FACE TOUCH DETECTION BASED ON HAND GESTURE RECOGNITION USING WEARABLE MOTION SENSORS AND DEEP LEARNING

2022
MASTER THESIS
COMPUTER ENGINEERING

Abdullah ALESMAEIL

Thesis Advisor
Assist.Prof.Dr. Eftal ŞEHİRLİ

FACE TOUCH DETECTION BASED ON HAND GESTURE RECOGNITION USING WEARABLE MOTION SENSORS AND DEEP LEARNING

Abdullah ALESMAEIL

T.C.
Karabuk University
Institute of Graduate Programs
Department of Computer Engineering
Prepared as
Master Thesis

Thesis Advisor
Assist.Prof.Dr. Eftal ŞEHİRLİ

KARABUK
Jan 2022

I certify that in my opinion the thesis submitted by Abdullah Alesmaeil titled "FACE TOUCH DETECTION BASED ON HAND GESTURE RECOGNITION USING WEARABLE MOTION SENSORS AND DEEP LEARNING" is fully adequate in scope and in quality as a thesis for the degree of Master of Science.

Assist.Prof.Dr. Eftal ŞEHİRLİ ....
Thesis Advisor, Department of Computer Engineering

This thesis is accepted by the examining committee with a unanimous vote in the Department of Computer Engineering as a Master of Science thesis. Jan 20, 2022

Examining Committee Members (Institutions)		Signature
Chairman	: Prof.Dr.İsmail Rakıp KARAŞ (KBU)	....
Member	: Assist.Prof.Dr. Birsen GÜLDEN ÖZDEMİR (Doğuş)	....
Member	: Assist.Prof.Dr. Eftal ŞEHİRLİ (KBU)	....

The degree of Master of Science by the thesis submitted is approved by the Administrative Board of the Institute of Graduate Programs, Karabuk University.

Prof. Dr. Hasan SOLMAZ ....
Director of the Institute of Graduate Programs

"I declare that all the information within this thesis has been gathered and presented in accordance with academic regulations and ethical principles and I have according to the requirements of these regulations and principles cited all those which do not originate in this work as well."

Abdullah ALESMAEIL

ABSTRACT

M. Sc. Thesis

FACE TOUCH DETECTION BASED ON HAND GESTURE RECOGNITION USING WEARABLE MOTION SENSORS AND DEEP LEARNING

Abdullah ALESMAEIL

Karabük University
Institute of Graduate Programs
The Department of Computer Engineering

Thesis Advisor:
Assist.Prof.Dr. Eftal ŞEHİRLİ
Jan 2022, 60 pages

Wearable devices like fitness bands and smartwatches have increased in popularity in recent years. Those devices are fitted with wide range of health, fitness, and motion sensors that can be utilized to analyze and monitor body and hand activities. Being worn on the wrist or arm make them a good candidate for hand activity monitoring applications like Hand Gesture Recognition (HGR). With the worldwide spread of COVID-19 pandemic, many recommendations were issued by World Health Organization (WHO), to avoid touching the face as it was a main method for viral infections. However, most face touches are done unconsciously that is why, it is difficult for people to monitor their hand moves and try to avoid touching the face which opens the need for automatic Face-Touch Detection (FTD) solution. This thesis proposes a smartwatch application that uses small, efficient, and end-to-end Convolutional Neural Networks (CNN) models to classify hand motion and identify

Face-Touch moves. The application provides a real-time feedback and alerts the user with vibration and sound whenever attempting to touch the face, which leads to lower unconscious face touches and lower infection rates. The obtained results for recall, precision, F1-Score, and accuracy were calculated as 96.75%, 95.1%, 95.85%, 99.70% respectively, with low False Positives Rate (FPR) of 0.04%. By using efficient configurations and small models, the application can run for long hours without significant impact on battery which makes it applicable on most out-of-the-shelf smartwatches.

Key Words : Convolutional Neural Networks, Face Touch Detection, Hand Gesture Recognition, Human Activity Recognition, Smartwatch Sensors.

Science Code : 92432

ÖZET

Yüksek Lisans Tezi

GİYİLEBİLİR HAREKET SENSÖRLERİ VE DERİN ÖĞRENME KULLANARAK ELİN HAREKETLERİNİ TANIMAYA DAYALI YÜZE DOKUNMA TESPİTİ

Abdullah ALESMAEIL

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr.Öğr.Üyesi Eftal ŞEHİRLİ
Ocak 2022, 60 sayfa

Fitness bantları ve akıllı saatler gibi bileğe takılan cihazların popülaritesi son yıllarda artmıştır. Bu cihazlar, vücut ve ellerin aktivitelerini analiz etmek ve izlemek için kullanılabilecek çok çeşitli sağlık, fitness ve hareket sensörleriyle donatılmıştır. Bileğe veya kola takılmaları ellerin hareketini tanıma gibi el aktiviteleri izleme uygulamaları için bu cihazları iyi bir seçenek yapmaktadır. COVID-19 pandemisinin dünya çapında yayılmasıyla birlikte, Dünya Sağlık Örgütü (WHO) viral enfeksiyonlara sebebiyet verebilen el ile yüze temas etmekten kaçınmanın önemi üzerinde durmuştur. Fakat, yüze dokunuşların birçoğu bilinçsizce yapıldığından dolayı, insanların ellerinin hareketlerini izlemesi ve yüzlerine dokunmaktan kaçınması zordur. Bu durum otomatik olarak yüze dokunma tespiti ihtiyacını doğurmaktadır. Bu tezde ellerin hareketini sınıflandırmak ve yüze dokunma hareketlerini tanımlamak için küçük, verimli ve uçtan uca Evrişimsel Sinir Ağları (CNN) modelini kullanan bir akıllı saat uygulaması önerilmektedir. Geliştirilen uygulama gerçek zamanlı bir geri bildirim sağlamakta ve kullanıcıyı yüze dokunmaya çalıştığında titreşim ve sesle uyarmaktadır. Bu çözüm bilinçsizce yüze dokunuşlarının azalmasına ve enfeksiyon oranlarının düşmesine olanak sağlayacaktır. Elde edilen sonuçlara göre, önerilen çözüm %0.04'lük düşük Yanlış Pozitif Oranı, %96.75 ile ortalama hassasiyet, %95.1 ile ortalama kesinlik ve %95.85 ile ortalama F1-Skoruna ulaşmıştır. Verimli ayarlamalar ve küçük modeller kullanarak geliştirilen uygulama pil üzerinde önemli bir etki yaratmadan uzun saatler boyunca çalışabilir ve bu da onu kullanıma hazır akıllı saatlerin çoğuna uygulanabilir hale getirmektedir.

Anahtar Kelimeler : Akıllı Saat Sensörleri, Ellerin Hareketini Tanıma, Evrişimsel Sinir Ağları, İnsan Hareketi Tanıma, Yüze Temas Tespiti.

Bilim Kodu : 92432

ACKNOWLEDGMENT

First of all, I would like to give special thanks to my advisor, Dr. Eftal ŞEHİRLİ for his great advices, helpful insights, and assistance during the thesis project from the beginning until the end. Additionally, I would like to express my gratitude to my wife for her support and for helping me along with my friends to gather thousands of training samples while working for countless days and hundreds of hours collecting the training dataset.

CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

SYMBOLS AND ABBREVITIONS INDEX

ABBREVIATIONS

ADAM	: Adaptive Moment Estimation
ADL	: Activities of Daily Living
ANN	: Artificial Neural Networks
AR	: Augmented Reality
CNN	: Convolutional Neural Networks
DT	: Decision Trees
DTW	: Dynamic Time Warping
FFT	: Fast Fourier Transform
FN	: False Negative
FP	: False Positive
FPR	: False Positives Rate
FTD	: Face Touch Detection
GV	: Gravity Vector
HAR	: Human Activity Recognition
HCI	: Human Computer Interaction
HGR	: Hand Gesture Recognition
HR	: Heart Rate
HRT	: Habit Reversal Therapy
Hz	: Hertz
IMU	: Inertial Measurement Unit
IR	: Infra-Red
KNN	: K-Nearest Neighbor
LOSO	: Leave One Subject Out
LSTM	: Long-Short Term Memory
MCD	: Minimum Covariance Determinant

MEMS	: Micro Electro-Mechanical Systems
NB	: Naïve-Bayes
OS	: Operating System
RF	: Random Forest
RFID	: Radio Frequency Identification
RMS	: Root Mean Square
RMSPROP	: Root Mean Square Propagation
RNN	: Recurrent Neural Networks
SARS-CoV2	: Severe Acute Respiratory Syndrome Coronavirus 2
SGD	: Stochastic Gradient Descent
SNR	: Signal to Noise Ratio
SpO_2	: Peripheral Oxygen Saturation
SVM	: Support Vector Machines
TN	: True Negative
TP	: True Positive
VIO	: Visual Inertial Odometry
VR	: Virtual Reality
WHO	: World Health Organization

PART 1

INTRODUCTION

1.1. INTRODUCTION

In the current technological age, smart devices have become small enough that can be fitted in our pockets or easily worn in our bodies. The market of wearables has grown many folds in the past decade, which includes range of devices from smartphones, Augmented-Reality headsets, to wrist-worn devices which include fitness trackers, wrist bands, and smartwatches. People use wrist-worn devices to track their daily exercise, steps taken, and monitor their vitals from Heart Rate (HR), blood-oxygen, and calories spent during exercise or sport activities like walking, running, cycling, gym classes, or during playing all kinds of sports.

Nowadays wrist-worn devices like smartwatches have become fully-fledged devices that can rival the bigger smartphones in terms of hardware and software features. They are supported by major OS platforms like iOS and Android and started to add support for third-party applications. Most of these devices are equipped with wide range of useful sensors like health sensors, motion sensors, and external environment sensors.

These sensors can be utilized for Human Activity Recognition (HAR) like walking, running, sitting, or whole-body activities in general. In addition to whole-body activities, they can be used to classify hand moves and activities which have many applications such as sport actions [1] and daily-life actions [2].

Wrist-worn smart wearable devices for Hand Gesture Recognition (HGR) have been used in many studies. Example of HGR applications are Sign-Language gesture recognition where gestures can mean words or actions [3], and Human Computer

Interaction (HCI) where hand gestures can give commands to a remote computer interface [4-5].

Although, Face Touch Detection (FTD) has received more focus recently due to COVID-19 pandemic, there were many studies that took advantage of smartwatch sensors to identify hand moves towards the face. Face-touch hand move can be considered as special type of hand gestures. It shares many characteristics with other HGR gestures. However, it differs in some ways that makes it more challenging to detect. Thus, it requires additional works to overcome the obstacles.

1.2. MOTIVATION

According to World Health Organization (WHO), the Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2) which causes the COVID-19 infection, can be transmitted among people in two major ways. The first one is through respiratory droplets when a person stays in close contact with an infected person. The second one is when people touch a surface contaminated with the virus and touch their faces specially their eyes, noses, or mouths. The second way is our point of interest where an automatic FTD application can be useful. This is because when people are outside, they touch all kinds of surfaces with their hands, which can be contaminated with viruses or bacteria. After that, when they touch their faces, these germs can be transmitted through the respiratory system to their bodies.

To help in lowering infection rates, many recommendations have been issued by national health organizations to avoid face touches as much as possible. Unfortunately, many studies show that people tend to touch their faces more than 20 times per hour on average, and many of these touches are done unconsciously [6]. This makes it difficult for people to monitor their hand moves by themselves in order to avoid undesired face touches. This opens the need for automatic detection and alert solution.

The proposed solution detects face touch moves and alert users in real-time whenever they attempt to touch their faces. This helps people to lower their unwanted face-touch attempts especially when used in public places. The goal of the application is not only to alert the user when trying to touch the face, but also to train his unconscious mind using haptic feedback similar to Habit Reversal Therapy (HRT) [7]. This assists to develop a habit of avoiding unnecessary face touches even when the application is not running, which in turn leads to fewer infection rates.

1.3. NOVELTIES

The contributions of this thesis are:

- It provides complete and applied solution that runs completely on a smartwatch and uses only IMU motion sensors without requiring extra sensors, devices, or hardware components.
- Instead of approaching the problem as binary classification (Touch/No-Touch), it proposes a smart approach of dividing hand motions into 5 classes (Touch, Up, Abdominal, Normal, Stationary) which is proved to have significant impact on improving precision and minimizing false positives.
- The use of controlled data collection sessions and automatic labeling algorithm based on peak-valley analysis allows to collect large training dataset with 14k samples for each hand, which to the best of our knowledge is the largest dataset for face-touch detection.
- This work is designed to address and solve battery consumption limitations on a small device like smartwatch especially when running the application continuously for long hours. It uses the lowest configurations compared with literature from small window size, to extremely compact Convolutional Neural Networks (CNN) models.

1.4. THESIS OUTLINES

Part one (current part) of this thesis gives a brief introduction of this work like motivation, novelties, and outlines. In second part, the domain of human and hand activity recognition, its applications, and wearable devices is introduced. Third part contains a review for the literature of automatic face touch detection, considering the face-touch hand-move as a special gesture type. In addition, HGR and FTD literature for solution design, approach, and methods are reviewed. In fourth part, the proposed solution is introduced, which is a smartwatch application that uses small, efficient, and end-to-end CNN models to classify hand motion and identify Face-Touch moves. This part describes the approach and methods used to accomplish these results. Chosen setup, sensors, features, and configurations are detailed. Additionally, the collection process of a large dataset is detailed for both left and right hands and represents multiple hand motion types, body positions, and hand orientations. In fifth part, all experimental results are detailed. Then in the last part, the obtained results are compared with state-of-the-art research on face touch detection. Finally, this thesis is summarized at the end.

PART 2

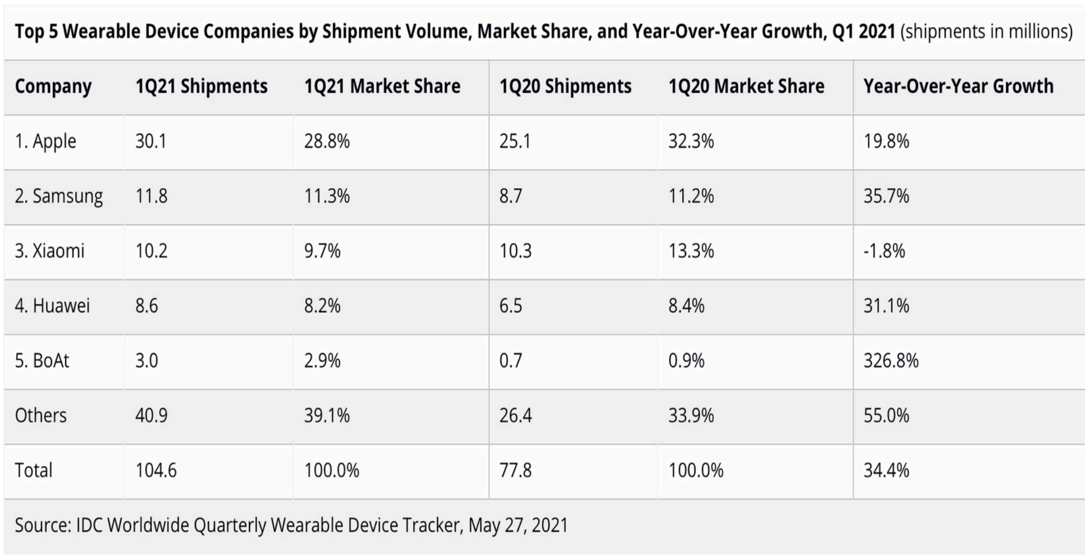
BACKGROUND

This part introduces the domain of human and hand activity recognition using wearable devices and sensors, and gives overview of its applications in health, sports and other fields. It describes wearable devices and sensors, activity recognition types, and outlines different challenges and obstacles for FTD in comparison with other activity types.

