UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY
ADVANCED PROGRAM IN COMPUTER SCIENCE

NGUYỄN HIỀN TUẤN DUY - NGUYỄN TRƯƠNG VĨNH THUYÊN

TOWARDS A GRATIFYING INTERACTIVE MODALITY FOR SMART ENVIRONMENTS BASED ON UBIQUITOUS SENSING

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

HO CHI MINH CITY, 2021

UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY
ADVANCED PROGRAM IN COMPUTER SCIENCE

NGUYỄN HIỀN TUẤN DUY - 1751001
NGUYỄN TRƯƠNG VĨNH THUYÊN -1751042

TOWARDS A GRATIFYING INTERACTIVE MODALITY FOR SMART ENVIRONMENTS BASED ON UBIQUITOUS SENSING

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

THESIS ADVISOR:
ASSOC. PROF. TRẦN MINH TRIẾT

ACADEMIC YEAR 2017-2021

ACKNOWLEDGEMENT

Foremost, we want to deeply thank our advisor – Assoc. Prof. Tran Minh Triet. You are a continual source of wisdom and advice to us. It has been a privilege of working along side you and being given the freedom to explore our own direction. We are deeply grateful for your tremendous supports, both intellectually and emotionally, and all of the inspiration you have been giving us since the day we started and throughout our university life.

We would also like to thank our thesis reviewer, Dr. Nguyen Thi Minh Tuyen, not only for being very patient and supportive towards our thesis progress but also for your kind and thoughtful feedback.

We express our gratitude to our thesis committee for the insightful and encouraging feedback – Prof. Duong Nguyen Vu, Assoc. Prof. Vu Hai Quan, and Dr. Tran Thai Son.

We would also like to express our gratitude to the faculties and teaching assistants of the Advanced Program in Computer Science (APCS) and the Faculty of Information Technologies (FIT), especially Prof. Nguyen Huu Anh, Assoc. Prof. Dinh Dien, Assoc. Prof. Dinh Ngoc Thanh, Dr. Dinh Ba Tien, Dr. Nguyen Van Vu, Dr. Nguyen Ngoc Thao, Dr. Nguyen Phuc Son, Dr. Nguyen Tuan Nam, Dr. Nguyen Vinh Tiep, Nguyen Huu Tri Nhat MSc., Tran Anh Duy MSc., and Ho Tuan Thanh MSc. We are indebted to your brilliant and engaging teaching, and your unwavering support throughout our university life.

To senior members of SELab – Nguyen Hai Dang, MSc., Nguyen Thanh An, MSc., Tran Quang Tanh, MSc., Do Trong Le, Tran Ngoc Dat Thanh, Hoang Trung Hieu, Tran Mai Khiem, Huynh Viet Tham, Vu Le The Anh; and senior members of Robotics and IoT lab – Cao Xuan Nam, MSc. and Dang Hoai

Thuong, we are indebted to all the supports and guidance you have given us. Thank you for sharing with us all the wonderful moments at the labs.

To our peers – Nguyen Ho Huu Nghia, Dao Hieu, Nguyen Quang Thuc, Nguyen Ngoc Minh Huy, Tran Gia Huan, Do Tri Nhan, Nguyen Minh Tri, Nguyen Diep Xuan Quang, Tran Bao Phuc, Vo Trung Thanh, Tran Thi Anh Thu, Nguyen Thi Kim Phuong, Nguyen Hoang Tan, Nguyen Phuc Minh, Le Tan Dang Tam, Truong Tuc Luan, Nguyen Van Quang Tien, Trinh Huu Duc, Tran Trieu Thanh, Huynh Thanh Vy, Do Nhat Huy, Nguyen Le Nguyen, Tran Cong Hoang Trong, Tran Duc Huy, Vo Kien, Doan Ngoc Anh Khoa, Dinh Nguyen Hoang Kim, Nguyen Hoang Thanh Tuan; our juniors – Pham Bang Dang, Nguyen Ho Thang Long, Duong Anh Kiet, Le Thanh Danh, Huynh Tuan Luc, Trinh Van Minh, Nguyen Phu Van, Ho Thi Ngoc Phuong, Nguyen Ngoc Bang Tam, Vo Tien Dung; and the whole APCS class of 2017-2021. You are an important part to this life chapter of us and a huge source of inspiration. We are grateful and adore the hard work and discussion we had together during the course of 4 years. Thank you for sharing with us all of the moments.

We thank Nguyen Minh Tuan and Shad Roi from Google, and Ms. Hoang Thanh Tu of the APCS for helping us establishing the Google Developer Student Club at Ho Chi Minh City University of Science (GDSC-HCMUS), which has been an important experience during our time at the university. We thank the members of GDSC-HCMUS for sharing the experience with us, you are both our hope and inspiration in building a meaningful and lasting alumni network.

Tuan Duy dedicates this thesis for his parents – ba Tuan and me Thao, his guide parents – bac Thanh and bac Yen, and his brother Bao Duy. He is forever grateful for their cares, sacrifices, and tender love through everything, and for being his lifelong inspiration. Tuan Duy would also like to express deep gratitude towards Prof. Park Chan-ik, his advisor during the time he exchanged at Pohang

University of Science and Technology (POSTECH, South Korea) for the supports and cultivation he has given Tuan Duy, which has encouraged him greatly to pursue the path of becoming a good researcher. Tuan Duy is forever indebted with the trust and favors Prof. Park has given him, especially the chance of attending POSTECH and the chance to start have his first attempts in true scientific research at SSLab. Tuan Duy would also like to thank his industry advisors, Assistant Prof. Vo Duy Tin (Lakehead University, Ontario, Canada) and Luong Tien Manh, MSc., for their patience and mentorship during Tuan Duy's time at VinAI Research, introducing him to many interesting research in speech signal processing and engineering practices that have taken an influence in this thesis. Tuan Duy also thanks Prof. Karl Aberer and Duong Chi Thang, MSc. for their endeavor during his internship with the École polytechnique fédérale de Lausanne (EPFL, Switzerland), which has helped widen his knowledge horizon and developed crucial principles in doing research and engineering. Tuan Duy thanks all of his friends, including members of the Pathfinder social startup – Le Duy Luat, Nguyen Ho Huu Nghia, Banh Thanh Son, Dao Hieu, Lai Phan Quynh Anh, and Do Nguyen Phuong Quynh; ones he met at POSTECH – Dr. Nguyen The Kha, Dr. Dinh Thi Hong Nhung, Dr. Truong Thi My Nhung, Dr. Hong Minh Triet, Nguyen Van Tu MSc., Tran Diem Nghi, Tim Bui, Dr. Ma Jeong-hyun, Dr. Lee Un-sung, Hong Sang-won, MSc., Noh Yong-du, MSc., Chae Byung-hoon, MSc., Shin Dong-min, MSc., Park Hae-sung, MSc., Jung Woo-chang, MSc., Jo Yongrae, MSc., Krishna Gopal Rajput, Syed Nurul Hasan, Anoop Gopal Singh, Tinting Ji, Sara Chaoufi, Clemens Reinhardt, MSc., Mostafa Bestamy, Inha Lee, Raul Ruiz, Juan Limon, Lorenzo Torres, Marlon Pillasagua Nemer MSc., Kim Eun-sik MSc., Larissa Miasiro MSc., Nikita Sterlyus, Kristeen Neuman, Song Sihao, Kevin Garcia, Ousseynou Fall, Hector Dang-Nhu MSc., Dr. Juval Tongco, Dr. Tim King, Dr. Jingeol Lee, Ong Wei Hua, and Tan Wei Xuan; and everyone that is yet to be mentioned here, for going with him through important

chapters of his life and giving him invaluable lessons. Last but not least, Tuan Duy thanks his girlfriend – Le. He cherishes the fact that they have always been learning from each other, and have been maturing together for a long time. He would like to thank Le for complementing him in every ways, and for all the love and supports.

Vinh Thuyen can't even explain how much belief and love of his parent means to him. He is grateful eternally for his mother's sacrifices; for his dad who has tried his best to live and to be a good father every single moment for exact 10 years. He is proud of his two big brothers' love and support for him. During Thuyen's high school time, his homeroom teaches Miss. Phan Lan and his school's vice president Nguyen Trung Nhan always encourage him to follow his passion. Moreover, to meet Miss. Dang Thi Tuong Vy in his eleventh grade is a turning point, she explored his potential and encouraged him to learn coding and computer science. Vinh Thuyen values and respects Assoc. Prof. Dinh Dien for his efforts and commitments in natural language processing in Vietnamese and for the supports he has given Thuyen, which always encourages him greatly to keep Thuyen's passion in research journey. Vinh Thuyen would like to thank Assoc. Prof. Tran Minh Triet once again for the encouragement, he has provided Thuyen at the moment Thuyen's lost which has helped Thuyen to has confidence to get through tough times. Thuyen is grateful for having a first job being a teaching assistant at RocketAcademy, he has gathered a lot of knowledge and experience by working for a startup company; surprisingly, he worked with the founder Kai-Yuan Neo who is the drummer of his favorite song's band. Vinh Thuyen would like to thank Do Nhat Huy, Tran Trieu Thanh, Tran Nguyen Hien, Doan Le Cat Uyen for our little music band. Thuyen truly appreciates efforts and commitments on dotosave project with his friends Nguyen Hoang Thanh Tuan, Tran Bao Phuc, Nguyen Pham Dan Anh, Phan Kim Thai. He also would like to thank ever familiar faces in 135B Tran Hung Dao dormitory including my roommates Tran

Cong Hoang Trong, Pham Huynh Nhat, Tran Bao Khanh, Nguyen Dang Khoa, Dinh Vu Quynh, Truong Nhat Ninh, Duong Cong Hau, Nguyen Phong Hao, Pham Toan, Nguyen Nhat Ninh, and dormitory staffs and assistants, especially chu Loc for taking care of Thuyen and other students live in 135B dormitory for many years. Vinh Thuyen is so grateful for having a lot of good friends in his university time, especially APCS friends. Thuyen truly wants to say thank to an APCS friend who always is a motivation for Thuyen to keep going. Finally, Thuyen wants to say thank to Kita, Luna, Tuna, Giga, Sola, Xuka and especially Fugi for being with me.

Finally, we would like to dedicate a special thank to Prof. Duong Nguyen Vu, who is not only a member of our thesis committee, but also the creator of the wonderful APCS. You are the father figure and source of inspiration for generations of students enrolling in the program. None of this would have happened without you. Your first-day guidance means the world to us. Not only it launched us to committing good and meaningful engineering and scientific work, but also introducing us to a positive living attitude of always putting in 100% of effort in all deeds of life in general.

THESIS SYLLABUS

Thesis title: Towards a Gratifying Interactive Modality for Smart Environments based on Ubiquitous Sensing
Advisor: Assoc. Prof. Trần Minh Triết
Duration: December 1st, 2020 to July 31th, 2021
Students: Nguyễn Hiền Tuấn Duy (1751001) - Nguyễn Trương Vĩnh Thuyên (1751042)
Introduction: The rapid development of telecommunication technology, computing technology and wearable technology increases the need of new modality for interaction between human and these systems. Many interaction techniques are proposed using either proof-of-concept wearable devices devices or off-the-shelf wearable devices. The advantage of building device prototypes for new interaction techniques is that helps researchers easier way to implement their ideas by adding necessary components. While, techniques that are proposed using off-the-shelf devices are limited due to manufacturers' design. Studies using off-the-shelf devices often for investigating the feasibility of new applications. An advantage of these studies is that devices are used in these studies available to buy and easier to use. Smart watches are the most popular off-the-shelf wearable devices and suitable for everyday-carry and daily usage. They contains various types of sensors suitable for various applications. With the development of technologies that relate to smart watches such as micro-computing, battery technology, etc., modern smart watches can yield a gratifying interactive modality.

Content:

This thesis presents a smart watch inertial signal processing method for activity recognition based on an approach for speech recognition. Following that, we introduce off-the-shelf Android smart watch enhancement method and signal transmission method for high frequency signal sequence transmission. Finally, we propose an interaction scenario that is performing authentication on computer platform using smart watch. In order to demonstrate, we build an multiple platforms authentication system.

Main tasks:

- Literature Review:

  - We gather knowledge from studies on Human-Computer Interaction topics and find our interests.

  - We explore user study method and interactive system design from Human-Computer Interaction studies.

  - We focus on gathering knowledge and understanding studies on smart environments, ubiquitous sensing and wearable technology.

- Feature Enhancement for Smart watch

  - We gather knowledge to understand method to enhance Android smart watch for support higher sensors sign frequency.

  - We gather knowledge to understand Android smart watch hardware.

  - We perform hardware and firmware interference for devices enhancement.

  - We implement application for signal transmission from smart watches to personal computers.

- Inertial Signal Sequence Understanding

  - We explore state-of-the-art methods in smartwatch-based human activity and gesture recognition.

  - We explore state-of-the-art methods in learning-based sequential signal processing.

  - We design human activity and gesture recognizers with design choices that facilitate deployment and data collection.

  - We implement human activity and gesture recognizers.

  - We evaluate proposed human activity and gesture recognizers.

- Smart Interaction System With Smart Watch

  - We explore usage scenarios for interaction techniques, propose new usage scenarios if possible.

  - We research and understand authentication systems on popular computer platforms and method to implement authentication modules for these platforms.

  - We implement shared component for a smart watch interaction system for authentication supports multiple platforms.

  - We implement locally credential component for following platforms: Windows, Linux, and Web-based.

Results:

We transform existing smart watch to be used as smart input devices. Then, we propose an activity recognition algorithm using inertial signal sequence inspired from speech recognition domain. Finally, we build a flexible authentication system for multiple platforms authentication using smart watch that system is customizable, upgradable and able to extend to new platform.

Research timeline

- December 2020 - February 2021: do a research on related topics to gather knowledge and write literature review.
- February - March 2021:

  - Purchase necessary devices and study methods to modify hardware and firmware of the devices.

  - Perform hardware and firmware inference and test the frequency results.

  - Implement data transmission application.

  - Search for benchmark datasets to perform experience in inertial signal
- March - April 2021:

  - Keep performing hardware and firmware inference and test the frequency results.

  - Perform more experience in inertial signal sequence.

  - Perform a research for authentication system on popular platform.

- April - June 2021:

  - Purchase other devices if previous devices can not be overclocked.

  - Implement shared component of our authentication system,

  - Study method to implement credential providers, authentication modules on each platform.

  - Deploy a demo of activity recognition model.

  - Perform more experience in activity recognition model and evaluate using numerical benchmark.

- June - July 2021:

  - Implement custom Windows Credential Providers.

  - Implement Google Extension for autologin to support web-based application.

  - Implement custom PAM module for supporting Linux platforms.

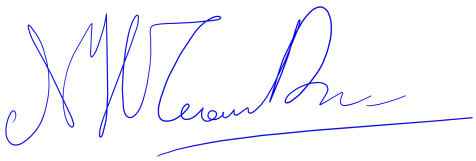
	July 14th, 2021
Advisor	**Students**
	
Assoc. Prof. Trần Minh Triết	Nguyễn Hiền Tuấn Duy
	
	Nguyễn Trương Vĩnh Thuyên

TABLE OF CONTENTS

Page

Acknowledgement ii

Thesis Syllabus vii

Table of Contents xii

List of Tables xvii

List of Figures xviii

Abstract xx

CHAPTER 1 – INTRODUCTION

1.1 Smart Environments 1

1.2 Human-Computer Interaction with Ubiquitous Sensing and Wearable Technology 4

1.2.1 Ubiquitous Sensing 6

1.2.2 Wearable Computing 7

1.3 Motivation 13

1.4 Objective 14

1.4.1 Objectives 14

1.4.2 Tasks 14

1.4.3 Contributions 16

1.5 Outline 16

CHAPTER 2 – FEATURE ENHANCEMENT FOR SMART ANDROID DEVICES WITH INERTIAL SENSORS

2.1 Hardware and Firmware Interference 17

2.1.1 LG G Watch W100 18

2.1.2 Motorola 360 (2nd generation) 19

2.1.2.1 Build a kernel 22

2.1.2.2 Smelt kernel source: M4 sensorhub 24

2.2 Signal Collecting 24

2.2.1 Client 26

2.2.1.1 UDPSender 26

2.2.1.2 Sensor Event Listener 28

2.2.2 Server 29

CHAPTER 3 – INERTIAL SIGNAL SEQUENCE UNDERSTANDING FOR SMART INTERACTION

3.1 Signal Processing 31

3.1.1 Heterogeneities and time-domain vs. frequency domain features 32

3.1.2 Discrete Fourier Transform 32

3.1.3 Short-time Fourier Transform 33

3.2 Learning-based Signal Processing 34

3.2.1 Foundational Methods 34

3.2.1.1 Convolutional Neural Network 34

3.2.1.2 Separable Convolutional Filters 36

3.2.1.3 Depthwise Separable Convolutional Layer 37

3.2.1.4 Time-depth and time-channel Separable Convolutional Layer 37

3.2.1.5 Connectionist Temporal Classification Loss 39

3.2.2 Compound Methods 41

3.2.3 QuartzNet - Separable Convolutional Network for Sequential Signal 41

3.2.3.1 QuartzNetBxR-1D 43

3.2.3.2 QuartzNetBxR-3D 43

3.3 Implementation Details 43

3.3.1 QuartzNet 44

3.3.1.1 QuartzNetBxR-1D 44

3.3.1.2 QuartzNetBxR-3D 48

3.4 Numerical Benchmarks 49

3.4.1 The WISDM-HARB Dataset 49

3.4.2 Interleaved activity segmentation without overlaps 52

3.4.3 Numerical Evaluation 53

3.4.3.1 Sequence Error Rate 53

3.4.3.2 Numerical Results 53

CHAPTER 4 – SMART INTERACTION PLATFORM WITH SMART WATCH

4.1 Authentication On Common Computer Platforms 56

4.1.1 Windows Credential Providers 57
4.1.1.1 Architecture 58
4.1.1.2 Credential Providers Authentication Process 59
4.1.2 Linux Pluggable Authentication Modules 61
4.1.2.1 Architecture 62
4.1.2.2 PAM configuration 63
4.1.2.3 Authentication Process 67
4.1.3 Web-based Applications Authentication 68
4.2 Authentication System Design and Implementation 68
4.2.1 Authentication Techniques 69
4.2.2 System Design 70
4.2.3 Module Implementation: Custom Windows Credential Provider 72
4.2.3.1 Windows Credential Provider Interfaces Implementation 72
4.2.3.2 Authentication Process 74
4.2.4 Module Implementation: Linux PAM Module 76
4.2.5 Module Implementation: Google Chrome Extension 78
4.2.6 Module Implementation: Authentication server 81
4.2.7 Module Implementation: Database Design 82

CHAPTER 5 – CONCLUSION

5.1 Main results 84

5.2 Samples of Usage Scenarios 85

5.3 Future Work 86

References 88

List of Publications 110

LIST OF TABLES

Table 1.1	Common wearable devices with locations and applications	10
Table 3.1	QuartzNet5x3-1D convolution block hyperparameters	48
Table 3.2	WISDM dataset activities	50
Table 3.3	Interleaved segmentation evaluation result	54
Table 3.4	Fixed time-frame segmentation evaluation result	54

LIST OF FIGURES

Figure 1.1	Health wearable device classification	9
Figure 2.1	Layers of the Android sensor stack	18
Figure 2.2	The first attempt at soldering on the USB connector, without opening the back.	20
Figure 2.3	Opening the back of the Moto360v2 using a heat gun	21
Figure 2.4	The second attempt at soldering on the USB connector, with opened back	22
Figure 2.5	Communication and client-side and server-side	25
Figure 2.6	Android socket client application is installed on our Moto 360 watch	26
Figure 3.1	LeNet-5 – the original CNN reproduced in [1]	34
Figure 3.2	Normal Convolution	36
Figure 3.3	Depthwise Convolution	38
Figure 3.4	Pointwise Convolution	38
Figure 3.5	Time-channel Separable Convolution	39
Figure 3.6	CTC algorithm for speech recognition [2]	39
Figure 3.7	QuartzNet architecture adapted from [3]	42
Figure 4.1	Interactive Login Architecture	58
Figure 4.2	Credentials Processes in Windows Authentication	59
Figure 4.3	Components on a Windows Logon screen	60
Figure 4.4	PAM architecture and core components relationship	62

Figure 4.5 List of files in `pam.d/` folder on the author's Linux system 65

Figure 4.6 PAM configuration example: Authentication process 67

Figure 4.7 Web-based authentication techniques examples 69

Figure 4.8 Web-based authentication using local devices and email addresses 70

Figure 4.9 Authentication techniques 71

Figure 4.10 Future work: Authentication techniques 72

Figure 4.11 Authentication system architecture 73

Figure 4.12 Custom Credential Provider class diagram 74

Figure 4.13 Custom Credential Provider process 75

Figure 4.14 Custom PAM module process 77

Figure 4.15 Extension execution process 80

Figure 4.16 Database diagram 82

ABSTRACT

Despite computers are created to serve humans and come for their needs, interaction with computers nowadays is an obstruction to our societal life. Computers, being it your laptop or smartphone, require significant attention from us to operate. Every time you look down to text on your phone screen is a moment you disengage from reality. This is due to the lack of intuitive and implicit input capabilities for computer and such minuscule but repetitive behavior have turned people to prefer keyboards and screens over talking and interacting with the physical world. Over the past decade, there have been significant efforts in addressing this challenge via brave new interactive modalities that could weave seamlessly into our life such as smart glasses, smart rings, smart speakers, etc.

In this thesis, we set forth to extend this line of research for intuitive, effortless, and enjoyable computer interaction by employing a natural everyday-carry object – the smartwatch. We look for extending the capability of the smartwatch beyond what it already is, i.e., a timepiece and mostly a health monitor, and become a mean to facilitate an effortless human-computer away from mice and keyboards. We follow through, first by enhancing the capability of the smartwatch; then we propose a technique for activity recognition using inertial motion signals from the smartwatch; and finally, we implement a flexible authentication framework for multiple platforms authentication using wearable devices that are highly customizable.

CHAPTER 1
INTRODUCTION

In this chapter, we briefly introduce the overview about smart environments, together with smart interaction between users and computing systems. Then we present the motivation and main objectives of our thesis. Finally, we describe summarize the outline content of this thesis report.

1.1 Smart Environments

Exactly three decades ago, Mark Weiser published an important landmark paper "The Computer for the 21st Century" describing the future of computing in everyday life, in which each person is surrounded by hundreds of computers [4]. These computers are mobile; they know their location, and they communicate with each other creating a cyber-physical environment around the user. Computers are no longer mere passive terminals for the user to interact with cyberspace, but rather coherent partners that can be active or reactive with respect to the environment, as well as to the user's intent. Indeed, Weiser's prevision has proved to be somewhat of the truth as the reality we are experiencing now looks a lot like its manifestation.

Following Moore's Law on computing power [5], the development of application-specific integrated circuits (ASICs) has powered the success of embedded edge and cloud computing in recent years and brought about the cyber-physical environment that is now commonly referred to as the Internet-of-Things (IoT) or simply the smart environment. Each computer in the current smart environment is exponentially more powerful than the most powerful computer in 1991, the year by which Weiser's article was published. Yet, their sizes can shrink to the unimaginable size of watches. The embedded edge is the relatively compact

or mobile computers that interact directly with the user as peripherals of the cyber-physical environment [6]. The cloud, on the other hand, is the collective of high-performance mainframes whose computing power can flexibly expand at the cost of power consumption. The user, regardless of whether or not aware of the cloud existence, never actually encounters this part of the cyber-physical environment or needs to care for its details, hence the "cloud" abstraction. Such computing power allows the smart environment to be not only active but also proactive and reactive by perceiving the world through machine learning algorithms executed on both the edge and the cloud. These many components of the smart environment form a network of diverse devices capable of communicating with agility, thanks to the high-speed transmission technologies (e.g., optics fibers, 4G, 5G), and acts in harmony, thanks to efficient distributed consensus protocols (e.g., Paxos, federated learning), all the while conceding a significant mass of communication wirelessly as if no literal network exists.

Unfortunately, the invisible Internet-of-Things is the boundary before the resemblance between our reality and Weiser's vision breakdown. Because, as Weiser put it, "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." This condensed and powerful statement of the vision implies that means of interaction, instead of only links between such means, should seamlessly integrate with the user's natural behaviors to the extent that they are no longer noticeable. As for us now, having the invisible mesh of interconnected devices is as transparent as we can get. Instead of having the human-computer interaction weave cyberspace seamlessly into our lives and enhance our natural and social interaction, we are having advanced cyberspace fasten to and augment our physical world. This means we still have to intermittently switch between the two worlds every day through non-ignorable devices: dashboards, PCs, tablets, smartphones, etc. All of which are essentially rectangular terminals of various

sizes.

Regardless, there has been a large body of work in smart environments. Although works in smart environments and ubiquitous sensing – presented below – are closely related with overlaps, one can arguably consider smart environment research to be attempts focused on designing multi-component systems such that each component embodied a technique in ubiquitous computing or, more specifically, ubiquitous sensing. As such, state of the art in smart environments research spans from large-scale systems such as climate monitoring and smart agricultural systems [7] to domestic and healthcare interactive systems [8]. Within the scope of this work, we pay attention to the human-centric use cases of the field, particularly in domestic assistive living and personal cybersecurity.

