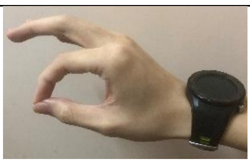
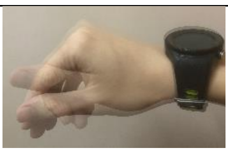
5.3. System Components

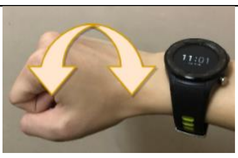
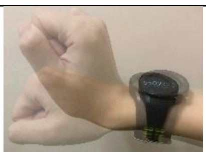
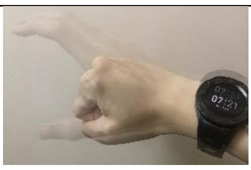
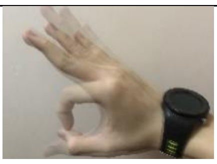
5.3.1. Gesture Vocabulary

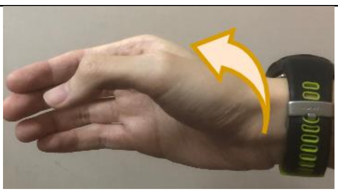
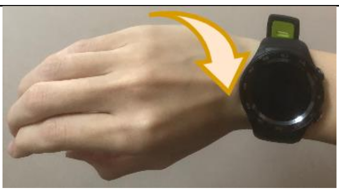
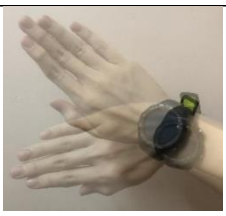
There are different operational functions required for operating smartwatches. To implement the low-effort smartwatch interactions, 15 fine-grained gestures are designed with symbolic meanings for ease of performing and memorizing, so it can avoid arm fatigue and maintain visibility of the watch face compared with the large-scale gestures. *Table 2* shows all the supporting essential operational functions with the proposed gestures.

Table 2. Design of 15 gestures for corresponding operational functions

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

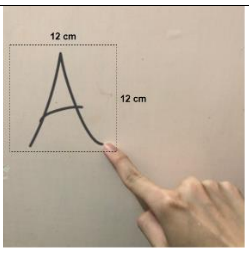
Operations	Select Previous	Select Next	Confirm selection	Back / Backspace
Gesture Vocabulary				
	Finger Waving	Finger Snapping	Finger Pinching	Finger Rubbing

Operations	Exit	Search Function	Copy	Paste
Gesture Vocabulary				
	Hand Rotation	Knocking	Hand Squeezing	Finger Flicking

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

Besides the ordinary operational functions, 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 in-air fingertip writing gestures and map to the first fifth upper-case English characters for the text entry demonstration. *Table 3* shows all the supporting basic operational functions with the proposed writing gestures.

Table 3. Design of 5 finger pointing 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 are expected to be performed in variable-length since the performance speed and writing speed to complete a specific finger gesture, or fingertip writing are different. Thus, the maximum time to complete a gesture will be set 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 classification of the deep learning models and ensures that users can complete the gestures without any burden.

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

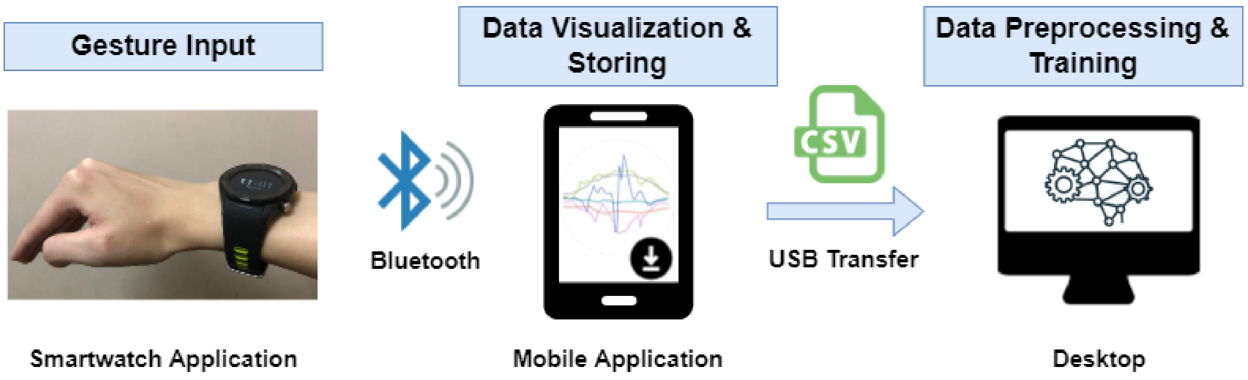
5.3.2. Data Collection Component

Since the sensor-based deep learning model is adopted for gesture recognition, a smartwatch data capture application and paired mobile application are developed to collect and store the raw data sequences from the smartwatch accelerometer and the gyroscope unit on the x, y, and z-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), represent the values of x, y, z-axis from the accelerometer and gyroscope unit according to the smartwatch linear position and angular velocity from a specific time step from 1 to T as shown as below.

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

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

Figure 4. Design of Data Collection Flow



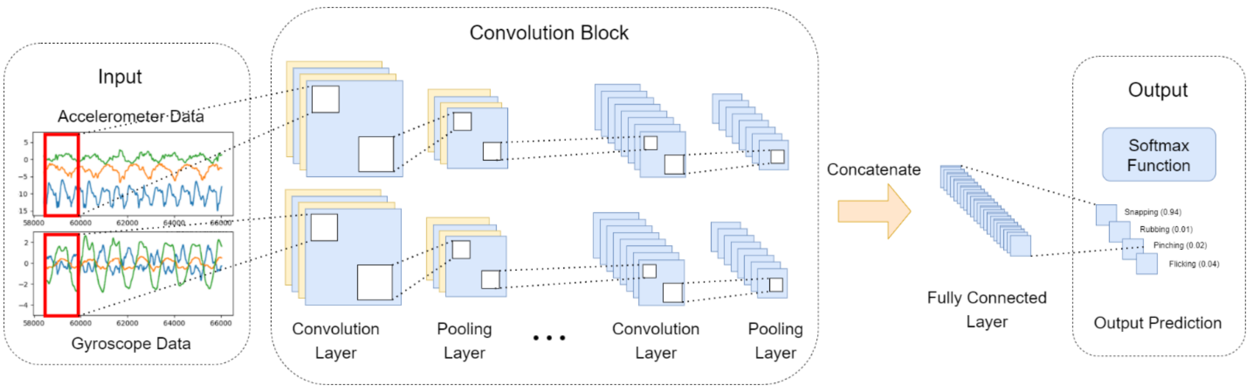
The smartwatch is used to priorly collect sufficient motion data samples with designed gesture labeling for model training. Due to the concern of computation overload and difficulty of data access with the Wear OS device, the motion data from the smartwatch will be transmitted to the paired smartphone at the end of each sample collection instead of real-time data transmission. As shown in *Figure 4*, after users complete each gesture sampling with the smartwatch, the corresponding 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 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 streams. Since all motion data will be captured under a fixed data length of two seconds, it will be more suitable to be processed by the CNN algorithm rather than LSTM. Due to the advantages of rotational and positional invariance, CNN can reduce the overall loss of the raw data to achieve high-level feature extractions under different dimensions. One-dimensional CNN (1D-CNN) architecture is superior at deriving the hidden signal features in fixed-length under smaller 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 CNN architectures 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 CNN will receive the 3-axis data of the accelerometer, while the second CNN will receive the 3-axis data of the gyroscope. The multi-head 1D-CNN model is trained with a supervised approach by associating gesture labels with the extracted features. It captures the temporal characteristics of two separated inputted motion data to extract and learn the extracted features from sensor data by associating the window to a gesture pattern.