2.1. WEARABLE DEVICES AND SENSORS

2.1.1. Wearable Devices

The market of wearables has grown many folds in the past decade, and fueled by the pandemic, it has grown more than 35% in first quarter of 2021 alone compared with 2020 as shown in Figure 2.1. Smart wearables include a range of devices from smartphones, Augmented-Reality headsets, to wrist-worn devices which in turn include fitness trackers, wrist bands, and smartwatches. People use wrist-worn devices to track their daily exercises, steps taken, and monitor their vitals from HR, blood-oxygen, and calories spent during exercises like walking, running, cycling, gym classes, or while doing sports activities.



Top 5 Wearable Device Companies by Shipment Volume, Market Share, and Year-Over-Year Growth, Q1 2021 (shipments in millions)

Company	1Q21 Shipments	1Q21 Market Share	1Q20 Shipments	1Q20 Market Share	Year-Over-Year Growth
1. Apple	30.1	28.8%	25.1	32.3%	19.8%
2. Samsung	11.8	11.3%	8.7	11.2%	35.7%
3. Xiaomi	10.2	9.7%	10.3	13.3%	-1.8%
4. Huawei	8.6	8.2%	6.5	8.4%	31.1%
5. BoAt	3.0	2.9%	0.7	0.9%	326.8%
Others	40.9	39.1%	26.4	33.9%	55.0%
Total	104.6	100.0%	77.8	100.0%	34.4%

Source: IDC Worldwide Quarterly Wearable Device Tracker, May 27, 2021

Figure 2.1. Worldwide sales of smart wearable device.

Wearable devices mainly fall into three categories as shown in Figure 2.2:

Figure 2.2. Smart wearable devices types.

Smart bands and fitness trackers are smaller and cheaper than smartwatches. They are just equipped with the most essential sensors and features to track person fitness like steps and calories spent during the day.

Smartwatches are more expensive, have more features and sensors, and most importantly they have more capable processor and larger battery. They have become powerful devices that have features similar to the bigger smartphones in terms of hardware and software but still lack behind in terms of processing power and battery capacity. They are supported by major OS platforms like iOS and Android and started to add support for third-party applications. Being able to support the development and deployment of third-party applications makes them ideal choice for this thesis.

Augmented Reality (AR) glasses and Virtual Reality (VR) headsets can be considered as a special type of wearable devices. Along with motion sensors, they have depth sensors and advanced camera systems that allow them to do Visual Inertial Odometry (VIO) that incorporates vision data with motion data to accurately estimate speed and position in 3D space. Estimating position, orientation, and speed make such devices ideal for immersing experiences with the real world.

2.1.2. Wearable Sensors

Most of wearable devices are equipped with wide variety of useful sensors that come in three groups: health related sensors, motion sensors, and external environment sensors.

2.1.2.1. Health Sensors

From its name, these sensors measure person health vitals during the day while wearing a device like a smartwatch. Most common types of these sensors are the heart rate and oxygen level sensors:

HR Sensor measures heart beats per minute and it is based on non-invasive optical sensors.

Blood-Oxygen Saturation Sensor (SpO_2) measures the percentage of oxygen saturation in the blood and it is based on non-invasive optical and Infra-Red (IR) sensors.

2.1.2.2. Inertial Motion Sensors

Inertial motion sensors or units (IMU) measure resistance of acceleration due to external forces (hence the name inertial) and based on Micro-Electro-Mechanical-

Systems (MEMS). Such sensors are very small in size which is measured in micrometers or less as shown in Figure 2.3. The most common types of such sensors are the Accelerometer and Gyroscope.

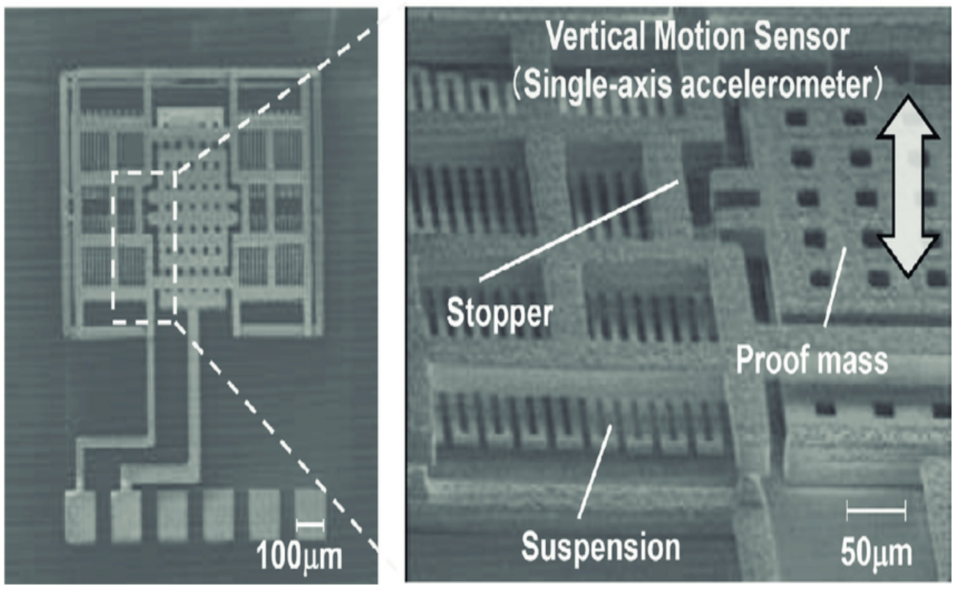


Figure 2.3. Schematic of embedded MEMS based accelerometer.

Accelerometer measures the total acceleration of the device in *g-force* the earth acceleration which is: 1g = 9.8 m/s^2. Its measurements contain not only device acceleration, but additionally, the earth gravitational acceleration component. That is why additional operations must be done to separate both components and get only real device linear acceleration. Using this sensor alone to get device speed by integrating acceleration has large bias. Furthermore, making double integration steps to get position has huge bias and error which make it totally impractical for getting speed and position. This sensor can be found in most smart wearable devices and consumes less power compared with other sensors [8].

Gyroscope measures the rotation rate of the devices in radians/second. It consumes more power than accelerometer and has more noise. Using this sensor alone to get the angles of the device is impractical and carries huge error rate.

2.1.2.3. External Environment Sensors

This type of sensors measures readings related to the outside world like the magnetic field or air pressure. The most two common sensors of this type are the Magnetometer and the Barometer:

Magnetometer measures surrounding earth magnetic field (measured in microtesla) and can detect magnetic north and directions. Its drawback is that the local fields caused by nearby electronic devices can interfere with its measurements and makes it hard to remove this interference without additional calibration every time. Another problem of this sensor is that it consumes a lot of battery and it is supported by only few high-end smartwatches. That is why using this sensor in any design means that the solution will not run on most wearable devices currently in the market.

Barometer is another example of external environment sensors. It measures current air pressure changes. It can be useful for detecting changes in device vertical motion. However, its accuracy is measured in several meters which makes it impractical for most wrist-worn hand motion applications.

2.2. ACTIVITY RECOGNITION TYPES

In this thesis, three types of human activity recognition are studied HAR, HGR, and FTD. These types can be thought of as levels, and each type can be considered as a subtype of its parent type. HAR is the most general and it contains all types of human activities. HGR is special type of HAR. It is only concerned about hand activities rather than body activities. FTD is a subtype of HGR. The face-touch move is a special type of hand gestures.

2.2.1. Human Activity Recognition (HAR)

This type can be considered as the parent of all activity recognition applications that may include full-body activities or activities for specific body-parts. Activities usually include walking, running, sitting, standing, laying, moving upstairs or downstairs. In this type, single or multiple wearable devices are worn on the body (not necessarily the hand) to measure and classify the whole-body motion. This motion is almost 2D in nature like moving on flat plan (the ground). Generally, it is run on long recognition period, and it is much easier to classify compared with the other two types.

2.2.2. Hand Gesture Recognition (HGR)

HGR is about recognizing all gesture types and actions done by the hand. For example, in Sign-Language gesture recognition, gestures can mean words or actions. Another application is remote interaction like in HCI where hand gestures can give commands to a computer interface. In addition, people can make remote interaction with devices like TVs using only hand gestures.

HGR can be divided into two categories, gesture recognition and action recognition. Gesture recognition involves the recognition of very specific gestures made by hand like drawing shapes, letters, or doing specific complex motion patterns. Although it seems hard, however, it is much easier job for the classifier to distinguish since those gestures are very different from each other, and each gesture has specific long path full of distinctive patterns.

Hand actions are specific moves done by hand and can be generally seen in hand sports like Tennis strokes (forehand, backhand, serve, volley), swimming types, or other sport moves such as throwing or catching. Another example is Activities of Daily Living (ADL), which include actions like opening and closing doors, picking and putting objects, washing hands, etc. [2]. Hand actions are even harder to classify than hand gestures because hand actions are short, fast, and have less distinctive path and much less distinctive features.

2.2.3. Face Touch Detection (FTD)

Face touch hand move can be considered as special type of hand gestures. It shares many characteristics with other HGR gestures, although it differs in some ways that make it more challenging task. It is done in shorter period with faster pace and has no complex differentiating patterns like other gestures. Moreover, it shares other similarities with normal hand moves done during regular daily living situations. All of them make it more challenging to identify face-touch moves among many other regular hand-moves. Although, Face-Touch detection has received some focus recently due to COVID-19 pandemic, there were many studies that took advantage of smartwatch sensors to monitor hand activities which involve a hand-to-face motion. Examples include smoking habits monitoring [9-10], toothbrushing style recognition [11], or bite counting while having a meal [12-13].

2.3. CHALLENGES AND LIMITATIONS

There are three types of obstacles for HAR, HGR, and FTD applications. The first obstacle is the limitation of embedded MEMS sensors. The second obstacle is the limited processing power and battery capacity of wearable devices compared with larger devices like smartphones. The third obstacle is that there are additional challenges for FTD applications in particular. Evaluating and addressing these limitations have big effects on the proposed solution and the choices done from setup, configurations, devices, sensors, and overall solution design.

2.3.1. MEMS Sensors Limitations

Accelerometer and Gyroscope sensors found on smartwatches are inertial sensors based MEMS systems and come with their own limitations [14-15]. They are listed as below:

- Raw data are noisy and carry an increasing error over time. These errors are related to its micro mechanical nature, that is why it can be easily affected by nearby chips and surrounding environment.

- Sensor readings are relative to a local moving reference frame by default attached to the watch screen. Using external fixed reference frame requires fusing data from additional third sensor (the Magnetometer) which has higher power consumption [8]. Moreover, its data is not reliable and can be affected by watch metal frame or band.
- MEMS sensors cannot be used for estimating position or distance travelled because these estimates suffer from double integration drift over time (for example, for a not moving device laying on a table, the integration drift will be more than 5 meters after only 10 seconds). Using additional algorithms like Kalman filter to minimize the error, does not guarantee the required accuracy and computationally expensive to be applicable on a small device like a smartwatch.

2.3.2. Limited CPU and Battery

Running machine learning models on a wearable device like smartwatch poses many challenges because such devices have very small battery and limited CPU power compared with smartphones. In addition, the solution needs to be running continuously in the background. Like in FTD, it must have real time response with delays and execution time measured in a few milliseconds

2.3.3. FTD Challenges compared with HGR and HAR

The problem of detecting face-touches is two levels harder than HAR. HAR uses relatively longer window size with sparse model calls and does not require real-time response. Moreover, the motion is mostly in 2D plane, device orientations are less variant, and motion classes are largely different and easier to distinguish.

On the other hand, FTD requires sub-second window size, multiple model calls per second, and real-time performance. The motion is in 3D space with variable device-orientations. Besides, the motion-types are similar with minor differences. They require differentiation between several normal daily life hand move types like touching body parts, moving up and down, picking items, moving objects, or closing and opening door, etc. Table 2.1 outlines all differences in sample size, execution time, motion and device orientation types between all three types of activity recognition HAR, HGR, and FTD.

Table 2.1. Comparison between HAR, HGR, and FTD tasks in various aspects.

	HAR	**HGR**	**FTD**
Window Size	3+ seconds	2+ seconds	0.5-1.0 seconds
Model Call Every	1+ seconds	1+ seconds	250-300 milliseconds
Model Min Required Execution Time	500 milliseconds	500 milliseconds	< 10 milliseconds
3D/2D Motion Coordinates	Motion mostly in 2D space	Motion mostly in 3D space	Motion in 3D space
Device Orientation	Fixed orientation	Variable Orientations	Variable Orientations
Variance Between Classes	Large Difference with Low Confusion	Large Difference with Medium Confusion	Small Difference with High Confusion

PART 3

LITERATURE REVIEW

This part contains a review of HGR and FTD literature, considering the face-touch hand-move as a special hand gesture type with its unique characteristics compared with other hand gestures. HGR and FTD literature are reviewed for solution design, approach, and methods. There are mainly four parts involved in solution design: hardware setup, dataset collection, data processing, and classification methods as shown in Figure 3.1. Classifications methods include classical methods and deep learning methods such as CNN networks.

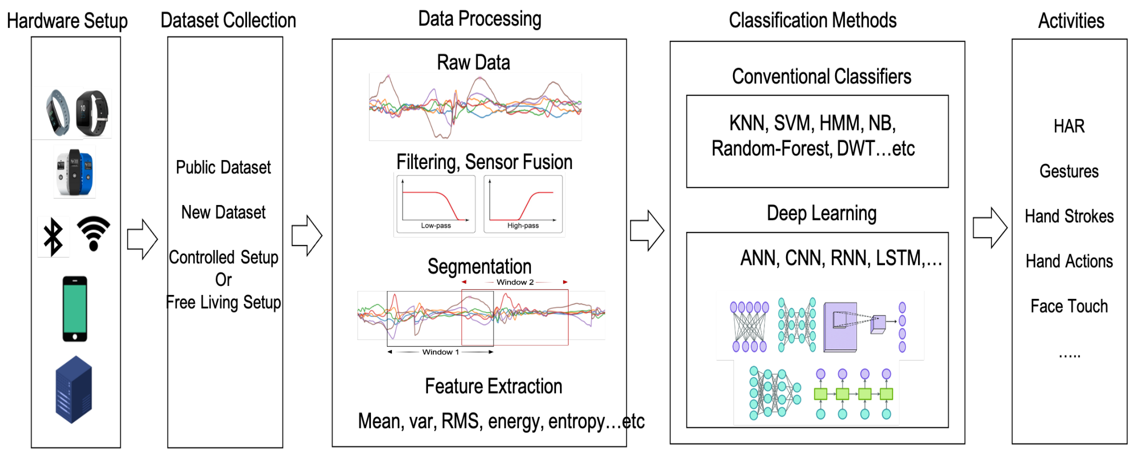


Figure 3.1. Four different phases of HGR and FTD solutions.

3.1. HARDWARE SETUP

Depending on the requirements, different studies used different hardware configurations, like single wearable device or multiple connected devices. Besides, single or multiple sensors working together have been used in literature.

3.1.1. Hardware Devices

Solution design may run on the wearable device directly without using smartphone or any other extra hardware [16-20]. This setup is more practical, cheaper, and provides immediate response to users. It can work offline without requiring any connection to other devices. However, in some situations it may lack the needed processing power. To overcome this problem, the second design utilizes a smartwatch connected to a smartphone where the smartwatch can send sensors data to a connected more capable smartphone for processing and classification [4]. Sometimes the implemented solution requires far more processing power than what smartwatch or phone can handle. In such case, data from smartwatch can be relayed to a PC over the network for processing and classification [21-23].

3.1.2. Used Sensors

For activity classification, generally a mix of three sensors is used in literature: Accelerometer, Gyroscope, and Magnetometer. Some studies used only Accelerometer to save on power consumption and requirements [19,21], while others used combination Accelerometer and Gyroscope to get more data [4,16-18,20,22]. Authors in [24] used Magnetometer along with magnets mounted on glasses or necklace to measure proximity between hand and face to detect potential face touches.