Most smart environment systems share a common scheme of 6 general layers: the sensor and actuator layer, the communication layer, the information service layer, the management layer, the security layer, and the application layer [9]. Current commercial offerings such as Amazon Echo, Google Home, and Apple HomePod are often shipped as a ubiquitous sensing agent that can manage and interoperate with other devices and various services that take a focal point in enhancing the daily indoor experience with regards to entertainment, online shopping, control of heating and appliances, monitoring energy usage, and smart personal assistants [10]. Such systems allow the user to conveniently interact with home entertainment systems like TVs and other home appliances, including lightnings and indoor climate control simply using their voice. Another class of common smart assistive environment systems is elderly health monitoring systems, most of which the most common are elderly fall detection systems [11, 12]. Notable works are the CodeBlue project. These systems often provide specialized devices for fall detection, e.g., Apple Watch Series 4, iLife fall detection sensor, or Linksys Aware set of WiFi routers, and the integration for emergency services [12]. The emergency alarms can be triggered automatically

using ubiquitous sensing and human activity recognition methods with immediate notifications to other family members and emergency services. Although the effectiveness of these systems has yet to be fully reported due to the private nature of clinical data, Apple Watches instances of saving lives have made multiple headlines across intensive care units [13, 14, 14, 15]. This also demonstrates the willingness of user adaptation with respect to personalized wearable devices. Moreover, the absence of vision-based systems in the commercial landscape of these health monitoring systems has proven the importance of privacy pervasiveness and ubiquitousness in actual useful systems.

1.2 Human-Computer Interaction with Ubiquitous Sensing and Wearable Technology

Human-Computer Interaction (HCI) is a field of study focusing on how people interact with computers. HCI is a combination of the "human side" and the "computer side". The "human side" consists of psychology, design, human factors, and ergonomics. While the "computer side" is about technology [16]. The dogma of HCI is that computers should enhance our welfare instead of being hinders to our natural way of life. This demands computers to be interactive systems that have incrementally refined usability, which in turn, is encompassed by their safety, effectiveness, and usefulness for human development. More importantly, as any computer systems were created to serve humans and come for their needs, these systems must be designed and developed in the context of user-centric, with emphasis on their enjoyability and ease of use. To achieve such goals, ever since its establishment, HCI research has been investigating the factors that determine the way users interact with technology, as well as exploring brave new modalities of interaction. Such effort has brought about the undeniable success of the Graphical User Interface (GUI), and the "see, point, and click" interaction in making computers usable and approachable. Despite its de facto

status in ubiquitousness – designing GUI has grown into an integral industrial discipline of the modern world – they are unnatural and demanding. The instance we engage with the GUI is also when we disengage with the real world and tap into cyberspace. This undoubtedly does not conform with our natural behavior and obstructs our interpersonal communication. This de facto "see, point, and click" is also the root of "desk job" physiological problems such as carpal tunnel syndrome, cervical spondylosis, lower back pain, etc.,

The pioneering vision depicted in [4] has inspired a new branch that seeks to reinstate the de facto of human-computer interaction with modalities that extend and weave into physical spaces freeing users from rectangular terminals. In order words, the goal of this research branch, coined ubiquitous computing or ubicomp, is to design and implement interactive modalities that conform with our natural physiological and social conditions, making our everyday objects an easy-to-access, or ubiquitous, interactive modality [6, 17]. Such modalities rely on robust and reliable sensing techniques for user intents, activities, events, and context in physical space. Noteworthy body of research in ubiquitous sensing includes a voice user interface, gesture recognition, gaze tracking, and biosignal monitoring, etc., all of which consider different contexts (e.g., indoor, ambient, wearable, wide-area, etc.,) and sensing hardware (e.g., microphones, camera, inertial measurement units, WiFi signals, etc.), and all of which are linked to in state-of-the-art directions in computer science such as mobile, machine learning, deep learning, speech and signal processing, etc. One can say ubicomp has blent itself entirely to the modern landscape of computer science and engineering, yet it is still subtle with respect to the research community [17]. This is due to the fact that traditional ubicomp research demonstrates algorithmic novelties using complex and interdisciplinary proof-of-concept, while most modern research only focuses on a niche subproblem without an integrated demonstration. Yet, not much ubicomp research is done with the framework design mindset, limit-

ing the reusability of such proof-of-concept prototypes and reduce many novel works to be chasing the numerical benchmarks. This is a critical final barrier for ubicomp as a research community, of which the responsibility lies our generation of researchers.

1.2.1 Ubiquitous Sensing

Although computer vision is being the most popular research direction as of 2021, with IEEE/CVF Conference on Computer Vision and Pattern Recognition ranked as the 4th most cited outlet across all scientific venues by Google Scholar Metrics [18], there are limited vision-based assistive living systems due to concerns for privacy intrusion [11, 10]. In commercial offerings such as Amazon Echo, Google Home, and Apple's HomePod, speech recognition, natural language processing, and voice user interface play an important role as it provides an intuitive and less pervasive interactive modals. Expensive SOTA research such as BERT and variants of pre-trained language models [19, 20, 21, 22] and wav2vec 2.0 pretrained speech representation [23, 24, 25, 26] demonstrate an emphasis of corporates in developing such seamless user experience that can easily become part of users' lives. However, the voice modality is not free of limitations. An obvious limitation is that they do not work in quiet environments. Besides, there are performance deficits in multi-occupant environment [27], wide area scenarios, e.g., across the room, biases in gender, age, and accent [28, 29]. Critically, there are privacy concerns for these voice-controlled systems as they require users to agree with the always-listening and data collecting agreements in order the achieve the best performance and convenience [30, 31]. Although the always-listening is less pervasive than always-watching conditions of vision-based systems, audio is still a human-comprehensible and rich modality of information that entails serious security breaches.

More recent and advanced approaches consider low-level and unconventional

information modalities such as WiFi, radio reflection signals, wearable motion sensor signals, or biosignals. WiFi-based human activity recognition is an attractive modality as it is completely "invisible" to the user. WiFi-based sensing draws from a core principle of ubicomp, which is "your noise is my signal". The modality relies on the fluctuation when propagating the channel state information (CSI) in a multiple-input multiple-output (MIMO) WiFi configuration [32, 33]. The interference of the signal is exploited to extract environmental landmarks and even human gestures and activities. Another related wireless sensing modality that operates on the same principle is radio reflection, which uses dedicated radio-frequency transceivers instead of commodity routers. Applications using this modality include motion detection, gesture recognition, identification/authentication, anonymous indoor localization. Notably, WiFi-based sensing is an effective healthcare monitoring as it has gave encouraging results in fall detection [32], biosignal monitoring [34], and emotion recognition [35]. This modality offers a truly ubiquitous experience while being minimally pervasive as it does not expose human-comprehensible data compare to audio and visual data. As a prior to this thesis, we have also investigated this modality for fall detection earlier since our first year with a publication by the end of our second-year [36]. As the wearable modality is a focal point of this work, we dedicate a more more comprehensible review for it in the subsection below.

1.2.2 Wearable Computing

In 1966, Edward O. Thorp et al. announced a system for roulette prediction that the first wearable computer. Thorp devised the idea of the system in 1955 when a question was in his mind about the possibility of beating the roulette wheel, then Claude Shannon joined the project in 1960. They kept the secret of the project until the 1966 announcement [37]. That the origins of wearable technology, a technology that is currently leading the Internet of Things (IoT)

field. Wearable technology uses different kinds of wearable devices that have the capability for receiving, storing, analyzing, and transmitting data for various purposes and applications by using many types of embedded sensors [38].

The term wearable devices refer to such devices that can be worn, attached, or mounted on a human body or animal or placed in clothing [38, 39]. Wearable devices can be smart watches , eye wears, body straps, headsets, foot-worn devices, and jewelry [39]. These devices can be electronic devices containing sensors to collect various types of data [38, 39], they may further include processor and storage to perform calculating task by itself [38]. Electronic wearable devices can communicate with each other and communicate with other computing devices via different protocols to perform data fusion algorithms, cloud computing or big data [38, 39]. Ometov et al. categorize wearable devices using many factors: application or functionality types, device types, or on-body locations [40].

Wearable technology innovations are affected by the robust development of mobile computing, and the wearable technology market is growing together with the growth of smartphone users [39]. In 2013, the year of smart watches when the technology was sufficient to make these devices cheaper and providing an experience of wearing a regular watch. Product-based tech companies have put much effort into releasing a lot of smart watch models with new technology that is a combination of engineering, design, and fashion [39]. In contrast, the prices of these devices are getting lower and lower. Nowadays, smart watches are not only fashionable but also offer a lot of utilities such as notifications, health monitoring, fitness tracking, and GPS. These features can be compatible with smartphone apps to increase usability.

The wearable technology development is encouraged by the rapid development in sensors, material sciences, telecommunication and microelectronics [39] that introduce new opportunities for wearable technology research topics and applica-

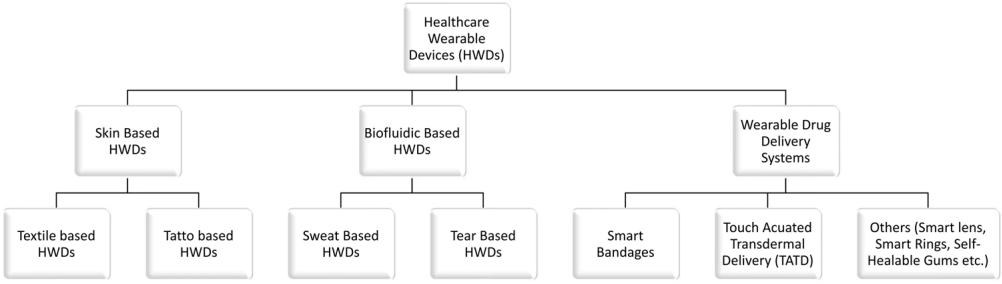


Figure reproduced from [38].

Figure 1.1: Health wearable device classification

tions in human-body interaction using wearable devices [39]. Because the variety of wearable sensors, many types of data can be collected [39]. The collected data from wearable devices can be biometric and health information [41], motion information, temperature and humidity data, visual and audio data [38]. The diversity of wearable sensors and the development of related fields help wearable technology can be applied to many different fields such as healthcare and health treatment [38], personalized drugs delivery [42], life logging [43], and applications lists in Zhang et al. 2020 [39] such as motion capture [44], elders monitoring [41], privacy [39], education [45], law enforcement [46], activity monitoring [47] etc.

As a result, many people investigate the feasibility of supporting tasks such as freehand controlling, authentication, lifelog data collecting, and even more. In 2018, Al-Naffakh et al. [49] showed the potential of a smart watch as a user-friendly individual recognizer. Hence, they collected data from 60 users for multiple days and applied a segment-based approach dividing gyroscope and acceleration data to perform authentication. smart watches are also useful to collect sensitive lifelog data. However, collecting data using a smart watch faces serious privacy issues. It follows that Kim et al. 2019 [43] proposed and evaluated a framework to collect health lifelog data from a smart watch with the collected data privacy protection by leveraging local differential privacy. Another effort to prove the feasibility for supporting freehand 3D interaction for a smartwatch, K

Table 1.1: Common wearable devices with locations and applications

Types	**Location**	**Applications**
Helmet	Head	GPS tracking, camera, microphone, inbuit earphone
VR headset	Head	Virtual environment interaction
Glasses	Head: eye	Information display, interact with mobile devices
Earphones	Head, ear	Fitness tracking, headphone
Throat Tatto	Neck	Inbuit microphone
Sports clothing	Upper body, Under body or Full body	Motion capture, step counting, health information collection, GPS trackiung
Gloves	Hand	Health information collection, sign language recognition, hand motion capture
Watch	Hand	Health information collection, display information, mobile application, GPS tracking, activity tracking
Socks and Shoes	Foot	GPS tracking, fitness tracking, activity tracking, foot diseases monitoring, step counting and health information collection
Implants	Under skin	RFID chips for authentication
Jewellery	Head, hand, foot, neck, ear	Use for many types of tracking and monitoring

Information on this table based on [48] with additional information

Pietroszek et al. 2017 [50] introduced a technique called Watchcasting that supports target selection and translation provided by mapping z-coordinate position to forearm rotation and proved the performance of the technique by demonstration 3D interaction in a large display.

Despite having only become mainstream by the mid-2010s, the niche research of wrist wearable motion-sensing dates back to the early 2010s with Chernbumroong et al. attempt to recognize five activities sitting, standing, lying, walking, and running using the accelerometer on the Ez-4310 Chronos hardware platform using artificial neural network and decision tree [51]. Scholl and van Laerhoven achieve encouraging results in smoking recognition using a Gaussian classifier and their self-developed Hedgehog sensor platform [52]. da Silva and Galeazzo return to Ez-4310 Chronos' accelerometer with their experiments on recognizing daily living activities, including stair activities and working on computers, with multi-layer perceptron (MLP), k-nearest neighbors (kNN), and support vector machine (SVM) [53]. From 2013 to 2017, researcher start to use smartwatch sensors as the enabling platform but only consider statistical recognizers and sets of activities as before [54, 55, 56, 57]. Notably, public datasets are being release during this time with the popular WISDM dataset [58] with daily living activities recorded on smartphones and smartwatches. Stisen et al. [59] release an analysis of affecting factors for wearable-based activity recognition with different hardware models and environment contexts. Vaizman et al. raise the research community awareness on the lack of crowdsourced in-the-wild data that can actually be useful in practice through a series of publications [60, 61, 62]. More recently, many works are done to investigate the application of deep learning methods in human activity recognition and achieve impressive results [63, 64, 65, 66, 67, 68, 69, 70, 71].

Wearable devices as a truly immersive interactive modality only start to come into play around 2015, with EM-Sense by Laput et al. exploit electromagnetic

wave interference as environmental signature [72]. Later, by overclocking the smart watch accelerometer, ViBand – an overclocked LG G Watch with gesture recognizers – demonstrate the capabilities of smartwatch on include being an interactive modality and being a close-range data transmitter with many further downstream use cases [73]. The overclocked LG G Watch is later improved to recognized a wider range of human gesture and activity at different granularities by using deep learning [74].

Although wearable technology has many useful and important applications, it also has many challenges and limitations waiting to be solved. A list of technical limitations of wearable technology, we reused from Qaim el. al [39]:

- Transmission overheads
- Wireless technology-related issues
- Inefficient routing
- Security-related aspects
- Processing limitations
- Storage limitations
- Inefficient use of energy consuming modules
- Battery limitations

Limitations of wearable technology mostly relate to design and development. Saleem et al. 2017 [48] found highly important challenges are related to social and security that wearable technology are facing:

- Legislation
- Technical Compatibility
- Privacy
- Third party access

- Public by default
- Health information protection laws

Wearable technology is developing very fast with a lot of applications and a lot of research topics to explore. Zhang et al. [39] claimed that wearable systems is a crossroads between engineering, design, and fashion. In future, many wearable devices will be introduced, they will become a part of our daily life. Such devices will create a smart ecosystem, assistant for our daily life helping us in healthcare, security, productivity and other daily activities.

1.3 Motivation

Following the necessity of proposing and implementing new modality for interaction in order to enhance the way users interact computer systems. In this thesis, we aim to investigate the feasibility of enjoyable and comfortable interacting techniques. Particularly, we intend to use smartwatch, an unobtrusive everyday-carry object, coupled with human activity and gesture recognition as the modality for human-computer interaction. We believe the power of modern wearable computing and the ubiquitous nature of a watch can yield a truly gratifying interactive modality. Wu et al. study in 2016 encourages our work by showing that enjoyment is a factor affects users intent to use smart watches [75].

We aim to achieve this through investigating methods in exploiting inertial motion signals available on off-the-shelf commodity smart watches , e.g., accelerometers and gyroscopes, for understanding complex human gestures and activities. We also aim to consider the added gratifying value of this modality, even in its most constrained form, when used to augmenting daily human-computer interactions beyond "see, point, and click". Hence, normal computer using scenarios, such as user login, are investigated with regard to their feasibility in coupling with smartwatch-gesture interactive modality.

Finally, to address the final challenge of ubicomp as called out by Abowd in making ubicomp systems accessible to a wider community of developers [17], we take into account the customisabilty and modular approach in our design such that anyone with the will can modify, upgrade and extend features for the system. This, we believe, should not only bring in incremental development values to the developer community, but also add in personalized values for users as a whole.

1.4 Objective

In this section, we discuss about our objectives, tasks to be done, and our contributions in the scope of this thesis.

1.4.1 Objectives

In this thesis, our objective is to propose a technique for interaction between to users and computer devices (desktop, laptop, etc) with smart watches . Therefore, we study the method to enhance off-the-shelf smart watches for achieving fully potential of inertial sensors. We further aim to understand different gestures based on inertial signal sequences from wearable devices; Finally, we design and implement a smart and flexible system for computer environments (Linux, Windows, Web-based) to perform user authentication.

1.4.2 Tasks

In order to realize our objectives, the list of tasks we have done in our thesis:

1. Explore and understand method to modify off-the-shelf smart watches available on the market to increase sample rate for inertial sensors.
2. Survey technical datasheets related to smartwatch hardwares.
3. Perform hardware interference for our Moto 360 smartwatch.
4. Investigate and comprehend Linux kernels compilation process to build

custom Android Linux kernel with custom inertial sensors drivers.

5. Investigate and comprehend techniques to write and modify Linux drivers. This task includes learning related concepts and components to Linux drivers in general and in Android devices such as Board files, device tree files, etc.
6. Investigate and comprehend signal processing methods for smartwatch accelerometer.
7. Survey state-of-the-art methods in smartwatch-based human activity and gesture recognition.
8. Survey state-of-the-art methods in learning-based sequential signal processing.
9. Design human activity and gesture recognizers.
10. Implement human activity and gesture recognizers.
11. Evaluate proposed human activity and gesture recognizers.
12. Implement a client app for smart watches and a server on computers to transmit inertial sensor data from smart watches to personal computers.
13. Investigate and comprehend authentication systems architecture and process on Windows such as Windows Credential Provider Model and graphical identification and authentication (GINA).
14. Implement a custom credential provider for Windows logon to use smart watch to authenticate.
15. Investigate and comprehend Pluggable Authentication Modules (PAM) architecture and process on Linux system.
16. Implement a PAM module to support authentication using smartwatch.
17. Investigate and comprehend common authentication techniques for end-users and authentication systems that are used on web-based platforms.
18. Investigate and comprehend Google Extension development and implement an extesion for autologin with our authentication system.

19. Design and propose usage scenarios.
20. Build interative demo.
21. Write the thesis.

1.4.3 Contributions

In this thesis, our contributions can be summarize as follow: First, we propose a solution for interaction by transform existing smartwatch smarter, so they are able to be used as smart input devices. Second, we propose an approach to perform activity recognition using inertial signal sequence that method is inspired from speech recognition domain. Finally, we build a simple flexible authentication system for multiple platforms authentication using wearable devices that easy to customize, upgrade and integrate new modules or platforms.

1.5 Outline

The rest of the thesis is organized as follows. In **Chapter 1**, we give a literature review of state-of-the-art related work in smart environments, ubiquitous sensing, and wearable technology. In **Chapter 2**, we introduce smart watch models we use in this thesis, then we present step-by-step smart watch enhancement methods for the devices. In this chapter, an simple socket server-client implementation is also proposed. The following chapter - **Chapter 3** a signal processing approach for inertial sensor signal understanding is discussed. In this chapter, we evaluate the system using numerical benchmark. An technical details of authentication system on popular platforms is proposed in **Chapter 4**. Then, we introduce technical details of our implementation of authentication system using smart watches. Finally, we discuss the evaluation results and an outlook for further improvements and research opportunities in **Chapter 5**, before concluding our work in the same chapter. Appendices related to our previous work toward this thesis attached after the conclusion.

CHAPTER 2

FEATURE ENHANCEMENT FOR SMART ANDROID DEVICES WITH INERTIAL SENSORS

In this chapter, we discuss the characteristics of off-the-shelf Android Wear devices including Android sensor stack architecture and inefficient power consumption limitation, then we introduce the possibility of receiving higher sample rate from the devices' sensors by hardware and firmware interference. Finally, we introduce an implementation for receiving sensor signal transmission from Android smartwatches.

2.1 Hardware and Firmware Interference

We use off-the-shelf Android Wear devices to collect data. Due to the limitation of smart watches that are inefficient power consumption to achieve better battery saving, so these smart watches sensors never run at its highest performance. To get a higher sample rate on these devices, we use custom Linux kernels for these devices to unlock a higher sample rate. In a typical Android device, sensor management is performed to help an application in Android OS can access data sensors, or manage sensors power consumption, the Android sensor stack show us how an Application which on the highest layer of an Android device communicate with the lowest layer with different types of sensors.

When an application request a highest sample rate for data of a sensor, then the sensor itself, the sensor drivers and related drivers, and the sensor hub are factors that decide the sample rate, these layers are 3 lowest layers in Figure 2.1. The sensor hub is a optional component that may support data batching, power saving, controlling and running fusion algorithm [76]. A device with sensor hub including more functionality, however, harder to be modified.

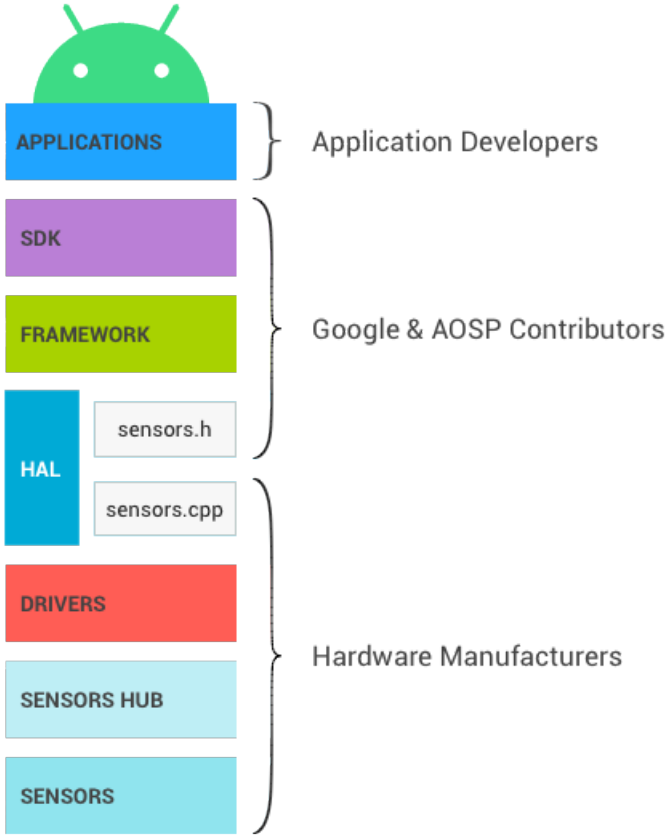


Figure is public in the Android sensor stack document[76].

Figure 2.1: Layers of the Android sensor stack

2.1.1 LG G Watch W100

LG G Watch W100 is an Android Wear-based smart watch released in June 25, 2014 by LG Electronics. The smart watch uses an InvenSense 9-Axis MPU including an accelerometer, a gyroscope and a compass sensors. The device's accelerometer normally runs at 100Hz; however, the accelerometer can reach 4000Hz sample rate that sample rate can be achieved by changing the kernel that uses another accelerometer driver supports higher sample rate. A modified version of the Android Dory kernel called FASTACCEL including kernel source and plug-and-play boot image are available on github that is released in 2016 by Laput et al [73].

Where flashing stores the boot image in the watch storage and booting load the kernel and run it in the watch without storing anything. The kernel boot image can be booted or flashed into a LG G Watch model W100 device after unlock bootloader using fastboot tool. Using Android Debug Bridge and Fast-Boot toolchain running on a bashshell environment, the custom kernel flashing can be done follow these steps:

- Install Android 5.0.1 (LWX48P) on the LG G Watch, it can be done by updating the android or flashing the system
- Using Android Debug Bridge (ADB) to open bootloader. In order to use ADB we have to enable developer features in the watch

```
adb reboot bootloader
```

- Using fastboot to unlock the bootloader

```
fastboot oem unlock
```

- Using fastboot to flash the boot image

```
fastboot flash boot <path-to-boot-image>.img
```

- If we want to boot the image instead of flashing it - the image will be booted without being stored in the watch storage.

```
fastboot boot <path-to-boot-image>.img
```

2.1.2 Motorola 360 (2nd generation)

In September, 2015, the second generation of Moto 360 smart watch (Moto360v2) is released after LG G Watch W100 by one year. The maximum sample rate of the device, however, is just 50Hz – half of LG G Watch model W100. This makes us spending much effort of this thesis on learning Linux kernel device

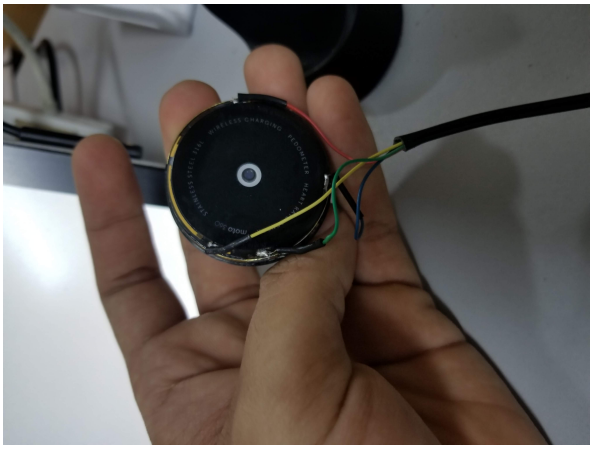


Figure 2.2: The first attempt at soldering on the USB connector, without opening the back.

driver programming and analyzing the kernel source of this device in order to create a kernel version that can reach a higher value of maximum sample rate.

Unlike the LG G Watch model W100, the Moto 360v2 does not provide an official wired connector to personal computesr, which is crucial to customize low-level software components. To adress this challenge, we implement a makeshift connector by soldering a USB cable to exposing copper strips corresponds to connector pins. At first, only remove the case and rim of the watch to expose these copper strips and the first version of our connector is made as in Figure 2.2. Unfortunately, the copper strips are highly delicate while we cannot provide additional protective support to the solder as it effect the dimension of the watch and interfere with the wireless charging – the only way the charge the watch. Thus, the copper strips eventually got ripped off the watch by accident, which force us to reinvent to makeshift connector. In the second attempt, we open the back of the watch using a heat gun and solder the USB cable to the remnants of the copper strips (Figure 2.3). This time we are able to provide additional protection to the solder as the dimension of the watch has been reduced after removing the glass back. The charging coil is also exposed, letting us to charge

it easier (Figure 2.4). When we successfully solder the watch, we can connect it to a PC and perform low-level interference.