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



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 CNN processes the segmented data across the entire sensor data on 1D convolutional and max-pooling layers in parallel. The convolutional layers will continuously extract the 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 as a noise suppressant by reducing the size of the feature maps.

After 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 for gesture classification with the output vector sequence. The resulting model will be launched into the application for real-time motion data receiving and classification.

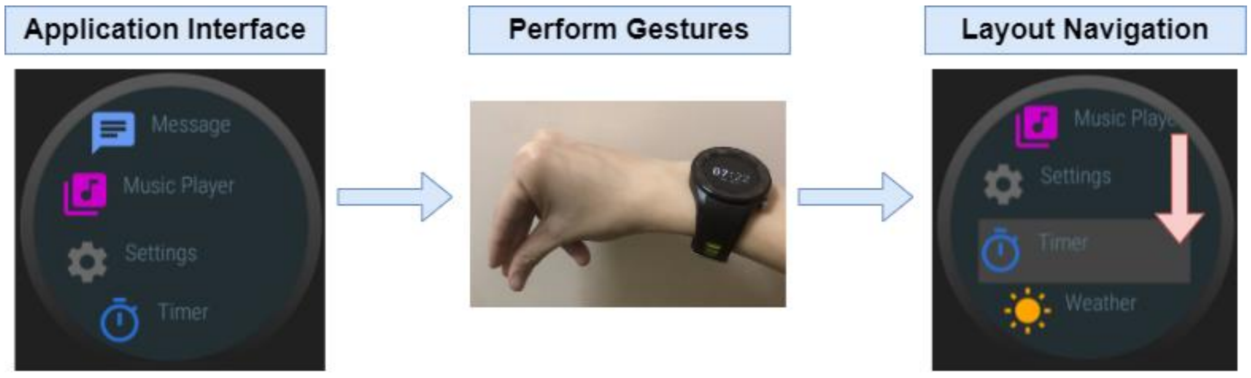
5.3.4. Application Component

During real-time gesture interaction, it is also necessary to recognize the occurrence of the performed gesture. In this project, the smartwatch will detect the motion iteratively in the background before classification. 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 classification. This mechanism aims to receive users' gestures at any starting point and prevent unnecessary classification computation if no hand motion is detected.

To demonstrate the capability of the design gestures to perform ordinary operation functions in existing smartwatch applications, the conversational and music applications will serve as instances to integrate with the single-hand interaction scheme. The proposed system will capture the hand motion for the classification model and instantly return the gesture result. The classification result will be mapped to the corresponding operation functions to control and browse layout elements, including editing input text, selecting buttons, scrolling, and switching pages in the ordinary conversational and music applications 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 on the current view, the application 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 in the gesture vocabulary. Thus, users can directly trigger the functions from the applications to save the time for locating the buttons and the space for displaying it on a tiny screen.

Figure 6. Implementation of Gesture Interaction Application



6. Methodology & Implementation

This section introduces the detailed project implementation regarding the adopted methods to address the problem. The data collection procedure, data preprocessing algorithms, the network architecture are further discussed.

6.1. Development Environment

6.1.1. Application Development Environment

As this project aims to develop a robust touch-free interaction application 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 the embedded accelerometer and gyroscope are used to capture the force and angular rate of different hand movements. Wear OS is the standard operating system for this commodity-level smartwatch for software and hardware information access. 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 gesture-based interaction application with Java programming languages. It allows the native application to utilize Android APIs for sensor management, data gathering, and control layout elements.

6.1.2. Model Development Environment

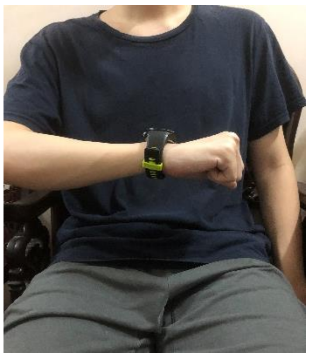
For the deep learning model development and the data preprocessing stage, TensorFlow is used for training a deep learning model with an open-source deep-learning library Keras in Python. The offline neural network is implemented and trained with TensorFlow on the desktop platform and converted to the 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 important to construct a sufficient and comprehensive dataset by collecting gesture samples from multiple users, which is also vital to evaluate 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 & Rise Mounted Hand