Although these three sensors have been widely used in literature, some other studies have used additional special sensors. In [23], in addition to Magnetometer, authors have used Radio-Frequency-Identification (RFID) tags to read radio signals and detect face touches. Other researchers have used depth IR cameras to get depth image and measure hand position [25].

3.2. DATASET COLLECTION METHODS

Multiple public datasets exist for HAR activities that can be used for training. However, depending on the situation, these datasets might lack some features, or use different methodology. In such cases, a new training dataset needs to be collected from scratch.

Training data can be collected in two ways. The first way is to use controlled sessions, where participants are asked to do activities repeatedly with specific positions, scenarios, and orientations. The second way is to use free-living setup in which participants are told to practice their lives naturally while wearing the smartwatch.

3.2.1. Controlled Sessions

In controlled sessions, training samples for particular class, type, or with specific conditions are collected in a separate session. Many HGR and FTD studies have used controlled sessions because data can be gathered in a faster pace. For example, different sessions can be allocated to gather samples for each class, scenario, or device orientation [16,18,20].

3.2.2. Free Living Sessions

To make training data more representative to real world scenarios, some studies have collected training data in free living setup. In this setup, volunteers are asked to practice their lives normally while wearing a smartwatch and the data logging application is already running during the session [25]. However, this adds an extra challenge for extracting the target actions. One way to solve this problem is to make video recording for all data collection sessions. Then manual cross-checking of videos with logged data is performed to find start and end points for target action samples. This method may yield more accurate data; however, it requires too much time to collect large training dataset.

3.3. DATA PROCESSING

Data processing is a crucial phase after dataset collection. It consists of four steps: sampling raw data, data filtering, data segmentation, and feature extraction.

3.3.1. Getting Raw Data

The number of times the sensors are sampled is measured in Hz or number of readings per second. Frequency ranges from 25Hz for low-end wearable devices, up to 200Hz in high-end smartphones. Sampling at higher frequency provides more data although it consumes more power. Accelerometer data is measured in *g* ($g = 9.81/s^2$). Gyroscope data is measured in radians/second. Magnetometer data is measured in microtesla. Raw data values come as vectors with three values that represents readings across 3D axes X, Y, and Z [26].

3.3.2. Data Filtering

Sensor readings are raw data that need to be filtered and processed. Filtering can be used to eliminate anomalies and reduce bias. Low-pass and high-pass filters are commonly used to reduce noise and break accelerometer total acceleration into real device linear acceleration and earth gravity acceleration [18,20]. Those filters work by transforming the signal from time domain to frequency domain. Then, they pass signals that have frequency below or above some cut-off threshold.

3.3.3. Data Segmentation

The next step after data filtering is segmentation. Reading sensors data at moment (t) is considered a single frame. Sliding-Window algorithm is a common method to group multiple frames together over specific window of time [16-19,21,23-24]. However, sometimes the target activity can start in the middle of one window and continue to the next window. To overcome such a problem, a sliding-window with overlapping is applied as shown in Figure 3.2.

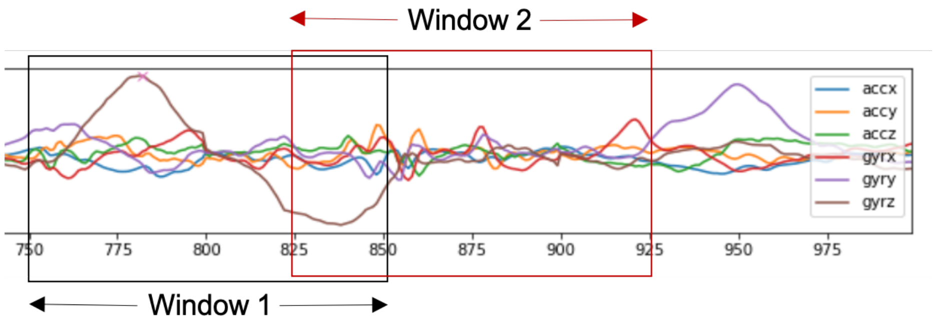


Figure 3.2. Sliding window with 100 frames size and overlapping of 25 frames.

3.3.4. Feature Extraction

In the case of end-to-end classification, data are served directly to the classifier without using any hand-crafted features or feature extraction phase. This is usually done when using deep learning methods. For example, CNN layers learn to extract features automatically, then feed those features to dense fully connected layers for classification [4,21].

Alternatively, feature extraction phase can be used before classification. Hand-Crafted features come in two types: Time-Domain features and Frequency-Domain features [16-17,19-20].

A few examples of hand-crafted features are mean, standard deviation, variance, quantiles, median, root-mean-square (RMS), entropy, energy, signal to noise ratio (SNR), skewness, kurtosis, etc. Those features are calculated for the whole sample window of time (Time-Domain), or after applying Fast-Fourier-Transform (FFT) (Frequency-Domain). Some studies have calculated these features, then used them as an input to deep learning models which calculated higher-level features.

3.4. CLASSIFICATION METHODS

There are two approaches of used by researchers: conventional approaches and deep learning approaches.

3.4.1. Conventional Approaches

Many studies have used traditional classifiers for their simplicity and fewer computations compared with deep learning methods. Traditional classifiers can be listed as K-Nearest-Neighbor (KNN), Support-Vector-Machines (SVM), Decision Trees (DT), Random-Forests (RF), Naïve-Bayes (NB), and Dynamic-Time-Warping (DTW).

KNN classifier operates by classifying the data based on how the neighbors or nearby points are classified. Any new data point is classified based on a similarity measure of all the available cases or data points. It is a versatile, common, easy to implement, and supervised machine learning algorithm that can be used for both regression and classification. KNN does not have a training phase, however, this comes at a cost of making prediction step little more expensive. Every time a prediction is made, it searches for the nearest neighbor in the whole training set [27-28].

The main objective of SVM classifier is to draw a line or hyperplane that divide data points based on their class. If a dataset is not separable by a single straight line, a kernel trick can be applied to transform the dataset from 2D dimension to 3D dimension. SVM can work with non-linear separable data because of its ability to operate on higher dimensions where it can make the data linearly separable [29-30].

RF depends on DT which builds a tree of questions based on particular features to separate a dataset into classes. In DT it is needed to know what features can to split the data in a way that the result classes are as different from each other as possible, and the members of each class are similar to each other as possible. In other words, the question is what features give the most information gain at what threshold. RF is a supervised classification method which consists of collection of decisions trees. It uses bagging and feature randomness when building each individual tree to create an un-correlated forest of trees whose prediction is more accurate than any single decision tree [31-32].

NB is a classification method for binary and multiclass classification problems. It is named Naive Bayes because the calculations of the probabilities for each class are simplified by making naïve assumptions. Rather than trying to calculate the conditional probabilities of each attribute value, they are assumed to be conditionally independent. Although this assumption is unlikely in real world, the approach performs surprisingly well even if this assumption does not hold. This makes NB easy to implement and calculate.

DTW is a classification method that depends on comparing the data time-series on hand with pre-calculated templates of time-series data. Any two time series can be compared using euclidean distance or other distance measured on a one to one basis along the time axis. Values of first time series at time T are compared with values of second time series at time T. However, this may result in a very poor similarity score when the two time-series data are out of phase even if they are very similar in shape. DTW solves this problem by comparing amplitude of first signal at time T with amplitude of second signal at time T+1 and T-1 or T+2 and T-2. This assures that it does not yield low similarity score for signals with similar shape but in different phase [4,33-34].

Traditional classifiers have been used frequently in literature to classify gestures. Authors in [16] compared the performance for SVM and KNN. They found that SVM had better accuracy for classifying seven different hand gestures. DTW classifier have been used in many studies to classify hand gestures [4,18]. Researchers in [17] compared performance of different classifiers like KNN, SVM, and NB. They found that each classifier performs better on certain gestures by choosing the right classifier based on target gestures.

3.4.2. Deep-Learning Approaches

While traditional classifiers are convenient for small wearable devices due to less required computations, deep learning methods like regular Artificial Neural Networks (ANN), CNN networks, Recurrent Neural Networks (RNN) generally outperform other classifiers. Deep Learning methods have an ability to learn features automatically even when fed with raw data. When fed with hand-crafted features, they can learn correlations between several features and induce extra higher features.

CNN networks take input data like images and apply convolution layers where it convolves over images with learned filter to produce another image with same or less dimensions. Pooling layers are usually applied after each convolution to reduce size and fight overfitting. Convolution blocks can be considered as feature extraction layers. Those layers are followed by dense layers at the end which serve as final classification layers. Figure 3.3 shows the structure of the popular VGG-16 CNN network.

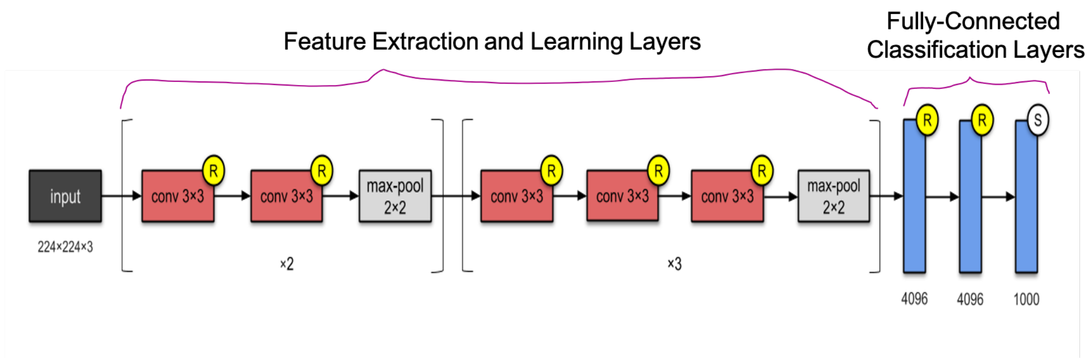


Figure 3.3. Architecture of the VGG-16 CNN network.

RNN networks are suitable for sequence of data. In such networks, the output of hidden layer is fed again as additional input in the next step. Figure 3.4 shows the structure of RNN network and it is unfolded which provides clearer image about its structure.

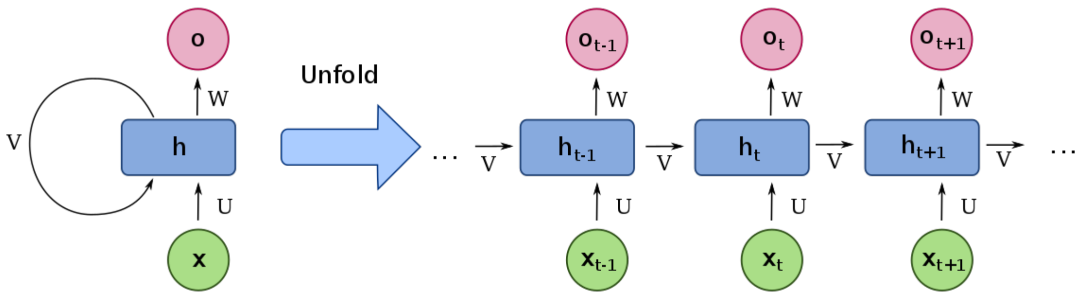


Figure 3.4. Unfolded RNN network structure.

Some studies use hybrid approach which combines CNN and RNN together. Long-Short Term Memory (LSTM) layers which are a flavor of RNN can be placed on top of CNN layers. In such approach, CNN layers extract current window features, and the LSTM layer keeps track of hidden layer output from previous samples as shown in Figure 3.5. This is especially useful when hand gestures or motion types have temporal dependencies [35].

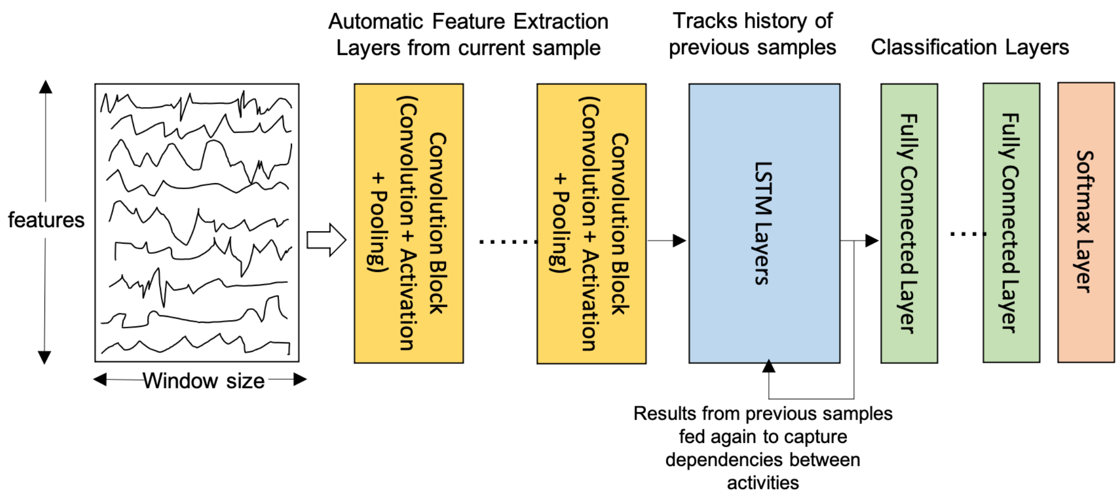

Automatic Feature Extraction Layers from current sample
Tracks history of previous samples
Classification Layers
features
Window size
Convolution Block (Convolution + Activation + Pooling)
Convolution Block (Convolution + Activation + Pooling)
LSTM Layers
Fully Connected Layer
Fully Connected Layer
Softmax Layer
Results from previous samples fed again to capture dependencies between activities

Figure 3.5. Architecture that combines LSTM layers on top of CNN layers.

Deep learning methods have been used frequently in the literature. Authors in [20] compared classification performance between CNN and several conventional classifiers like SVM, iForest, Minimum Covariance Determinant (MCD). They found that CNN outperformed other classifiers in terms of recall and precision. Authors in [21] have used end-to-end CNN network to classify 10 different gestures and achieved 97.32% F1-score.

PART 4

MATERIALS AND METHODS

This part describes the materials and methods used to implement the proposed solution. Used materials like hardware, software, and volunteers are explained. Chosen setup, sensors, features, and configurations are detailed. The collection process of a large dataset is detailed for both left and right hands, then all post processing steps are elaborated. Classification process based on deep learning and the validation methods used to test the trained models are mentioned.

4.1. MATERIALS

In this section, all required hardware, devices, sensors, software, operation systems are detailed. Additionally, the number of volunteers and characteristics used to obtain training dataset are described. The size and features of the dataset are outlined.

4.1.1. Used Hardware and Software

During dataset collection process, two different smartwatches were used. First one is Apple Watch Series 3 which was released in 2017. This watch tests the ability of the proposed solution to run even on old hardware with limited capabilities. The second one is a new version called Apple Watch SE released in 2020 with better capabilities in terms of CPU and other hardware. During data collection, a special recorder application was run on both watches where logs were saved, then sent to a connected local PC. All subsequent data processing and training models were done on local PC. Although the smartwatches were equipped with several sensors, only two sensors were used which are the Accelerometer and the Gyroscope. During recording training data, the maximum sensor sampling frequency were set to 100Hz. During data processing phase, down-sampling such as 50Hz and 75Hz were applied.

All the data collection, processing, and testing data were done on Apple WatchOS operating system. The solution can completely run on the watch alone offline without requiring internet connection or connecting to any server or nearby smartphone.

4.1.2. Volunteers

Training dataset were collected using three different volunteers with variable age and body characteristics. Collecting the Test dataset was done by the help of three volunteers as well. Two of them were not participating in the training dataset collection process. For beta-testing on real app, eight volunteers were asked to install the application and test both its accuracy in real life usage and battery usage. Those volunteers have wide range of smartwatches versions. They have different characteristics and even live in different countries. They use their smartwatches in different real-world environments like work, household, shopping, and public transportation environment.