Figure 2.3: Opening the back of the Moto360v2 using a heat gun

When we flash the fastaccel kernel into the Moto 360, unfortunately, the watch get stuck at the boot state, creating a boot loop. We subsequently find out that the Linux kernel and the hardware are not compatible. The solution is creating a custom kernel using the kernel source of Moto360v2. Fortunately, the Linux kernel source for the Moto360v2 – codename smelt– is published on Github by Motorola themselves [77].

After boot image that we create using the source code on Github is successfully flashed into our device, we are then left with modifying the kernel to overclock accelerometer sampling rate.

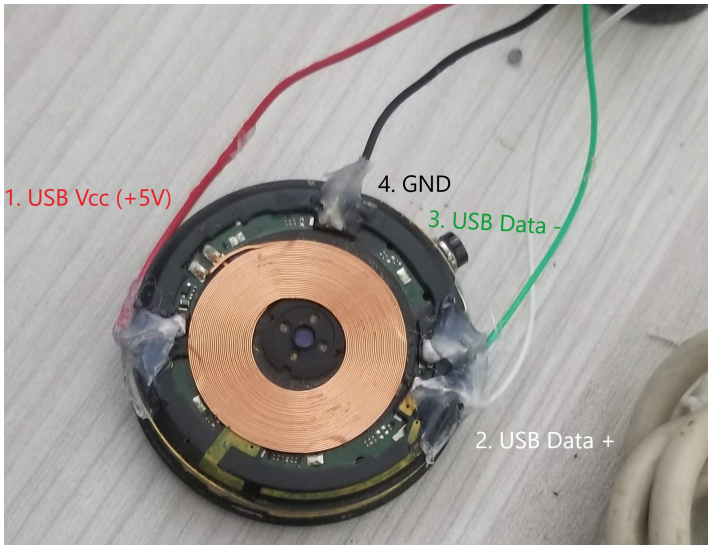


Figure 2.4: The second attempt at soldering on the USB connector, with opened back

2.1.2.1 Build a kernel

In the following text we introduce techniques to compile an Android Linux kernel for Moto360v2 other devices with the ARM architecture.

To build an ARM kernel using x86_64 intel's CPU architecture personal computer, cross compile is required. We use Linaro toolchain, specifically `GCC-Linaro-4.8-2016.06-x86\_64\_arm-linux-gnueabi` version for cross-compiling in order the support the antique kernel version 3.10 of the Moto360v2.
The first step is to provide the target architecture for the cross complier we are using. Inside Makefile of the Linux kernel, the developers provide 2 variables named `CROSS_COMPILE` for cross compile toolchain prefix path specifying and ARCH variable to specifying the architecture. So we can change the value inside the Makefile or export 2 corresponding environment variables. The prefix in this case is `arm-linux-gnueabi-` corresponding to the version we are using:

```
export CROSS_COMPILE=<path-to-toolchain>/bin/<prefix>
export ARCH=arm
```

Next, we need to create the configuration file for the build. The content of the file provide modules, subsystem, and drivers which will be built with the kernel. Inside the `arch/arm/configs` directory is a list of configurations for some devices, the one we need to pay attention to is `smelt_config` that is the default configuration by Motorola. To apply the config to the build we run:

```
make smelt_config
```

Then we can perform build step, with `number-of-jobs` is the number of processes we allow for multiprocessing at once.

```
make -j<number-of-jobs>
```

Once we successfully built the kernel. We receive 3 files placing in `arch/arm/boot`:

- `Image`: a generic kernel image
- `zImage`: a self-extracting compressed image
- `zImage-dtb`: a compressed image with device tree blob

We need `zImage-dtb` kernel version to be booted into our Moto360v2 because this image provides hardware support configuration. Unfortunately, our device can only boot these kernel but cannot flash them into the boot sector of the device. To perform flashing we have to create a boot image, which is an image that wraps the ramdisk and the image we have built. We retrieve the ramdisk by unpacking the stock boot image of the Moto360v2 with the support of `unpack_bootimg` tool available in Android Open Source Project repository. The tool can extract the ramdisk and the `zImage-dtb` of the Moto360v2 stock boot image and provide information such as offset values, command line, and the page size. Once we have the ramdisk and these information, the `mkbootimg` tool is used to create the boot image which can be flashed into the device. The following script can be used:

```
python mkbootimg.py \
--kernel <path-to>/zImage-dtb  \
--ramdisk <path-to>/ramdisk \
--cmdline '<commandlines are given>' \
--base 0x00000000 \
--pagesize 2048 \
--kernel_offset 0x00008000 \
--ramdisk_offset 0x02000000 \
--tags_offset 0x01e00000 \
--second_offset 0x00f00000 \
-o output.img
```

2.1.2.2 Smelt kernel source: M4 sensorhub

All sensors on the Moto360v2 are connected to and regulated by and ARM Cortex-M4 Microprocessor. This microprocessor is a co-procesor of the watch beside the main CPU and is referred to as the sensor hub.

The core driver of M4 processor is a simple multi-function devices driver (MFD) which controls contain client drivers for accelerometer, gyroscope, batching, compass, heart rate, step-counter, etc. The goal of this subsystem to help the watch to use power more efficient, which also means that it threshold the sampling rate that the sensor can operate.

2.2 Signal Collecting

Sensor data transmission is an essential task to facilitate the smart watch as an input device. We also need transmit data from sensor in smart watch and smart phone into our personal computers for various purposes in the developing process, including data collection and offsetting computational intensive task to other devices connected to the watch.

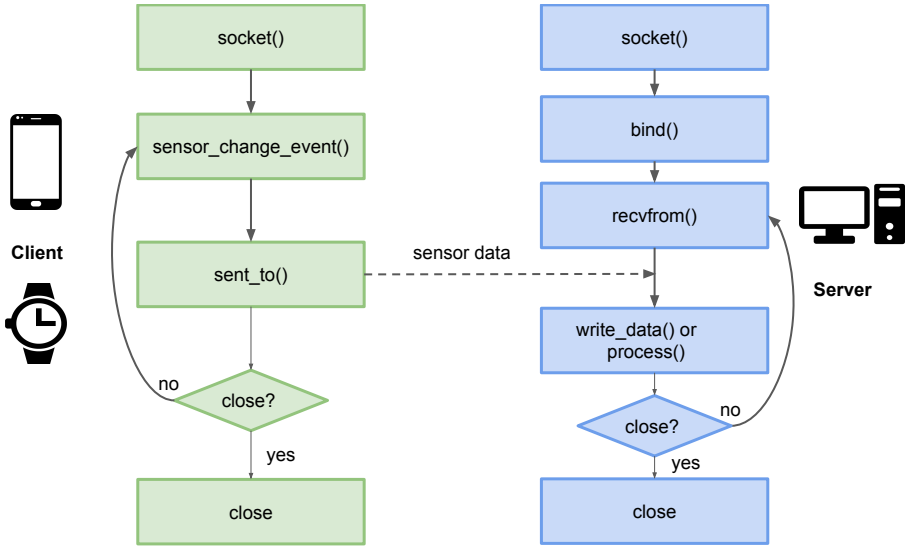
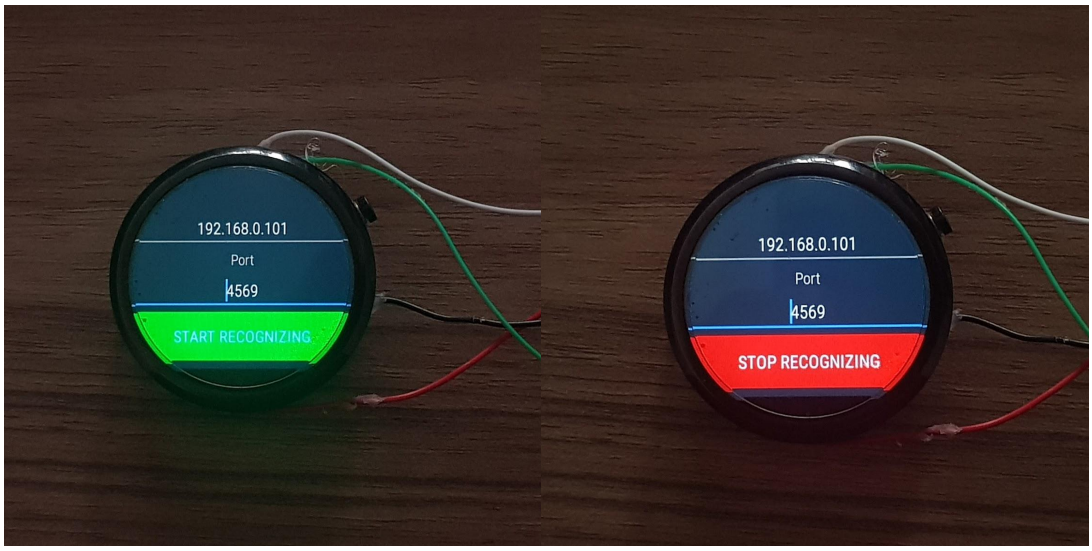


Figure 2.5: Communication and client-side and server-side

Therefore, we implement a client/server application to stream recorded data from the sensor into our personal computer. The implementation requires socket programming and Android development knowledge. The client side application is an Android application, while the server side is a Python application. The client transmits the data into the server via a WiFi network and communicate with the server using User Datagram Protocol (UDP). A diagram of the client-server communication is presented in Figure 2.5. The UDP is chosen to simplify the current developing process so that we can bypass three-way handshake and develop a much simpler data collecting application.

In order to reduce coding effort, we inherit an Android wear gesture project that is published on github by Ziwei Zhu [78]. We reuse message structure and GUI implementation in the project, but the UDP message sender is re-implementation to meet our requirement. Essential details to our implementation are discussed further in the following subsections.



The application has the "START RECOGNIZING" button to run the socket client. While the process is running, it can be stopped by pressed on the "STOP RECOGNIZING" button.

Figure 2.6: Android socket client application is installed on our Moto 360 watch

2.2.1 Client

A watch with the socket client installed is depicted in Figure 2.6. The socket client is an Android Wear Application that is implemented using AndroidSDK, compose of sensor event listener and an UDP message sender.

2.2.1.1 UDPSender

The `UdpSender` implementation is using classes in `java.net` library.

The message sender is a simple socket client that byte arrays over the the network. The `UdpSender` class contains a string attribute with a value being the server's IP address and a `SentTo()` method which send a byte array to the corresponding server IP address if the method is called.

```
1 public class UdpSender {
```

```
  final String host = "<server ip>";
    public void SendTo(byte[] msgBytes) {...}
  ...
}
```

The `SentTo()` method creates and starts a new thread to send the collected data as a byte array to the server. Under the created thread, we create a `DatagramSocket` object and try to enable socket broadcast if the socket broadcast is not enabled. When the socket broadcast is ready, we make new a object named `packet` which is an instance of the `DatagramPacket` class to create a new packet containing the IP address of the server, the port of the server, the byte array data, and then length of the byte array. Finally, we send the packet and close the connection. We make use of `try-catch` statement around the code to handle unexcepted exception.

```
public void SendTo(byte[] msgBytes) {
    final byte[] buf = msgBytes;
    //Start new thread with a function for datagram sending
    new Thread(new Runnable() {
        public void run() {
            try {
            // An InetAdress object is initialized
            // to represent an Internet Protocol(IP) address
                InetAddress serverAddress =
                    InetAddress.getByName(host);
                DatagramSocket socket = new DatagramSocket();
                if (!socket.getBroadcast())
                    socket.setBroadcast(true);
            // Create a packet object contains bytes array
            // and specify IP-port destination
                DatagramPacket packet =
```

```
                    new DatagramPacket(buf, buf.length,
                                        serverAddress, 4569);
            // Send packet and close the socket
                socket.send(packet);
                socket.close();
            }
            catch(UnknownHostException e){
            //exception handling
        }}}).start();
}
```

2.2.1.2 Sensor Event Listener

The purpose of the application is to get data from sensor and transmit them into server using the `UdpSender`.

The main activity is an implementation of abstract class `SensorEventListener` and an extended class of the `Activity` class in AndroidSDK API. We override the `onSensorChanged()` method from `SensorEventListener`. Inside the `onSensorChanged()` method, there are 3 `if`-statements. The first `if`-statement is using to verify the sensor's reliability. The second and the last `if`-statement is use to guarantee the sensor is an accelerometer. In the second `if`-statement, the sensor data is assigned to data node and inside the last `if`-statement block, the packet number and timestamp are also added to `dataNode` before sending the `dataNode` structure in JSON format using `UdPSender`.

```
@Override
public void onSensorChanged(SensorEvent event) {
    // If accStatus is false
    if(!accStatus) {
        // Create new data node structure
```

```
6           dataNode = new DataNode();
7       }
8       // verify the sensor event is from an accelerometer
9       if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
10          // Update accStatus for next step
11          accStatus = true;
12          // assign event data to data structure
13          dataNode.setACC(event);
14      }
15      // Verify the sensor is an accelerometer
16      if (accStatus) {
17          accStatus = false;
18          // Get current timestamp and assign to data structure
19          long currentTime =  System.currentTimeMillis();
20          dataNode.setTimeStamp(currentTime);
21          // Set data node number
22          dataNode.setPktNum(++pktNum);
23          // Convert dataNode to JavaScript Object Notion String
24          String json = new Gson().toJson(dataNode);
25          // Use UdpSender to send converted byte array
26          sender.SendTo(json.getBytes());
27        }
28  }
```

2.2.2 Server

The server is a Python socket application that initializes the socket connection and receives UDP packets sent by the client. By using Python to create a socket server, we can easily integrate other Python packages for additional functionalities such as data processing on-demand. The structure of the main process is simple. The first 2 lines are declarations for the IP address and the socket

`PORT` number of the server. In the next line, the `socket.socket()` method use to create mew socket takes 2 parameters: the first parameter is the address family and the second one is the socket type. In this case, the socket is created as an UDP socket with `AF_INET` address family (IPv4). A while loop is placed after the socket binding step to listen to sensor data from the client. When sensor data is received, a data processing function or data storing function can be called. The following lines are our server implementation in Python:

```
IP = <server-ip>
PORT = <port-number>
# Create socket use Address Family type AF_INET (IPv4)
# Specify the socket is datagram socket with SOCK_DGRAM value
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
# Bind the socket
sock.bind((IP, PORT))
# Open a data listen loop
while True:
    # receive data using a buffer with size equals to 1024
   bytes
  data, addr = sock.recvfrom(1024)
  # After data is received
    # data can be processed or written into disk
```

CHAPTER 3

INERTIAL SIGNAL SEQUENCE UNDERSTANDING FOR SMART INTERACTION

In this chapter, we present our proposed method for analyzing inertial signal sequences to understand users' gestures. Inspired by the processing model for speech recognition, we propose our method to work on frequency domain. The proposed method is capable of both recognizing a single gesture from a frame of sensor's signals, and recognizing a sequence of gestures. We present the experimental results on WISDM-HARB dataset in the last part of this chapter.

3.1 Signal Processing

When considering the signal used for gesture and human activity recognition, the literature have consistently reported that the accelerometer outperform the gyroscope on smartwatches in term of stability, and hence often yield better performance [58, 79, 60, 80, 59]. Thus, we only focus on developing the processing pipeline for accelerometer signals. Despite most of previous works only consider frame-based recognition of human activities, i.e., casting the problem in to a classification of sensor reading segments, we argue that such approaches constraint real-time interactive use case. In fact, it has been acknowledged that the frame-based approach can only work for limited simple activities as the frames does not consider the variance in time of different activities, as well as different phases of activities. Hence, this approach is not able to recognize compound dynamic activities, e.g., sports [81], or similar activities with common phases of action, e.g., eating activites [79, 58]. Therefore, we aims to develop a more general solution using the sequence-to-sequence approach, which is detailed below.

3.1.1 Heterogeneities and time-domain vs. frequency domain features

It has been shown in [59] that (1), the practical sampling rate of smartwatches varies in real-time due to different CPU workloads, and (2), frequency-domain features are more resilient to sampling rate variation compare to time-domain features. As we aim for a robust real-time algorithm, it is reasonable for us to opt for preprocessing and training our learning-based algorithm using frequency-domain features.

However, unlike previous work, which only focus on static time-steps classification, we aim for a dynamic and continuous activity recognition, which means the data will unavoidably be processed in a time-based manner. After many considerations, we decided to employ sequence-to-sequence models with input features preprocessed to be in time-frequency domain, i.e., time-based frames with localization of frequency-domain features.

3.1.2 Discrete Fourier Transform

The Fourier Transform is a change of basis transform that describes a continuous function in terms of a Fourier series defined in terms of periodic functions, which is an infinite sum of cosines and sines of increasing frequency. Using Euler's formula, the Fourier transform of a continuous real-valued function $x(t) \in \mathbb{C}, t \in (-\infty, \infty)$ can be written as

$$X(\omega) \triangleq \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt$$

In practice, however, we are rarely able to access continuous functions, thus the above form of the Fourier function is not particularly useful. Instead, what we actually deal with are series of sensor readings over timesteps, which are regarded as discrete samples of a continuous-time function that describe the underlined

phenomena. Thus, in order to make the Fourier Transform useful, we need the discretized version of it, which is defined using the Riemann sum over the length of the signal sample instead of infinite integral

$$X(\omega_k) \triangleq \sum_{n=0}^{N^-1} x(t_n) e^{-j\omega_k t_n}$$

where t_n is the nth sampling instance and N is the number of samples.

In practice, we almost always use the Cooley-Tukey Fast Fourier transform (FFT) algorithm [82], which achieves the time complexity of $O(N \log N)$ instead of $O(N^2)$ for the exact DFT. Thus, the terms FFT and DFT are often used interchangeably.

3.1.3 Short-time Fourier Transform

The short-time Fourier transform (STFT), also known as the Gabor transform [83, 84], computes a windowed FFT in a moving window. This enables the localization of frequency content in time, resulting in the spectrogram, which is a plot of frequency versus time.

In our application of accelerometer signal processing, each raw sensor readings is a 3-tuple of real values (x, y, z). A general segment of data can consist of multiple readings, which can be preprocessed using the STFT. For each sample, we compute the power spectrum using the FFT with a Hanning window instead of a Gaussian to minimize spectral banding. To make the frequency features robust across initial conditions, each of the FFT bin is normalized by dividing by the frame length to remove the DC component.

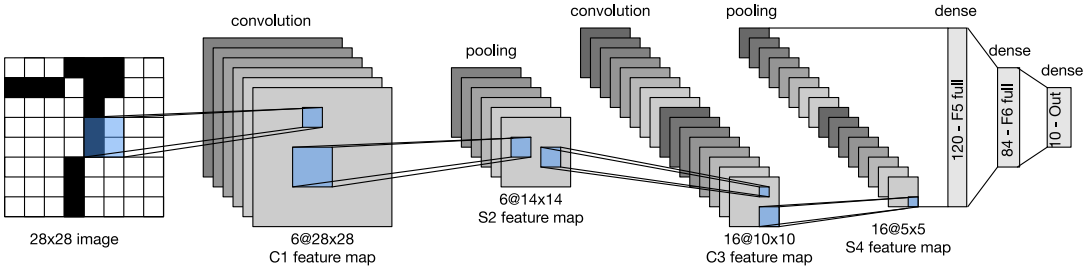


Figure 3.1: LeNet-5 – the original CNN reproduced in [1]

3.2 Learning-based Signal Processing

3.2.1 Foundational Methods

3.2.1.1 Convolutional Neural Network

The Convolutional Neural Networks (CNN) is arguably one of the most powerful ideas in modern computer science. Not only it has achieved the state-of-the-art results of its time but also has lent itself to become an important foundation in modern deep learning. The original CNN – LeNet-5 (Figure 3.1) – architecture consists of convolution and pooling operations followed by a multi-layer perceptron [85]. The weight-shared convolutional and pooling operations are the key ingredients for the model effectively extract and aggregate local invariants while remain to be compact with respect to the number of parameters.

The convolutional filter layers are interleaved with activation functions, followed by spatial feature pooling operations such as subsampling. For each 2D input matrix I, the convolutional layer computes the cross-correlation by convolving a trainable filter kernel of size $K \times K$ across the input.

$$F(i,j) = (I * K)(i,j) = \sum_{m=0}^{K} \sum_{n=0}^{K} I(m,n)K(i-m, j-n)$$

This results in a smaller output matrix often referred to as the feature map as each element of this matrix is a feature extracted from a connected neighborhood of elements in the input. Each convolutional layer C_i often contains a given number of different kernels that map to different feature maps. A trainable bias is added to the results of each convolutional mask, and a hyperbolic tangent function, used as an activation function is applied. These feature maps are stacked together as channels, resemble the color channels of images, and are used as the "input image" for the next layer. The feature map produced by the convolutional layers is highly effective in extracting low-level features as they take advantage of the translational symmetric of the cross-convolution operator. As such, the resulted feature maps are translational equivariant, meaning if the input is translated, the feature is also translated in the feature space to reflect that change. This helps the learned kernel to generalize edge, texture, and shape features in different relative locations. Once a feature has been detected, its exact location is less important for some tasks such as classification. Hence, each convolutional layer C_i is typically followed by a pooling layer S_i that takes the average (or maximum) values over a neighborhood and multiplies it by a trainable coefficient and adds a trainable bias. Pooling layers aggregate the equivariant features to be invariant, i.e., enforcing location-varied inputs to result in consistent feature maps, which is helpful for the final layers of multi-layer perception for classification.

Despite in the original paper, non-linear activations are used, it has been shown that performant results are achievable through longer training time without non-linear activation functions. For the sake of brevity when comparing with the separable convolution layer, we will demonstrate the complexity of a convo-

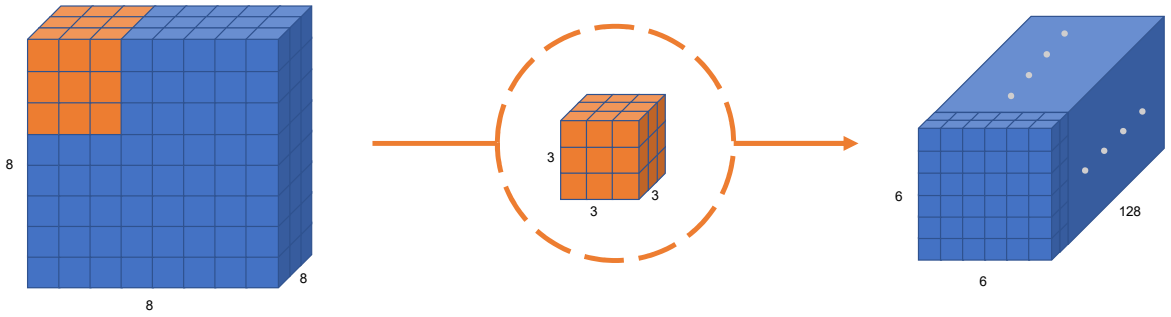


Figure 3.2: Normal Convolution

lutional layer with identity activation function. Such convolutional layer C_I with a kernel size K_i and an input of size $W_{in} \times H_{in}$ would yield an output map of size $W_i \times H_i$ where $(W_{in}, H_{in}) = (W_{in} - K_i + 1, H_{in} - K_i + 1)$.

The complexity of the back-propagation delta-rule algorithm for a given element is proportional to its output map size and the cardinal of its connections with the following layers. That means the complexity is proportional to $(W_i \times H_i)$, The number of weight-shared parameters is proportional to $(W_i \times H_i \times K_i^2)$. Notice that for a fully connected layer in an MLP that does not employ the weight sharing kernel trick, the number of parameters for such given input would have been proportional to $(W_i \times H_i \times W_{in} \times H_{in})$, which is considerably larger.

3.2.1.2 Separable Convolutional Filters

One common way to simplify the convolutional layers, and hence reduce the number of parameters, is by using separable convolutional filters. Separable convolutions can be expressed as the outer product of two vectors: $C_i = Ch_i * Cv_i = Cv_i * Ch_i$ where Ch_i is a row and Cv_i is a column vector of size K_i. This technique has been well-known in image processing with the most famous example being the separable Sobel filters. In normal feed-forward computation applied over a $W \times H$ input image, this transformation leads to a $K_i^2/(2K_i)$ speedup factor. This type of 2D convolution has been applied since at least 2012 [86] to speedup LeNet (Figure 3.1).

3.2.1.3 Depthwise Separable Convolutional Layer

Although the 2D separable convolution has been a well-known technique in, its version for 3D tensors are widely used only after the Xception paper [87], based on the extreme Inception hypothesis for convolutional neural network, introduced in [87]. In a convolutional neural network, a convolution layer attempts to learn filters in a 3D space with 2 spatial dimensions and 1 channel dimension implies that it is tasked with simultaneously mapping cross-channel correlations and spatial correlations. The Inception hypothesis postulates that cross-channel correlations and spatial correlations are sufficiently decoupled that it is preferable not to map them jointly. Hence, the cross-channel correlations can be captured by 1×1 convolutions and spatial correlations can be captured via regular 2D convolutions. The Xception hypothesis take this postulate to the eXtreme, coined Xception, and assume that cross-channel correlations and spatial correlations can be mapped completely separately, and subsequently perform 1×1 on every channel before the 2D convolution on each of the cross-channel output. As the convolution is commutative and the two convolution operators under the Xception hypothesis is always pairwise coupled, the order of the operator is not important. This pairwise operator is called depthwise separable convolution and is implemented as a 2D depthwise convolution (Figure 3.3) followed by a pointwise convolution (Figure 3.4)in practically every modern deep learning frameworks [88, 89, 90].