Standing & Rise Mounted Hand

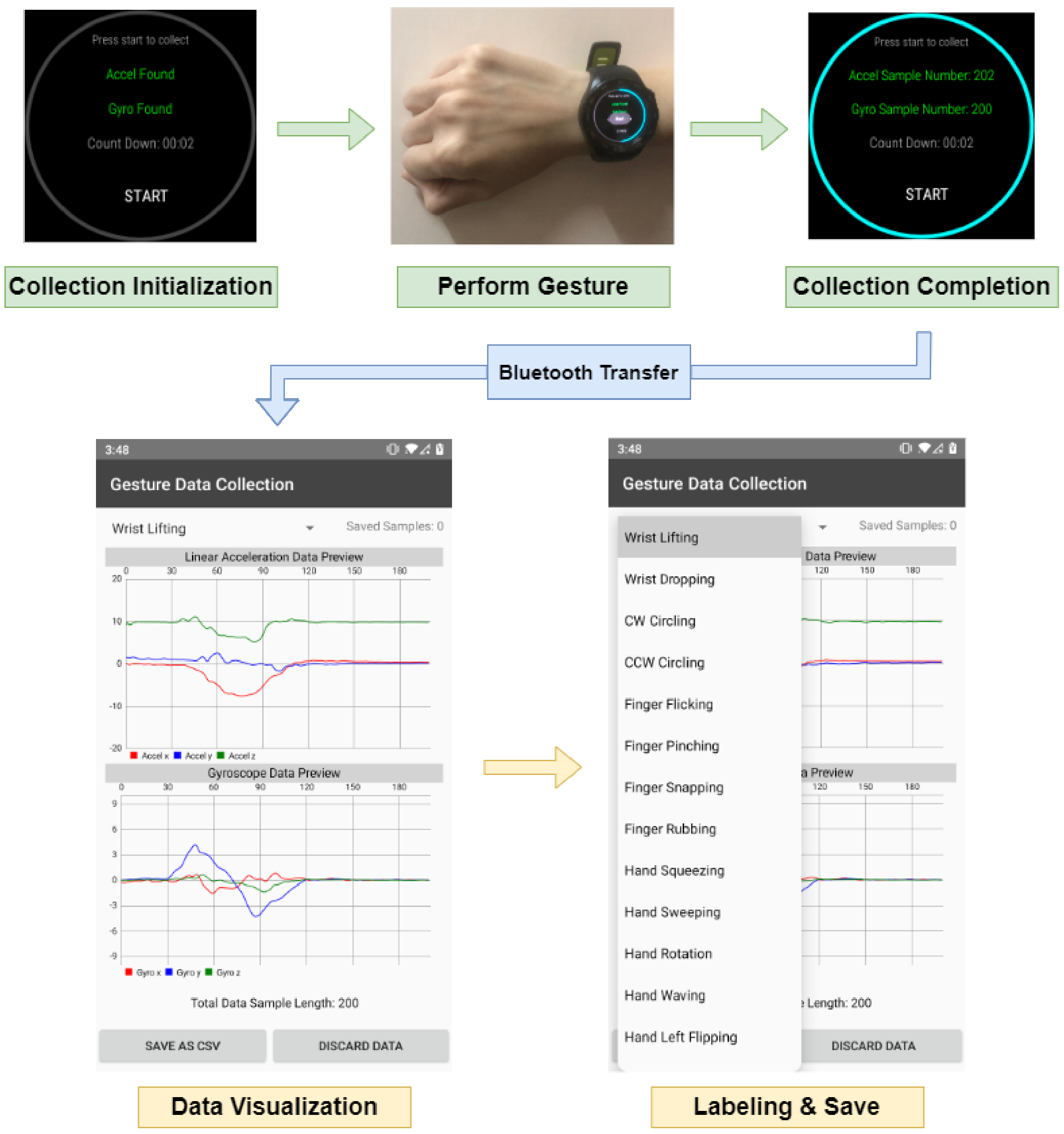
Since smartwatch users need to raise the mounted hand to focus on the watch face when interacting with the ordinal applications, the gestures were expected to be performed when users are raising the mounted hand. Thus, participants were asked to raise the mounted hand when performing the gesture during 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 on the dominant hand to collect precise hand motions data samples, so it was further assumed that the right hand is the dominant hand for this project. All the participants were confirmed to be right-handed to ensure that the sensor data were related to right-hand gestures only. The experiment was monitored by the experiment moderator to ensure the integrity of data samples.

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 with 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 current gesture sampling progress. When the timer reaches the end, the data sequence will be sent to the mobile application for further 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. Detailed Interfaces of Data Collection Flow



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 than 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 preprocessing 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 stream, which slightly exceeds the expected size (100Hz * 2 second = 200 rows) because the buffer difference of smartwatches sensors will cause an inconsistent sampling latency. Since the 1D-CNN can only interpret 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.

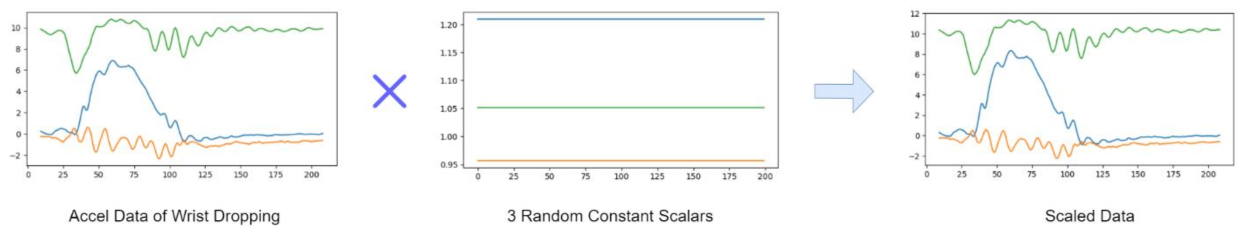
6.3.2. Data Augmentation

As the collected dataset is relatively small for training a deep learning model, it will cause the model overfitting, where the model closely corresponds to the small training data. Thus, the model will have poor performance in evaluating new data from different users or slight motion differences resulting in generalization error. Hence, to overcome the deficiency of training data, several data augmentation approaches were applied on the original acceleration and gyroscope data to create some slightly modified data copies for training dataset enrichment, which helps to avoid overfitting and improve recognition accuracy. The adopted data augmentation approaches include data scaling, time-warping, and magnitude-warping, which increase the variations to 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 in a specific window 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*).

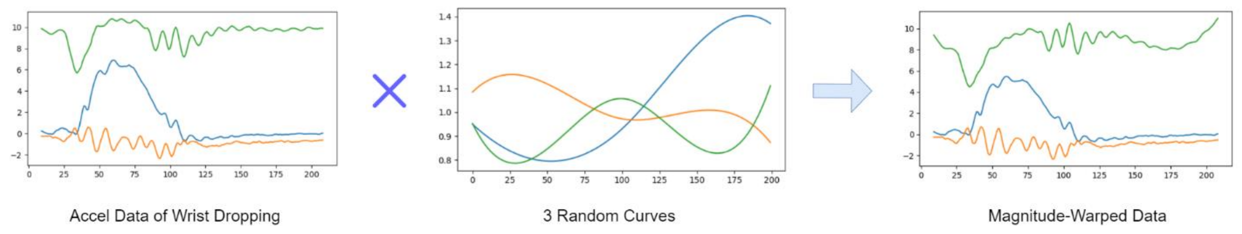
Figure 9. Illustration of Data Scaling Process



- 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 position. It can simulate more strengths variety of fingers, hands, and arms movements (*Figure 10*).

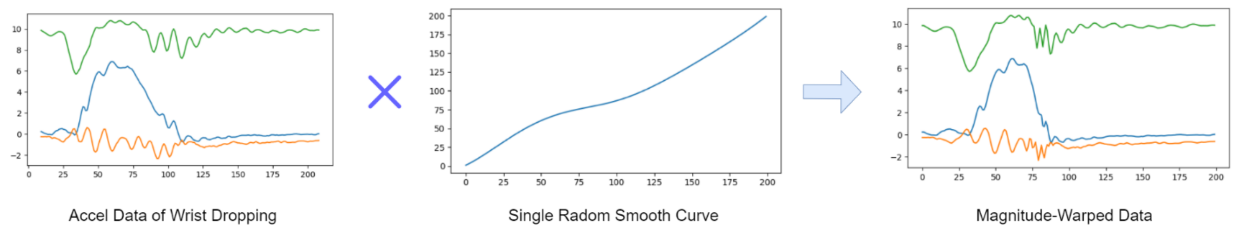
Figure 10. Illustration of Magnitude-Warping Process



- Time-Warping Approach

Time-warping approach disturbs the temporal position of a specific sample. It smoothly distorts the time intervals between samples by enlarging or compressing a specific range of 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 speeds and paces when different users are performing the same gesture (*Figure 11*).