4.1.3. Dataset Properties

Two different datasets were collected from scratch for left and right hands with the exact methodology. Each dataset contains approximately 14000 training samples distributed evenly on five classes. Each training sample contains 100 frames (because 100Hz frequency is used). Although subset of features was used in final solution, during training data recording, all 25 features were saved for further experiments as shown in Table 4.1.

Table 4.1. List of 25 raw features saved while recording training data.

Type	**Num. Features**	**Description**
Device Real Acceleration	3 features	accx, accy, accz which are acceleration on X, Y, Z axes measured in g-force
Rotation Rate	3 features	gyrx, gyry, gyrz which are rotation rate on X, Y, Z axes in radians/per-second
Current Gravity Vector	3 features	gravx, gravy, gravz which are gravity force values on the current X, Y, Z axes measured in g-force
Attitude – Represented in Euler Angles	3 features	Roll: rotation on Y axis in radians Pitch: rotation on X axis in radians Yaw: rotation on Z axis in radians
Attitude – Represented in Quaternions	4 features	qx, qy, qz, qw A quaternion offers a better way to parameterize and represent attitude.
Attitude – Represented in Rotation Matrix	9 features	[m11 m12 m13 m21 m22 m23 m31 m32 m33] A rotation matrix in linear algebra describes the rotation of a body in three-dimensional Euclidean space.

4.2. SENSOR FUSION

Sensor-Fusion, which is an important technique is explained in this section. It has a large effect on features and configurations. The goals and flavors of sensor fusion is firstly described. Then the used reference frames are detailed.

4.2.1. Goals and Types

Sensor Fusion is the process of combining data from two or more sensors to generate more robust and accurate readings when compared with reading each single sensor data. Sensor fusion between multiple sensors achieves multiple goals like reducing bias, separating linear acceleration and gravity, and estimating device orientation in

3D space as shown in Figure 4.1. Device orientation can be given as Euler-Angles simple form, quaternions, or a rotation matrix that have more accuracy and less problems [22]. Fusion process is usually done by using a flavor of Kalman-Filter which is time consuming process specially when applied on wearable devices. That is why, modern smartwatches have hardware-accelerated sensor fusion using dedicated motion co-processor.

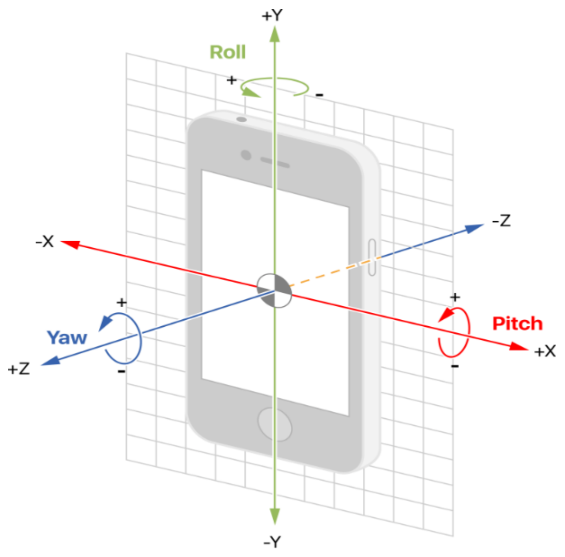


Figure 4.1. Attitude angles: Roll, Pitch, and Yaw.

There are two types of sensor fusion. First type uses fusion between Accelerometer and Gyroscope. It can achieve many goals such as correcting rotation bias, isolating linear and gravity acceleration, and estimating device angles.

Second type uses fusion between Accelerometer, Gyroscope, and Magnetometer. The second type of sensor-fusion achieves goals which are achieved by the first type as well. In addition to that, the second type has external fixed reference frame. However, the second type is notably more expensive in terms of battery usage and required processing power.

4.2.2. Reference Frames

The reference frame used when reading acceleration and rotation data is local reference-frame pinned to the watch screen as shown in Figure 4.2. Reference frame axes can be explained as below:

- X-Axis is going horizontally on the watch screen from left to right.
- Y-Axis is perpendicular on X-Axis and going from top to down on watch screen.
- Z-Axis is vertical, virtually penetrating watch screen, and perpendicular on X and Y axes.

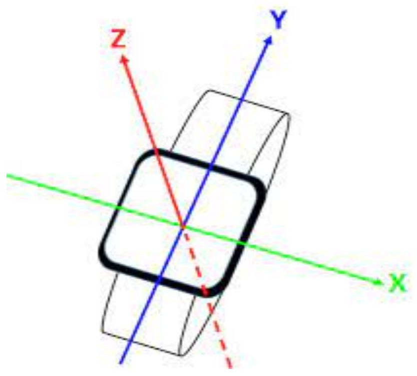


Figure 4.2. Local reference frame attached to watch screen.

When reading device orientation, those values must be based on fixed external reference frame. There are two types of reference frames that can be used depending on which type of sensor fusion is used.

When using Accelerometer and Gyroscope sensor-fusion, the Z-axis is vertical along with earth gravity vector and X-axis is chosen randomly in horizontal plane, and Y-axis is perpendicular on both X and Z as shown in Figure 4.3. This random choice of X-axis may lead to different patterns for the same move during training and inference. The solution is to save initial attitude at first frame of the input window. Then the solution is to convert all subsequent attitude readings to be relative to the initial value. This is done by multiplying with inverse of initial attitude rotation matrix as shown in formula Eq. 4.1.

$$Attitude_{Relative} = Attitude_{current} * Inverse(Attitude_{Initial}) \tag{4.1}$$

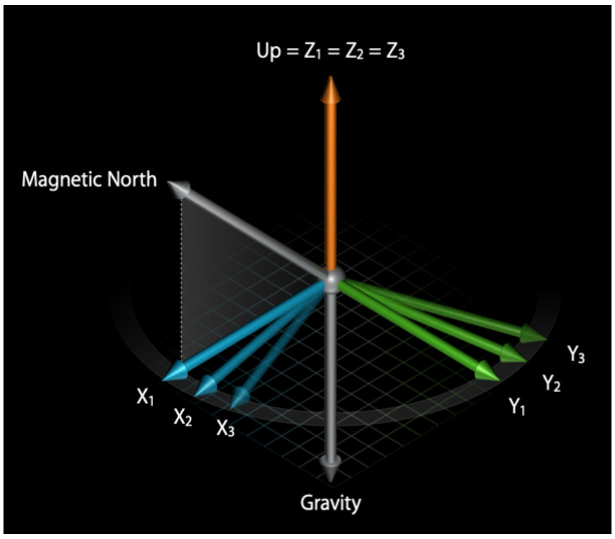


Figure 4.3. Reference frames where Z-axis is fixed, X and Y axes are random.

When using the second type of sensor fusion, the reference-frame is totally fixed where Z-axis is vertical, Y-axis points towards the magnetic north, and X axis is perpendicular as shown in Figure 4.4. In such case, a conversion for each attitude value is not required.

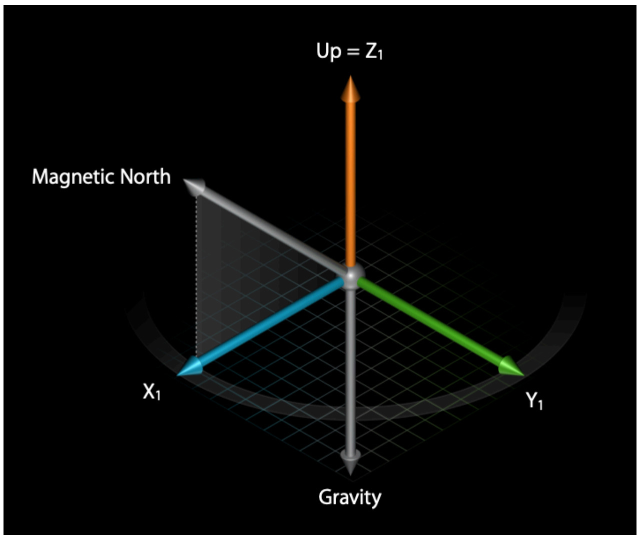


Figure 4.4. Fixed reference frame where X, Y, Z axes are always the same.

4.3. CHOSEN CONFIGURATIONS

In this section, the chosen setup from sensors, configurations like sampling frequency, sliding-window size, and features are explained.

4.3.1. Chosen Sensors and Configurations

Only Accelerometer and Gyroscope inertial sensors were used because they are supported on all smartwatches and they consume much less power compared with other sensors. The use of Magnetometer sensor was dropped because it is not supported on all smartwatch models, it has higher power consumption [8], and can be affected by watch metal frame. Thus, it requires manual calibration to be done by user. All sensor readings are relative to a local moving reference frame pinned to the device screen. Sliding-Window method is used with window-size set to a small number of 30 frames or 0.6 seconds in order to reduce the calculations required in each model call and save on power consumption. During runtime inference, 50% overlapping is applied which results in 3.33 model-calls per second.

Instead of reading raw sensors data, Sensor-Fusion between Accelerometer and Gyroscope was used at low sampling frequency of 50Hz. Sensor-Fusion achieves two main goals. The first one is correcting Gyroscope data by removing accumulated bias. The second one is separating linear device acceleration from acceleration due to earth gravity.

4.3.2. Raw Features

Sensor fusion between Accelerometer and Gyroscope allows for recording of 25 features. Three features represent real device linear acceleration. Three features represent the corrected rotation rate. Three features represent current gravity vector. Rotation rate is measured in radians/second whereas both device acceleration and gravity are measured in g ($g = 9.81/s^2$). This makes total of 9 features, and the rest 16 features are reserved for device orientation. Device orientation can be represented in 3 angles (Euler angles: Roll, Pitch, and Yaw), Quaternions, and rotation matrix with shape 3x3. All 25 features are shown in Table 4.1.

Gravity can capture device tilt because data are relative to device's reference frame. Current gravity vector (GV) is always vertical and aligned with earth GV which means GV angles change with the device or hand tilt as shown in Figure 4.5.

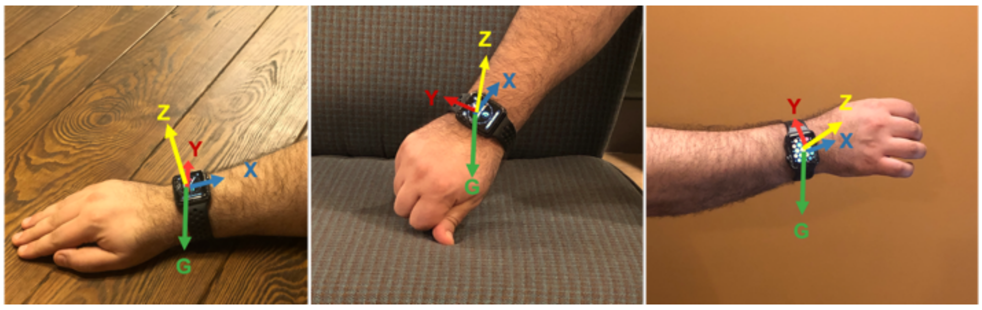
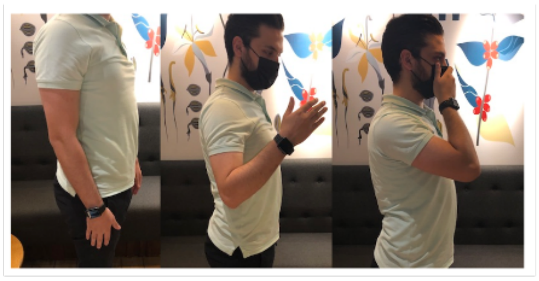
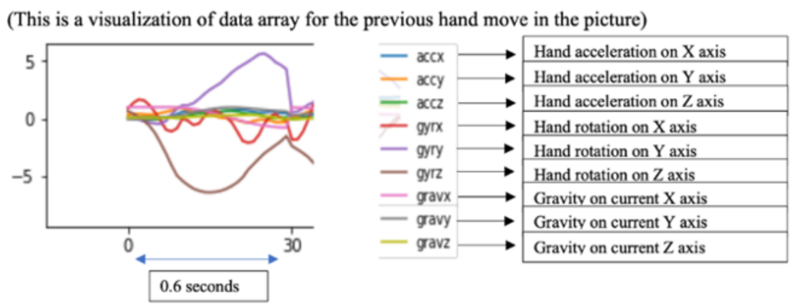


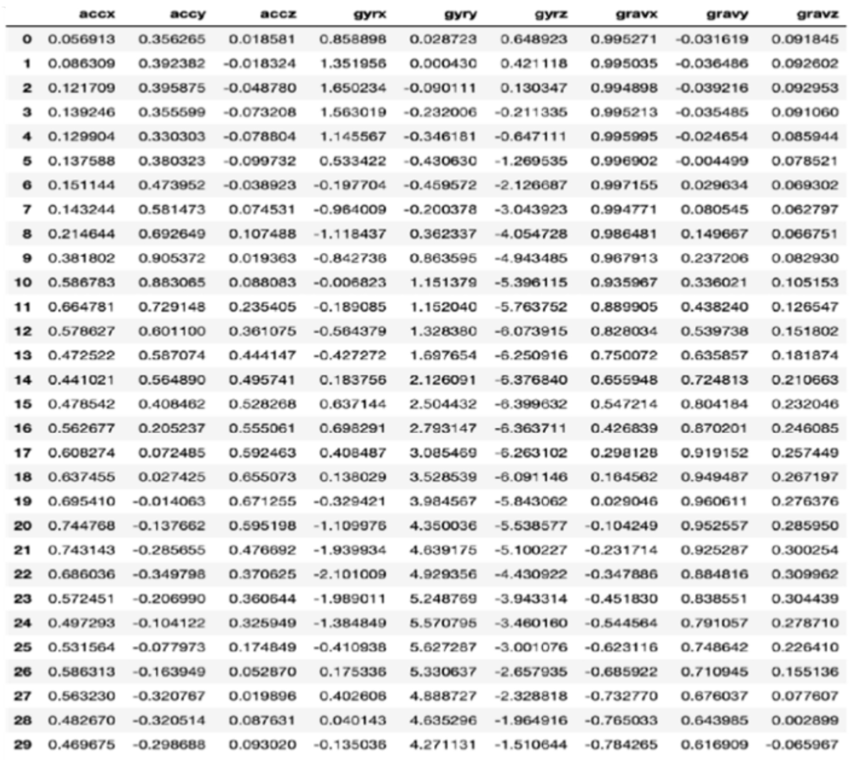
Figure 4.5. GV relative to current device reference frame.

Figure 4.6 shows data for one hand motion sample of a person trying to touch his face while wearing a smartwatch. It explains how one sample of face touch motion is translated to a window of 30 frames and each frame contains 9 data points or features.

In Figure 4.6, while this hand moves, every 20 milliseconds single data frame is read from motion sensors. When 30 readings are collected, the window is fed to the CNN model.