3.2.1.4 Time-depth and time-channel Separable Convolutional Layer

The concept of time-depth separable convolution (TDSConv) is first proposed in [91]. Hannun et al.'s TDSConv block operates over an input of shape $T \times w \times c$ where T is the number of time-steps, w is the input width and c is the number of channels. The basic TDSConv block is composed of a 2D convolutional layer where kernels are size $k \times 1$ over $(T \times w)$ and a fully-connected block, consisting

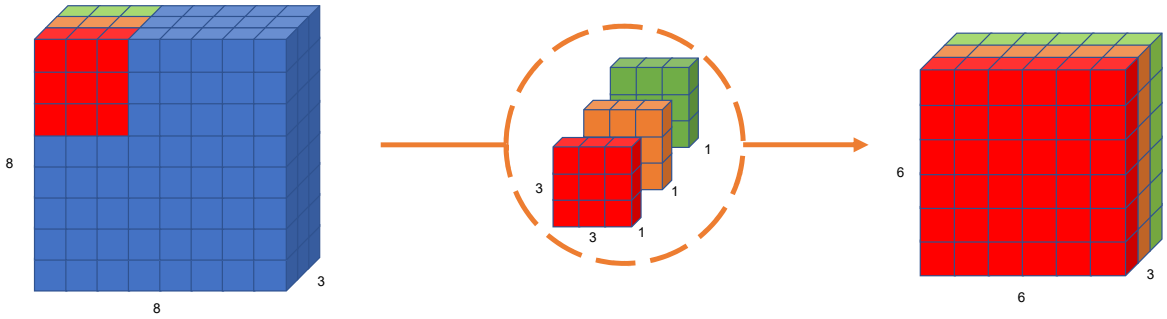


Figure 3.3: Depthwise Convolution

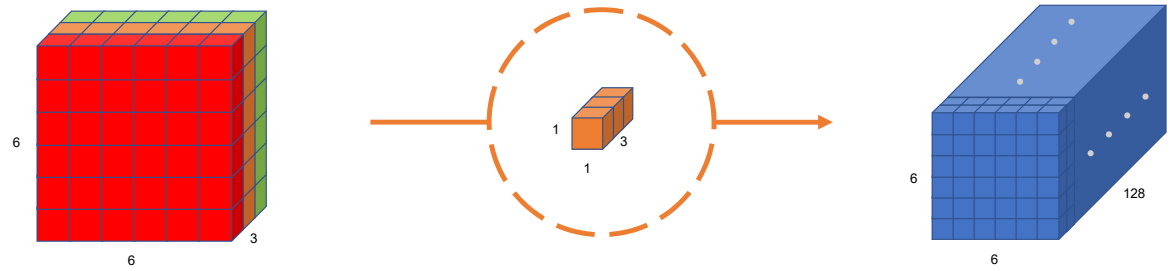


Figure 3.4: Pointwise Convolution

of 1×1 pointwise convolutions operating on $(w \cdot c)$ channels interleaved with layernorm layers. The fist 2D convolution layer take in an input of shape $T \times w \times c$ and also ouput of shape $T \times w \times c$. Then the output is recast into shape $T \times 1 \times wc$ and apply a sequence of two pointwise 1×1 with ReLU non-linearity in between. The total number of parameters in this layer is $k \times c^2 + 2 \times (w \cdot c)$.

The separable block employ in our work, in contrast, operates on data in time-channel format $(T \times c)$ and completely decouples the time and channel-wise parts of convolution and is called time-channel separable convolution (TCSConv). This means instead of employing a 2D separable convolutional layer followed by 2 pointwise convolutions as of the TDSConv, we substitute it with a 1D separable convolution follow by 1 pointwise convolution (Figure. 3.5). Particularly, the input is casted as a multi-channel time sequence of shape $T \times c$ with a depthwise convolution layer and a pointwise convolution layer, reducing the number of parameters to be $k \times c + c^2$.

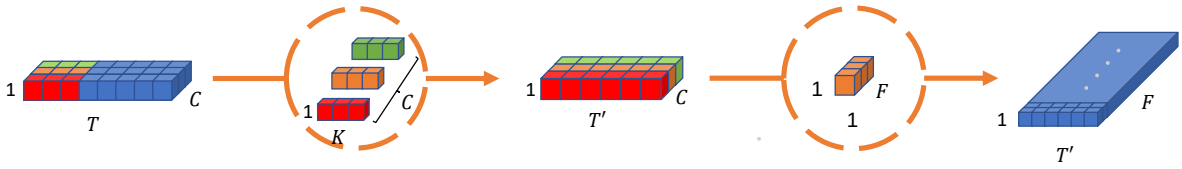


Figure 3.5: Time-channel Separable Convolution

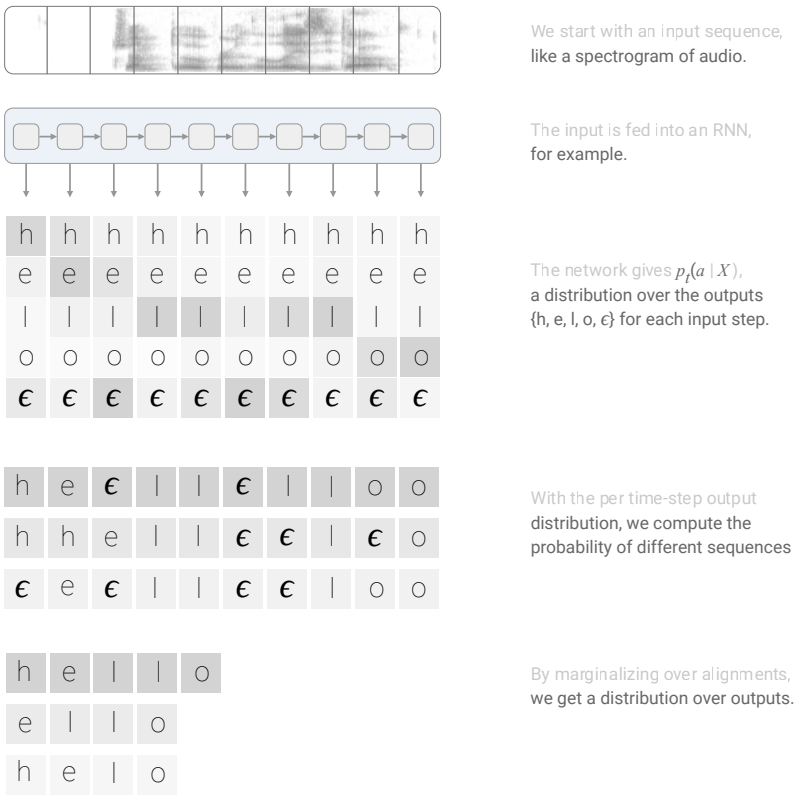


Figure 3.6: CTC algorithm for speech recognition [2]

3.2.1.5 Connectionist Temporal Classification Loss

Based one of the authors' experience in working with industry speech recognition systems, we are inspired to design our sequence-to-sequence model to be a Connectionist Temporal Classification (CTC) loss.

In previous approaches and public datasets for accelerometer-based human activity recognition, each accelerometer recordings during the course of an activity A are assigned a label A, regardless of which phase of the action it currently is.

Similarly, traditional training data used for speech recognition problems, where the recognizer transcribes a sequence of audio signals to a sequence of texts, requires every time frame of a voice recording to be labeled with a token. The limit of such approach is two-fold. First, such approach is not semantically sound for complex data with different phases as we have discussed above. For speech-to-text, it is sometimes impossible to have a concrete alignment between the modalities of source data and text-based labels, e.g., the alignment between phonemes and morphemes or character. Second, and more important to the task of human activity recognition, the intensive labeling effort constraint the collection of data to be mostly "in vitro" with organized sessions and prevent the acquisition of crowdsourced "in-the-wild" data as annotating the data along their natural routine can be bothersome for experiment participants [61, 60, 80].

To address this alignment problem, people have proposed the Connectionist Temporal Classification approach. For a given input sequence $X = [x_0, x_1, \ldots, x_T]$ correspond to an output sequence $Y = [y_0, y_1, \ldots, y_T]$.

The CTC algorithm gives us the an output distribution of all possible Y given an X. In other words, this is a probability distribution for all valid alignments from X to Y, where a valid alignment A from X can be collapsed into Y. For example, ccaaat, caaat, and catttt are valid alignments for cat.

Formally, with a set of learnable parameter Θ we consider $P(Y|X;\Theta)$ with $p(Y|X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t|X)$ is the objective probability for a single (X, Y) pair computed by propagating the probability of token alignment step-by-step. T is the sequence length, $\mathcal{A}_{X,Y}$ is the set of valid alignment for the pair (X, Y) drawing from the set of pairs propose by the recognizer, and A is a single valid alignment.

Using this probability distribution, we can either infer a likely output or assess the probability of a given output. Based on this distribution, a differentiable loss

function is computed as follow:

$$\mathcal{L} = \sum_{(X,Y)\in\mathcal{D}} -logp(Y|X)$$

where $\mathcal{D}$ is the training set.

3.2.2 Compound Methods

3.2.3 QuartzNet - Separable Convolutional Network for Sequential Signal

QuartzNet is an encoder-decoder CTC model originally designed for speech recognition tasks [3]. QuartzNet is famous for achieving near state-of-the-art results on large English speech recognition corpus while being very compact. Thus, it is highly favored in production settings where processing delay and deployment resource requirement is crucial. During one of the author experience in working with industry speech recognition systems, QuartzNet is regularly considered as deployment candidates, achieving good performance across both English and Vietnamese corpi. However, as mentioned systems and data corpi are confidential information, we are prohibited from discussing them in more details. Inspired by such performance given by QuartzNet, we adapt this model to the domain of human activity recognition using accelerometer data with the aim of achieving a good trade-off between performance and deployment constraints.

The building block of QuartzNet is the time-channel separable convolution (TCSConv) described in Subsection 3.2.1.4. The architecture of QuartzNet defines the TCSConv-BN-ReLU block to consist of a K-sized depthwise convolution layer followed by a pointwise convolution layer and batch norm. The output of the TCSConv-BN-ReLU block is computed via the non-linear ReLU activation function.

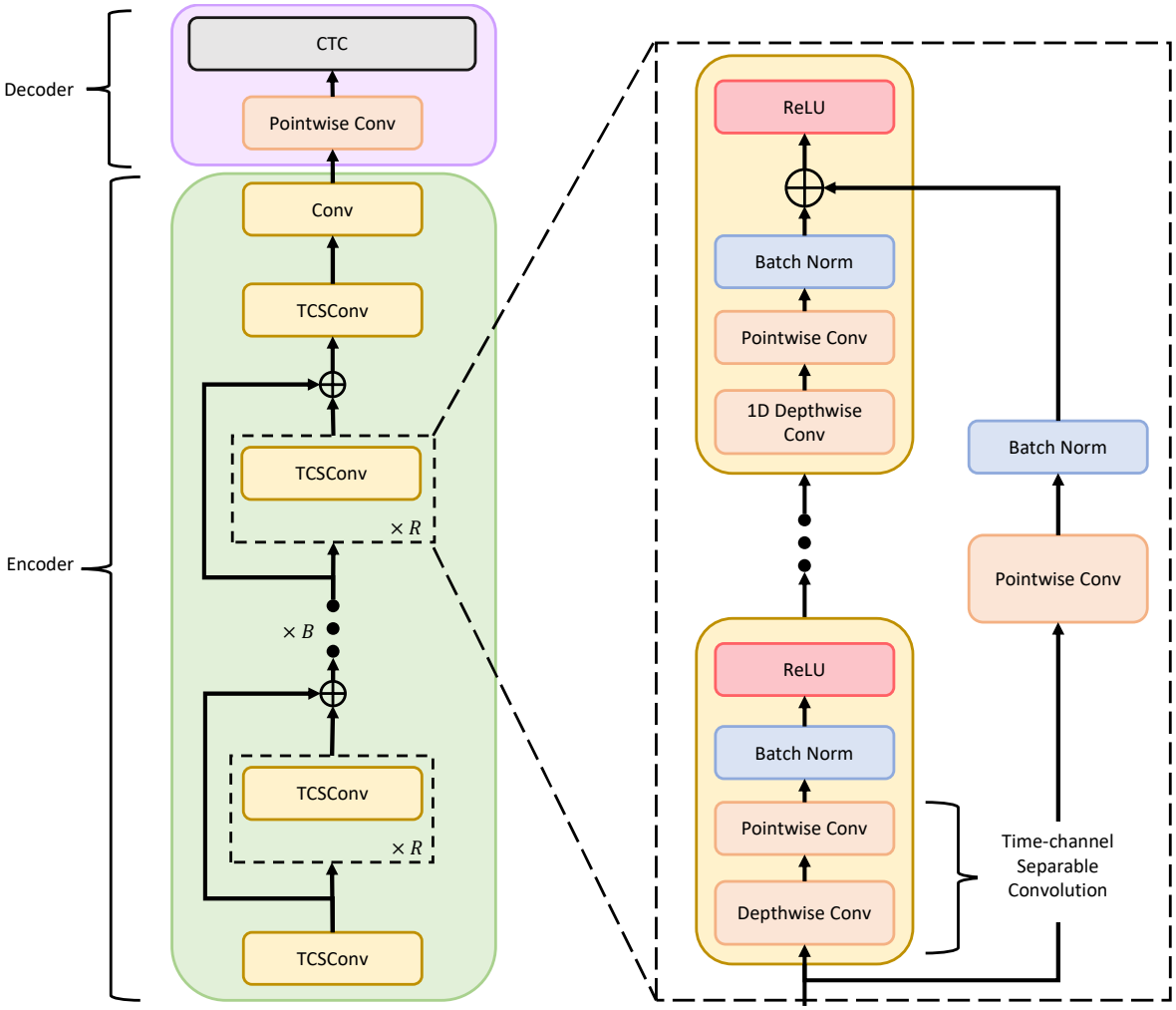


Figure 3.7: QuartzNet architecture adapted from [3]

The encoder-decoder architecture of QuartzNet is designed as a sequence of blocks. Each $B \times R$ configuration of QuartzNet consists of B block, each blocks, in turn, is made up of R repeated TCSConv-BN-ReLU sub-blocks. Each block input is connected directly to the last sub-block via a residual connection. Along this residual connection, the feature maps are projected through a 1×1 pointwise convolution to adapt with different numbers of input and output channels, before getting normalized through a batch norm layer. This residual connected feature is also added to the output of the batch norm layer in the final sub-block within each block. Finally, the result of each block is passed through an activation

function and dropout layer before getting forwarded to the next block.

We propose two approaches to adapt QuartzNet for the tri-axial accelerometer data as presented in the following subsections:

3.2.3.1 QuartzNetBxR-1D

We coin the first approach as QuartzNetBxR-1D. In this approach, we compute the STFT features of accelerometer axes separately before concatenating them channel-wise. Doing this allow us to reuse the TCSConv as in the orignal QuartzNet, with custom hyperparameters considered below.

3.2.3.2 QuartzNetBxR-3D

In the second approach, the STFT features of axes are stacked depth-wise, creating a 3D tensor of shape $(T, n\_feature\_bins, 3)$. We then implement a TDSConv block based on the TCSConv, where 2D depthwise convolution and pointwise convolution is used. Our implementation resemble the TDSConv block implemented by Hannun et al. TDSConv [91] in term of operating dimension. However they are simpler as the depthwise separable convolution is only follwed by one pointwise convolution instead of two. Its features map is compensated through the residual the pointwise convolution along the residual connection to the final layer of each B_i block and between B_i blocks instead.

3.3 Implementation Details

Unlike the original QuartzNet which use mel-filterbank features calculated from the STFT power spectra with 20ms windows over 16kHz mono audio with a 10ms overlap, we adapt QuartzNet for tri-axial accelerometer first at the preprocessing stage. To adapt with the $20Hz$ sampling rate of the WISDM dataset, we adjust the STFT window size to be of 3 seconds, with a stride of 0.5 second, which yields 120 samples per input frame. The number of FFT bins is chosen to be

60 to achieve $0.5Hz$ resolution adapted from the preprocessing in [74]. However, only the first 20 bins, correspond to frequencies $0-10Hz$ are used as these are the only bands with information captured in full, according to Whittaker-Nyquist-Shannon sampling theorem [92].

3.3.1 QuartzNet

We implement the described QuartzNet by extending the open-source NVIDIA NeMo Toolkit [93] and Pytorch library [89]. All of models used in experiments are trained using the NovoGrad optimizer, also proposed in the original QuartzNet paper.

3.3.1.1 QuartzNetBxR-1D

For the QuartzNetBxR-1D approach, we choose to start our experiment with the smallest QuartzNet configuration reported with good results, with $B = 5$ and $R = 3$ [3]. Next, as the original kernel size of QuartzNet is too large for the our samples, we perform manual parameter sweep to find a suitable kernel. The kernel size start at 3 and is increased by 6 for every consecutive layers, except for the first two layers whose kernel sizes are different by 3. The tuning data configuration used is the first splitting configuration described in Subsection 3.4.1. In each tuning iteration, the kernel sizes are increased by 2. After many iterations of not encouraging results, we restart the procedure by keeping the first kernel size be the original 33 while kernels in the rest of layers follow the sweeping process.

Next, as we proceed to evaluate the model on other configuration of data, we need to continue tuning the model as the abundant capacity of the model make it prone to be overfitted to the training set distribution and while perform poorly on the validation distribution only after a couple hundreds steps. To address

this we start another parameter sweeping procedures for the number of filters in each layers. We replace the layers with 256 filters by 8 filters and layers with 512 filters by 32 and double them after each iteration. The number of filters in the final pointwise convolution layer is set to be double that of its previous layers, and the number of layers in between is also reduced, starting with only 1 B_i layer and increase every iteration. For a given number of layer, the procedure consider increasing the number of filters until reaching the original size before increasing the number of layers and restart the sweeping process for kernel sizes.

We observe that more restrictive models are less prone to overfitting but also less likely to reach a good loss. Given the low resolution dataset with a limited size, it is hard to nail a sweet spot of trade-off. We then restart the procedure again, this time taken into account the dropout rate to randomly sparsify the model by randomly skip layers. After iterations, we identify a good trade-off point is using a large model with an aggressive dropout rate of 0.25. This configuration let model to have enough complexity to extract complex patterns from low resolution data while being able to counter the overfitting behavior.

This refined configuration of the architecture is defined using the Hydra-based listing as follow:

```
name: &name "QuartzNet5x3-1D"
model:
    sample_rate: &sample_rate 20
    repeat: &repeat 3
    dropout: &dropout 0.25
    separable: &separable true
    batch_size: &batch_size 256
    use_cer: true

```

In the heading of the model, we declare resusable fields such as `sample_rate`

which can be used across the configuration of dataloaders as well as in the preprocessor, `separable` to be used in declaring convolution blocks, and `use_cer` to signify the the use of the predefined character error rate to evaluate our model as we consider each of our label is a character.

Then for the preprocessor configuration, we have

```
preprocessor:
      _target_: modules.preprocessing.
 AudioToSpectrogramPreprocessor
      normalize: "per_feature"
      window_size: 3
      window_stride: 0.5
      sample_rate: *sample_rate
      window: "hann"
      onesided: false
      n_fft: 60
      fft_head: 20
```

The preprocessor is instantiated as a custom class `modules.preprocessing.AudioToSpectrogramPreprocessor` we implement to facilitate our custom STFT procedures. Particularly, our custom class consider the `onesided` parameter to facilitate asymetric FFT around 0 that can save the computation effort, and `fft_head` parameter to faciliate our selection of feature bins.

Next, the encoder is defined as an instance of the `ConvASREncoder` module – a NeMo `NeuralModule` for Convolutional Encoder. Specifically, as QuartzNet extends the implementations of the Jasper family [94], it can be implement by redefining specific arguments of Jasper's base configuration for convolution blocks.

```
   encoder:
      _target_: nemo.collections.asr.modules.ConvASREncoder
      feat_in: 60 #20 * 3
      activation: relu
```

```
        conv_mask: true

        jasper:
        - dilation: [1]
          dropout: *dropout
          filters: 256
          kernel: [33]
          repeat: 1
          residual: false
          separable: *separable
          stride: [2]
          ...

```

Each block in the `jasper` represent a convolution block in Figure 3.7. The last pointwise convolution is the decoder and is instantiate as the `nemo.collections.asr.modules.ConvASRDecoder` as follows:

```
    decoder:
        _target_: nemo.collections.asr.modules.ConvASRDecoder
        feat_in: *enc_filters
        num_classes: 18
        vocabulary: ["A", "B", "C", "D", "E", "F", "G", "H", "I
    ", "J", "K", "L", "M", "O", "P", "Q", "R", "S"] #WISDM
    dataset does not have class "N"

```

Details configuration of these blocks is summarized in Table 3.1.

In Table 3.1, each block B_i consists of R repeated TCSConv blocks. Each TCSConv block, in turn, has a kernel size of K, F filters, a dilation factor of D, and a stride of S.

Table 3.1: QuartzNet5x3-1D convolution block hyperparameters

Block	**R**	**K**	**F**	**D**	**S**
C_1	1	33	256	1	2
B_1	3	3	256	1	1
B_2	3	6	256	1	1
B_3	3	9	512	1	1
B_4	3	12	512	1	1
B_5	3	15	512	1	1
C_2	1	18	512	1	1
C_3	1	1	1024	2	1
C_4	1	1	$\|\|labels\|\|$	1	1

3.3.1.2 QuartzNetBxR-3D

To faciliate the design of QuartzNetBxR-3D, we implement two custom classes `modules.conv_asr.ConvASREncoder` and `modules.conv_asr.ConvASRDecoder` from the base for `nemo.collections.asr.modules.ConvASREncoder` and `nemo.collections.asr.modules.ConvASRDecoder`. In our preliminary experiments with QuartzNetBxR-3D models, they are able to converge much more quickly compare to their 1D counterparts. However, the performance is not different from a converged QuartzNetBxR-1D by a significant margin. Being that, the effects of configuration found for the QuartzNetBxR-1D is also applicable for the QuartzNetBxR-3D except for the operating dimension with comparable results. Extending the filters to new dimension, however, significantly slow down the time processing each epoch, suggesting that the processing time for each sample is significantly longer. For that reason, we do not proceed with this approach beyond preliminary findings.

3.4 Numerical Benchmarks

3.4.1 The WISDM-HARB Dataset

The "WISDM Human Activity Recognition and Biometric Dataset" (WISDM-HARB Dataset, also known as the WISDMv2 dataset) obtained from the UCI Repository is one of the few publicly available datasets in the field of smartwatch-based human activity recognition and is commonly used as a standard dataset across studies. The dataset contains tri-axial accelerometer and tri-axial gyroscope data captured at a rate of 20 Hz from smartphones running Android 6.0 (Google Nexus 5/5X and Samsung Galaxy S5), as well as a smartwatch running Android Wear 1.5 (LG G Watch) from 51 participants. The data is collected in controlled sessions where participants conduct 18 courses of predetermined physical human activities in each session. The activities include six general living non-hand-oriented activities, seven general living hand-oriented activities, and five eating-related activities with hand-oriented implications. A detailed summary of the activities in the dataset is given in Table 3.2. Each activity is carried out repeatedly for roughly 3 minutes during the recording session before moving on the the next activity. In order to assess our models, the data need to be segmented, of which we employed several schemes.

Table 3.2: WISDM dataset activities

Type	Subtype	Activity	Label
General activities	Not hand-oriented	Walking	A
		Jogging	B
		Climbing Stairs	C
		Sitting	D
		Standing	E
		Kicking Soccer Ball	M
	Hand-oriented	Typing	F
		Brushing Teeth	G
		Dribbling Basketball	P
		Tennis Ball Catch	O
		Handwriting	Q
		Clapping	R
		Folding Clothes	S
Eating activites	Hand-oriented	Eating Soup	H
		Eating Chips	I
		Eating Pasta	J
		Drinking	K
		Eating Sandwiches	L

Interleaved activity segmentation with overlaps:

As each activity in the dataset are organized into discrete recording courses while our model aims to recognize dynamic continuous sequences of activities, we need to emulate the data to facilitate our experiments. We segment continuous courses of activity into 3-second frames at , i.e., 60 sensor readings, each labeled by the activity of its original course. Then, we randomly shuffle these frames for each recording session – consists of 18 activities from a participant – and then segment them in sequences of various length. Then the set of samples is randomly splitted for the training set, validation set, and testing set with a ratio of $(0.7, 0.1, 0.2)$ and a pseudo-random seed of 65.

At first, we considered this splitting set to share the characteristics of normal

splittings used with the WISDM datset, where the samples are single 3-second frames with significant overlaps between $50\% - 80\%$ between frames [95, 96, 97, 98, 99], and is used to evaluate initial configurations of our model. However, after more careful consideration, we acknowledge that our model should not be trained on overlapping datasets as our model does not share the characteristics with other previously proposed models. Compare to our model, previous models operate on a single-frame basis, hence they might need temporal shifting contexts to better recognize the time-series patterns. Our model, however, works on a sequence basis with temporal shifts and alignment issues already been handled by the CTC loss. Thus, considering data with overlaps is taking an unnecessary risk of leaking data between the training set, the validation set, and the testing set. Even though no two samples are exactly the same, bits of shared information can still be embedded within each other, results in potential data leaks in randomly splitted subsets. Overlapped samples also yield more similar distribution between each data split compare to non-overlaped samples.

Given our goal is to develop and deploy a practically useful model, we choose not to proceed with this data configuration beyond tuning for the kernel sizes, even though our models were able to achieve a sequence error rate less than 0.0025.

3.4.2 Interleaved activity segmentation without overlaps

In this data configuration, we also split activities into 3-second frames and randomly shuffle frames for each recording session. However, we carefully create sequences by non-replacement sampling, i.e., making sure there is no frame exists in two different sequences, regardless of their sequence length. We introduce two sub-configuration for evaluation with this segmentation schemes: sequences with randomly varied length and sequences with a fixed 60-second length.

