Sensing enhanced mobile computing with smart devices

Jiang, Shiqi

2017

Jiang, S. (2017). Sensing enhanced mobile computing with smart devices. Doctoral thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/73055>

<https://doi.org/10.32657/10356/73055>

NANYANG TECHNOLOGICAL UNIVERSITY

Sensing Enhanced Mobile Computing with Smart Devices

By

Shiqi Jiang

Supervisor: Dr. Mo Li

School of Computer Engineering

This thesis is submitted to Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

2017

Abstract

Today’s mobile devices not only serve as the key communication tools, but become important sensing and computing platforms as well. Sensing capacities promise the emergence of *sensing enhanced mobile computing on smart devices*, which leverages various on-board sensors, applies sophisticated machine learning techniques for intelligent reasonings according to the context inference of surrounding environments. In this report, we introduce two systems we built based on this idea, which are PDS, a Phantom Data Usage Detection System and Memento, an Emotion Driven Lifelogging System.

First we introduce PDS, a phantom data usage detection system. With the wide development of smartphones, mobile data usage has enjoyed rapid growth in recent years. Unfortunately many users are plagued with *Phantom Data Usage* (PDU), which refers to the unexpected mobile data usage that does not accord with user perception. We investigate real PDU issues and find the causes of PDU are not only the exceptions of applications, e.g., software bugs or malware, but also the user’s personalized misuse. Based on the observations that each user preserves specific data usage patterns under particular environmental context, we present PDS, which automatically detects whether the current data usage is consumed as expected.

Then we present Memento, an emotion driven lifelogging system on wearables. Due to the increasing popularity of mobile devices, the usage of lifelogging has been dramatically expanded. People collect their daily memorial moments and share with friends on the social network, which has been an emerging lifestyle. We see great potential of lifelogging applications along with rapid growth of recent wearable market, where more sensors are introduced to wearables, i.e., electroencephalogram (EEG) sensors, that can further sense the user’s mental activities, e.g., emotions. We present the design and implementation of Memento, an emotion driven lifelogging system on wearables. Memento integrates

EEG sensors with smart glasses. Since memorable moments usually coincides with the user's emotional changes, Memento leverages the knowledge from the brain-computer-interface (BCI) domain to analyze the EEG signals to infer emotions and automatically launch lifelogging based on that. Towards building Memento on COTS wearable devices, we study EEG signals in mobility cases and propose a multiple sensor fusion based approach to estimate signal quality. We present a customized two-phase emotion recognition architecture, considering both the affordability and efficiency of wearable-class devices. We also discuss the optimization framework to automatically choose and configure the suitable lifelogging method (video, audio or image) by analyzing the environment and system context. Finally our experimental evaluation shows that Memento is responsive, efficient and user-friendly on wearables.

Acknowledgments

First and foremost, I would like to sincerely thank my supervisor, Prof. Mo Li for his support and encouragement. I really appreciate his guidance with his patience, valuable experiences and insightful opinions.

I would like to thank Prof. Wentong Cai, Prof. Chengzheng Sun and all the members in Parallel & Distributed Computing Center (PDCC) and Computer Networks & Communications Lab (CNCL), for give us such a great research environment. I also highly appreciate the financial support from research scholarship program of NTU.

I would like to acknowledge the members and alumni from Wireless and Networked Distributed Sensing (WANDS) group: Dr. Yuanqing Zheng, Dr. Wan Du, Dr. Jiajue Ou, Dr. Yuxiao Hou, Dr. Yaxiong Xie, Chu Cao, Jansen Liando, Panrong Tong, Dr. Shuyu Shi. I would like give my special thanks to Dr. Pengfei Zhou, Dr. Zhenjiang Li and Dr. Zhidan Liu, for the help and collaboration on such exciting projects. It's so fantastic to have the opportunity to work majority of research in your facilities.

And finally, I must express my gratitude to my family for providing me with unfailing support and continuous encouragements throughout my entire life, always. Thanks so much.

Contents

Abstract	i
Acknowledgments	iii
List of Figures	vi
List of Tables	viii
1 Introduction	1
1.1 Detecting Phantom Data Usage on Smartphones	2
1.2 Building an Emotion Driven Lifelogging System with Wearables	5
1.3 Organization of This Report	9
2 Sensing Enhanced Mobile Computing	10
2.1 Sensors on smart devices	10
2.2 Techniques of context inference	13
2.3 Mobile sensing applications and systems	14
2.4 System implementation issues	16
3 Detecting Phantom Data Usage on Smartphones	18
3.1 Phantom Data Usage Issues	18
3.2 Context and Data Usage	21
3.3 System Design and Implementation	25
3.3.1 Information Collector	26
3.3.2 Data Usage Model	27
3.3.3 Decision Engine	28
3.4 Evaluation	31
3.5 Related Work	35
3.6 Discussions	37