Figure 11. Illustration of Time-Warping Process



As the data might be over-distorted if the modification impact was too large, 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 robustness of the model accuracy by improving the generalization performance of gesture recognition.

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.

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 Pool	-	-	-	-	2 * (64)
2-head Drop out (0.5)	-	-	-	-	-

Concatenate	-	-	-	-	(128)
Fully Connected	64	-	ReLU	8,256	(64)
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 the 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, while the global average pooling is used to replace the flattened 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 directly feed the result into next layer.

A dropout with a percentage of 0.5 is used for each network head before outputting for concatenation. 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.

The generated feature maps from the two network 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 before passing to the softmax layer for the classification result. 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 superior 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. System Application Instance

6.5.1. Application Framework

After launching the resulting TFLite model, the proposed system is implemented as the application framework to support the gesture-based interaction scheme with gesture detection and classification 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 targeted motion data, so the gesture detection algorithm is significant to keep tracking the user's hand movement and identify the sudden motion changes. The proposed system will perform gesture detection with a high-frequency motion detection mechanism described in *Algorithm 1*. It first collects sensor data to construct a small cache window with 10 rows (100 milliseconds) and calculates the noise average for each axis stream inside the window. If the average noise value of any axis channel is larger than (initial channel value + threshold) or smaller than (initial channel value - threshold), it considers that the hand motion is detected. The threshold here was set to 0.5, so the fine-grained 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 motion classification. Otherwise, the cache window will be reset for the next motion detection to prevent unnecessary classification.

Algorithm 1. Gesture Detection

```
$\boldsymbol{Cache\ Window\ in\ 100\ ms: D \leftarrow \{a_1, \dots, a_{10}\} \in \{Ax, Ay, Az, Gx, Gy, Gz\}}$
$\boldsymbol{Threshold: t \leftarrow 0.5}$
$\boldsymbol{Average\ noise\ value\ of\ Cache\ Window: Avg(D) \leftarrow \frac{a_1 + a_2 + \cdots + a_{10}}{10}}$
$\boldsymbol{if\ (Avg(D) \geq a_1 + t)\ or\ (Avg(D) \leq a_1 - t)}$
        $\boldsymbol{MoveDetection \leftarrow True}$
$\boldsymbol{else}$
        $\boldsymbol{MoveDetection \leftarrow False}$
```

- Gesture Classification & Operations Mapping

After a complete motion collection cycle, the system passes the collected data to the embedded TFLite model for the probability inferencing of the matching gesture. 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 according to the mapping table to update the application layout elements or execute the corresponding functions. After that, the motion detection mechanism will be triggered again to start a new collection cycle for the next gesture input.

6.5.2. Application Instance

This gesture-based application framework provides a quick and single-handed way to control smartwatches without physical touching. This project created two commonly used smartwatch application instances, including the music player and conversational application, to demonstrate the usability of the proposed system in real-world settings. Those applications are controllable by the designed gestures to replace the deficient touch-based interaction scheme.

- Music Player

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

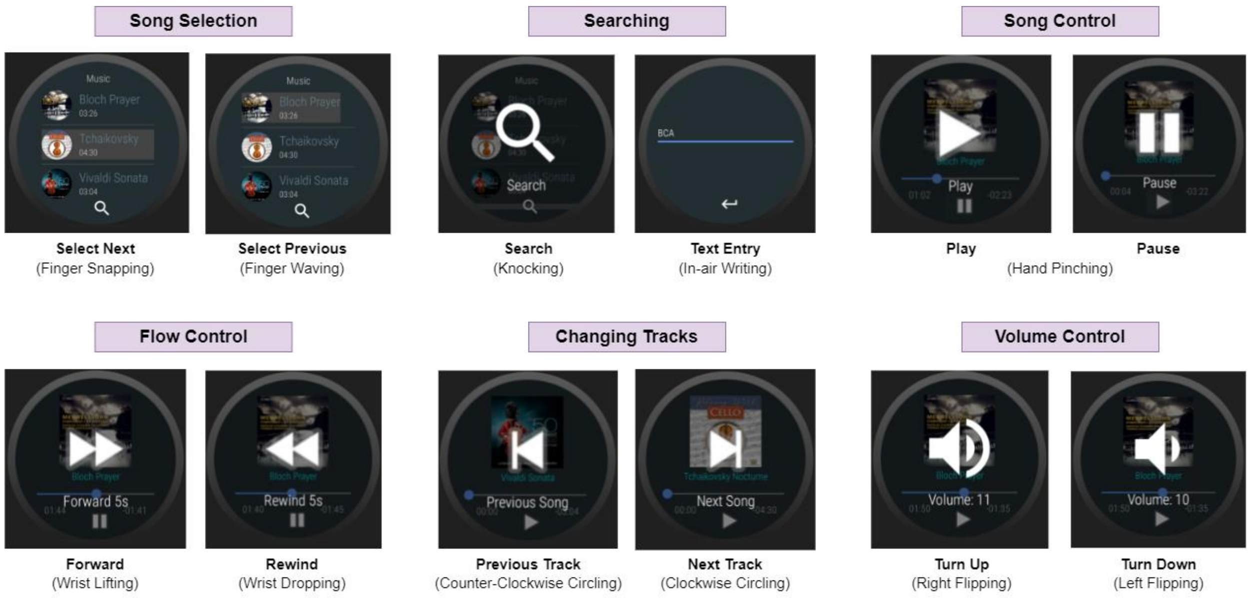
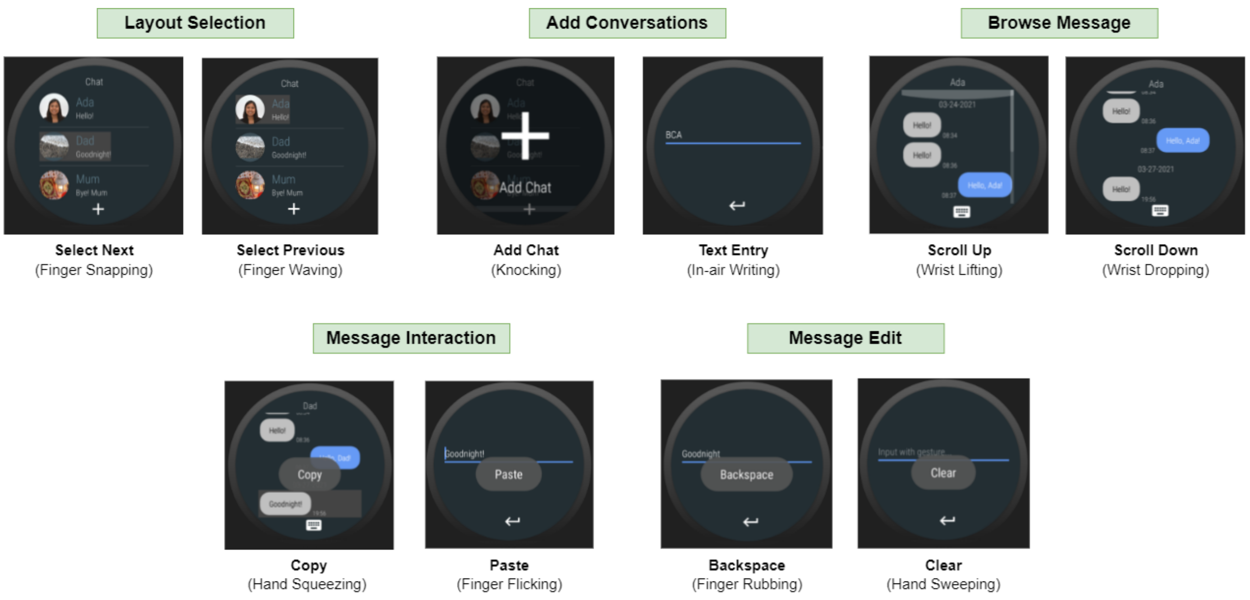