Fixed time-frame segmentation:

As discussed, the standard treatment for the WISDM dataset is to sample all recordings into 3-second frames with significant overlaps. Each of these frames are treated as a single sample for discrete activity recognizers. Despite our model is designed for sequence-to-sequence approach with not overlap between samples, we figure that we can evaluate the model on a similar, but more restrictive, ground with previous studies by using degraded 1-length sequences. In this case, the error rate metrics computed using edit distance is equivalent to the complement of accuracy by definition. This "on-point" evaluation and inference mode is applicable for power-saving mode of human activity recognition, e.g., wake-up motion, where the device only intermittently record and analyze samples "at a glance", instead of recording the data continuously. However, as the configuration is not completely the same as in other studies, we cannot consider our result to be on a fair direct comparison ground to previously proposed models. Moreover, as our method rely on the conditional probability distribution of sequences, it is expected that our result is not as good as specialized frame-based models.

3.4.3 Numerical Evaluation

3.4.3.1 Sequence Error Rate

Sequence Error Rate (SER) is a common metrics in evaluating sequence-to-sequence tasks. In our experiments, each activity label is considered as a token of the sequence. Given a sequence is made of several tokens, the metrics is derived from Levehnstein edit distance and is calculated as

$$SER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

in which,

- S is the number of substitutions
- D is the number of deletions
- I is the number of insertions
- C is the number of correct tokens
- N is the number of tokens in the reference truth ($N = S + D + C$)

In other words, the sequence error rate is the number of minimum correcting steps averaged by the sequence length. Hence, in the case of 1-length sequences, the SER is exactly the complement of the accuracy, calculated by averaged number of correctly predicted samples, or more concretely

$$Acc_{1-length} = 1 - SER_{1-length}$$

3.4.3.2 Numerical Results

The results of our experiments is presented in Table 3.3 and Table 3.4.

For the interleaved segmentation experiments, we evaluate our model with varied sequence lengths as discussed above. We also further our evaluation using

Table 3.3: Interleaved segmentation evaluation result

Model	(train, val, test)	Seg. length	Test SER
QuartzNet5x3-1D	(0.7, 0.1, 0.2)	random	0.3248
		60 seconds	0.3226
	(0.6, 0.1, 0.3)	random	0.3205
		60 seconds	0.3104

different data splits to examine the robustness of the model with different distribution.

As shown in Table 3.3, the results on different splitting ratios only deviate slightly, suggesting that our model is robust and have successfully overcomed the overfitting problem. The SERs on the order of 0.3 also suggest a positive outlook for such a compact sequence-to-sequence model as similar behavior have been observed in our experience with training the speech-to-text models on small corpi. Such results suggest that the performance can easily be enhanced in future work with better quality and quantity of data.

For the fixed time-frame segmentation experiments, we evaluate our models with 10-second frames and 3-second frames, with similar data ratio variants to assess the model's robustness. The results are reported in Table 3.4. To our surprise, our model is able to reach an equivalent accuracy of up to 0.93, without any drastic degeneration between experiments.

Table 3.4: Fixed time-frame segmentation evaluation result

Model	(train, val, test)	Seg. Length	Test SER	Equiv. Acc.
QuartzNet5x3-1D	(0.7, 0.1, 0.2)	10 seconds	0.1086	0.8914
		3 seconds	0.1839	0.8161
	(0.6, 0.1, 0.3)	10 seconds	0.0652	0.9348
		3 seconds	0.0945	0.9055

Despite the result is "on par" with some previous works with overlapping samples [99], we cannot directly compare results since we do not consider overlapped samples. Yet, the results are encouraging and suggest a positive outlook for unified model for both task of power-saving and always-on human activity recognition in deployment.

CHAPTER 4

SMART INTERACTION PLATFORM WITH SMART WATCH

This chapter introduces authentication systems on popular computer platforms. We discuss about system architectures and authentication processes on operating systems. While, on web-based platforms, an overview of popular authentication techniques is introduced. We further introduce a simple and flexible authentication system for multiple platforms. Additionally, technical detail for implementing and implementation detail of the system are provided including separately modules on Windows, Linux and web-based platforms.

4.1 Authentication On Common Computer Platforms

In computing context, authentication is the process for verifying the identity of an object, service or person. Authentication processes can be different based on authentication scenarios, but the goal is always to know who the user is or what the service/ object is. Authentication scenarios are various [100] and may have specific requirements [101], so many authentication techniques are available to support a wide range of authentication scenarios [101]. Username-Password, ID tokens, public key certificates, and biometrics are examples for authentication techniques [101]. As set forth by Schmidt [102] regarding the interactive modality for interaction, we should not "blame the users" for their bad habit in keeping credentials but rather focus on improving the modality to be effortless for them.

In this subsection, we introduce an implementation for authentication technique using smartwatches. Our implementation can be used for authentication on following platforms/ environments:

- Windows operating system environments - any version released after Win-

dows Vista

- Linux distros such as archlinux, ubuntu, gentoo, etc.
- Web applications: Facebook, Gmail, Youtube, Amazon, etc

Before we introduce our system in this section, we would like to describe architectures of authentication system/module on Windows and Linux distros and demonstrate step-by-step processes that Windows and Linux distros are using to authenticate users when users perform logon process. With web applications, we would like to introduce popular techniques are used to authenticate on web environment.

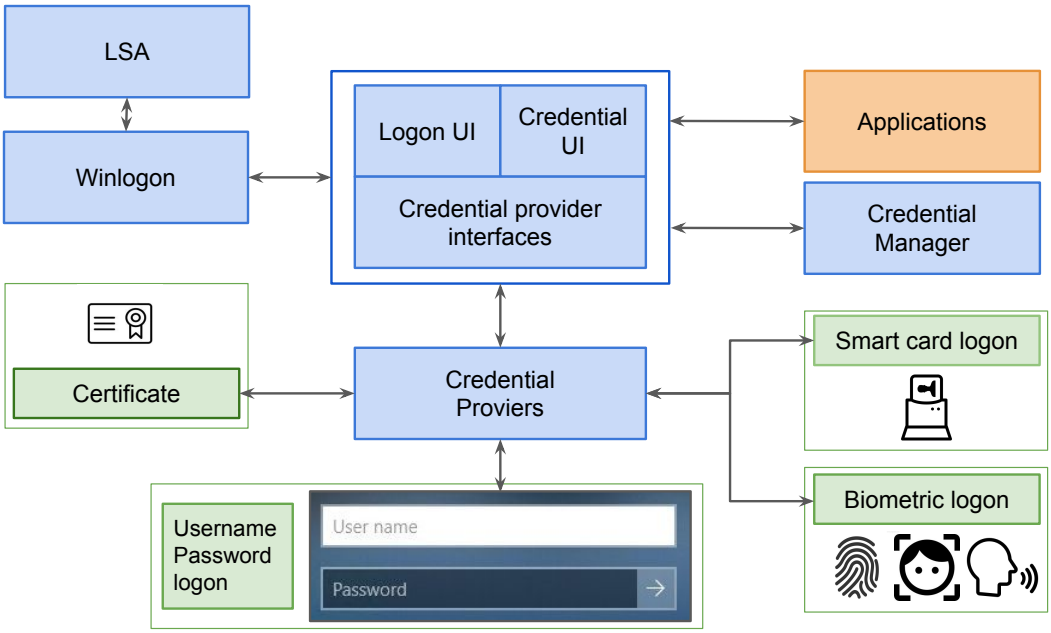
4.1.1 Windows Credential Providers

Legacy Windows version use GINA-based architecture for authentication. GINA (Graphical Identification and Authentication) was dropped from long-term support when Windows Vista released, making it is hard for developers to understand and implement. For Microsoft, the cost to keep supporting GINA is expensive. The replacement of GINA-based architecture is the Credential Provider model that is used on Windows Vista, Windows 7, Windows 8, Windows 8.1, Windows 10 and the server operating systems such as Windows Server 2008, Windows Server 2008 R2 and Windows server 2012 and later. The new credential provider reduce the develop effort, improve the way to implement and more robust security [103].

Each credential provider is built as a dynamic-link library (DLL) file, typically located in `system32` folder on a Windows environment. A credential provider is registered in Registry at following path:

```
HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows
\CurrentVersion\Authentication\Credential Providers
```

4.1.1.1 Architecture



Redraw based on [100] in Microsoft Windows documentation

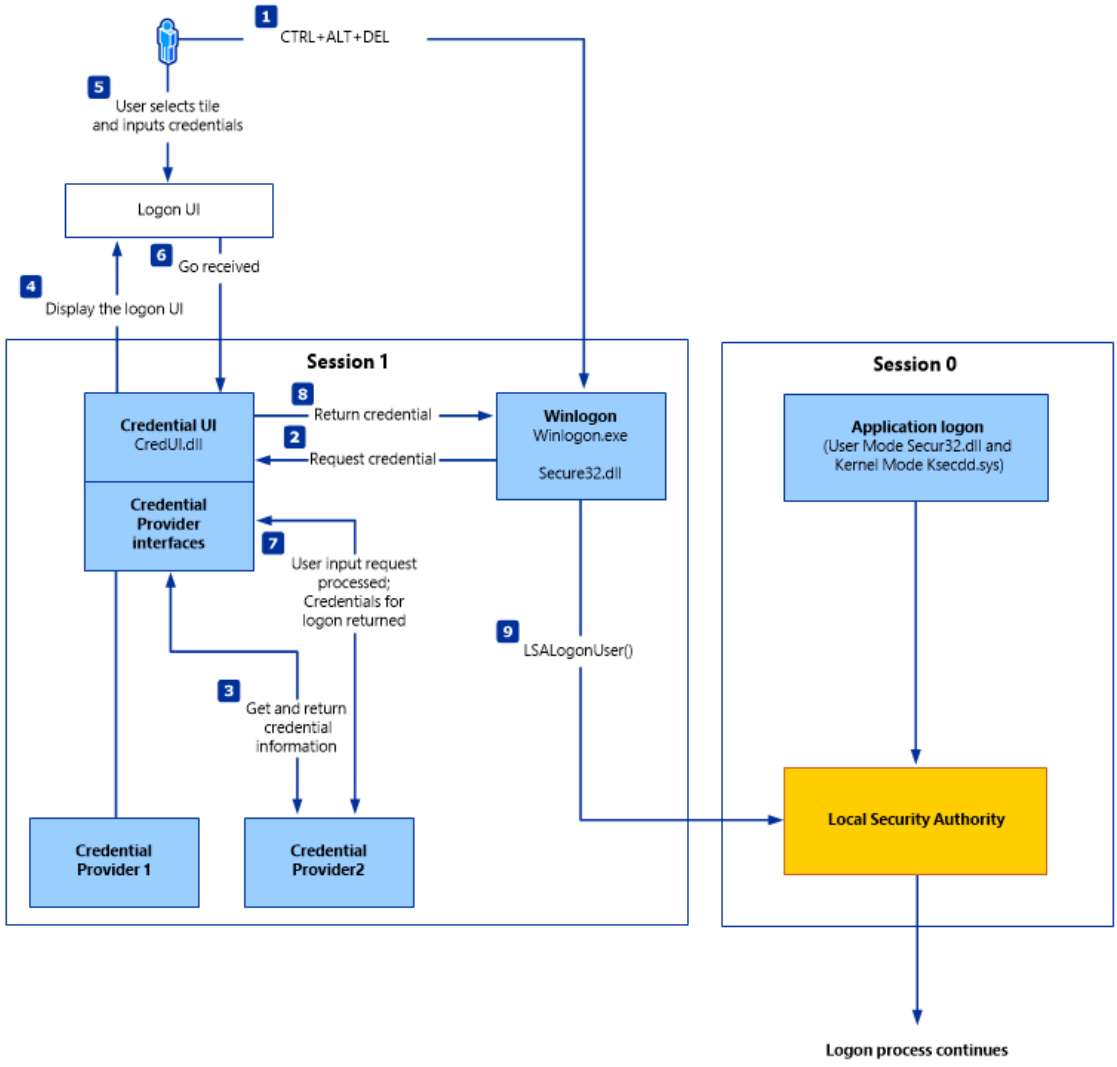
Figure 4.1: Interactive Login Architecture

The Interactive Login Architecture shows in Figure 4.1. The following list is the components in the interactive logon architecture Microsoft, which is used for their operating systems [100]:

- **Logon UI**: providers interactive UI rendering. For example: display Username Password, form, display credential provider tiles for users to select the logon mechanism (smartcard, PIN, fingerprint, etc).
- **Winlogon**: provides interactive logon infrastructure, and instruct the logon UI to display
- **Credential providers**: describe and serialize the credential information required for authentication. A credential for a user can be username and password or credential information contained on chip of a smart card.

- **Local Security Authority**: Process logon credentials. Load authentication packages [104].
- **Authentication packages**: Analyze logon data [104] and communicate with server authentication. Include 2 authentication protocols Windows New Technology LAN Manager (NTLM) and Kerberos.

4.1.1.2 Credential Providers Authentication Process

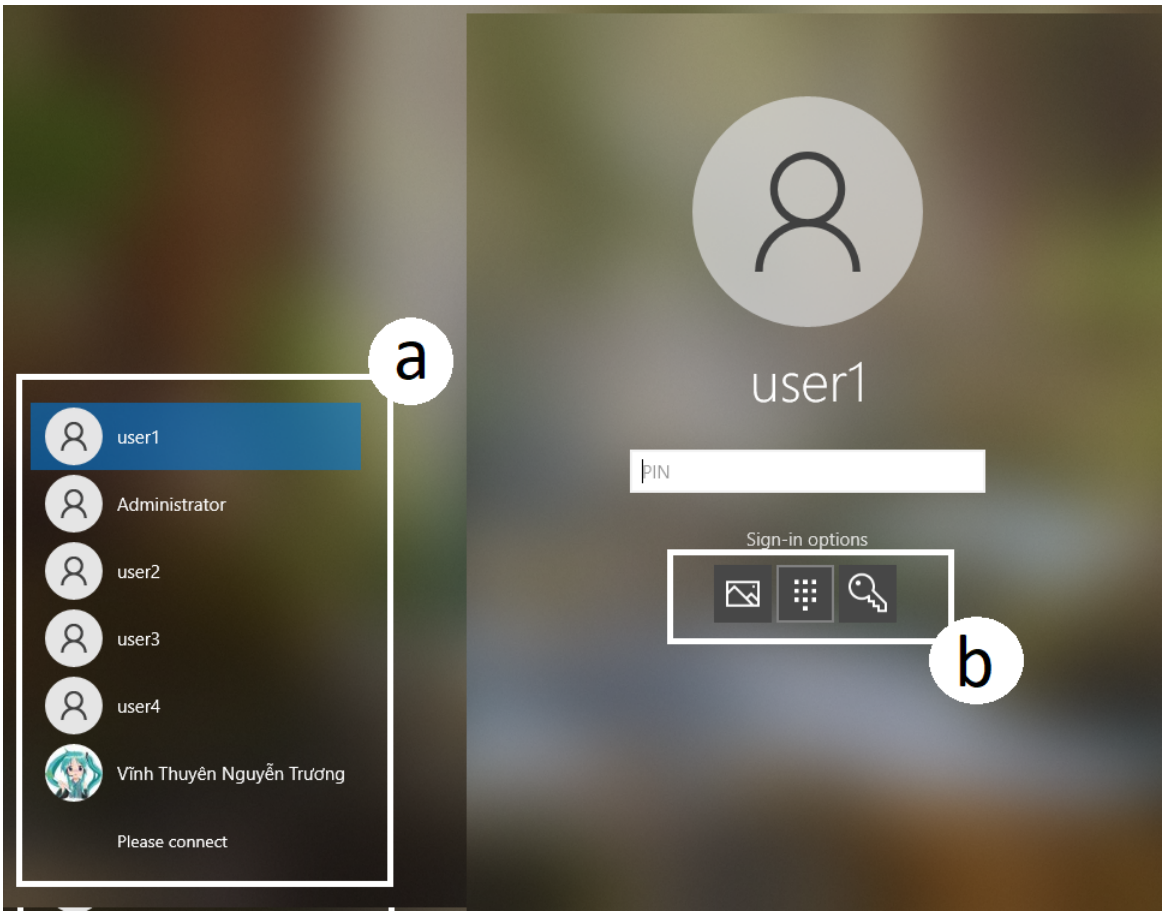


Reproduce from Microsoft Windows documentation [105].

Figure 4.2: Credentials Processes in Windows Authentication

We reproduce a diagram designed by Microsoft in their official documentation in Figure 4.2. The diagram demonstrates the credential process for their developed operating systems.

We describe the process step-by-step as showing in the diagram as the following 9-step process:



a) Tiles list, 5 items from the top are tiles for each users on local machine and can authenticate with local accounts. The sixth tile is a user that can be authenticated using Microsoft account. The last tile "Please connect" is our implemented custom credential provider tile. b) Credential provider lists, user1 can use 3 authentication techniques: the first credential provider is picture password, the second provider is PIN-authentication, and the last provider is password-based authentication.

Figure 4.3: Components on a Windows Logon screen

1. Winlogon is called when the system start up, when the user signs out of an account, or when the user presses CTRL+ALT+DEL combination.
2. Winlogon request Credential UI for credential
3. Credential UI can request Credential Providers information via the Credential Provider Interface.
4. Information from step 3 is used to display on Logon UI
5. User selects the tile and the credential provider then inputs corresponding credential information. A visualization for the tiles list and the credential provider list show in Figure 4.3. For example: user selects fingerprint tile on Logon screen, then he/she puts their finger on the fingerprint scanner.
6. The input data will be sent from Login UI to Credential UI.
7. The user's input is processed depend on the selected Credential Provider and prepare credentials for logon return.
8. Credential that is received from previous step is sent back to Winlogon.
9. Credential Providers is used to collect and serialize credential data and not not enforcement mechanisms [105] So, Winlogon call `LSALogonUser()` that sent serialized data LSA to do enforce security.

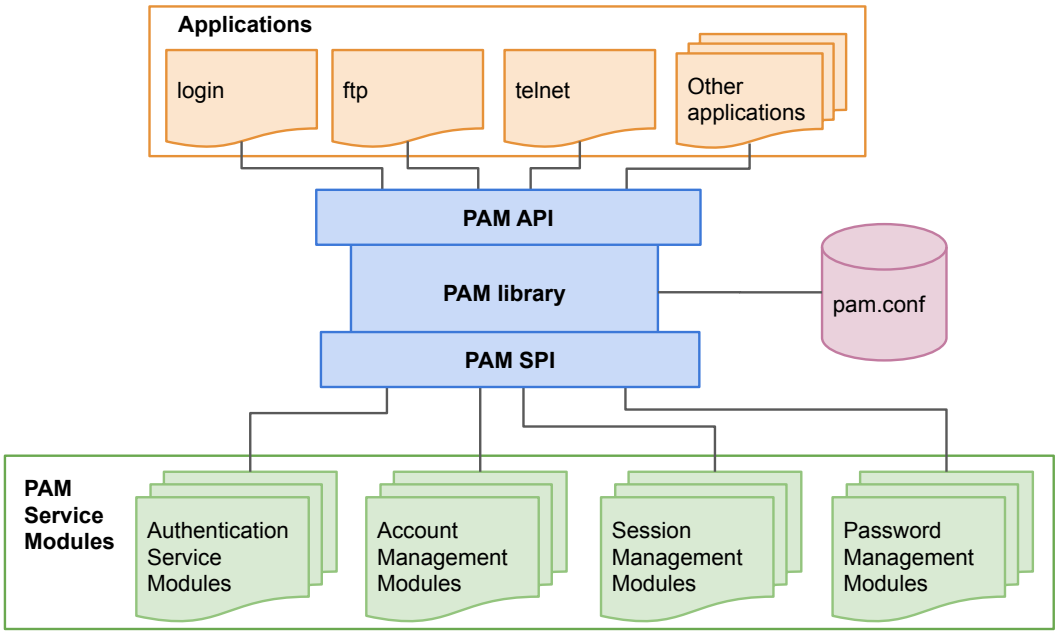
4.1.2 Linux Pluggable Authentication Modules

Traditionally, to implement authentication, Linux developers can use the standard libraries approach. However this approach has many limitations [106]:

- Not allow to set a system-wide standard for the encryption protocol.
- Require access to the encrypted password.
- Require to prompt logon information every single time access resource require authentication.
- Require Password-based technique. More implementation is required for additional techniques such as biometrics, smartcard, etc.

To solved these limitations, the Pluggable Authentication Modules (PAM) was proposed as a new standard in 1995 by Sun Microsystems. PAM is a system of shared libraries for authentication (Open Software Foundation RFC 86.0 [107]) [106], it also was published at CCS '96: Proceedings of the 3rd ACM conference on Computer and communications security [108]. The architecture and design of PAM provides for both pluggability and ease-of-use [108]. The idea is to centralization module for applications and credential techniques. When an application requires authentication, it has to be linked against PAM to verify the user identity. That design helps developers easy to add system-wide support for additional techniques like biometrics or ID tokens [106] and also help to stack these techniques to create multiple user authentication techniques [108].

4.1.2.1 Architecture



The diagram is redrawn from [109] documentation.

Figure 4.4: PAM architecture and core components relationship

The diagram shows the relationship between core components in the PAM architecture in Figure 4.4.

The architecture of the PAM framework consist of following core components :

- **PAM Library** - Authentication library API: a library contains PAM API providing interfaces to applications to load and control modules. PAM service modules communicate with PAM library through this interface [109].
- **PAM Service Modules/Providers** - Authentication mechanism-specific modules: developers can write custom modules to provide authentication techniques, and management such as account, session and password.
- **Applications**: an application use PAM to authentication can be called a PAM consumer. Applications communicate with PAM library through the PAM API [109].
- **PAM Configuration**: in old system PAM configuration is a text file named `pam.conf`. A new version of PAM configuration is a folder named `pam.d/` containing multiple configuration files. PAM configuration files allow users to configure a particular service module independently with other modules interfaces [109].

4.1.2.2 PAM configuration

PAM configuration files have 2 version: a single file named `pam.conf` and `pam.d/` which is a folder contains configuration files for each application [110].

`pam.conf` - the legacy configuration file in old systems that contains configuration for multiple applications in a single file and the file's format has following columns [110]:

- The first column is a application name.
- The second column is a module interface/ function type.
- The third column is control flag/ control argument.

- The fourth column is a module you want to load.
- The last column is optional, this is the module arguments if the module requires

A `pam.conf` sample from Oracle's documentation [111]:

```
#
# login service (explicit because of pam_dial_auth)
#
login    auth    required    pam_dhkeys.so.1
login    auth    required    pam_dial_auth.so.1
login    auth    binding     pam_unix_auth.so.1 server_policy
login    auth    required    pam_ldap.so.1 use_first_pass
#
# rlogin service (explicit because of pam_rhost_auth)
#
rlogin   auth    sufficient pam_rhosts_auth.so.1
rlogin   auth    requisite  pam_authtok_get.so.1
rlogin   auth    required   pam_dhkeys.so.1
rlogin   auth    binding    pam_unix_auth.so.1 server_policy
rlogin   auth    required   pam_ldap.so.1 use_first_pass
...
```

In a system uses new version of PAM configuration, each application configuration is placed on a single text file, and they are placed in the `pam.d/` folder. An example for list of files in `pam.d/` folder on the author's local Linux system in Figure 4.5. The folder contains PAM configuration for basic build blocks of a Linux system like `systemd` and other applications such as `postgresql` (database manager), `lightdm` (desktop display manager). And each file's format contains the last 4 columns of the old `pam.conf` file. The following code below is an example of a file in `pam.d/` folder, this is a file named login - the configuration file of login application:

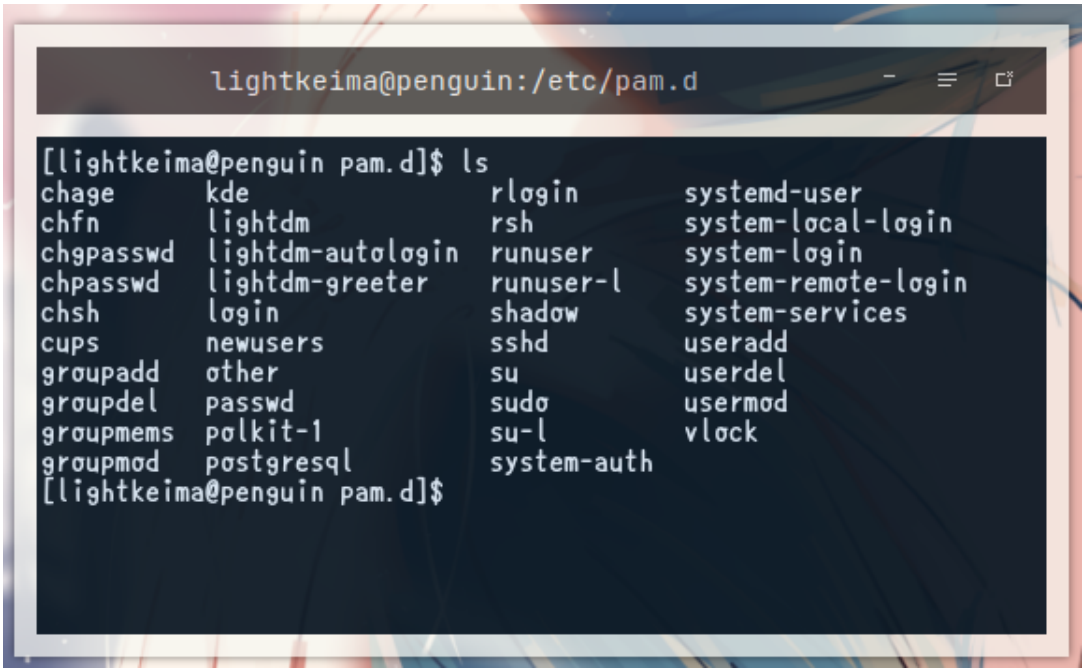


Figure 4.5: List of files in `pam.d/` folder on the author's Linux system

```
#%PAM-1.0
auth        required      pam_securetty.so
auth        requisite     pam_nologin.so
# include system-local-login configuration file in pam.d/
auth        include       system-local-login
account     include       system-local-login
session     include       system-local-login
```

Module Interfaces:

We introduce 4 types of PAM service modules showing in Figure 4.4. Therefore, four types of PAM module interfaces are available:

- `auth`: use for authenticating a user.
- `account`: use for verifying access.
- `password`: use for modifying user passwords or other credentials.
- `session`: configure and manages user session like creating new home direc-

tory when an user's account successful login for the first time.