(This is a visualization of data array for the previous hand move in the picture)
5
0
−5
0
30
0.6 seconds
accx
accy
accz
gyrx
gyry
gyrz
gravx
gravy
gravz
Hand acceleration on X axis
Hand acceleration on Y axis
Hand accleration on Z axis
Hand rotation on X axis
Hand rotation on Y axis
Hand rotation on Z axis
Gravity on current X axis
Gravity on current Y axis
Gravity on current Z axis



	accx	accy	accz	gyrx	gyry	gyrz	gravx	gravy	gravz
0	0.056913	0.356265	0.018581	0.858898	0.028723	0.648923	0.995271	-0.031619	0.091845
1	0.086309	0.392382	-0.018324	1.351956	0.000430	0.421118	0.995035	-0.036486	0.092602
2	0.121709	0.395875	-0.048780	1.650234	-0.090111	0.130347	0.994898	-0.039216	0.092953
3	0.139246	0.355599	-0.073208	1.563019	-0.232006	-0.211335	0.995213	-0.035485	0.091060
4	0.129904	0.330303	-0.078804	1.145567	-0.346181	-0.647111	0.995995	-0.024654	0.085944
5	0.137588	0.380323	-0.099732	0.533422	-0.430630	-1.269535	0.996902	-0.004499	0.078521
6	0.151144	0.473952	-0.038923	-0.197704	-0.459572	-2.126687	0.997155	0.029634	0.069302
7	0.143244	0.581473	0.074531	-0.964009	-0.200378	-3.043923	0.994771	0.080545	0.062797
8	0.214644	0.692649	0.107488	-1.118437	0.362337	-4.054728	0.986481	0.149667	0.066751
9	0.381802	0.905372	0.019363	-0.842736	0.863595	-4.943485	0.967913	0.237206	0.082930
10	0.586783	0.883065	0.088083	-0.006823	1.151379	-5.396115	0.935967	0.336021	0.105153
11	0.664781	0.729148	0.235405	-0.189085	1.152040	-5.763752	0.889905	0.438240	0.126547
12	0.578627	0.601100	0.361075	-0.564379	1.328380	-6.073915	0.828034	0.539738	0.151802
13	0.472522	0.587074	0.444147	-0.427272	1.697654	-6.250916	0.750072	0.635857	0.181874
14	0.441021	0.564890	0.495741	0.183756	2.126091	-6.376840	0.655948	0.724813	0.210663
15	0.478542	0.408462	0.528268	0.637144	2.504432	-6.399632	0.547214	0.804184	0.232046
16	0.562677	0.205237	0.555061	0.698291	2.793147	-6.363711	0.426839	0.870201	0.246085
17	0.608274	0.072485	0.592463	0.408487	3.085469	-6.263102	0.298128	0.919152	0.257449
18	0.637455	0.027425	0.655073	0.138029	3.528539	-6.091146	0.164562	0.949487	0.267197
19	0.695410	-0.014063	0.671255	-0.329421	3.984567	-5.843062	0.029046	0.960611	0.276376
20	0.744768	-0.137662	0.595198	-1.109976	4.350036	-5.538577	-0.104249	0.952557	0.285950
21	0.743143	-0.285655	0.476692	-1.939934	4.639175	-5.100227	-0.231714	0.925287	0.300254
22	0.686036	-0.349798	0.370625	-2.101009	4.929356	-4.430922	-0.347886	0.884816	0.309962
23	0.572451	-0.206990	0.360644	-1.989011	5.248769	-3.943314	-0.451830	0.838551	0.304439
24	0.497293	-0.104122	0.325949	-1.384849	5.570795	-3.460160	-0.544564	0.791057	0.278710
25	0.531564	-0.077973	0.174849	-0.410938	5.627287	-3.001076	-0.623116	0.748642	0.226410
26	0.586313	-0.163949	0.052870	0.175336	5.330637	-2.657935	-0.685922	0.710945	0.155136
27	0.563230	-0.320767	0.019896	0.402606	4.888727	-2.328818	-0.732770	0.676037	0.077607
28	0.482670	-0.320514	0.087631	0.040143	4.635296	-1.964916	-0.765033	0.643985	0.002899
29	0.469675	-0.298688	0.093020	-0.135036	4.271131	-1.510644	-0.784265	0.616909	-0.065967

Figure 4.6. Sample data of a person trying to touch his face.

4.4. DIVIDING HAND MOTION TYPES

Hand motion in 3D space is chaotic and some hand moves share many similar acceleration and rotation patterns with the face-touch move. In this binary classification problems (touch/no touch), high recall was obtained, however, precision was relatively low due to many false positives. By closely examining false positives, it was found that they were mostly caused by specific types of hand moves that are elbow-based. This happens when the forearm moves around the elbow making a sharp angle with the upper arm as shown in Figure 4.7. This is in contrast to moving the whole arm together or when the angle between the forearm and upper arm is large as shown in Figure 4.8.

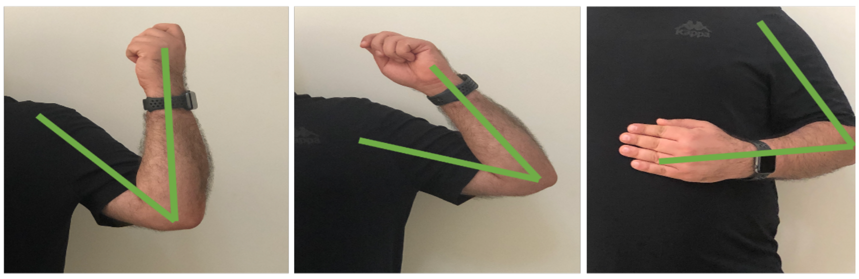


Figure 4.7. The angle between forearm and upper arm is sharp.

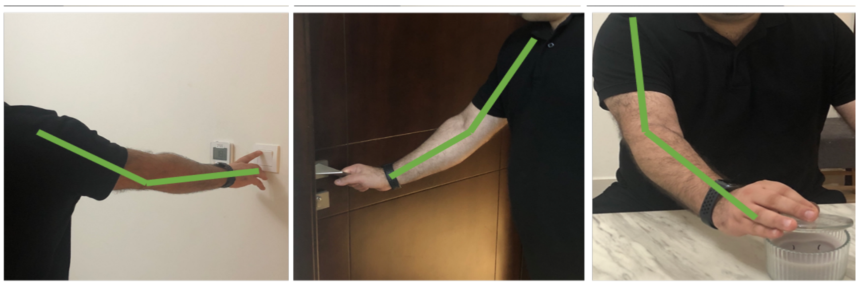


Figure 4.8. The angle between forearm and upper arm is large.

4.4.1. Hand Motion Classes

To minimize false positives, additional two elbow-based motion classes were added. “Up” class represents when the hand moves up with close level to the head, and “Abdominal” class represents all hand moves towards the abdominal area like stomach and chest. Besides, another two classes were added, “Stationary” class for when the hand is stationary or slightly moving, and “Normal” class for all other hand moves which brings the total of hand motion classes to five.

4.4.2. Scenarios for Each Class

To achieve fine-grained division and make the classifiers more robust, data are collected for each of those classes in five scenarios: standing, walking, and three different sitting positions as sitting while hand in a neutral or lower position, in a medium position like resting on the legs or couch, and in a high position like resting on a table as shown in Figure 4.9. The three sitting positions serve for another purpose which is recognizing face touch move even if it was initiated from different locations as shown in Figure 4.10.

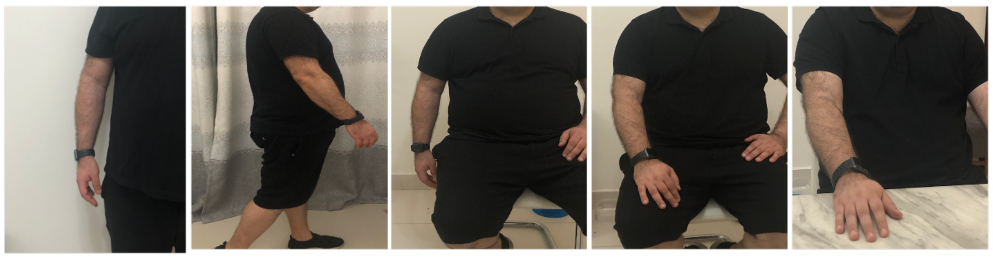


Figure 4.9. Five different positions: standing, walking, and three different sitting.

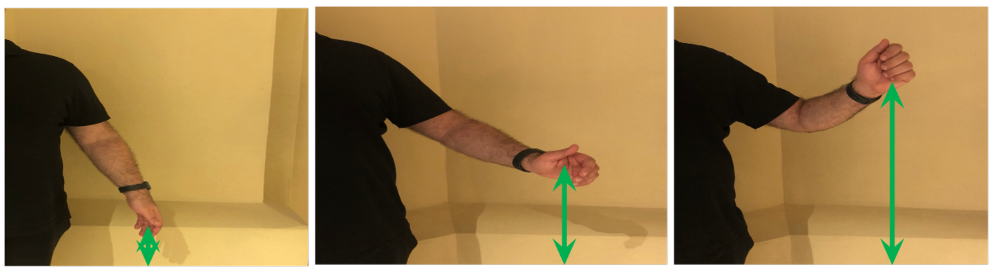


Figure 4.10. Initiating hand move from three different heights.

4.4.3. Orientations for Each Scenario

Since local moving reference-frame is used instead of external fixed one, any change in hand orientation or any small shift in watch position around the wrist caused by loose bands give different data patterns. To overcome this problem, for each of the above five scenarios, data are collected with multiple hand orientations. Two hand orientations are defined at the move start, and three at the move end that represents three different ways for touching the face either with hand palm, back, or side. This makes the total 6 combinations of hand orientations as shown in Figure 4.11. Those hand orientations are not rigid. A range of hand rotation in either direction is allowed during samples collection to mimic real-life cases especially with loose bands where the watch can move around the wrist.

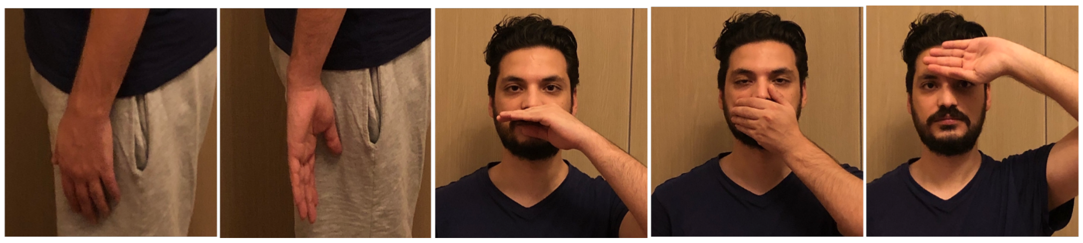


Figure 4.11. Hand start/end orientations when touching the face.

Any change in hand orientation at the start may cause different motion patterns like acceleration or rotation on a different axis. Similarly, touching the face with different orientation or different hand-part causes different rotational patterns. Figure 4.12 shows different acceleration and rotation patterns for the same hand move in the same standing position, but in 6 different hand orientations.

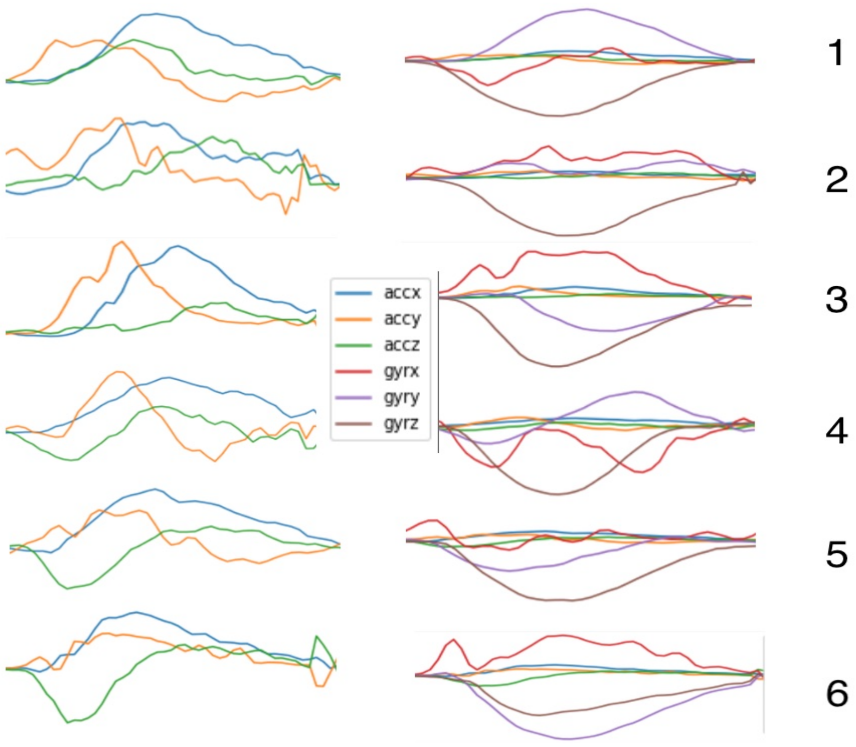


Figure 4.12. Sample motion patterns for 6 hand orientations of "Touch" move.

By applying the 3-level division of hand motion, the models are allowed to be trained on a wide range of motion patterns in real-life situations which improves detection accuracy and minimize false positives.

4.5. COLLECTING DATASETS

Training and Test datasets were collected from scratch. This section describes the methodologies used in the collection process.

4.5.1. Training Dataset

Training datasets were collected using controlled sessions and automatic sample extraction algorithm.

4.5.1.1. Controlled Sessions

The methodology used for training dataset collection and processing has a huge impact on achieving both high accuracy and high efficiency. Training data are collected in controlled sessions where each session is performed by one person for a specific motion class, and only for one scenario, and hand orientation. That means up to 30 different sessions for each motion class. In each session, the volunteer does repetitive hand moves with around 1 second rest in between. Each session contains between 40 and 150 samples. Data are recorded using a special application, where the user presses start-button to start the session and presses the stop-button to finish the session.

All data collected in a single session are logged and saved in a separate file. This file includes motion data for the target touch move along with other irrelevant hand moves like moving the hand back from the face to the neutral position. In addition, it contains some intervals where the hand is in the resting state. To automatically extract only the target face-touch move, a novel algorithm was applied.

4.5.1.2. Automatic Sample Extraction Algorithm

The novel algorithm was developed to extract training samples automatically which helped the collection of large number of samples in less time. Visualizing the three elbow-based hand moves (Face-Touch, Up, Abdominal), it was observed that they cause a spike in rotation rate around Z or Y axes resulted from moving the forearm around the elbow. Using this prominent feature besides the fact that there is only one targeted move type in each session, an algorithm that uses peak-valley analysis was developed to extract samples automatically. Each hand-move starts with an acceleration phase until it peaks at the middle, then a deceleration phase starts and continues until the hand stops. Sample is extracted by taking the peak value as the middle point as shown in Figure 4.13.

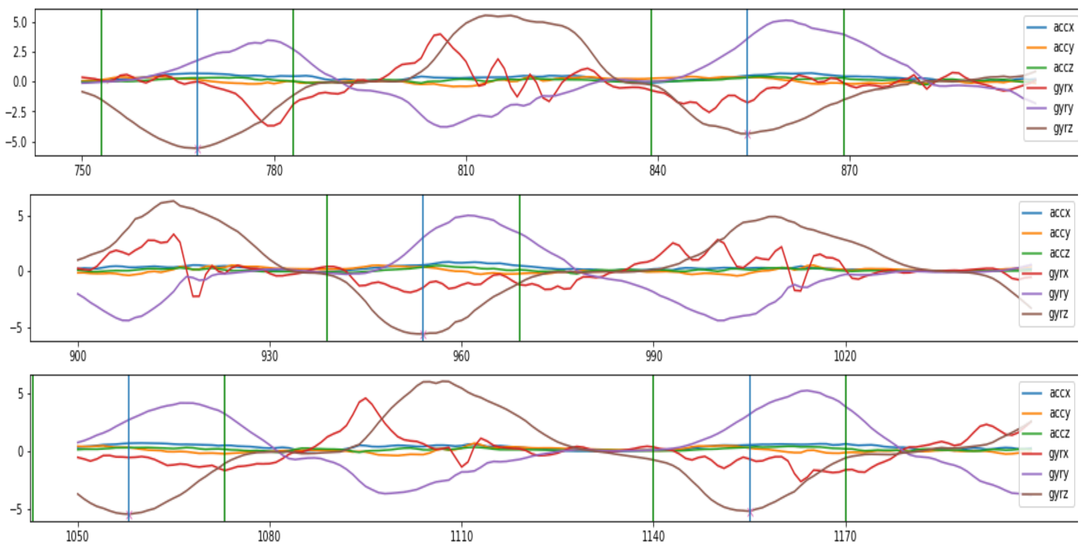


Figure 4.13. Example of samples automatically detected by the algorithm.