Figure 12 shows an overview of all the ordinary functions demonstration for the gesture-based music player, including selecting a song from the list, searching a song with keyword, playing, or pausing the song, controlling the music flow, changing tracks, and controlling volume with the corresponding single-hand gesture. Most frequently used functions can be triggered quickly with

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

- Conversational Application

Figure 13. Functions of the conversational tool supported by the proposed system



Compared with the music player, smartwatch conversational applications focus on the message retrieving and text-entry functions. As shown in *Figure 13*, The gesture-based version also allows users to perform common functions as other modern conversational tools, including selecting a chat from the list, adding conversations, browsing message history, message editing, copying, and pasting, with the corresponding single-hand gesture. The proposed application provides a more intuitive operation with writing gestures for message input instead of using a tiny soft keyboard to improve the text-entry efficiency and prevent overshooting key buttons.

It is believed that most of the existing smartwatch applications could also be controlled with the designed gesture-based interaction scheme. The proposed system remains capable of performing precise selection on the small interface with a sequence of gestures or quickly accessing the essential functions with a single gesture.

7. Result & Future Improvement

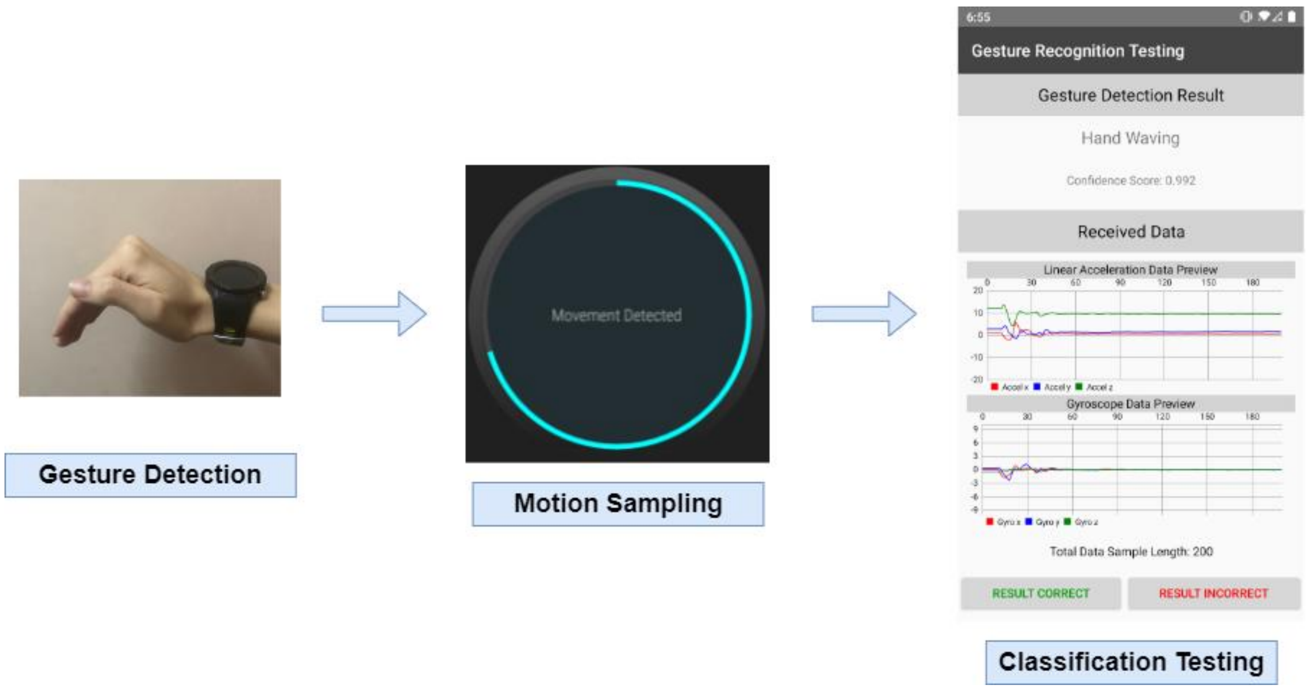
7.1. Classification Model Performance

7.1.1. Performance Testing Procedures

After successfully training the deep learning model, it is significant to evaluate the generalization performance on the data samples by measuring different metrics. The recognition accuracy of each batch was first verified by leave-one-person-out cross-validation (LOOCV). It split the dataset from the specific individual participant into 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 classifications 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 the testing interface



Hence, 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

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

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

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

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

7.1.2. 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.

- 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	98.81%	98.88%	98.88%	98.81%
Manual Testing	99.29%	99.34%	99.29%	99.28%

The first measurements 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 16* from the appendix 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 gestures. Therefore, the model without adopting data augmentation is not suitable to be launched to the proposed smartwatch application.

The second measurement for the proposed model with data augmentation achieves a relatively high average accuracy and f1-score of 98.81% for the LOOCV and 99.28% for the manual testing. This model has an outperformed performance on recognizing gestures from unfamiliar users than the previous model by increasing the accuracy by around 7%. Most of the confusion pairs from the previous model can be successfully classified with data augmentation. It is proved that the proposed data augmentation effectively prevents the overfitting problem and generalize across most users. Hence, this resulting model allows the proposed smartwatch application 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, and there is no need to manually extract the feature compared with traditional machine learning methods.

- Network Performance Comparison

As the proposed network requires two separate sensor data streams as input, the state-of-the-art single-head 1D-CNN architecture that only receive a single stack of data streams was also implemented for the performance comparison and evaluation. The single-hand 1D-CNN shares the same configuration with the proposed network but removes one parallel CNN block, 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 interpret the accelerometer and gyroscope data sequentially. Those results were used to compare with the proposed network architecture.