Control Flags: Control flags help PAM to know what to do if a module run successful or failure. The list of available control flags in PAM configuration:

- `required`: The result must be successful to continue. Otherwise, the process stops and notifies a failure message to users.
- `requisite`: The result must be successful to continue
- `sufficient`: If the result is fails, it will be ignored. If the results of a module with sufficient flag has successful result and all previous modules with flagged required have successful results, then the user is authenticated to the service without requiring more other results.
- `optional`: The result is ignored. A module with optional flagged only needs when no other modules reference the interface for successful authentication.

An example of configuration file that we reproduce from How Linux Works (3rd Edition) book [106]:

```
1 auth sufficient pam_rootok.so
2 auth requisite  pam_shells.so
3 auth sufficient pam_unix.so
4 auth required   pam_deny.so
```

We would like to describe step-by-step the authentication process of this configuration file. First, we describe the meaning of each module:

- `pam_rootok`: verify root (highest permission user in Linux system) is trying to authentication.
- `pam_shells`: verify `/etc/shells` file is available, that is a file stores a list of available shells in Linux system.
- `pam_unix`: verify the user enter correct password.
- `pam_deny`: this module always returns fail result.

A flow diagram for this example configuration in Figure 4.15. The steps of the authentication process when the application calls for PAM library:

4.1.2.3 Authentication Process

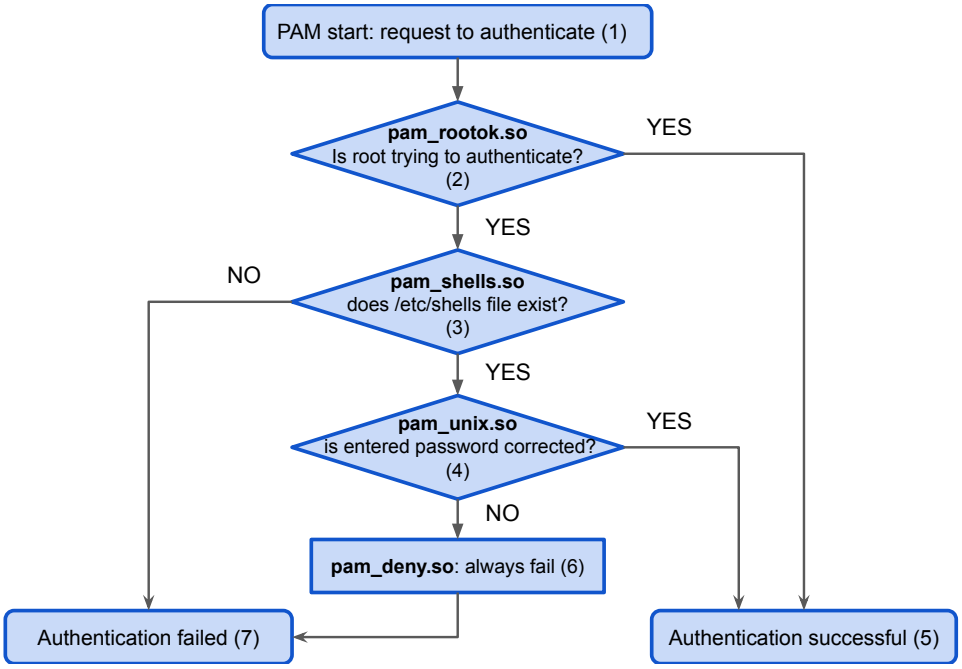


Figure is redrawn from How Linux Works (3rd Edition) book [106]

Figure 4.6: PAM configuration example: Authentication process

1. PAM start when an application requests to authenticate
2. Run `pam_rookok.so` module and check the results
3. Run `pam_shells.so` module, if `/etc/shells` file is found and contains any shell in it, the result is success, then move to step 3. If /etc/shells file is missing or no shell available in the shell list, the the result is fail and move to step 7.
4. Run `pam_unix.so` module. If the user enter correct password, move to step 5 else move to step 6.
5. Successful result, user is authenticated.

6. `pam_deny.so`, this module always return a fail result.
7. Authentication is failed.

4.1.3 Web-based Applications Authentication

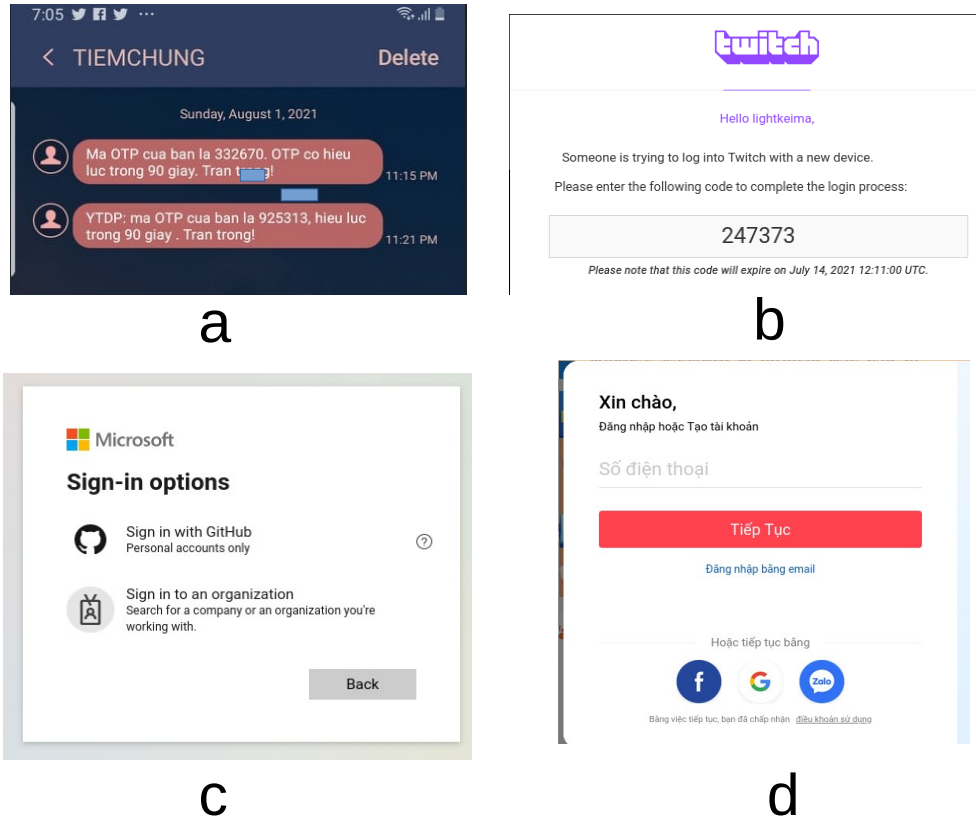
Different web-based applications can have different authentication architecture using different technologies and mechanism. If we ignore the technologies and security techniques behind these applications for authentication processes and just care about the way users input their credentials, the popular technique every websites and web applications are using is password-based technique (do not include web service for developers which typically using token API or not user-friendly credential) . These websites and web applications can provide multiple factors to improve security of their system such as:

- One-time password via email or short message service.
- Email addresses.
- Single sign-on authentication via third party account.
- Authentication with local devices.

Figures 4.7 4.8 demonstrate popular authentications on real web-based applications. In this web authentication subsection, the architecture inside a website or a web application is an interesting topic; however it is out of the scope of our topic. We don't have to modify a web application authentication architecture to do authentication process and only care about the credentials a user have to provide if they want to access a web service, to achieve that goal by create a web browser extension that we will introduce in next section.

4.2 Authentication System Design and Implementation

In this section, we want to introduce a design of our authentication system. The authentication system design goal and core components implementations will be



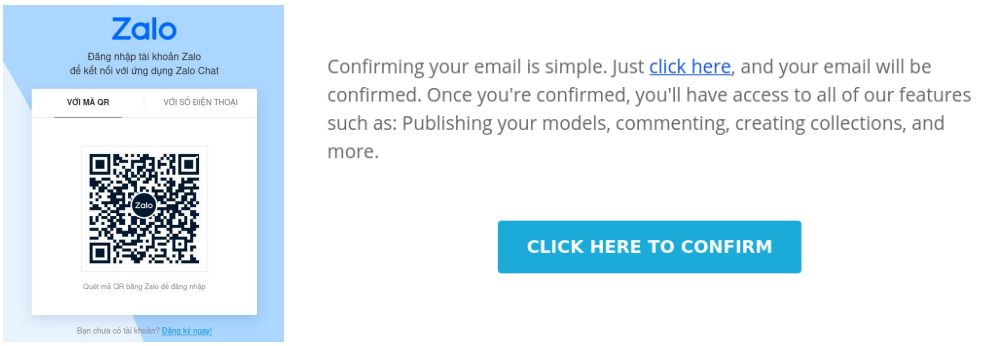
a) An example for OTP: OTP is sent via SMS to authenticate for vaccination registration. b) An example for OTP: twitch.tv - a video streaming service sends an OTP via email. c) An example for third party authentication: the first authentication option to Microsoft account login is using Github account. d) An example for third party authentication: Tiki.vn - an e-commerce website allows to login using Facebook, Google and Zalo account.

Figure 4.7: Web-based authentication techniques examples

discussed in next subsection.

4.2.1 Authentication Techniques

We propose a list of authentication techniques that can be used in our system. Figure 4.9 shows techniques we focus on in the scope of this thesis. Figure 4.10 describes techniques we leave for future work.



a) Zalo - a realtime chat application authenticate a user by scanning a QR code using another device that already signed in. b) An email to conform the email address.

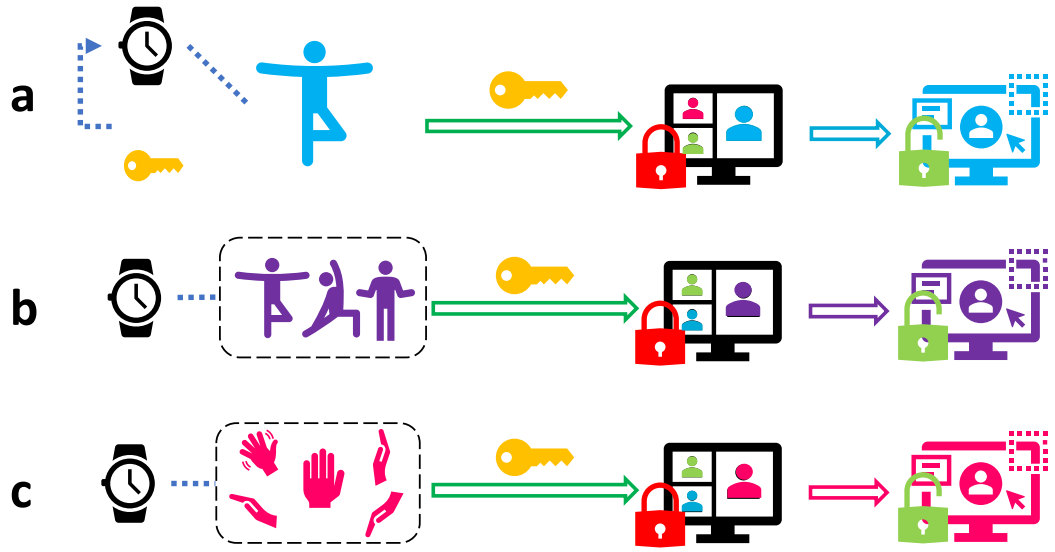
Figure 4.8: Web-based authentication using local devices and email addresses

4.2.2 System Design

The goals for our authentication system in summary:

- Supports multiple platforms.
- Easy integrate new platform.
- Various authentication techniques can be used.
- Authentication server shares data and functionalities among platforms.
- Communication protocols can be replaced for advanced security techniques.

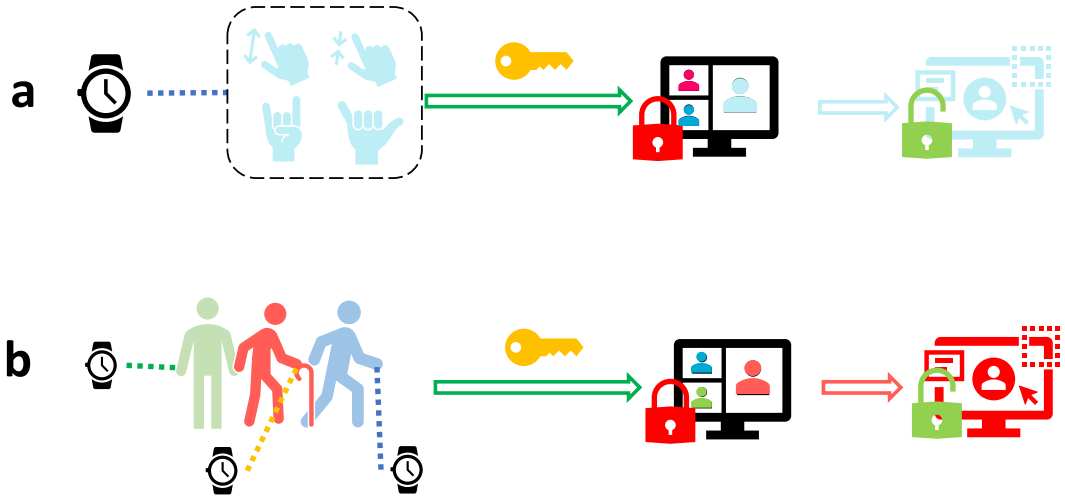
In order to support multiple platforms, core components are implemented using various technologies and tools. We work with web development technology for web platform and write modules for opensource operating system (Linux system) and closed source operating system (Windows) that have different design aspects. Therefore, to we have to learn the design of authentication system for each platform in order to implement external modules for authentication on the platforms. The communication between components in our system shows



a) Token-based authentication, a token is stored in smartwatch's storage. b) Movement-based authentication. c) Gesture-based authentication.

Figure 4.9: Authentication techniques

in Figure 4.11. The system aims to support new platforms without modifying old components, so we separate our system into 2 main parts: a part for credential information collection, and a part for authentication. The first part is implemented as an additional module for local authentication system of each supported platform, To deploy on a new platform, we can implement a module to collect credential information and connect it to our authentication server. The second part is an authentication server, we separate this part sharing between platforms. Additionally, the information stores in the database can be accessed by multiple machines. The implementation details for each component in our system are provided in the following subsections.



a) Subtle gesture-based authentication. b) Gait-based authentication.

Figure 4.10: Future work: Authentication techniques

4.2.3 Module Implementation: Custom Windows Credential Provider

In this subsection, we describe implementation detail for our Custom Credential Provider on Windows environment.

4.2.3.1 Windows Credential Provider Interfaces Implementation

In order to build a custom Windows Credential Provider, we need to implement two interfaces that are provided by Windows SDK. The methods of two interfaces will be called by LogonUI and CredentialUI processes to perform authentication with user credentials. The required interfaces to implement are :

- `ICredentialProvider` interface [112]: use to setup and manipulate a credential provider.
- `ICredentialProviderCredential` interfaces [113]: contains methods help to handle a credential.

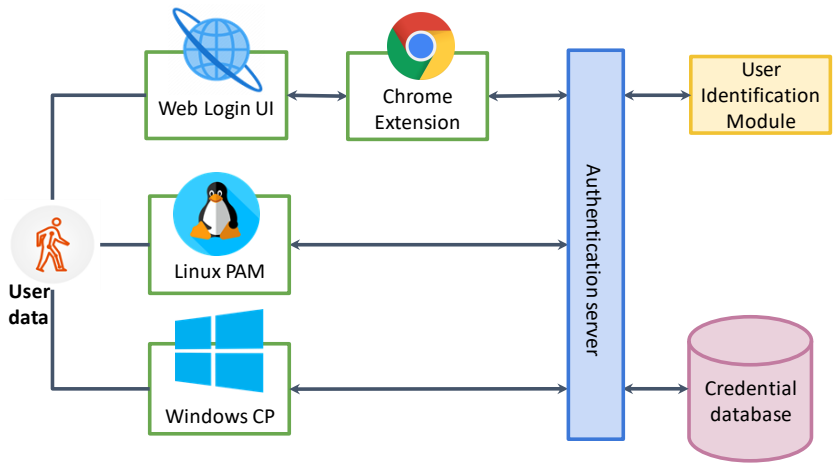


Figure 4.11: Authentication system architecture

`ICredentialProviderCredential` interfaces have 2 versions that the newest version inherits all methods from the first version except one named `OnCreatingWindow()` and introduces batch update of fields in LoginUI and CredentialUI [114]. In order to do automatically logon, we use the first version based on the remark in WindowsSDK document that recommends the first version is used to do automatically logon [114]. A class diagram shows the relationship between classes in our implementation in Figure 4.12. `CProvider` is an implementation for `ICrentialProvider` interface, while `CMessageCredential` and `CCredential` are implementations for `ICredentialProviderCredential` interface. We also implement a `SocketListener` to listen credential data and notify change on credential state.

`CProvider` has 2 attributes `CMessageCrential` and `CCredential` types that help `CProvider` to access the instance of these 2 classes. `SocketListener` contains pointer to an intance of `CProvider` class to access change state methods to notify update in credential provider, while `CProvider` contains pointer to an object of `SocketListener` to create that object in initialization step.

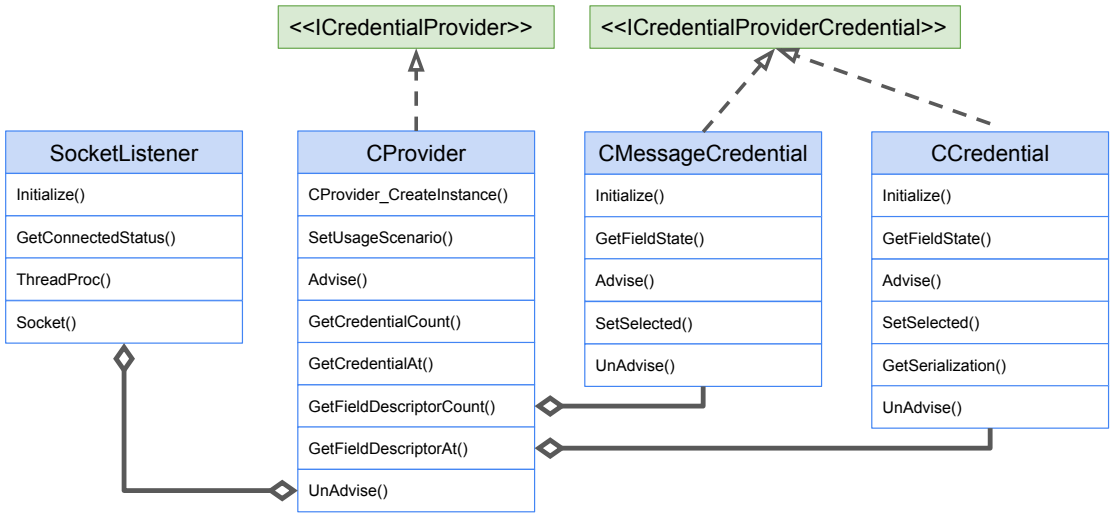


Figure 4.12: Custom Credential Provider class diagram

4.2.3.2 Authentication Process

When user select this credential provider tile - the "Please connect" tile in the tile list on Figure 4.3, our custom credential provider will execute. The process contains 2 separate threads:

- Thread 1: This thread has 2 state: "Initialization" state to setup `CProvider` , `CCredential`, `CMessageCredential`, `SocketListener` instances. Thread 2 also is started in this state. In "Initialization" state `CCredential` is initialized but not be used. When the state change to "Process credential" state, `CCMessageCredential` instance is released and `CCredential` instance is used to process and package credential information in order to send to Local Security Authority subsystem to enforce authentication.
- Thread 2: A socket is bind and wait for data. If the socket receives data, then it run `OnConnectStatusChanged()` method of a `CProvider` object to notify the state change.

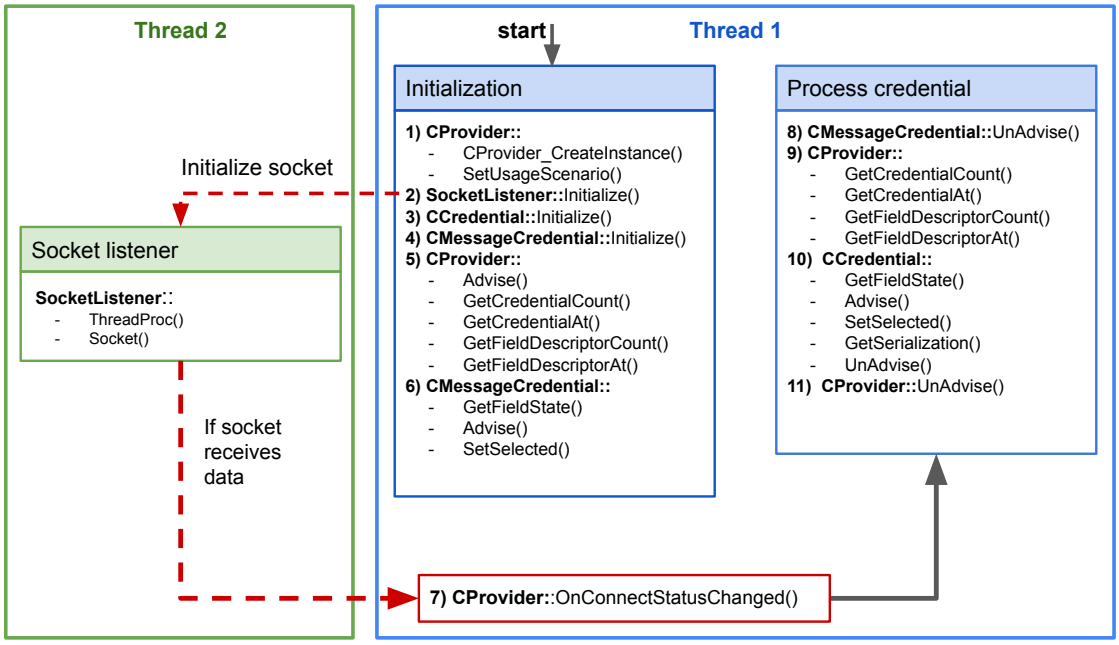


Figure 4.13: Custom Credential Provider process

In Figure 4.13, we propose a list of called methods during the whole process in order:

1. 2 methods of `CProvider` are called: `CProvider_CreateInstance()` initializes a `CProvider` instances and `SetUsageScenario()` determines that a user uses logon screen.
2. `SocketListener` calls `initialize()`, It creates new thread runs `threadProc()` method. In new thread, we bind a socket to listen to data and notify state change to `CProvider` instance.
3. `CCredential` runs `Initialize()` to setup its instance, and do nothing in this state.
4. `CProvider` runs `Advise()` method, this method helps `CProvider` to assign `ICredentialProviderEvents` from `Logon UI` and stinlineCredential UI to its appropriate pointer. `GetCredentialCount()` receives number of Credentials available in this Credential Provider, after that it can get a Credential by called `GetCredentialAt()` by passing an index. `GetFieldDescriptorCount()`

are called to receive the number of needed information for a Credential and then it can get each information by called `GetFieldDescriptorAt()` by passing the indexes.

5. `CMessageCredential` get information from fields to display in user tile by calling `GetFieldState()`, after that `Advise()` is executed to enable synchronous callback communication with `CProvider` by passing value to the pointer `IcredentialProviderCredentialEvents` . The `SetSelected()` then called to setup logon characteristics, in this case, it set autologon is false.
6. If socket receives data and successful verifies user identification to receive logon credential from the Authentication Server, `OnConnectStatusChanged()` method of `CProvider` will be called and start the next state.
7. `CMessageCredential` instance is no more in use, so `UnAdvise()` is called to release it.
8. The credential instance is pointed by `CProvider` corresponding pointer has been changed, so re-enumeration is needed.
   `GetCredentialCount()`, `GetCredentialAt()`, `GetFieldDescriptorCount()` and `GetFieldDescriptorAt()` run to perform the same task at the step 5.
9. `CCredential` runs a list of methods like step 6. However, when `SetSelected()` runs it set the autologon characteristic is true. `GetSerialization()` method package processed data and send it to Local Security Authority subsystem to enforce authentication. `UnAdvise()` also is called to release the object.
10. `CProvider` releases itself by calling `UnAdvise()` method.