4	Building An Emotion-driven Lifelogging System on Wearables	39
4.1	Lifelogging and Emotions	39
4.2	Memento System Design	41
4.2.1	Signal preprocessing	42
4.2.2	Two phase emotion recognition	47
4.2.3	Lifelogging Engine	51
4.2.4	Computation offloading	56
4.3	Implementation	57
4.4	Evaluation	60
4.4.1	Dataset	60
4.4.2	Emotion Recognition Performance	61
4.4.3	Lifelogging performance	64
4.4.4	Energy consumption	66
4.5	Related Work	68
4.6	Discussions	70
5	Conclusion and Future Work	72
	Appendix	75
A	Author's Publications	76

List of Figures

1.1	Emotion-driven lifelogging system.	6
3.1	Root case distribution of the data usage issues in the survey.	19
3.2	Download rankings of applications in unknown cases.	20
3.3	The temporal data usage patterns of two users.	21
3.4	The spatial data usage patterns for different applications.	23
3.5	The data usage patterns in different movement statuses.	24
3.6	Phantom detection system overview	25
3.7	The similarity of 20 operations on 6 applications.	29
3.8	Detection results of PDS.	31
3.9	The similarity distribution of the PDU cases confirmed by users.	32
3.10	The growth of data usage history.	34
3.11	Instant battery consumption of PDS.	35
4.1	System architecture of Memento	41
4.2	Frequency distribution of raw EEG signals.	43
4.3	Detecting blinks from raw EEG signals using kernel correlation	44
4.4	Band power correlation across EEG electrodes when drifting.	45
4.5	Resultant accelerometer readings under different usage cases.	46
4.6	Using the arousal trends as the trigger to perform lifelogging	51
4.7	The hardware and software implementation of Memento prototype.	58
4.8	Emotion recognition on the data of DEAP and collected by Memento.	62
4.9	How the emotion-driven lifelogs fit the users' expectations.	65
4.10	The coverage ratio of the emotion-driven lifelogs against the clips selected by users.	65

4.11 Lifelog quality evaluation using face count.	67
4.12 Energy consumption of each component.	68

List of Tables

3.1	The trace samples collected in the information collector.	26
3.2	Breakdown of detected PDU cases.	31
3.3	Energy and memory consumptions of PDS in different usage states. . . .	33
4.1	Energy profiles of different lifelogging methods on Google Glass	52
4.2	Pre-defined lifelogging configurations in Memento	56
4.3	Breakdown emotion recognition performance.	63

Chapter 1

Introduction

Forty-four years after the first mobile phone was demonstrated, mobile devices have revolutionized the lifestyle of millions of individuals. According to [1], the number of mobile phone users is about to reach 4.77 billion in 2017. Nowadays mobile devices including smartphones, tablets and smart wearables serve for social network service (SNS), travel planning, health-care, e-business, online shopping, mobile payments and numerous other functionalities are being invented. Despite more and more powerful processing capacities supported by multi-core CPU and gigabytes of memory, current mobile devices are equipped with a rich set of sensing functions: a smartphone can record audio, take photos, videos and sense geo-location, gravity, air pressure and device attitude, etc. In addition, new types of sensors are being incorporated with mobile devices to extend the capacities. Today's mobile devices not only serve as the communication tools, but become important sensing and computing platforms as well.

Such capacities promise the emergence of *sensing enhanced mobile computing with smart devices*, which leverages various on-board sensors, applies machine learning techniques for intelligent reasoning based on the context inference. Smart devices such as smartphones, wearables are usually carried by users and exposed to the ambient environment. Thus they could be perceptive devices to bridge the gap among the user, the environment and the device.

By understanding the context of surrounding environments, a mobile sensing system could make intelligent actions. For instance, the screen brightness could be automatically adjusted according to the current light condition sensed by the ambient light sensor. Smart devices are also capable to extend human senses. With rich sensors on smart devices, human activities are recognized and recorded, which are valuable for various services e.g. health-caring and personal assistant. Finally these devices are connected with each other via Internet. The data and knowledge from individuals could be shared, gathered and used to solve the problem on a large scale such as urban traffic estimation, pollution monitoring and so on.

The idea of sensing enhanced mobile computing is being exploited to optimize many existing services and bringing out innovative applications and systems. Around this key idea, in this thesis we contribute our experience of designing and building two real sensing enhanced mobile computing systems, which are PDS, a Phantom Data Usage Detection Systems on smartphones and Memento, an Emotion-driven Lifelogging System on wearables. In PDS, we make attempts to detect Phantom Data Usage which doesn't accord with users' perception by sensing and analyzing context information. In Memento, with the help of various sensors and sensing techniques on mobile, we introduce emotion-sensing into lifelogging systems and build an emotion-driven lifelogging system on wearable-class platforms while considering efficiency, affordability as well as user-friendly.