Table 6. Result Summary of comparing different model

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	98.81%	98.88%	98.88%	98.81%
	Manual Testing	99.29%	99.34%	99.29%	99.28%

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 in the appendix (*Figure 17 & Figure* 18), 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, it is apparent that the proposed multi-head network architecture slightly improves the recognition accuracy by around 2% compared with the single-head 1D-CNN architecture. It indicates that separating features extraction from different sensor sources allows the network to have a more accurate and dedicated features extraction on the specific data stream, which is beneficial for classification that involves multiple data streams.

- 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 plot in *Figure 15*. It shows that the model can accurately recognize most of the gesture classes, especially resulting in nearly 100% accuracy on recognizing "Wrist Dropping / Lifting," "Hand Flipping," "Hand Rotation," "Hand Sweeping," and "Knocking." Those highly recognizable gestures propagate massive displacement and angular velocity changes to the sensors and have unique motion characteristics. Therefore, it is proved that the proposed model has excellent classification performance in distinctive coarse-grained gestures.

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

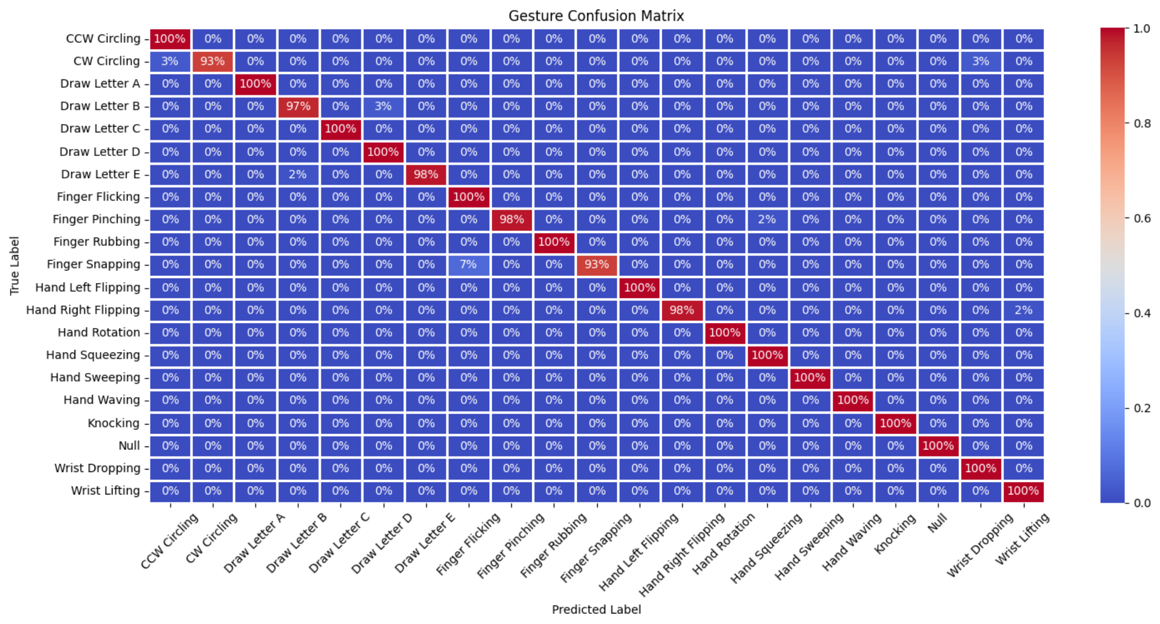


Table 7. Most Confusing Pairs Ranking

Rank	Gesture Confusing 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 the fine-grained gestures. The confusion table given in *Table 7* points out that the “Finger Flicking” and “Finger Snapping” are the most confusing pair. It fails to recognize “Finger Snapping” accurately, while 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 displacement and angular velocity 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 inherent motion characteristics of the fine-grained gestures and the deficiency of only relying on the 3-dimensional inertial sensors to measure data samples 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 10%, it is still necessary to mitigate the impact of the classification error for the confusing gesture pair when using the proposed system. Those gestures from the confusing pairs will be mapped to separated operational functions that are rarely be used sequentially. It aims to prevent unexpected operations when the recognition error happens after performing those confusing gestures.

7.2. Future Improvement

7.2.1. Leverage Additional Data Channels

Currently, the proposed system only leverages the motion sensors data, including accelerometer and gyroscope data, to classify the designed gestures. However, some distinct gestures that produce similar motion data confuse the model recognition due to deficiency of sensor-based recognition. Thus, the use of additional data channels, including audio data, helps to improve the recognition accuracy since each designed gesture produces a unique sound of the finger or hand friction. Most of the off-the-shelf smartwatches are equipped with a high-quality microphone, so they can capture those small volumes of sounds for another data channel to differentiate the gesture patterns. The audio channel can also be used to detect the occurrence of the target gesture by keep measuring the ambient sounds. As the audio data requires a professional extraction process, it will be further studied and applied to the system in the future.

7.2.2. Enrich Data Samples

The number of experiment participants and the collected gesture motion samples are relatively small due to the time and capability limitations. The recognition accuracy of the proposed system is still restricted, while some gesture confusions still cannot be classified due to the insufficient amount of data. Also, the proposed system might have a decreasing accuracy on recognizing the gesture from the completely unfamiliar users who are not acknowledged about the gesture techniques. Therefore, more participants can be invited to produce more samples to enrich the dataset and improve the generalization and classification performance. Moreover, the experiments only involve right-handed users, so the system can only receive right-hand gestures. It is desired to collect another dataset produced by the left-handed users to improve the model for supporting the ambidextrous users.