4.2.4 Module Implementation: Linux PAM Module

On Linux system, PAM modules are easier to build than a Credential Provider on Windows system based on the architecture and samples of implementations from online repository. A PAM module is built as an object file and placed in `/etc/security/` directory, then any applications that uses PAM are able to use the

module by modifying their configuration in `/etc/pam.d`. Instead of creating our custom PAM module as an object file, we use a PAM module named `pam_python` [115] that module runs the Python interpreter to allow running Python code for writing Python PAM module. `pam_python` module provides a faster method to implement PAM module, and supports importing other Python modules such as pytorch, Tensorflow, openCV, etc. As a result, many facial recognition authentication PAM module implementations use `pam_python` package. In order to

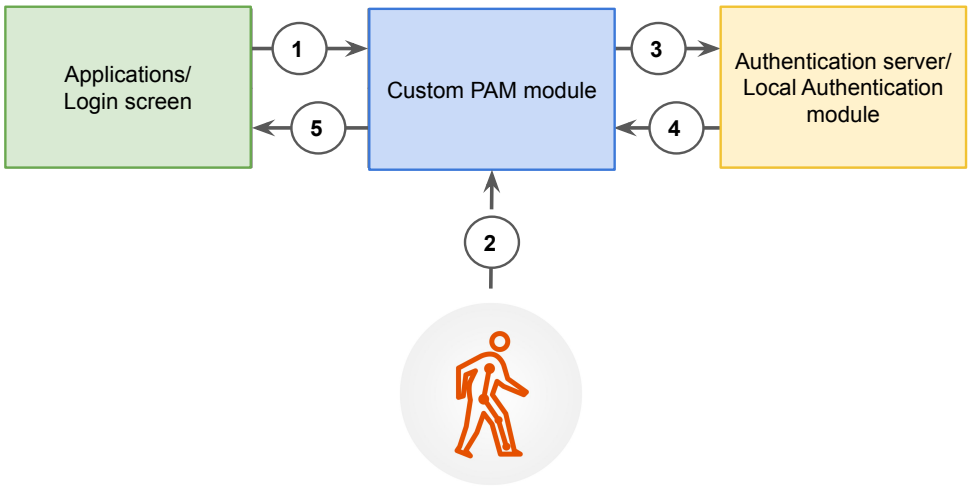


Figure 4.14: Custom PAM module process

perform authentication, we implement `pam_sm_authenticate()` method provided by `pam_python` to wrap the actual method in PAM library. We create a socket listener inside the method for receiving credential data and a connection to authentication server for user identification. Additionally, we can easy run a identification model into this PAM module to authenticate users without authentication server connection when multiple platforms authentication does not require. The process of the PAM module shows in Figure 4.14, the process is

run following these steps:

1. Applications or login screen need authenticate a user and calls `pam_start()` to initialize a PAM transaction. If our custom PAM module is configured for the applications or login screen, then it must be called.
2. Module's socket receive data from the user.
3. PAM module sends received data from to authentication server or a local authentication module to identify the user.
4. Authentication server/local authentication module send the identification result back to custom PAM module.
5. PAM module sent the result back to requesting Applications or login screen.

4.2.5 Module Implementation: Google Chrome Extension

We build a Google Chrome extension to support autologin for websites that uses password-based authentication technique. When a website supports login using password-based authentication, they often has a login page. A login page has a form containing username/email filed, password field and a submit button. We can login by automatically filling the form (autofill) and click the button, so if we can autofill the form and simulate a button click event, we can login automatically.

An implementation for a Google Chrome Extension uses web development technologies such as HTML, CSS and JavaScript [116]. An Chrome Extension project contains a manifest and source code. The manifest file is JSON (JavaScript Object Notation) format file for specifying characteristics of the extension and source code to implement extensions' functionality.

Our extension contains 4 JavaScript files:

- `background.js`: this script executes as a service worker - a background process. The process use to call script in content scripts to interact with web-

site's content.

- `urlsupport.js`: this process verifies an url is supported. If that url is supported, the process returns names of fields and button of this url's login form, these information can be used to access HTML elements in JavaScript code through Document Object Model (DOM) tree .
- `getprofile.js`: this process gets an account corresponding to a profile and current url.
- `autofill.js`: this content script uses returned information from `urlsupport.js` process and `getprofile.js` process to access HTML elements, fill information and simulate button click event.

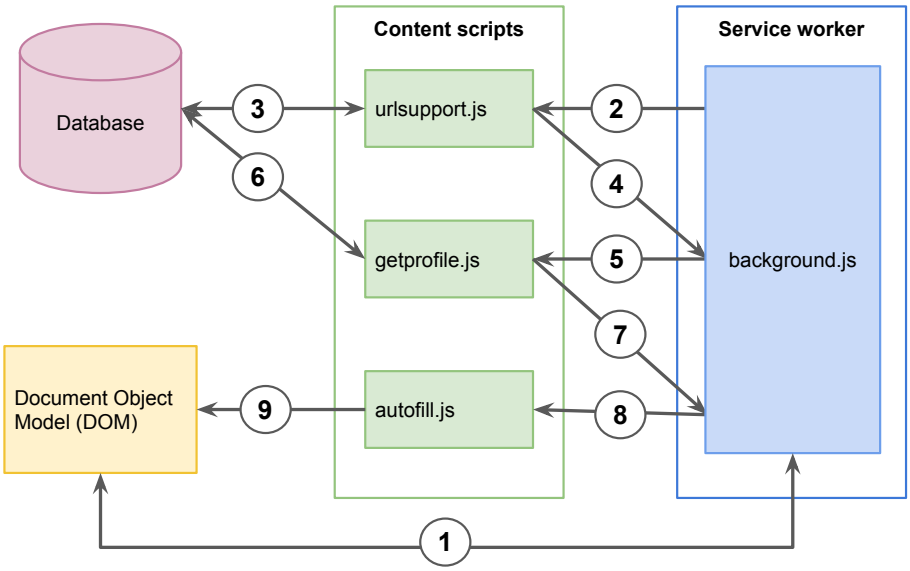


Figure 4.15: Extension execution process

The process of the extension:

1. Service worker `background.js` requests active tab url and execute `urlsupport.js`.
2. `urlsupport.js` uses the url to query in the database. If the url is found in the database means it is supported. Information that is received from the database is packaged and send within a message to service worker.
3. `background.js` after receives the message from `urlsupport.js` will call `getprofile.js` script.
4. `getprofile.js` script queries the current user data from the database. The current user can reuse recent profile from Windows logon or PAM authentication step instead of performing another authentication. User data of the current url is sent to `background.js`.
5. `background.js` receives the message from `getprofile.js` process data from `getprofile.js` and `urlsupport.js` and execute `autofill.js` script.
6. `autofill.js` uses information from previous step to access the login form using DOM. After these elements are received, profile information is used to fill in corresponding fields and create a submit event.

4.2.6 Module Implementation: Authentication server

The authentication server helps query data in database and provide an interface to interact with credential providers and authentication modules that are introduced in previous subsections.

The server contains 2 components. The first component is a Python application that sends credential information from credential provider to user identification module and provide communicate to database service. While the second components is a REST API that supports database operations and provides https protocols for security connection. We implement the second component using Nodejs [117] - an opensource JavaScript backend runtime environment. The following packages and libraries are used:

- **Express** [118]: a lightweight framework to create web application or application programming interfaces.
- **Mongoose** [119]: a package helps Nodejs applications to use MongoDB. This package support object modeling and asynchronous.
- **cors** [120]: a middleware that can be used with Express to enable Cross-Origin Resource Sharing (CORS) - an HTTP-header mechanism which enables controlled access to resources located outside of a given domain [121].
- **CryptoJS** [122]: a package contains secure cryptographic algorithm implementations. We use for encrypting usernames, passwords and other credential information before store them in database.
- **jsonwebtoken** [123]: implementation of Json Web Token (RFC7519) [124]. Enable JWTToken credential to access URLs that require. We use token credential to guarantee only User Identification Module can request for credential data on database.

By separating into 2 parts, the Python component can be either deployed within the REST API as a part of authentication server or as a system service on operating system. To deploy this component as a system service, we can register the Python component as a system service using systemd [125] - the basic service manager on Linux systems or NSSM (the Non-Sucking Service Manager) - third party service helper [126] on Windows systems. The REST API component can be placed on users' local machine or can be deployed as a server that can be accessed by multiple machines to share credential information among them.

4.2.7 Module Implementation: Database Design

To quickly implement the Nodejs server, we use MongoDB is a cross platform document-oriented database to be a credential database. This database manager has well support package that is Mongoose. A database diagram shows in Figure 4.16.

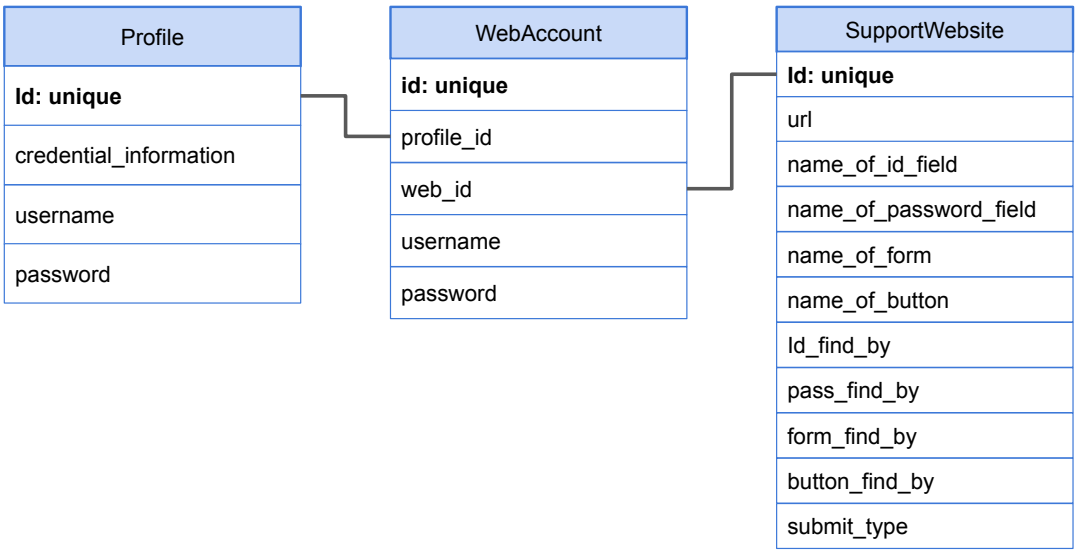


Figure 4.16: Database diagram

The database is designed to support multiple platform, so it stores information that supports 3 platforms we concern and other platforms in future. The database contains 3 schemas which maps to MongoDB as 3 collections:

- `Profile`: for each profile is stored, an unique `id` that is a random unique 256 bytes string. The next 3 values are optional, they can be used depend on the way we implement the user identification module. For example, we can store `credential_information` which is an object value as a token string. If the platform does not support custom authentication techniques and support password-based technique only, we can store `username` and `password` , and pass them to the platform's build-in authentication after successfully authenticate using the user identification module. We keep in mind that the stored username and password are also encrypted.
- `WebAccount`: an user might need authenticate one time, then they can use autofill feature of the extension we propose. This collection stores credential information of web accounts linking with user profiles.
- `SupportWebsite`: this collection stores urls of login pages supported by the extension. This collection requires augmentation for wider range of supported websites. `name_of_*` attributes are used to determine the name of appropriate elements in DOM tree. Attributes that contains `*_find_by` specify corresponding components can be find by `name` or `id` through DOM tree, The datatype of these values are string type. The `submit_type` has a value of either true or false, if the value is false, then the form is submit by a button click event, else the form is submit by a submit event.

CHAPTER 5
CONCLUSION

This chapter presents the conclusions of our thesis. We review the results for each objective. Additionally, we propose sample usage scenarios that can be applied with our technique. In final section, we propose a short proposal of future work.

5.1 Main results

Off-the-shelf wearable devices can be made useful for smart environment interaction, so that devices such as smartwatches can be used as an input for computer systems or electronics systems. Additionally, these off-the-shelf devices mostly provide programming interfaces like Android SDK in order to create useful applications to increase usability for the devices. To create applications in user-space for smartwatches that provides more features; however, applications are limited due to resource access limitation for power saving purpose. To overcome the limitation from stock devices, we learn that off-the-shelf devices can be modified to achieve higher sensor sampling rate, the modification can be easier in some devices and some devices would cost more effort to do so. These modifications can make inertial sensors of smartwatches reach the highest rate possible, yet at the cost of consuming more power than usual. With higher information resolution, signals sampled at higher sampling rate embed richer information and can be used for understanding movements, activities, and gestures – even very subtle ones such as wrist movements, finger movements, etc.

In order to process and understand these signal sequences, we introduce an approach inspired on speech recognition domain, the approach can be applied for a single frame (short sequence) case and also understand a continuous sequence of activity. Given devices available to the access of the authors, a Moto360v2

with low sampling rate and a LG G Watch W100 with higher sample rate, we aim to perform experiment and deployment in both two kind of frequencies.

Due to the COVID-19 pandemic, unfortunately, the device with higher sampling rate that we aim to be used for collecting high sampling rate data and deploy our high sampling rate models cannot not arrive in time. Thus, such experiments can not be done within the scope of this thesis. Yet, it is definitely a part of future work for this continuing project.

The interaction techniques from our work can be used for interaction with not only computers but also various devices such as electronics (lambs, fans, televisions, etc). The main interaction scenarios we introduce in this thesis is authentication on computer platforms with smartwatches. The system we have implemented for the scenarios is simple and flexible that can be integrates on various platforms. To demonstrate, we learn to understand the characteristics of authentication system on popular platforms that are Windows, Linux and web-based platforms. Based on that knowledge, we integrate 3 modules in order to support the 3 platforms separately with using recommended ways to implement.

Due to the long-term quarantine as an objective condition, some significant tasks are still remaining by the requirement on devices and people such as high sample rate model deployment, build more scenarios for user study, and user study performing. These remaining tasks are added into our future work.

5.2 Samples of Usage Scenarios

The technique using smartwatches that we propose in the scope of this thesis can be used in scenarios as follow:

- Use as an authentication technique: movement-based password, gait-based authentication.

- Use as a remote control for electronics: lambs, fans, televisions, projectors, or other electronics systems.
- Use as an optional freehand input device for computer and other systems.
- Collect activities data, collected data can be used for activity tracking and life logging.
- User interfaces of existing applications can also be re-designed to support the technique.

Usage scenarios for high sampling rate sensor signals are not listed in this section. By increasing sampling rate of the devices, the information from inertial sensors can understand minor movements and further acoustic information that support a wider range usage scenarios and applications.

5.3 Future Work

Due to challenges occur during conducting this thesis, various experiments and usage scenarios are left to our future work. With the goal of proposing useful and enjoyable interactive modalities, we aim to continue evaluating and improving our signal processing approach for high sampling rate signals, as well as considering the performance in-deployment across different smart watch models. Our previous works on information retrieval using representation learning embeddings [127, 128, 129] also suggest a promising outlook in a smart authentication system that employ only natural and unobtrusive human interaction.

Following the optimistic outlook is our concerns to verify comfort and enjoyment of proposed techniques. We acknowledge that user studies need to be conducted to explore suitable of movements and gestures for different scenarios, as well as to assess the usability, comfort, and enjoyment of proposed interactions with respect to the users' perspective. We aim to explore many types of movements

and gestures, yet with an emphasis on finger movements, because they are more subtle, implicit, and entails less obstruction from our normal activities.

Finally, as the world faces the third wave of COVID and Vietnam faces the all-time-high death toll, we plan to a investigate an alternative COVID-relief solution right after our thesis defence. Particularly, we are looking to apply bioacoustics sensing to smartphones as an alternative COVID-19 screening technique based on the capability of sensing bioacoustics signal, i.e, vibrations propagated through the human body. Despite being subtle, these signals are much less prone to ambient noise and might be able to overcome challenges in audio-based screening efforts that are being investigated by many Vietnamese developers.

REFERENCES

[1] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. arXiv preprint arXiv:2106.11342, 2021.

[2] Awni Hannun. Sequence modeling with ctc. Distill, 2017. doi: 10.23915/distill.00008. https://distill.pub/2017/ctc.

[3] Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, Jocelyn Huang, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, and Yang Zhang. Quartznet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions. In ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 6124–6128, Barcelona, Spain, May 2020. IEEE. ISBN 978-1-5090-6631-5. doi: 10/ghbz2x. URL `https://ieeexplore.ieee.org/document/9053889/`.

[4] Mark Weiser. The Computer for the 21st Century. SCIENTIFIC AMERICAN, page 12, 1991.

[5] Gordon E Moore. Cramming more components onto integrated circuits. Proceedings of the IEEE, 86(1):82–85, 1998.

[6] Gregory D. Abowd. Beyond Weiser: From Ubiquitous to Collective Computing. Computer, 49(1):17–23, January 2016. ISSN 0018-9162. doi: 10/gf2khx. URL `http://ieeexplore.ieee.org/document/7383147/`.

[7] Silvia Liberata Ullo and G. R. Sinha. Advances in Smart Environment Monitoring Systems Using IoT and Sensors. Sensors, 20(11):3113, May 2020.

ISSN 1424-8220. doi: 10/gmc67s. URL https://www.mdpi.com/1424-8220/20/11/3113.

[8] Partha P Ray. Home health hub internet of things (h 3 iot): An architectural framework for monitoring health of elderly people. In 2014 International Conference on Science Engineering and Management Research (ICSEMR), pages 1–3. IEEE, 2014.

[9] Partha Pratim Ray. A survey on internet of things architectures. Journal of King Saud University-Computer and Information Sciences, 30(3):291–319, 2018.

[10] Rob Dunne, Tim Morris, and Simon Harper. A Survey of Ambient Intelligence. ACM Computing Surveys, 54(4):1–27, July 2021. ISSN 0360-0300, 1557-7341. doi: 10/gk55v5. URL https://dl.acm.org/doi/10.1145/3447242.

[11] Naeem Iqbal, Shabir Ahmad, Do Hyeun Kim, et al. Health monitoring system for elderly patients using intelligent task mapping mechanism in closed loop healthcare environment. Symmetry, 13(2):357, 2021.

[12] Xueyi Wang, Joshua Ellul, and George Azzopardi. Elderly fall detection systems: A literature survey. Frontiers in Robotics and AI, 7:71, 2020.

[13] Man credits this apple watch feature for helping save his father. https://www.cbsnews.com/news/apple-watch-saves-life-hard-fall-apple-watch-series-4-falling-emergency-bob-burdett/.

[14] Apple watch calls 911 as middletown man falls down cliff. https://newjersey.news12.com/apple-watch-calls-911-as-middletown-man-falls-down-cliff-41211528.

[15] Smartwatch saved toralv (norwegian). https://www.nrk.no/norge/smartklokken-reddet-toralv-lordag-natt-1.14412266.

[16] Brad A. Myers. A brief history of human-computer interaction technology. Interactions, 5(2):44–54, March 1998. ISSN 1072-5520, 1558-3449. doi: 10.1145/274430.274436. URL https://dl.acm.org/doi/10.1145/274430.274436.

[17] Gregory D. Abowd. What next, ubicomp?: celebrating an intellectual disappearing act. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12, page 31, Pittsburgh, Pennsylvania, 2012. ACM Press. ISBN 978-1-4503-1224-0. doi: 10/gj2qzj. URL http://dl.acm.org/citation.cfm?doid=2370216.2370222.

[18] Google Scholar Metrics. https://scholar.google.com/citations?view_op=top_venues, . Accessed: 2021-08.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs], May 2019. URL http://arxiv.org/abs/1810.04805. arXiv: 1810.04805.

[20] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A Pretrained Language Model for Scientific Text. arXiv:1903.10676 [cs], September 2019. URL http://arxiv.org/abs/1903.10676. arXiv: 1903.10676.

[21] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. arXiv:1909.11942 [cs], February 2020. URL http://arxiv.org/abs/1909.11942. arXiv: 1909.11942.

[22] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and

lighter. arXiv:1910.01108 [cs], February 2020. URL http://arxiv.org/abs/1910.01108. arXiv: 1910.01108.

[23] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised Pre-Training for Speech Recognition. In Interspeech 2019, pages 3465–3469. ISCA, September 2019. doi: 10/ghpk8x. URL http://www.isca-speech.org/archive/Interspeech_2019/abstracts/1873.html.

[24] Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations. In Proceedings of the 8th International Conference on Learning Representations, April 2020. URL https://iclr.cc/virtual_2020/poster_rylwJxrYDS.html.

[25] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. In Advances in Neural Information Processing Systems, volume 33, pages 12449–12460, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-Abstract.html.

[26] Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. Unsupervised Cross-lingual Representation Learning for Speech Recognition. arXiv:2006.13979 [cs, eess], December 2020. URL http://arxiv.org/abs/2006.13979. arXiv: 2006.13979.

[27] Tae Jin Park, Naoyuki Kanda, Dimitrios Dimitriadis, Kyu J. Han, Shinji Watanabe, and Shrikanth Narayanan. A Review of Speaker Diarization: Recent Advances with Deep Learning. arXiv:2101.09624 [cs, eess], January 2021. URL http://arxiv.org/abs/2101.09624. arXiv: 2101.09624.

[28] Xian Shi, Fan Yu, Yizhou Lu, Yuhao Liang, Qiangze Feng, Daliang Wang, Yanmin Qian, and Lei Xie. The accented english speech recognition challenge

2020: open datasets, tracks, baselines, results and methods. In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 6918–6922. IEEE, 2021.

[29] Siyuan Feng, Olya Kudina, Bence Mark Halpern, and Odette Scharenborg. Quantifying bias in automatic speech recognition. arXiv preprint arXiv:2103.15122, 2021.

[30] Josephine Lau, Benjamin Zimmerman, and Florian Schaub. Alexa, are you listening? privacy perceptions, concerns and privacy-seeking behaviors with smart speakers. Proceedings of the ACM on Human-Computer Interaction, 2(CSCW):1–31, 2018.

[31] Yue Huang, Borke Obada-Obieh, and Konstantin Beznosov. Amazon vs. my brother: How users of shared smart speakers perceive and cope with privacy risks. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, pages 1–13, 2020.

[32] Yongsen Ma, Gang Zhou, and Shuangquan Wang. WiFi Sensing with Channel State Information: A Survey. ACM Computing Surveys, 52(3): 1–36, July 2019. ISSN 0360-0300, 1557-7341. doi: 10.1145/3310194. URL https://dl.acm.org/doi/10.1145/3310194.

[33] Siamak Yousefi, Hirokazu Narui, Sankalp Dayal, Stefano Ermon, and Shahrokh Valaee. A Survey on Behavior Recognition Using WiFi Channel State Information. IEEE Communications Magazine, 55(10):98–104, October 2017. ISSN 0163-6804. doi: 10.1109/MCOM.2017.1700082. URL http://ieeexplore.ieee.org/document/8067693/.

[34] Fadel Adib, Hongzi Mao, Zachary Kabelac, Dina Katabi, and Robert C. Miller. Smart Homes that Monitor Breathing and Heart Rate. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing

Systems, pages 837–846, Seoul Republic of Korea, April 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10/ggsrdr. URL https://dl.acm.org/doi/10.1145/2702123.2702200.

[35] Mingmin Zhao, Fadel Adib, and Dina Katabi. Emotion recognition using wireless signals. In Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, pages 95–108, 2016.

[36] Tuan-Duy H. Nguyen and Huu-Nghia H. Nguyen. Towards a robust wifi-based fall detection with adversarial data augmentation. In 2020 54th Annual Conference on Information Sciences and Systems (CISS), pages 1–6, 2020. doi: 10.1109/CISS48834.2020.1570617398.

[37] E.O. Thorp. The invention of the first wearable computer. In Digest of Papers. Second International Symposium on Wearable Computers (Cat. No.98EX215), pages 4–8, Pittsburgh, PA, USA, 1998. IEEE Comput. Soc. ISBN 978-0-8186-9074-7. doi: 10/ffqq8z. URL http://ieeexplore.ieee.org/document/729523/.

[38] Sheikh M. A. Iqbal, Imadeldin Mahgoub, E Du, Mary Ann Leavitt, and Waseem Asghar. Advances in healthcare wearable devices. npj Flexible Electronics, 5(1):9, December 2021. ISSN 2397-4621. doi: 10/gjtq2w. URL http://www.nature.com/articles/s41528-021-00107-x.

[39] Mengjie Zhang, Rehan Saeed, Safwan Saeed, Stevan Stankovski, and Xiaoshuan Zhang. Wearable Technology and Applications: A Systematic Review. page 13.

[40] Aleksandr Ometov, Viktoriia Shubina, Lucie Klus, Justyna Skibińska, Salwa Saafi, Pavel Pascacio, Laura Flueratoru, Darwin Quezada Gaibor, Nadezhda Chukhno, Olga Chukhno, Asad Ali, Asma Channa, Ekaterina Svertoka,

Waleed Bin Qaim, Raúl Casanova-Marqués, Sylvia Holcer, Joaquín Torres-Sospedra, Sven Casteleyn, Giuseppe Ruggeri, Giuseppe Araniti, Radim Burget, Jiri Hosek, and Elena Simona Lohan. A Survey on Wearable Technology: History, State-of-the-Art and Current Challenges. Computer Networks, 193:108074, July 2021. ISSN 13891286. doi: 10/gjptzc. URL https://linkinghub.elsevier.com/retrieve/pii/S1389128621001651.

[41] Eduardo Teixeira, Hélder Fonseca, Florêncio Diniz-Sousa, Lucas Veras, Giorjines Boppre, José Oliveira, Diogo Pinto, Alberto Jorge Alves, Ana Barbosa, Romeu Mendes, and Inês Marques-Aleixo. Wearable Devices for Physical Activity and Healthcare Monitoring in Elderly People: A Critical Review. Geriatrics, 6(2):38, April 2021. ISSN 2308-3417. doi: 10/gmcqhw. URL https://www.mdpi.com/2308-3417/6/2/38.

[42] James D. Brandt, Harvey B. DuBiner, Robert Benza, Kenneth N. Sall, Gary A. Walker, Charles P. Semba, Donald Budenz, Douglas Day, Brian Flowers, Steven Lee, Quang Nguyen, and David Wirta. Long-term Safety and Efficacy of a Sustained-Release Bimatoprost Ocular Ring. Ophthalmology, 124(10):1565–1566, October 2017. ISSN 01616420. doi: 10.1016/j.ophtha.2017.04.022. URL https://linkinghub.elsevier.com/retrieve/pii/S0161642017303883.