1.1 Detecting Phantom Data Usage on Smartphones

Nowadays smartphones serve as not only phones but also the primary social communication and entertainment tools involving the email services, Social Network Services (SNS), mobile games, etc. The number of third-party smartphone applications enjoyed years of explosive growth and many apps incur data traffic through the cellular or Wi-Fi network.

Global mobile data traffic grew 63% in 2016 and has reached 7.2EB per month. By 2021, the global mobile data traffic will increase sevenfold [2]. Due to the rapid technology evolution and wide coverage of cellular networks, e.g., 3G and LTE, most users buy the cellular network data plan for mobile data usage. Although the per unit data cost is becoming cheaper, users are paying more money as the total data consumption grows. While more money is spent, users feel more and more confused about their mobile data bills. According to a recent survey [3], most users believe only 50% of their data bills are indeed consumed by themselves and the rest are inexplicable. Similar complaints arise in the USA [4] that a number of users find that mysterious charges occur which is not noticed in time. Many users are paying for such unclear and inexplicable data usage, which is named *Phantom Data Usage* (PDU) in this thesis.

In the users' point of view, PDU refers to the exceptional data usage which does not accord with their normal perception. PDU is a real and emerging problem. Rather than being unusual and isolated cases, many PDU issues affect a significant number of users. When we randomly sample 100 mobile data usage issues, for 82 issues users fail to determine the causes, regarding them as PDU. Only 18 issues are due to the certain causes, e.g., software bugs, malicious features of applications, or the normal heavy usage of users. In this thesis, by analyzing about four hundred real PDU issues, we find that the users' personalized misuse is a major cause of PDU. The misuse of applications is a collateral consequence of the development of smartphone industry. On one hand, the smartphone gains its popularity and smartphone users widely spread across different age groups and professions. On the other hand, smartphone applications are mostly developed by third-party or individual developers with different level of programming skills and design considerations, leading to a wide variety of applications. As a result, users might be confused across different applications, which causes the misuse problems which were not noticed by most users. Therefore it would be beneficial if smartphones

can automatically tell if some applications are being misused by the certain user, in other words, detecting PDU.

Existing data usage monitors on smartphones can derive data usage information. Some third-party data monitoring applications like My Data Manager [5] can record the detailed data usage history for each app. However such tools cannot differentiate normal data usage and PDU. Large data usage consumption does not always indicate PDU. The data usage information need to be further analyzed to validate whether it accords with users' usage habits. There have been some approaches [6, 7] proposed in the literature for abnormal network traffic detection and they can be potentially adapted for PDU detection. However most of them are developed to detect the network intrusion on Internet. Directly applying those methods for PDU detection on smartphones is inappropriate due to the high uncertainty and the diversity of PDU issues. Firstly the root causes of PDU are highly different from that of the Internet. Traditional methods are mostly learning-based while the PDU characteristics are unknown. Secondly the situations on smartphones are more complicated and the usage behaviors are highly personalized, which makes it difficult to extract patterns in generic.

To the best of our knowledge, PDU issues on smartphones have been rarely studied. Inferring the occurrence of PDU from a large number of applications is challenging. First, a high data consuming app might not necessarily cause PDU. To differentiate normal data usage and PDU, we need to study how the data usage is supposed to be consumed, which is different across different applications. The basic observation is that the environmental context affects the way users use mobile applications. For instance, a Facebook user might be used to watching video at home but only browsing texts and images in office. By analyzing the traces from 7 volunteers for three weeks, we reveal that the user context (i.e., time, locations and movements) affects not only the application usage (which application is used) but also the data usage patterns (how the application is

used). Second, the context-related data usage patterns are highly personalized depending on the user’s usage patterns. In this paper, we learn each user’s usage habits from the historical data usage records, extract their data usage profiles and build a data usage model for each user.

We present PDS, a system tool to detect PDU on smartphones. PDS records the detailed data usage information of each application, the user context information (e.g., movements, time and locations) and relevant system events. By analyzing such information, PDS detects whether the current data usage accords with users’ usage habits. Specifically, PDS profiles the data usage in different circumstances. It then compares the current traffic with that in the most similar circumstance to dig out PDU. To make it practical, PDS meets several criteria including low monitoring overhead, high detection accuracy and flexibility to different applications and users.