7.2.3. Increase In-air Writing Gestures Categories

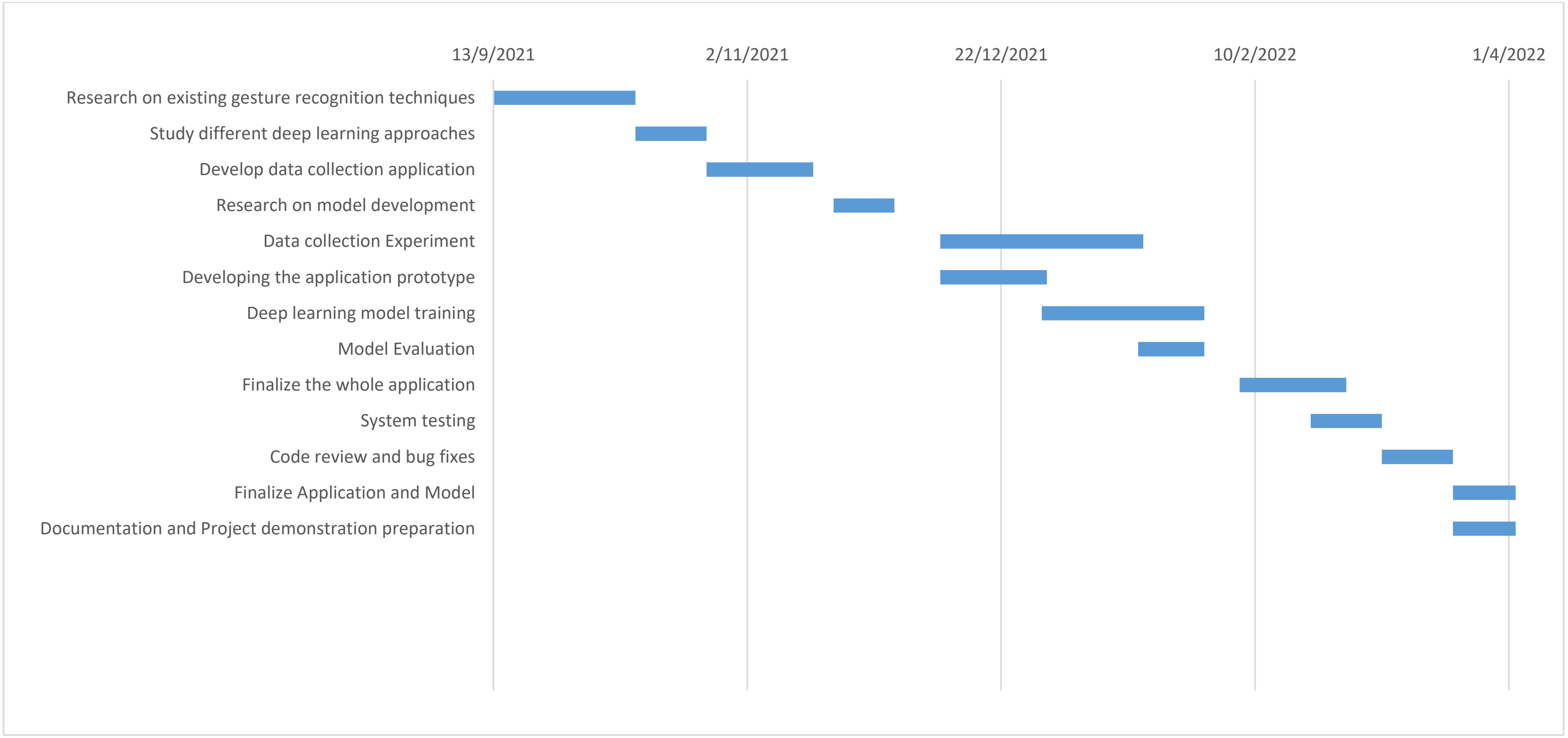
Moreover, although the proposed system successfully classifies the in-air writing related to partial English characters for text entry purposes, it only supports first 5 upper-case English characters due to the time limitations. To completely replace the traditional soft keyboard on the smartwatch, the writing gesture samples of the remaining 21 characters and numbers are also necessary to enrich the model. It is also interesting to further extend the system to support the complicated writing gesture of Chinese characters to extend the proposed text-entry method. Since the amount and keystroke of Chinese characters is more than English characters, it requires a large dataset to implement this extension in the future.

8. Plan & Schedule

8.1. 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	Developing the application prototype	10/12/2021	30/12/2021	21
7	Deep learning model training	30/12/2021	30/1/2022	32
8	Model Evaluation	18/1/2022	30/1/2022	13
9	Finalize the whole application	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 and Model	21/3/2022	3/4/2022	14
13	Documentation & Project demonstration preparation	21/3/2022	3/4/2022	14

8.2. Gantt Chart



9. References

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

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

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

Tai, T. M., Jhang, Y. J., Liao, Z. W., Teng, K. C., & Hwang, W. J. (2018). Sensor-based continuous hand gesture recognition by long short-term memory. *IEEE sensors letters,* 2(3), 1-4. https://doi.org/10.1109/LSENS.2018.2864963

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

10. Appendix

10.1. Monthly Logs

October 2021

- Completed the submission of 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 interactions 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 the submission of Project Plan
- 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

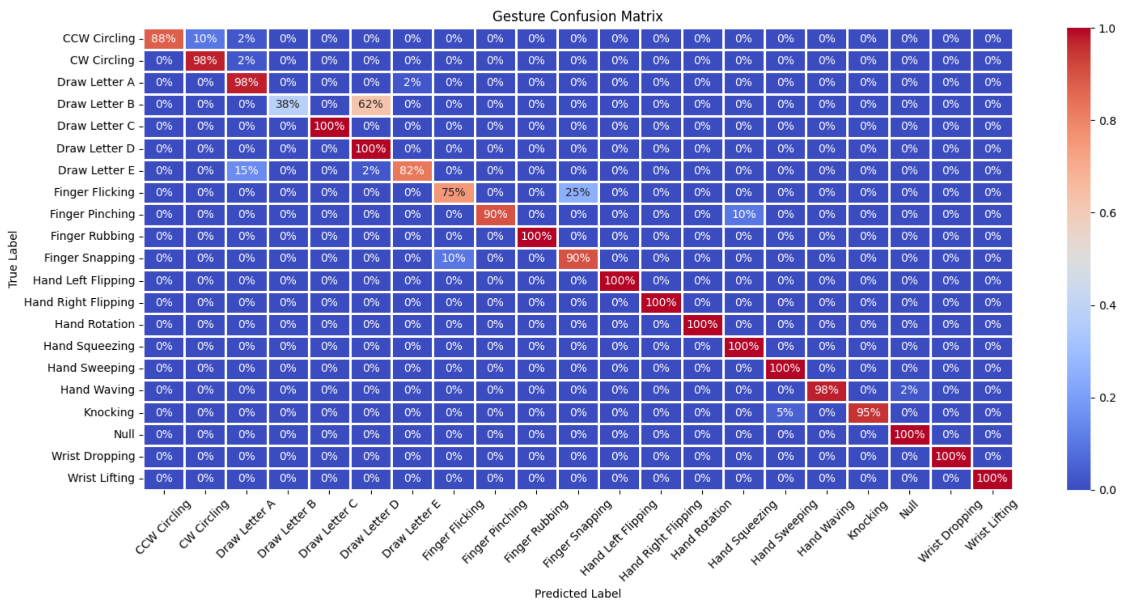
- Processing the Gesture Data Collection Stage
- Adjusted the Data collection component and associate it with the paired smartphone
- Completed the basic wear application prototype
- First attempt of constructing and applying the proposed CNN model on the smartwatch for real-time testing
- Refining the 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