A small window size of 0.6 seconds was chosen to minimize required calculations, save on power consumption, and allow early touch detection. However, this is mostly smaller than the actual touch move duration as observed in the dataset which mostly falls in the range of [0.55, 0.8] seconds. Random shift of the middle point was applied with up to 2 frames to the left, and up to 5 frames to the right, with more emphasis on right-shift because the end of touch move contains more distinctive features compared with the move-start. The formula for the extracted-sample-range at detected peak at moment (t) is given in Eq 4.2. where r is a random number in range of [-2, +5].

$$\mathrm{SampleRange} = \left[t - \left(\frac{\mathrm{window_{size}}}{2} \right) + r, \qquad t + \left(\frac{\mathrm{window_{size}}}{2} \right) + r \right] \tag{4.2}$$

Using controlled setup where each volunteer made over 80 data collection sessions, and the automatic sample extraction, helped to collect large training dataset for both left and right hands.

4.5.2. Separate Test Dataset

To test the trained models better, a Test-Dataset which was never used in training phase was collected. It was collected separately from three different volunteers doing real-life activities including face-touches while wearing two different watches. Each person did 50-100 face-touches in the 5 scenarios while doing normal activities for a total of 250 touch samples for each of the left and right hands. Each session was recorded as a video and then the dataset was labelled manually by cross-checking all recorded videos.

In order to make the test dataset more representative of real-world usage, additional moves that may cause false positives were added like moving hand up near the face or touching the chest. Those moves were added with one-to-one ratio compared with the face-touch move (also around 250 moves per dataset). Additionally, a wide range of normal daily-life activity moves were added like household activities and office activities. In total, each dataset contains 250 positive face-touch samples and over 5700 non-touch samples where the hand is doing other motion types. All moves are done within 90 minutes period. Finally, 50% overlapping was applied to match runtime conditions on the application.

4.6. POST-PROCESSING ON TRAINING DATASET

Additional processing steps were done on collected raw dataset. Those steps were used to generate multiple different datasets out of the original dataset. Each one has different characteristics or parameters as generating datasets based on Full-Motion-Wave or Half-Motion-Wave. Additionally, datasets with different window-sizes were generated to test more possible window sizes. Three different datasets with different window shift augmentations were used to increase samples or to test different window shift values. Finally, extra processing steps were done, specifically, applying jittering and data standardization.

4.6.1. Half-Wave vs Full-Wave

The hand motion can be divided into two phases. The first one is the acceleration phase where all readings start at low values and then explode quickly towards the peak as shown in Figure 4.14. The second phase starts after the hand motion speed peaks and then starts the deceleration phase until it stops on the face as shown in Figure 4.15.

Two types of datasets were processed. First one is where all samples extracted from considering the motion peak is the end of the window. This type allows for early detection as the application alerts the user well before the hand reaches the face. Second one is where all samples extracted considering the peak is the middle of the window.

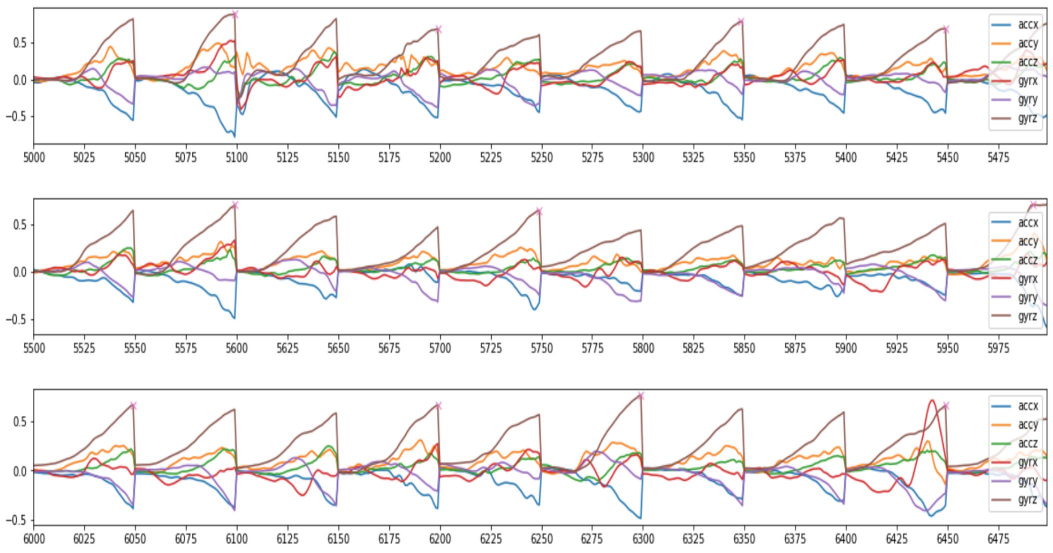


Figure 4.14. Extracted samples where only acceleration phase is taken.

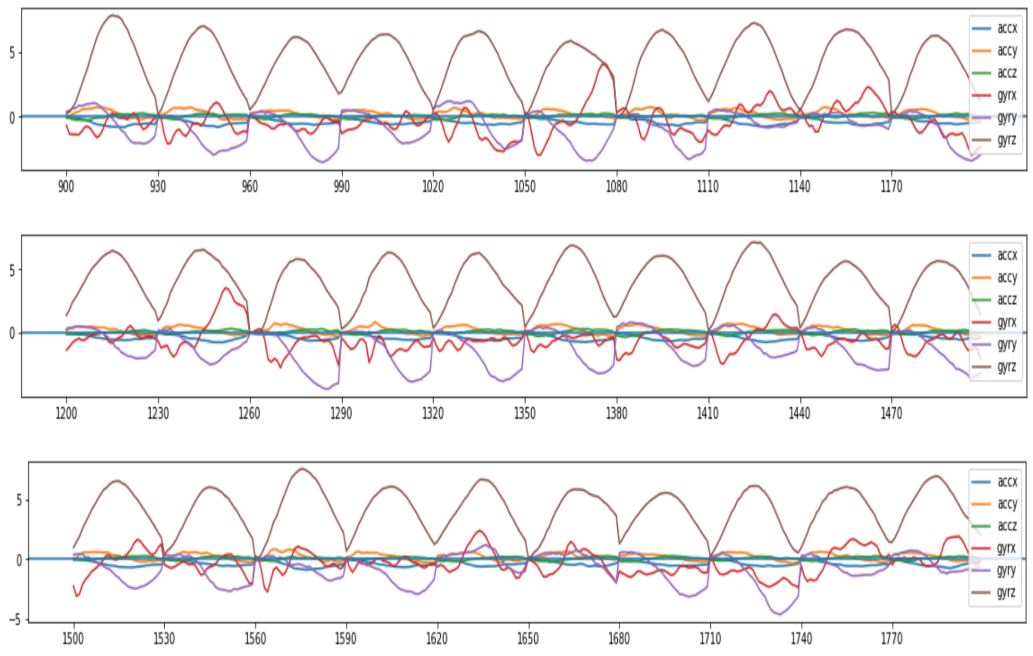


Figure 4.15. Extracted samples where both phases are taken.

4.6.2. Window-Sizes

Multiple datasets were processed with different window-sizes like 30, 40, and 50 frames sizes which are equivalent to 0.6, 08, 1.0 seconds, respectively. Having small window size lowers the necessary computations. That means faster classification and less consumed power. On the other hand, having larger window size means there are more input data points for the classifier which may result in better accuracy.

4.6.3. Window-Shifting

The original sample is extracted based on the detected peak. Additional window-shifting is applied to generate more samples and more robust training data. Window-Shifting serves as data augmentation where multiple training samples are generated from one sample. Shifting is done on both ways, to the right (positive) where window-center is moved couple of frames, or to the left (negative) where window-center is subtracted few frames as shown in Figure 4.16.

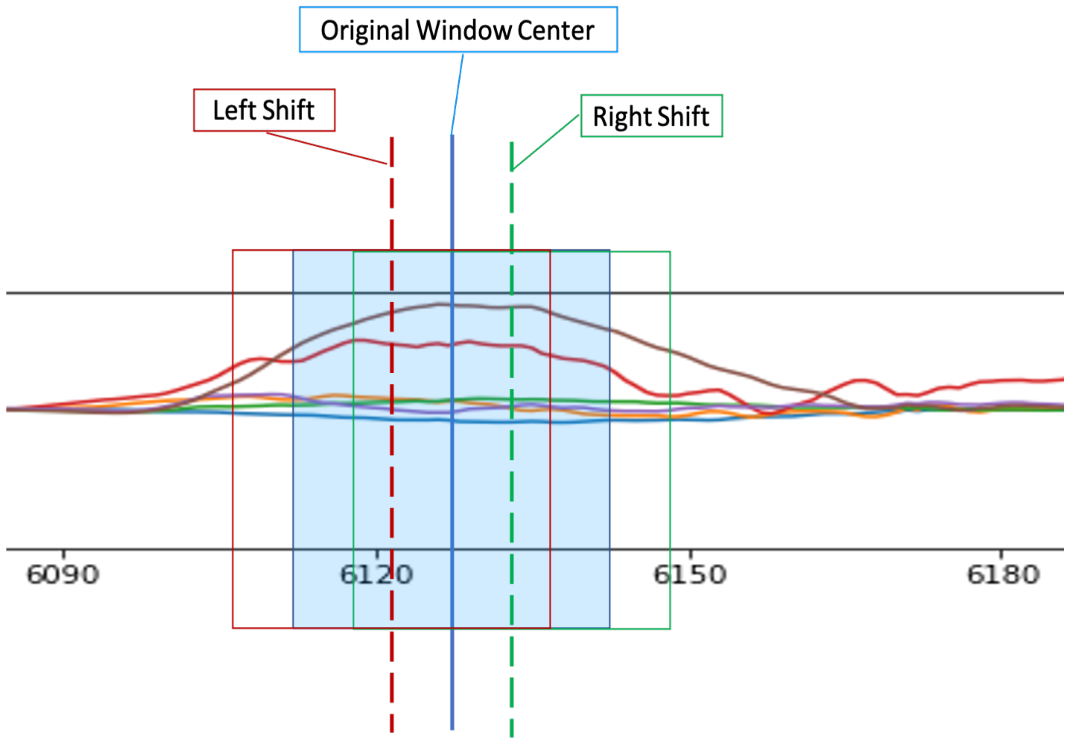


Figure 4.16. Original extracted sample window and 2 other shifted windows.

4.6.4. Additional Data Processing

To make the training dataset more robust, signal jittering and data standardization were applied. Signal jittering resembles jittery sensor readings when some high priority task is performed by the operating system. Data standardization is done on the final data by subtracting the mean and dividing by standard deviation.

4.7. TRAINING USING DEEP-LEARNING

4.7.1. End-To-End CNN Architecture

The proposed architecture uses end-to-end models based on CNN networks without any hand-crafted or computed features. After experimenting with adding LSTM layers on top of CNN layers, the performance gain was negligible compared with expensive computations of LSTM layers. Instead of 2D-Convolutions, 1D convolutions were used to convolve data along time axis with features set as depth channels. Average pooling layers were used for down sampling and translation invariance. CNN blocks automatically extract features before feeding them to dense layers for classification.

4.7.2 Hyperparameter Tunning Using Grid-Search

To find the best hyperparameters, a grid-search of 243 iterations was executed. In each iteration, a separate model was trained using one combination of hyperparameters like filters count, dropout rates, optimizer method, batch size, and epochs count.

Filters counts of 32, 64, and 128 were used. For dropout rates, three different combinations were used. Batch size of 32, 64, and 128 were tried. Three different counts were tried for epochs count. Adam, RMSPROP, and SGD optimizers were tried. This makes the total iterations or different trained models as 3^5.

Trying different optimizers like SGD, RMSProp, and Adam with different learning rates parameters is necessary because each problem might work better with different optimizer method and with better learning rate parameters. In SGD which stands for Stochastic Gradient Descent, weights are updated after one sample classification forward pass during training and learning rate is fixed. This means the path to optimal solution or global loss minima is noisy and slow. RMSProp which stands for Root Mean Square Propagation is an improvement over SGD. It normalizes the gradient by using the moving average of squared gradients which means the learning rate is adaptive and changes overtime. Adam which stands for Adaptive Moment is considered as an improvement on RMSProp. Adam is the latest state of the art optimizer and the most used optimizer in deep learning literature.

4.8. VALIDATION

Before presenting experimental results, this section outlines the validation for testing results used to test the trained models.

4.8.1. Confusion Matrix

Confusion matrix summarizes the prediction results for the trained classifier. It shows the counts of correctly classified or misclassified cases for each class. It can show better understanding of the trained model and its weak points. This is effective for misclassified samples for each class to display. It gives better insight not only on the mistakes done by the classifier, but also on the types of errors that were made.

In binary classification, classified samples can be categorized in four types. In the case of Touch/No-Touch problem, True-Positives (TP) are face-touches that are classified correctly by the classifier. True-Negatives (TN) are non-touch moves that are correctly classified as non-touch. False-Positives (FP) are normal moves but wrongly classified as touch moves by the classifier. False-Negatives (FN) are real face-touch samples but wrongly classified as Non-Touch by the classifier. The confusion matrix for the proposed solution is shown in Figure 4.17.

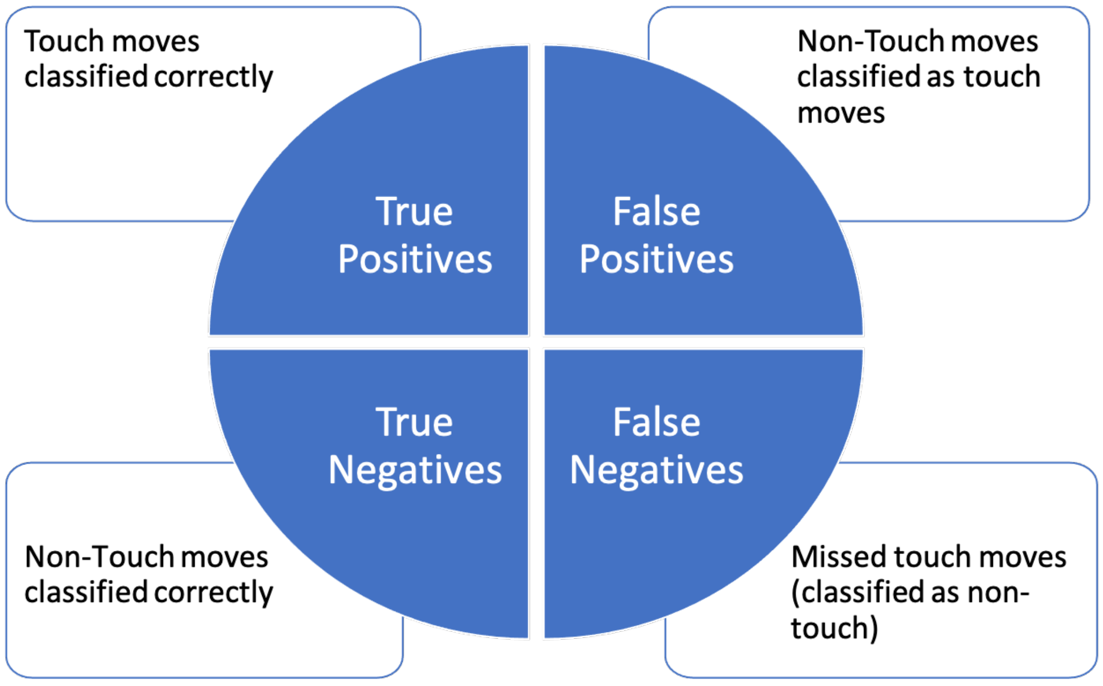

Touch moves classified correctly
Non-Touch moves classified as touch moves
True Positives
False Positives
True Negatives
False Negatives
Non-Touch moves classified correctly
Missed touch moves (classified as non-touch)

Figure 4.17. Definitions of TP, TN, FP, and FN for the proposed solution.