[43] Jong Wook Kim, Jong Hyun Lim, Su Mee Moon, and Beakcheol Jang. Collecting Health Lifelog Data From Smartwatch Users in a Privacy-Preserving Manner. IEEE Transactions on Consumer Electronics, 65(3):369–378, August 2019. ISSN 0098-3063, 1558-4127. doi: 10.1109/TCE.2019.2924466. URL https://ieeexplore.ieee.org/document/8744248/.

[44] Shiqiang Liu, Junchang Zhang, Yuzhong Zhang, and Rong Zhu. A wearable motion capture device able to detect dynamic motion of human limbs. Nature

Communications, 11(1):5615, December 2020. ISSN 2041-1723. doi: 10/gmcqhv. URL http://www.nature.com/articles/s41467-020-19424-2.

[45] Matt Bower and Daniel Sturman. What are the educational affordances of wearable technologies? Computers & Education, 88:343–353, October 2015. ISSN 03601315. doi: 10/f7wnxw. URL https://linkinghub.elsevier.com/retrieve/pii/S036013151530018X.

[46] Shubham Garg and Pradumn Joshi. Integrated Wearable Police Module for Fine Management and Law Enforcement. In 2014 Texas Instruments India Educators' Conference (TIIEC), pages 138–143, Bangalore, India, 2014. IEEE. ISBN 978-1-4673-8922-8. doi: 10.1109/TIIEC.2014.031. URL http://ieeexplore.ieee.org/document/7899224/.

[47] Haruka Murakami, Ryoko Kawakami, Satoshi Nakae, Yosuke Yamada, Yoshio Nakata, Kazunori Ohkawara, Hiroyuki Sasai, Kazuko Ishikawa-Takata, Shigeho Tanaka, and Motohiko Miyachi. Accuracy of 12 wearable devices for estimating physical activity energy expenditure using a metabolic chamber and the doubly labeled water method: Validation study. JMIR Mhealth Uhealth, 7(8):e13938, Aug 2019. ISSN 2291-5222. doi: 10.2196/13938. URL https://mhealth.jmir.org/2019/8/e13938/.

[48] Kashif Saleem, Basit Shahzad, Mehmet A. Orgun, Jalal Al-Muhtadi, Joel J. P. C. Rodrigues, and Mohammed Zakariah. Design and deployment challenges in immersive and wearable technologies. Behaviour & Information Technology, 36(7):687–698, July 2017. ISSN 0144-929X, 1362-3001. doi: 10/gmcwk2. URL https://www.tandfonline.com/doi/full/10.1080/0144929X.2016.1275808.

[49] Neamah Al-Naffakh, Nathan Clarke, and Fudong Li. Continuous User Authentication Using Smartwatch Motion Sensor Data. In Nurit Gal-Oz

and Peter R. Lewis, editors, Trust Management XII, volume 528, pages 15–28. Springer International Publishing, Cham, 2018. ISBN 978-3-319-95275-8 978-3-319-95276-5. doi: 10.1007/978-3-319-95276-5_2. URL http://link.springer.com/10.1007/978-3-319-95276-5_2. Series Title: IFIP Advances in Information and Communication Technology.

[50] Krzysztof Pietroszek, Liudmila Tahai, James R. Wallace, and Edward Lank. Watchcasting: Freehand 3D interaction with off-the-shelf smartwatch. In 2017 IEEE Symposium on 3D User Interfaces (3DUI), pages 172–175, Los Angeles, CA, USA, 2017. IEEE. ISBN 978-1-5090-6716-9. doi: 10/gjbz54. URL http://ieeexplore.ieee.org/document/7893335/.

[51] Saisakul Chernbumroong, Anthony S Atkins, and Hongnian Yu. Activity classification using a single wrist-worn accelerometer. In 2011 5th International Conference on Software, Knowledge Information, Industrial Management and Applications (SKIMA) Proceedings, pages 1–6. IEEE, 2011.

[52] Philipp M Scholl and Kristof Van Laerhoven. A feasibility study of wrist-worn accelerometer based detection of smoking habits. In 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, pages 886–891. IEEE, 2012.

[53] Fernando Ginez Da Silva and Elisabete Galeazzo. Accelerometer based intelligent system for human movement recognition. In 5th IEEE International Workshop on Advances in Sensors and Interfaces IWASI, pages 20–24. IEEE, 2013.

[54] Farzin Dadashi, Arash Arami, Florent Crettenand, Gregoire P Millet, John Komar, Ludovic Seifert, and Kamiar Aminian. A hidden markov model of the breaststroke swimming temporal phases using wearable inertial measurement

units. In 2013 IEEE international conference on body sensor networks, pages 1–6. Ieee, 2013.

[55] Muhammad Shoaib, Stephan Bosch, Hans Scholten, Paul JM Havinga, and Ozlem Durmaz Incel. Towards detection of bad habits by fusing smartphone and smartwatch sensors. In 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), pages 591–596. IEEE, 2015.

[56] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. Mole: Motion leaks through smartwatch sensors. In Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, pages 155–166, 2015.

[57] Serkan Balli, Ensar Arif Sağbas, and T Hokimoto. The usage of statistical learning methods on wearable devices and a case study: activity recognition on smartwatches. Advances in statistical methodologies and their application to real problems, pages 259–277, 2017.

[58] Gary M. Weiss, Jessica L. Timko, Catherine M. Gallagher, Kenichi Yoneda, and Andrew J. Schreiber. Smartwatch-based activity recognition: A machine learning approach. In 2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), pages 426–429, Las Vegas, NV, USA, February 2016. IEEE. ISBN 978-1-5090-2455-1. doi: 10/gmcqh3. URL http://ieeexplore.ieee.org/document/7455925/.

[59] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart Devices are Different: Assessing and MitigatingMobile Sensing Heterogeneities for Activity Recognition. In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, pages 127–140, Seoul

South Korea, November 2015. ACM. ISBN 978-1-4503-3631-4. doi: 10/bcgb. URL https://dl.acm.org/doi/10.1145/2809695.2809718.

[60] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. Recognizing Detailed Human Context in the Wild from Smartphones and Smartwatches. IEEE Pervasive Computing, 16(4):62–74, October 2017. ISSN 1536-1268. doi: 10/gk7smc. URL http://ieeexplore.ieee.org/document/8090454/.

[61] Yonatan Vaizman, Katherine Ellis, Gert Lanckriet, and Nadir Weibel. ExtraSensory App: Data Collection In-the-Wild with Rich User Interface to Self-Report Behavior. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pages 1–12, Montreal QC Canada, April 2018. ACM. ISBN 978-1-4503-5620-6. doi: 10/ghnn6s. URL https://dl.acm.org/doi/10.1145/3173574.3174128.

[62] Yonatan Vaizman. Behavioral Context Recognition In the Wild. PhD thesis, UC San Diego, 2018.

[63] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing. In Proceedings of the 26th International Conference on World Wide Web, pages 351–360, Perth Australia, April 2017. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-4913-0. doi: 10/gfx3gp. URL https://dl.acm.org/doi/10.1145/3038912.3052577.

[64] Serkan Balli, Ensar Arif Sağbaş, and Musa Peker. Human activity recognition from smart watch sensor data using a hybrid of principal component analysis and random forest algorithm. Measurement and Control, 52 (1-2):37–45, January 2019. ISSN 0020-2940. doi: 10/ggft3r. URL http://journals.sagepub.com/doi/10.1177/0020294018813692.

[65] Samaher Al-Janabi and Ali Hamza Salman. Sensitive integration of multilevel optimization model in human activity recognition for smartphone and smartwatch applications. Big Data Mining and Analytics, 4 (2):124–138, June 2021. ISSN 2096-0654. doi: 10/gmd9t6. URL https://ieeexplore.ieee.org/document/9343922/.

[66] Florenc Demrozi, Cristian Turetta, and Graziano Pravadelli. B-HAR: an open-source baseline framework for in depth study of human activity recognition datasets and workflows. arXiv:2101.10870 [cs, eess], January 2021. URL http://arxiv.org/abs/2101.10870. arXiv: 2101.10870.

[67] Yujiao Hao, Rong Zheng, and Boyu Wang. Invariant Feature Learning for Sensor-based Human Activity Recognition. IEEE Transactions on Mobile Computing, pages 1–1, 2021. ISSN 1536-1233, 1558-0660, 2161-9875. doi: 10/gmhhrz. URL https://ieeexplore.ieee.org/document/9372813/.

[68] Satya P. Singh, Madan Kumar Sharma, Aime Lay-Ekuakille, Deepak Gangwar, and Sukrit Gupta. Deep ConvLSTM With Self-Attention for Human Activity Decoding Using Wearable Sensors. IEEE Sensors Journal, 21(6):8575–8582, March 2021. ISSN 1530-437X, 1558-1748, 2379-9153. doi: 10/gjgc7w. URL https://ieeexplore.ieee.org/document/9296308/.

[69] Ghanapriya Singh, Mahesh Chowdhary, Arun Kumar, and Rajendar Bahl. A Personalized Classifier for Human Motion Activities With Semi-Supervised Learning. IEEE Transactions on Consumer Electronics, 66(4):346–355, November 2020. ISSN 0098-3063, 1558-4127. doi: 10/gmfcnn. URL https://ieeexplore.ieee.org/document/9249433/.

[70] Sakorn Mekruksavanich and Anuchit Jitpattanakul. Deep Convolutional Neural Network with RNNs for Complex Activity Recognition Using Wrist-Worn Wearable Sensor Data. Electronics, 10(14):1685, July 2021. ISSN

2079-9292. doi: 10/gmd9fg. URL https://www.mdpi.com/2079-9292/10/14/1685.

[71] Bolu Oluwalade, Sunil Neela, Judy Wawira, Tobiloba Adejumo, and Saptarshi Purkayastha. Human activity recognition using deep learning models on smartphones and smartwatches sensor data. In Proceedings of the 14th International Joint Conference on Biomedical Engineering Systems and Technologies - HEALTHINF,, pages 645–650. INSTICC, SciTePress, 2021. ISBN 978-989-758-490-9. doi: 10.5220/0010325906450650.

[72] Gierad Laput, Chouchang Yang, Robert Xiao, Alanson Sample, and Chris Harrison. EM-Sense: Touch Recognition of Uninstrumented, Electrical and Electromechanical Objects. In Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, pages 157–166, Charlotte NC USA, November 2015. ACM. ISBN 978-1-4503-3779-3. doi: 10/gmc67p. URL https://dl.acm.org/doi/10.1145/2807442.2807481.

[73] Gierad Laput, Robert Xiao, and Chris Harrison. ViBand: High-Fidelity Bio-Acoustic Sensing Using Commodity Smartwatch Accelerometers. In Proceedings of the 29th Annual Symposium on User Interface Software and Technology, pages 321–333, Tokyo Japan, October 2016. ACM. ISBN 978-1-4503-4189-9. doi: 10/gk75f5. URL https://dl.acm.org/doi/10.1145/2984511.2984582.

[74] Gierad Laput and Chris Harrison. Sensing Fine-Grained Hand Activity with Smartwatches. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pages 1–13, Glasgow Scotland Uk, May 2019. ACM. ISBN 978-1-4503-5970-2. doi: 10/ggft3w. URL https://dl.acm.org/doi/10.1145/3290605.3300568.

[75] Liang-Hong Wu, Liang-Chuan Wu, and Shou-Chi Chang. Explor-

ing consumers' intention to accept smartwatch. Computers in Human Behavior, 64:383–392, November 2016. ISSN 07475632. doi: 10.1016/j.chb.2016.07.005. URL https://linkinghub.elsevier.com/retrieve/pii/S0747563216304940.

[76] Sensor stack. https://source.android.com/devices/sensors/sensor-stack. Accessed: 2021-04-01.

[77] kernel-msm - Motorola Android Linux Kernel source repository. https://github.com/MotorolaMobilityLLC/kernel-msm. Accessed: 2021-03-25.

[78] Ziwei Zhu. Real-time gesture recognition demo with Android Wear device (moto360)[Software]. https://github.com/Zziwei/AndroidWear_Gesture_Recognition. Accessed: 2021-05-20.

[79] Gary M. Weiss, Kenichi Yoneda, and Thaier Hayajneh. Smartphone and Smartwatch-Based Biometrics Using Activities of Daily Living. IEEE Access, 7:133190–133202, 2019. ISSN 2169-3536. doi: 10/gmd86p. URL https://ieeexplore.ieee.org/document/8835065/.

[80] Yonatan Vaizman, Nadir Weibel, and Gert Lanckriet. Context Recognition In-the-Wild: Unified Model for Multi-Modal Sensors and Multi-Label Classification. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 1(4):1–22, January 2018. ISSN 2474-9567. doi: 10/gmhnw8. URL https://dl.acm.org/doi/10.1145/3161192.

[81] Zhendong Zhuang and Yang Xue. Sport-Related Human Activity Detection and Recognition Using a Smartwatch. Sensors, 19(22):5001, November 2019. ISSN 1424-8220. doi: 10/ggdg4m. URL https://www.mdpi.com/1424-8220/19/22/5001.

[82] James W Cooley and John W Tukey. An algorithm for the machine calcula-

tion of complex fourier series. Mathematics of computation, 19(90):297–301, 1965.

[83] Dennis Gabor. Theory of communication. part 1: The analysis of information. Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering, 93(26):429–441, 1946.

[84] Jonathan Allen. Short term spectral analysis, synthesis, and modification by discrete fourier transform. IEEE Transactions on Acoustics, Speech, and Signal Processing, 25(3):235–238, 1977.

[85] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. Advances in neural information processing systems, 2, 1989.

[86] F. Mamalet and Christophe Garcia. Simplifying convnets for fast learning. In ICANN, 2012.

[87] Francois Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1800–1807, Honolulu, HI, July 2017. IEEE. ISBN 978-1-5386-0457-1. doi: 10/gfxgtm. URL http://ieeexplore.ieee.org/document/8099678/.

[88] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke,

Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[89] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[90] François Chollet et al. Keras. `https://keras.io`, 2015.

[91] Awni Hannun, Ann Lee, Qiantong Xu, and Ronan Collobert. Sequence-to-Sequence Speech Recognition with Time-Depth Separable Convolutions. In Interspeech 2019, pages 3785–3789. ISCA, September 2019. doi: 10/gj9trd. URL `http://www.isca-speech.org/archive/Interspeech_2019/abstracts/2460.html`.

[92] Claude Elwood Shannon. Communication in the presence of noise. Proceedings of the IRE, 37(1):10–21, 1949.

[93] Oleksii Kuchaiev, Jason Li, Huyen Nguyen, Oleksii Hrinchuk, Ryan Leary, Boris Ginsburg, Samuel Kriman, Stanislav Beliaev, Vitaly Lavrukhin, Jack Cook, et al. Nemo: a toolkit for building ai applications using neural modules. arXiv preprint arXiv:1909.09577, 2019.

[94] Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M. Cohen, Huyen Nguyen, and Ravi Teja Gadde. Jasper: An End-to-End Convolutional Neural Acoustic Model. In Interspeech 2019, pages 71–75. ISCA, September 2019. doi: 10/gj765j. URL http://www.isca-speech.org/archive/Interspeech_2019/abstracts/1819.html.

[95] Min-Cheol Kwon, Hanjong You, Jeongung Kim, and Sunwoong Choi. Classification of Various Daily Activities using Convolution Neural Network and Smartwatch. In 2018 IEEE International Conference on Big Data (Big Data), pages 4948–4951, Seattle, WA, USA, December 2018. IEEE. ISBN 978-1-5386-5035-6. doi: 10/gj2q6w. URL https://ieeexplore.ieee.org/document/8621893/.

[96] Kenichi Yoneda and Gary M. Weiss. Mobile sensor-based biometrics using common daily activities. In 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), pages 584–590, New York, NY, October 2017. IEEE. ISBN 978-1-5386-1104-3. doi: 10/gmd86q. URL http://ieeexplore.ieee.org/document/8249001/.

[97] Isibor Kennedy Ihianle, Augustine O. Nwajana, Solomon Henry Ebenuwa, Richard I. Otuka, Kayode Owa, and Mobolaji O. Orisatoki. A Deep Learning Approach for Human Activities Recognition From Multimodal Sensing Devices. IEEE Access, 8:179028–179038, 2020. ISSN 2169-3536. doi: 10/gmd86s. URL https://ieeexplore.ieee.org/document/9209961/.

[98] Maria Cornacchia, Koray Ozcan, Yu Zheng, and Senem Velipasalar. A Survey on Activity Detection and Classification Using Wearable Sensors. IEEE Sensors Journal, 17(2):386–403, January 2017. ISSN 1530-437X, 1558-1748, 2379-9153. doi: 10/ggsp5b. URL http://ieeexplore.ieee.org/document/7742959/.

[99] Md Atiqur Rahman Ahad, Anindya Das Antar, and Masud Ahmed. Sensor-Based Benchmark Datasets: Comparison and Analysis. In IoT Sensor-Based Activity Recognition, volume 173, pages 95–121. Springer International Publishing, Cham, 2021. ISBN 978-3-030-51378-8 978-3-030-51379-5. doi: 10.1007/978-3-030-51379-5_6. URL http://link.springer.com/10.1007/978-3-030-51379-5_6. Series Title: Intelligent Systems Reference Library.

[100] Windows Interactive Logon Architecture. https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/ff404303(v=ws.10)?redirectedfrom=MSDN, . Accessed: 2021-07-25.

[101] Windows Authentication Overview. https://docs.microsoft.com/en-us/windows-server/security/windows-authentication/windows-authentication-overview, . Accessed: 2021-07-25.

[102] Albrecht Schmidt. Don't blame the user: toward means for usable and practical authentication. Interactions, 26(3):73–75, April 2019. ISSN 1072-5520, 1558-3449. doi: 10/gmg8bn. URL https://dl.acm.org/doi/10.1145/3320509.

[103] Dan Griffin. Create Custom Login Experiences with Credential Providers for Windows Vista. https://docs.microsoft.com/en-us/archive/msdn-magazine/2007/january/custom-login-experiences-credential-providers-in-windows-vista. Accessed: 2021-07-25.

[104] Authentication Packages. https://docs.microsoft.com/en-us/windows/win32/secauthn/authentication-packages. Accessed: 2021-07-25.

[105] Credentials Processes in Windows Authentication. https://docs.microsoft.com/en-us/previous-versions/windows/

it-pro/windows-server-2012-R2-and-2012/dn751047(v=ws.11)?redirectedfrom=MSDN. Accessed: 2021-07-25.

[106] Brian Ward. How Linux Works, 3rd Edition. No Starch Press, 2021. ISBN 978-1-71850-040-2. OCLC: 1242946803.

[107] Vipin Samar. UNIFIED LOGIN WITH PLUGGABLE AUTHENTICATION MODULES (PAM). http://www.opengroup.org/rfc/rfc86.0.html. Accessed: 2021-07-25.

[108] Vipin Samar. Unified Login with Pluggable Authentication Modules (PAM). In Proceedings of the 3rd ACM Conference on Computer and Communications Security, CCS '96, page 1–10, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897918290. doi: 10.1145/238168.238177. URL https://doi.org/10.1145/238168.238177.

[109] Introduction to the PAM Framework. https://docs.oracle.com/cd/E19120-01/open.solaris/819-2145/pam-01/index.html, . Accessed: 2021-07-25.

[110] Pluggable Authentication Modules (PAM). https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-en-4/ch-pam.html. Accessed: 2021-07-25.

[111] Introduction to the PAM framework. https://docs.oracle.com/cd/E19656-01/820-0386/aaqds/index.html, . Accessed: 2021-07-25.

[112] ICredentialProvider interface (credentialprovider.h). https://docs.microsoft.com/en-us/windows/win32/api/credentialprovider/nn-credentialprovider-icredentialprovider, . Accessed: 2021-07-25.

[113] ICredentialProviderCredential interface (credentialprovider.h). https://docs.microsoft.com/en-us/windows/win32/api/credentialprovider/

nn-credentialprovider-icredentialprovidercredential, . Accessed: 2021-07-25.

[114] ICredentialProviderCredentialEvents2 interface (credentialprovider.h). https://docs.microsoft.com/en-us/windows/win32/api/credentialprovider/nn-credentialprovider-icredentialprovidercredentialevents2. Accessed: 2021-08-01.

[115] Russell Stuart. pam_python - a PAM module runs Python interpreter [Software]. http://pam-python.sourceforge.net/doc/html/. Accessed: 2021-07-25.

[116] Extensions. https://developer.chrome.com/docs/extensions/, . Accessed: 2021-07-25.

[117] Node.js - a javascript runtime built on chrome's v8 javascript engine.[Software]. https://nodejs.org/en/. Accessed: 2021-07-25.

[118] Express - Fast, unopinionated, minimalist web framework for Node.js.[Software]. https://expressjs.com/. Accessed: 2021-07-25.

[119] Mongoose - Elegant MongoDB object modeling for node.js.[Software]. https://mongoosejs.com/. Accessed: 2021-07-25.

[120] CORS - a node.js package for providing a Connect/Express middleware.[Software]. https://github.com/expressjs/cors/, . Accessed: 2021-07-25.

[121] Cross-Origin Resource Sharing (cors). https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS/, . Accessed: 2021-07-25.

[122] CryptoJS - JavaScript implementations of standard and secure cryptographic algorithms.[Software]. https://cryptojs.gitbook.io/docs/. Accessed: 2021-07-25.

[123] jsonwebtoken - An implementation of JSON Web Tokens.[Software]. https://github.com/auth0/node-jsonwebtoken/. Accessed: 2021-07-25.

[124] Mike Jones. JSON Web Token (JWT). https://datatracker.ietf.org/doc/html/rfc7519/. Accessed: 2021-07-25.

[125] systemd -a suite of basic building blocks for a Linux system [Software]. https://systemd.io/. Accessed: 2021-08-01.

[126] NSSM - the Non-Sucking Service Manager [Software]. https://nssm.cc/. Accessed: 2021-8-01.

[127] Tuan-Duy H Nguyen, Huu-Nghia H Nguyen, and Hieu Dao. Recognizing families through images with pretrained encoder. In 2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020), pages 887–891. IEEE, 2020. doi: 10.1109/FG47880.2020.00130.

[128] Thuc Nguyen-Quang, Tuan-Duy H Nguyen, Thang-Long Nguyen-Ho, Anh-Kiet Duong, Nhat Hoang-Xuan, Vinh-Thuyen Nguyen-Truong, Hai-Dang Nguyen, and Minh-Triet Tran. Hcmus at mediaeval 2020: Image-text fusion for automatic news-images re-matching. In MediaEval 2020: Multimedia Benchmark Workshop 2020, 2020. URL http://ceur-ws.org/Vol-2882/paper73.pdf.

[129] Andrea Raffo, Ulderico Fugacci, Silvia Biasotti, Walter Rocchia, Yonghuai Liu, Ekpo Otu, Reyer Zwiggelaar, David Hunter, Evangelia I. Zacharaki, Eleftheria Psatha, Dimitrios Laskos, Gerasimos Arvanitis, Konstantinos Moustakas, Tunde Aderinwale, Charles Christoffer, Woong-Hee Shin, Daisuke Kihara, Andrea Giachetti, Huu-Nghia Nguyen, Tuan-Duy Nguyen, Vinh-Thuyen Nguyen-Truong, Danh Le-Thanh, Hai-Dang Nguyen, and Minh-Triet Tran. Shrec 2021: Retrieval and classification of protein surfaces equipped with physical and chemical properties.

Computers Graphics, 99:1–21, 2021. ISSN 0097-8493. doi: https://doi.org/10.1016/j.cag.2021.06.010. URL https://www.sciencedirect.com/science/article/pii/S0097849321001254.

LIST OF PUBLICATIONS

Prior works in smart environments are published in the following articles:

[1] Tuan-Duy H. Nguyen and Huu-Nghia H. Nguyen. Towards a robust wifi-based fall detection with adversarial data augmentation. In 2020 54th Annual Conference on Information Sciences and Systems(CISS), pages 1–6, 2020. doi: 10.1109/CISS48834.2020.1570617398

Other works in information retrieval using representation learning are published in the following articles:

[2] Tuan-Duy H Nguyen, Huu-Nghia H Nguyen, and Hieu Dao. Recognizing families through images with pretrained encoder. In 2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG2020), pages 887–891. IEEE, 2020. doi: 10.1109/FG47880.2020.00130.

[3] Thuc Nguyen-Quang, Tuan-Duy H Nguyen, Thang-Long Nguyen-Ho, Anh-Kiet Duong, Nhat Hoang-Xuan, Vinh-Thuyen Nguyen-Truong, Hai-DangNguyen, and Minh-Triet Tran. Hcmus at mediaeval 2020: Image-text fusion for automatic news-images re-matching. In MediaEval2020: Multimedia Benchmark Workshop 2020, 2020. URL http://ceur-ws.org/Vol-2882/paper73.pdf.

[4] Andrea Raffo, Ulderico Fugacci, Silvia Biasotti, Walter Rocchia,Yonghuai Liu, Ekpo Otu, Reyer Zwiggelaar, David Hunter, Evangelia I.Zacharaki, Eleftheria Psatha, Dimitrios Laskos, Gerasimos Arvanitis, Kon-stantinos Moustakas, Tunde Aderinwale, Charles Christoffer, Woong-Hee Shin, Daisuke Kihara, Andrea Giachetti,

Huu-Nghia Nguyen, Tuan-Duy Nguyen, Vinh-Thuyen Nguyen-Truong, Danh Le-Thanh, Hai-Dang Nguyen, and Minh-Triet Tran. Shrec 2021: Retrieval and classifica-tion of protein surfaces equipped with physical and chemical properties. Computers Graphics, 99:1–21, 2021. ISSN 0097-8493. doi: https://doi.org/10.1016/j.cag.2021.06.010. URL https://www.sciencedirect.com/science/article/pii/S0097849321001254