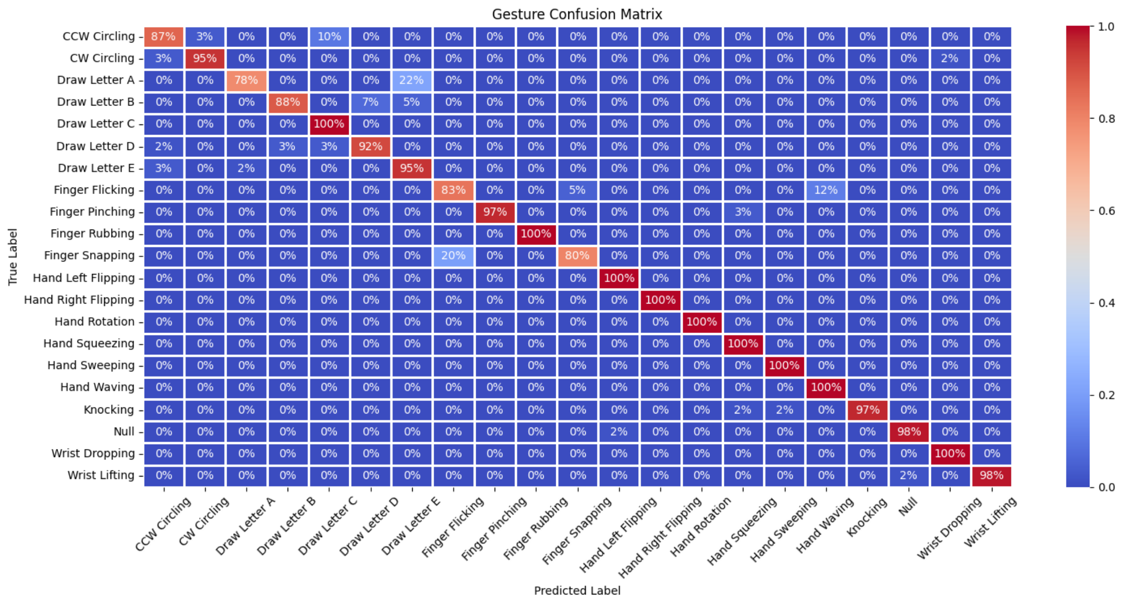
10.2. Supplements

Figure 16. Confusion plot of all validation results for the network without data augmentation



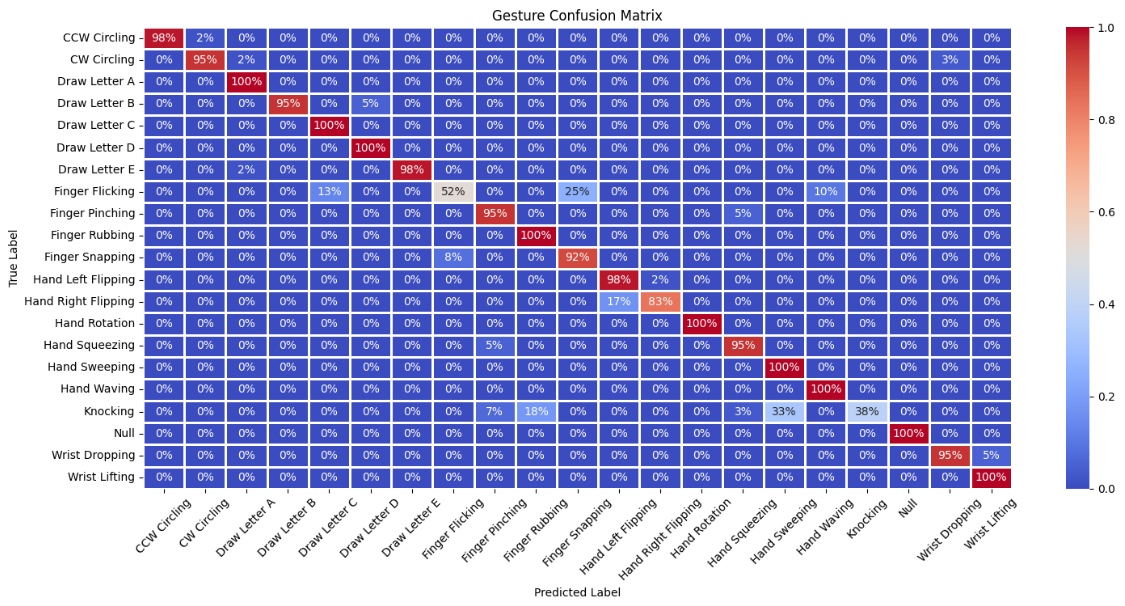
	Accuracy	Precision	Recall	F1-score
Leave-one-person-out	92.86%	94.36%	92.86%	92.54%
Manual Testing	90.95%	89.32%	90.95%	89.41%

Figure 17. Confusion plot of all validation results for the network with accelerometer only



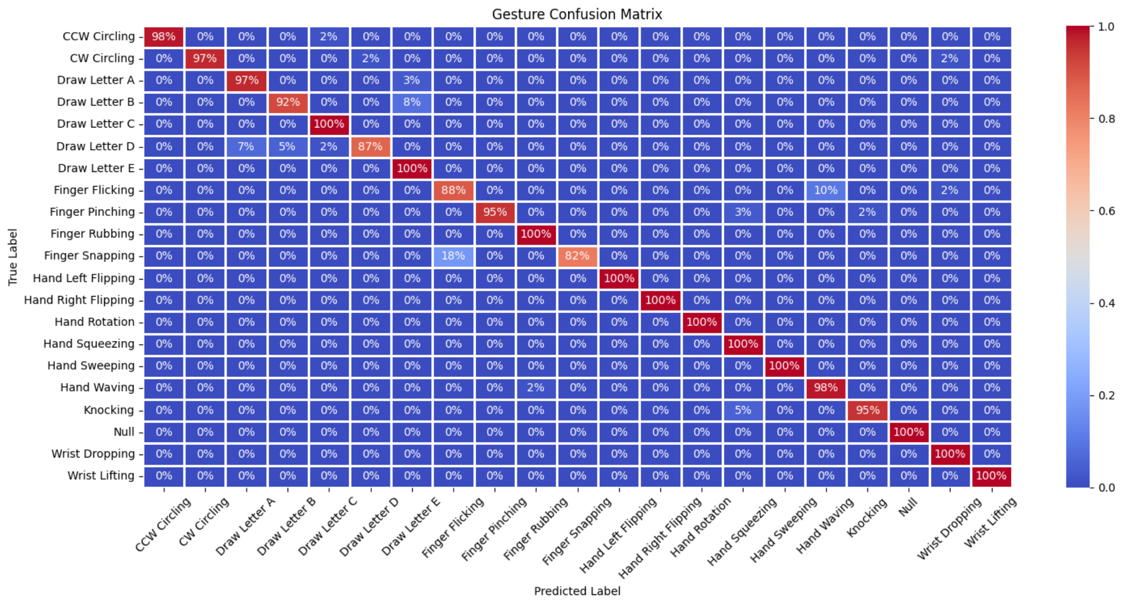
	Accuracy	Precision	Recall	F1-score
Leave-one-person-out	94.88%	94.61%	94.88%	94.85%
Manual Testing	94.29%	94.75%	94.29%	94.03%

Figure 18. Confusion plot 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 19. Confusion plot 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%