For evaluation, researchers apply several metrics like Recall (or sensitivity), Precision, Specificity, Accuracy, F1-Score, and False Positives Rate (FPR) to measure performance of trained models. In case of detecting face-touch hand moves, Recall given in Eq. 4.3 represents how many touches were detected out of all face touches made by the user. Precision given in Eq. 4.4 measures out of all face touches detected by the application how many real face touches were done. Specificity given in Eq. 4.5 gives the percentage of non-touch moves predicted correctly from all non-touch moves. Overall accuracy given in Eq. 4.6 calculates out of all samples in the dataset how many correctly classified were done by the model. F1-Score given in Eq. 4.7 is a measure that combines precision and recall. Positives-Rate (FPR) given in Eq. 4.8 represents the probability that a false alarm is raised.

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.4}$$

$$Specificity = \frac{TN}{TN + FP} = 1 - FPR \tag{4.5}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.6}$$

$$F1 - Score = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{4.7}$$

$$FPR = \frac{FP}{FP + TN} \tag{4.8}$$

4.8.2. Cross Validation

Cross Validation is a validation type where the training set is broken into parts. One of the parts is taken as test set while the rest is used as training set. Cross validation is useful when it is hard or expensive to collect a separate test dataset.

There are two types of cross validation, the first one is K-Fold cross validation. In this type, the training dataset is broken into K parts equal in size. Then, the model will be trained and tested K times where in each time one part is taken as test set and others combined as training set. In each time one different part is taken as test set until all K parts are tested. The second type of cross validation is Leave-One-Subject-Out (LOSO) where one subject data is used as test dataset and the rest as training set.

PART 5

EXPERIMENTAL RESULTS

In this part all experimental results are detailed based on three types of tests: Cross Validation testing, Separate Test Dataset testing, and Real-Life testing on the deployed smartwatch application.

5.1. CROSS VALIDATION TESTING

Two types of cross validation were done using the collected left and right training datasets, K-Fold and LOSO cross validation testing.

5.1.1. Five-Class Training Results

Although models are trained to fit the training set and it might overfit this dataset, training accuracy can give an initial view of the model accuracy. When looking at the confusion matrix particularly, percentage of misclassified samples with each class can be checked. Figure 5.1 shows for each class of the five motion classes, how many samples were misclassified with each of the rest four classes.

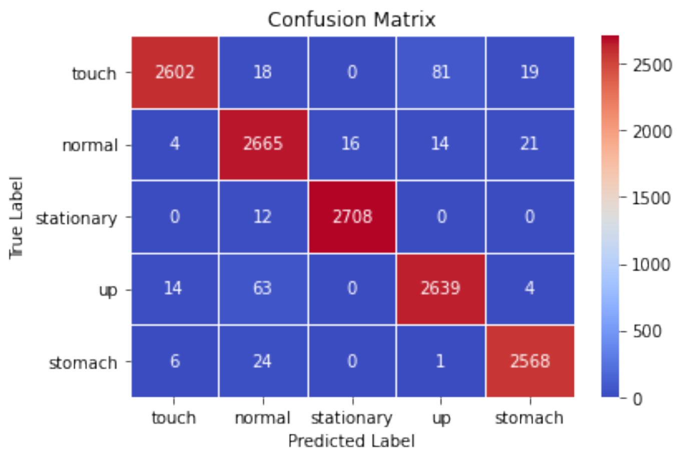


Figure 5.1. Confusion matrix for 5-class classification during training phase.

Figure 5.1. shows that most of missed face touches (False-Negatives) were classified as UP move. Besides, it can be seen that six stomach (Abdominal) samples were wrongly classified as Touch (False Positives).

Table 5.1 shows detailed precision, recall, f1-score for each class of the five classes: touch, normal, stationary, up, stomach (or abdominal).

Table 5.1. Precision recall and F1-score for all 5 classes.

Class	**precision**	**Recall**	**F1-score**
Touch	0.99	0.96	0.97
Normal	0.96	0.98	0.97
Stationary	0.99	1.00	0.99
Up	0.96	0.97	0.97
Abdominal	0.98	0.99	0.99

5.1.2. K-Fold Cross Validation Testing Results

To check the possibility of overfitting, two types of cross validation were done using the training dataset. K-Fold cross validation with 10-folds was firstly done by splitting training dataset to 10 equal parts. Test results show overall accuracy for 5-class classification between 95% and 97% across all 10 folds as shown in Figure 5.2.

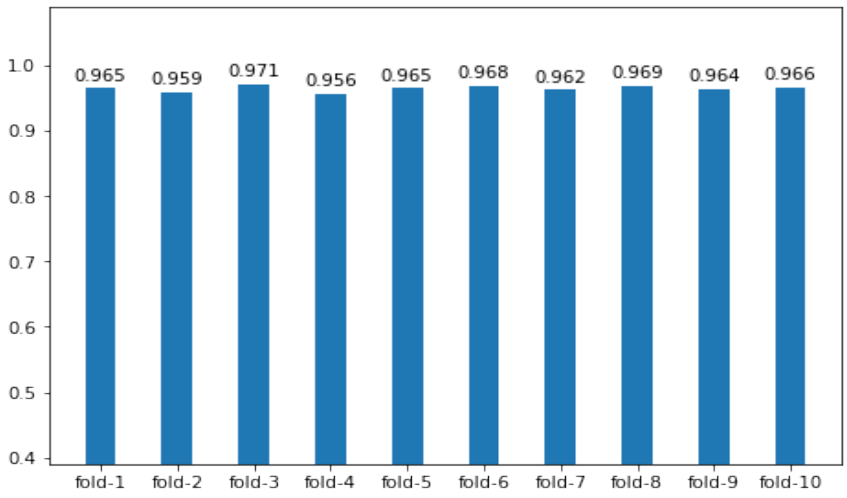


Figure 5.2. Accuracy across 10 folds in k-fold cross validation.

5.1.3. LOSO Cross Validation Testing Results

The second type of cross validation in this thesis is LOSO. The results for all three volunteers are close with accuracy between 95% and 96% as shown in Figure 5.3.

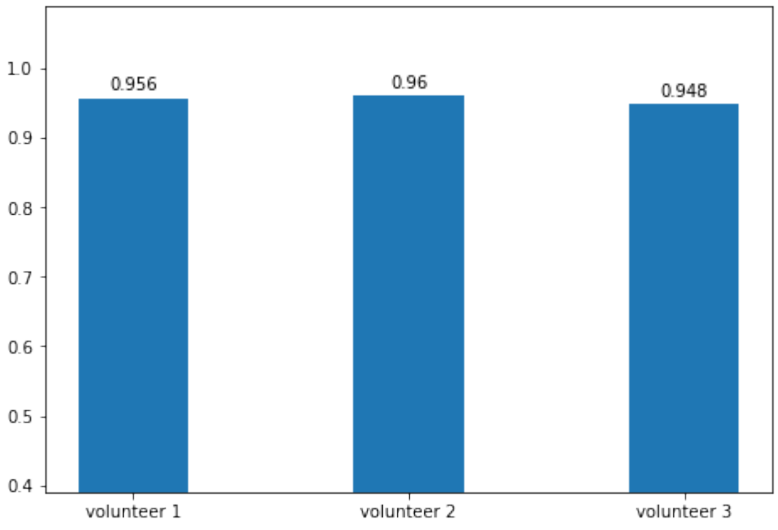


Figure 5.3. LOSO cross validation accuracy results for three volunteers.

5.2. TEST DATASET RESULTS

Since the training dataset was collected in controlled sessions where only one specific type of hand moves was done, testing results might not be representative to real-world testing. On the other hand, testing results on the Test Dataset give more representative results for real-world usage because it was collected in a free-living setup.

Test results show high recall and precision results with average F1-score of 95.85% as shown in Table 5.2.

Table 5.2. Face-Touch accuracy results on the Left/Right Test Datasets.

	Left Hand	**Right Hand**	**Average**
Recall	96.5%	97.0%	96.75%
Precision	93.7%	96.5%	95.10%
F1 Score	95.0%	96.7%	95.85%
Accuracy	99.7%	99.7%	99.70%

By using only linear acceleration and rotation rate features, the best recall score was 88.5% for F1-score of 86.7% (for left hand). However, when three gravity features were added, the recall jumped to 96.5% and F1-score reached to 95.0% as shown in Figure 5.4.

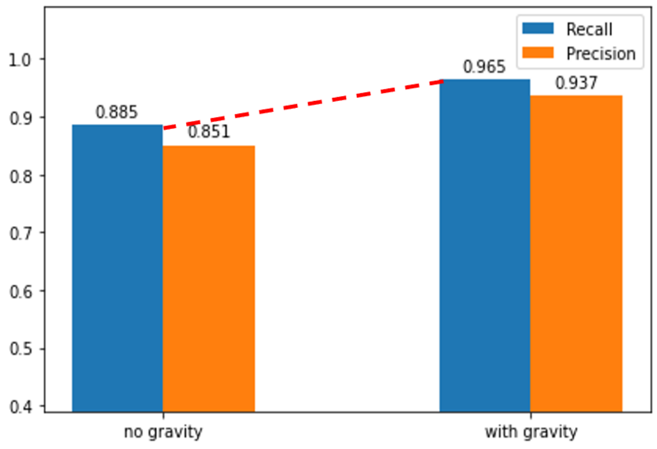


Figure 5.4. Results for two models with and without gravity.

Even when testing against over 200 different models trained with different hyperparameters during grid-search iterations, models with gravity consistently scored higher as shown in Figure 5.5.

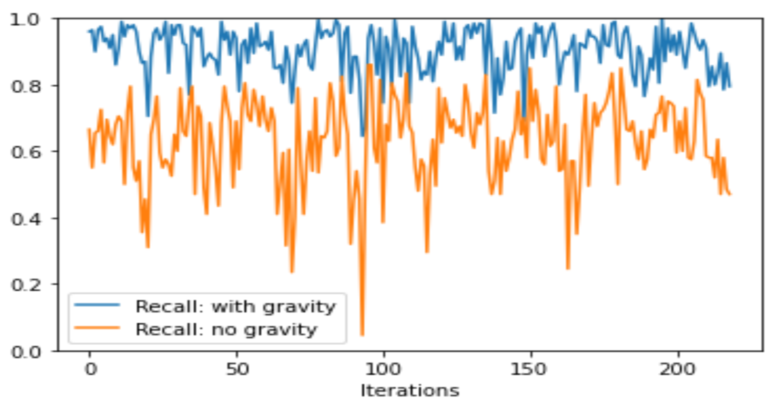


Figure 5.5. Results with/without gravity for multiple trained models.

Additionally, using five classes instead of binary classification had a big effect on minimizing false positives and improving precision and accuracy. Figure 5.6 shows a high precision improvement for 5-class models with a precision score of 93.7% compared with the best precision of 75.8% for 2-class models (for models with the best F1-score in both situations).

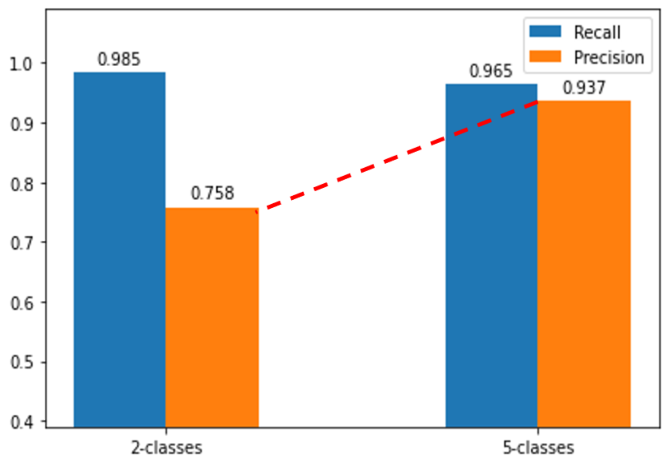


Figure 5.6. Recall/Precision scores for binary and 5-class classifiers.

Similarly, when testing against multiple models resulted from grid-search iterations, 5-class models consistently scored higher precision values compared with 2-class models with a big margin as shown in Figure 5.7. The precision score for binary classification was mostly in range of 0.5 and 0.7 which is considered very low and not practical for real-world usage on the application even if the recall score is high.

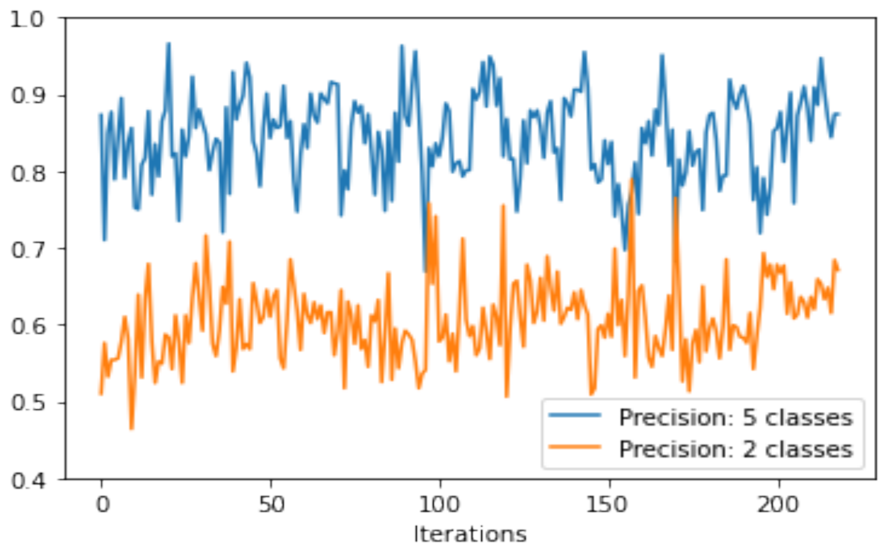


Figure 5.7. Precision for binary and 5-class classifiers for multiple trained models.

5.3. REAL TESTS ON THE APPLICATION

Testing was done on the final smartwatch application to test accuracy results, battery consumption, and percentage of false alarms over long periods.

5.3.1. Test Results of the Application

To check real-world performance, the final trained models were deployed on the developed application which was deployed on two different watches. The application was able to give real-time alert and haptic feedback and showed the current count of face touches. Five volunteers were asked to use the application extensively and do face touch attempts repeatedly in all five scenarios for each hand. The average detection accuracy was 97% for both hands.

5.3.2. False-Positives Test Results

To test false positives rate over long period, each volunteer was asked to use the application for five hours while doing normal daily-life activities.

For each volunteer data which contain around 30k hand motion samples, the average false positives rate was 0.04% or 1 false positive every 2500 samples which in our setup translates to 2.4 FPs/hour in normal daily life usage. Besides, specificity was calculated as 99.6%. Such very low FPR and high specificity mean that the application is alerting the user only when there is very high confidence of potential face-touch move. This makes the application suitable for long hours use without much disturbance for the user that can be caused by too many false alarms.

5.3.3. Battery Test Results

In terms of battery consumption, the application was power-efficient and battery drain was minimal. Testing results showed that running the application continuously in the background added less than 2% battery drain per hour as shown in Table 5.3. That means even running the application for 10 hours continuously consumes only 20% extra battery.

Table 5.3. Battery usage percentage with and without running the application.

	Battery Level After 5 Hours		
	Application is Not Running	**Application is Running**	**Consumption by the Application**
Apple Watch 3	93%	86%	7%
Apple Watch SE	91%	81%	10%

PART 6

DISCUSSION AND CONCLUSION

6.1. COMPARING RESULTS WITH LITERATURE

The methods and configurations used in the literature have an effect on classification of HGR, and FTD. Different studies used different approaches to overcome obstacles and achieve high results. Table 6.1 compares final results based on used devices, sensors, filtering, segmentation, feature extraction, and classification techniques.

Table 6.1. Literature comparison HGR and FTD applications.

Paper	Devices & Sensors	Filtering + Segmentation + Feature Extraction	Classification	Accuracy Results + Testing method
[21]	- Smartwatch + Phone +Server - Sensors: Accelerometer	- No filtering - Window: 10s, 10Hz - Features: raw data	End-to-End CNN	- Precision: 97.3%, Recall: 97.32%, F1-Score= 97.32% - 5-Fold Cross Validation
[22]	- wearable sensors on wrist and arm + wireless receiver + PC - Sensors: Accelerometer, Gyroscope	- No filtering - Features: Device Angles (Quaternions)	SVM + ANN	- Accuracy: na - K-Fold cross validation
[16]	-Smartwatch - Sensors: Accelerometer, Gyroscope	- No Filtering - Window: 1s, 5Hz - Features: 7 time-domain features + 10 frequency-domain bands	DWT + KNN/SVM	- F1: 87%
[17]	- Smartwatch - Sensors: Accelerometer, Gyroscope	- No Filtering - Window: 1s, 50Hz - Features: 11 features in Time-Domain and Frequency-Domain	KNN, SVM, NB	- Accuracy: na - Separate test dataset

[18]	Wrist-worn device Sensors: Accelerometer, Gyroscope	- Low-Pass filter - Window: 2.5s, 14s – 200Hz - Features: raw data	DTW + Decision Tree	- Accuracy: 95.6% - 10-Fold
[4]	Smartwatch + Phone Sensors: Accelerometer, Gyroscope	- No Filtering - Features: 3 acceleration + 3 gravity + 3 angles	DTW	- Accuracy: 71%
[23]	- Magnetic ring + RFID tags on goggles + Software on PC - Sensors: RFID + Magnetometer	- No Filtering - Window: 5s, 20Hz - Features: Radio signal and magnetic readings	KNN classifier	- 65% accuracy - Tests done on 1 person
[25]	- IR Camera + PC - Sensors: Depth Sensors, IR laser sensors	- Features: depth image data	Special Algorithm	- 90.96% accuracy
[24]	- Magnetic necklace + Smartwatch - Sensors: magnetometer	- MEKF filter - Window: 0.5s, 100Hz - Features: Raw magnetic data	Special algorithm with thresholds	- Recall: 91%, FPR: 3.8% - Tests done by (5-6) people in home setup doing ADL for 8h per day
[19]	- Smartwatch - Sensors: accelerometer	- No Filtering - Window: 1.5s, 100Hz, with 85% overlap - Features: (sum, mean, median, standard deviation, coefficient of variation, zero crossing, mean/median absolute deviation, skewness and kurtosis).	Random Forest	- Recall: 89%, FPR: 0.56% - Tests done by 3 people in home setup doing ADL
[20]	- Smartwatch - Sensors: Accelerometer, Gyroscope	- Low-Pass filter - Features: mean, std, mad, energy, correlation, skewness, entropy, kurtosis	Conventional classifiers (SVM, iForest, MCD, LOP) or CNN	- Recall: 86%, Precision: 90% - Tests done by 1 new person with 100 test samples

6.2. COMPARISON WITH THE STATE OF THE ART ON FTD

The proposed approach achieved overall accuracy of 99.7% on the test dataset for classifying Touch/No-Touch hand moves while keeping the models power efficient. In comparison, studies in [23,25] reported accuracy results for touching different face-parts. In [23] authors used RFID tags mounted on plastic goggles in addition to magnetic rings and measured touch detection accuracy for different face locations.

Similarly, authors in [25] used depth cameras to detect face touches and measure workers compliance with health practices and reported accuracy for touching multiple face parts. All reported accuracy results are shown in Table 6.2.

Table 6.2. Accuracy comparison with other approaches.

Study	Accuracy
Takayama et al. (2020) [23]	83.0%
Manghisi et al. (2020) [25]	90.7%
The proposed method	99.7%

In addition, the trained models have a better recall, precision, and lower FPR compared with the study [20] which used feature extraction and CNN networks, with the study [19] which used hand-crafted features and RF classifier, and with the study [24] which used magnets to measure the proximity between the hand and the face as shown in Table 6.3.

Table 6.3. Comparison for Recall, Precision, and False Positives Rate.

Study	Recall	Precision	FP Rate
Sudharsan et al. (2020) [20]	91%	--	3.8%
Xiang Chen (2020) [19]	89%	--	0.56%
Aurizio et al. (2020) [24]	86%	90%	--
The proposed method	96.75%	95.1%	0.04%

6.3. SUMMARY

In this thesis, it is presented a complete approach for classifying hand moves to detect face touches using only smartwatch IMU motion sensors and CNN networks without using any extra hardware equipment. The goal is to alert users and prevent unwanted face touches which can be one of the main causes for transmitting viral infections. The proposed approach utilizes smart data processing for automatic sample extraction to gather large training dataset with around 28k samples collected from left and right hand. Using sensor fusion assisted to eliminate bias and accumulated error. Adding gravity features was effective in improving touch detection accuracy. Besides, dividing hand moves into multiple classes, collecting data in multiple scenarios and multiple hand orientations minimized false positives. The proposed solution used the lowest configurations compared with the study [36] based on window size, layer count and model size. These efficient configurations allowed the models to run directly on the watch with real-time performance while preserving battery at the same time. Testing results show the proposed approach outperformed classical methods that use hand-crafted feature extraction and fared better than approaches that use extra hardware components. The application provides real-time feedback and alerts the user with vibration and sound whenever attempting to touch the face. The obtained results for average recall, precision, F1-Score, and accuracy were calculated as 96.75%, 95.10%, 95.85%, 99.70% respectively, with low FPR as 0.04%. By using efficient configurations and small models, the application achieves high efficiency and can run for long hours without significant impact on battery which makes it applicable on most out-of-the-shelf smartwatches.

REFERENCES

1. Z. Zhuang and Y. Xue, “Sport-related human activity detection and recognition using a smartwatch,” ***Sensors (Switzerland)***, vol. 19, no. 22, pp. 1–21, 2019, doi: 10.3390/s19225001.

2. G. Laput and C. Harrison, “Sensing fine-grained hand activity with smartwatches,” ***Conf. Hum. Factors Comput. Syst. - Proc.***, 2019, doi: 10.1145/3290605.3300568.

3. J. Hou et al., “SignSpeaker: A real-time, high-precision smartwatch-based sign language translator,” ***Proc. Annu. Int. Conf. Mob. Comput. Networking***, MOBICOM, 2019, doi: 10.1145/3300061.3300117.

4. D. Moazen, S. A. Sajjadi, and A. Nahapetian, “AirDraw: Leveraging Smart Watch Motion Sensors for Mobile Human Computer Interactions,” ***arXiv***, pp. 1–5, 2017.

5. M. Kim, J. Cho, S. Lee, and Y. Jung, “Imu sensor-based hand gesture recognition for human-machine interfaces,” ***Sensors (Switzerland)***, vol. 19, no. 18, pp. 1–13, 2019, doi: 10.3390/s19183827.

6. Y. L. A. Kwok, J. Gralton, and M. L. McLaws, “Face touching: A frequent habit that has implications for hand hygiene,” ***Am. J. Infect. Control***, vol. 43, no. 2, pp. 112–114, 2015, doi: 10.1016/j.ajic.2014.10.015.

7. K. S. Bate, J. M. Malouff, E. T. Thorsteinsson, and N. Bhullar, “The efficacy of habit reversal therapy for tics, habit disorders, and stuttering: A meta-analytic review,” ***Clin. Psychol. Rev.***, vol. 31, no. 5, pp. 865–871, 2011, doi: 10.1016/j.cpr.2011.03.013.

8. K. Katevas, H. Haddadi, and L. Tokarchuk, “Sensing Kit: Evaluating the sensor power consumption in iOS devices,” ***Proc. - 12th Int. Conf. Intell. Environ. IE 2016***, pp. 222–225, 2016, doi: 10.1109/IE.2016.50.

9. M. Abo-Tabik, N. Costen, J. Darby, and Y. Benn, “Towards a smart smoking cessation app: A 1D-CNN model predicting smoking events,” ***Sensors***

(Switzerland), vol. 20, no. 4, pp. 1–18, 2020, doi: 10.3390/s20041099.

10. V. Y. Senyurek, M. H. Imtiaz, P. Belsare, S. Tiffany, and E. Sazonov, “A CNN-LSTM neural network for recognition of puffing in smoking episodes using wearable sensors,” ***Biomed. Eng. Lett.***, vol. 10, no. 2, pp. 195–203, 2020, doi: 10.1007/s13534-020-00147-8.

11. S. Akther, N. Saleheen, S. A. Samiei, V. Shetty, E. Ertin, and S. Kumar, “mORAL: An mHealth Model for Inferring Oral Hygiene Behaviors in-the-wild Using Wrist-worn Inertial Sensors,” ***Proc. ACM Interactive, Mobile, Wearable Ubiquitous Technol.***, vol. 3, no. 1, pp. 1–25, 2019, doi: 10.1145/3314388.

12. Y. Shen, J. Salley, E. Muth, and A. Hoover, “Assessing the Accuracy of a Wrist Motion Tracking Method for Counting Bites Across Demographic and Food Variables,” ***IEEE J. Biomed. Heal. Informatics***, vol. 21, no. 3, pp. 599–606, 2017, doi: 10.1109/JBHI.2016.2612580.

13. P. W. Jasper, M. T. James, A. W. Hoover, and E. R. Muth, “Effects of Bite Count Feedback from a Wearable Device and Goal Setting on Consumption in Young Adults,” ***J. Acad. Nutr. Diet***., vol. 116, no. 11, pp. 1785–1793, 2016, doi: 10.1016/j.jand.2016.05.004.

14. A. Umek and A. Kos, “Limitations of Smartphone MEMS for motion analysis,” ***pp. 450–454, 2015***.

15. H. Guo and H. Hong, “Research on filtering algorithm of MEMS gyroscope based on information fusion,” ***Sensors (Switzerland)***, vol. 19, no. 16, 2019, doi: 10.3390/s19163552.

16. H. Wen, J. R. Rojas, and A. K. Dey, “Serendipity: Finger gesture recognition using an off-the-shelf smartwatch,” Conf. Hum. Factors Comput. Syst. - Proc., pp. 3847–3851, 2016, doi: 10.1145/2858036.2858466.

17. Y. Li, N. Yang, L. Li, L. Liu, and Y. Yang, “Finger gesture recognition using a smartwatch with integrated motion sensors,” Web Intell., vol. 16, no. 2, pp. 123–129, 2018, doi: 10.3233/WEB-180378.

18. J. Wu and R. Jafari, “Orientation independent activity/gesture recognition using wearable motion sensors,” ***IEEE Internet Things J.***, vol. 6, no. 2, pp. 1427–1437, 2019, doi: 10.1109/JIOT.2018.2856119.

19. X. Chen, “FaceOff: Detecting face touching with a wrist-worn accelerometer,” ***arXiv***, pp. 1–5, 2020.

20. B. Sudharsan, D. Sundaram, J. G. Breslin, and M. I. Ali, “Avoid Touching Your Face: A Hand-to-face 3D Motion Dataset (COVID-away) and Trained Models for Smartwatches,” ***ACM Int. Conf. Proceeding Ser***., no. October, 2020, doi: 10.1145/3423423.3423433.

21. M. C. Kwon, G. Park, and S. Choi, "Smartwatch user interface implementation using CNN-based gesture pattern recognition," ***Sensors (Switzerland)***, vol. 18, no. 9, pp. 1–12, 2018, doi: 10.3390/s18092997.

22. S. Alavi, D. Arsenault, and A. Whitehead, "Quaternion-based gesture recognition using wirelesswearable motion capture sensors," ***Sensors (Switzerland)***, vol. 16, no. 5, 2016, doi: 10.3390/s16050605.

23. Y. Takayama, Y. Ichikawa, T. Kitagawa, S. Shengmei, B. Shizuki, and S. Takahashi, "Touch Position Detection on the Front of Face Using Passive High-Functional RFID Tag with Magnetic Sensor," ***Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)***, vol. 12182 LNCS, pp. 523–531, 2020, doi: 10.1007/978-3-030-49062-1_35.

24. N. D'Aurizio, T. L. Baldi, G. Paolocci, and D. Prattichizzo, "Preventing Undesired Face-Touches with Wearable Devices and Haptic Feedback," ***IEEE Access***, vol. 8, pp. 139033–139043, 2020, doi: 10.1109/ACCESS.2020.3012309.

25. V. M. Manghisi et al., "A body tracking-based low-cost solution for monitoring workers' hygiene best practices during pandemics," ***Sensors (Switzerland)***, vol. 20, no. 21, pp. 1–17, 2020, doi: 10.3390/s20216149.

26. R. M. Al-Eidan, H. Al-Khalifa, and A. M. Al-Salman, "A review of wrist-worn wearable: Sensors, models, and challenges," ***J. Sensors***, vol. 2018, 2018, doi: 10.1155/2018/5853917.

27. M. L. Zhang and Z. H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," ***Pattern Recognit***., vol. 40, no. 7, pp. 2038–2048, 2007, doi: 10.1016/j.patcog.2006.12.019.

28. S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient kNN classification with different numbers of nearest neighbors," ***IEEE Trans. Neural Networks Learn. Syst***., vol. 29, no. 5, pp. 1774–1785, 2018, doi: 10.1109/TNNLS.2017.2673241.

29. C. Schuldt, L. Barbara, and S.- Stockholm, "Recognizing Human Actions : A Local SVM Approach ∗ Dept . of Numerical Analysis and Computer Science," ***Pattern Recognition***, 2004. ICPR 2004. Proc. 17th Int. Conf., vol. 3, pp. 32–36, 2004.

30. Z. He and L. Jin, "Activity recognition from acceleration data based on discrete consine transform and SVM," ***Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern***., no. October, pp. 5041–5044, 2009, doi: 10.1109/ICSMC.2009.5346042.

31. M. Pal, "Random forest classifier for remote sensing classification," ***Int. J. Remote Sens***., vol. 26, no. 1, pp. 217–222, 2005, doi: 10.1080/01431160412331269698.

32. T. Shi and S. Horvath, “Unsupervised learning with random forest predictors,” ***J. Comput. Graph. Stat.***, vol. 15, no. 1, pp. 118–138, 2006, doi: 10.1198/106186006X94072.

33. P. Senin, “Dynamic Time Warping Algorithm Review,” Science (80-.)., vol. 2007, no. December, pp. 1–23, 2008, [Online]. Available: ***http://129.173.35.31/~pf/Linguistique/Treillis/ReviewDTW.pdf***.

34. R. Varatharajan, G. Manogaran, M. K. Priyan, and R. Sundarasekar, “Wearable sensor devices for early detection of Alzheimer disease using dynamic time warping algorithm,” ***Cluster Comput.***, vol. 21, no. 1, pp. 681–690, 2018, doi: 10.1007/s10586-017-0977-2.

35. F. J. Ordóñez and D. Roggen, “Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition,” ***Sensors (Switzerland)***, vol. 16, no. 1, 2016, doi: 10.3390/s16010115.

36. W. S. Lima, E. Souto, K. El-Khatib, R. Jalali, and J. Gama, “Human activity recognition using inertial sensors in a smartphone: An overview,” ***Sensors (Switzerland)***, vol. 19, no. 14, pp. 14–16, 2019, doi: 10.3390/s19143213.

RESUME

Abdullah ALESMAEIL graduated from Damascus university in 2005 with bachelor's degree in computer engineering. He worked more than 8 years as a software engineer in different sectors such as telecom, banking, and enterprise solutions. In 2014 he co-founded a startup specialized in development of mobile and wearable applications. In 2020 he moved to Karabük and started his master degree in Computer Engineering in Karabük university.