We evaluate PDS on real users’ devices. We solicit 7 Android users with different smartphone experience and usage habits. In three weeks PDS detects 63 PDU cases in total and 45 of them (72%) are confirmed by users themselves. The system overhead is also carefully examined.

1.2 Building an Emotion Driven Lifelogging System with Wearables

Lifelogging digitizes human daily lives, which was widely adopted in the therapy for a series of neurodegenerative diseases using dedicated devices [8–10]. Due to the increasing number of smartphones, the use of lifelogging has been dramatically expanded. It has already become an emerging lifestyle for people to collect their memorial moments and share with friends. Lifelogging applications or services [11, 12] on smartphones are able to log users’ lives in various forms such as texts, images, audio clips, videos, etc.

Figure 1.1: Emotion-driven lifelogging system.

Recent wearable market has been rapidly growing in terms of both technology advances and penetration. Wearable devices, especially smart glasses are equipped with the first-person camera, microphone, and rich on-board sensors. They are always carried by the users and exposed to the ambient environment. These characteristics make us believe that wearables have the potential to provide better lifelogging services. It is, however non-trivial to build a lifelogging system on wearables. Rather than being directly migrated from the existing systems, lifelogging on wearable prompts new questions. In this paper, we make attempts to examine the questions we think important when designing an user-friendly, affordable, and efficient lifelogging system on wearables.

- (i) *The burden of human intervention.* Most existing lifelogging systems on mobile devices need manual operations [11,12], which require excessive human interventions. Though various user interaction methods are introduced on wearables, including the gesture, voice and even wink controls, the intervention overhead causes non-negligible burdens and may impair users' willingness to conduct lifelogging. Due to the lag of manual operations, many brief but valuable moments could be easily missed as well. Therefore, an automatic lifelogging system without requiring user intervention is expected.

- (ii) *The affordability of lifelogging on wearables.* Most passive lifelogging systems rely on dedicated devices [13, 14], which is, however, not appropriate for most COTS wearables devices. On the one hand, due to the limited size, the energy resources of wearables are bounded, which cannot afford continuous sensing. On the other hand, passive lifelogging usually leads to massive irrelevant lifelogs. Therefore the affordability of lifelogging systems on wearables should be carefully considered.
- (iii) *Inside context awareness.* Recent years, the commercial Electroencephalography (EEG) [15] headband comes into the market. EEG is a measure of brain waves, which could be a reflection of the environment and generally results in physical and psychological changes of people, influencing behaviors. EEG offers the opportunities for us to understand the user mental states. In this paper, we attempt to introduce EEG sensors in order to dig more information from the inside world of users and improve the lifelogging accuracy based on that.

We investigate the questions above through the design and implementation of an emotion driven lifelogging system on wearables, Memento. The goal of Memento is to provide an automatic seamless lifelogging experience on COTS wearable devices, while considering the affordability, user-friendly and efficiency in practice. Shown in Fig. 1.1, Memento integrates EEG electrodes with smart glasses. By leveraging the Brain-Computer-Interface (BCI) domain knowledge, Memento derives emotions from EEG signals and launches the lifelogging process automatically based on that. The implementation of such a system, however, entails several crucial challenges:

First, signals collected from wearable EEG sensors are not always reliable — EEG signals are usually mixed with various external noises, from the electric appliances and the interfering signals from the muscle activities such as the blink, jaw or heart beat. On the other hand, the electrodes of wearable EEG are not fixed on the scalp (see Fig. 1.1

as a reference). The movements of the user could thus cause a drift of the electrodes, which will in turn lead to a unpredicted change of the harvested EEG signals. In this paper we carefully study the frequency features of major noises and apply an efficient filter to remove the signals due to non-brain activities. Meanwhile we propose a sensor fusion based approach with the help of IMU readings to detect electrodes drifting and assist estimating signal quality.

Second, the high computational complexity of existing emotion recognition algorithms prohibits them to be adopted directly on wearable devices. To address such an issue, we split the emotion recognition into two phases and install them on smart glasses and private clouds (personal PC or smartphone). Instead of extracting emotions in real time, we trigger lifelogging by analyzing the emotional changes. The exact emotion information are obtained offline and tagged on the lifelogs.

Third, the environment impacts the lifelogging qualities. For instance, poor light conditions and high noise levels might influence the qualities of image lifelogs and audio lifelogs, respectively. Meanwhile lifelogging methods, image, audio and video with various configurations (e.g. resolution and frame rate) have different energy profiles. To best balance the logging quality and the energy consumption, we propose the optimization framework, which picks and configures the suitable lifelogging method with the analysis of both the environment and system context.

In summary, our contributions of this work include:

- The proposal of a new and natural way to automatically trigger lifelogging. We introduce the use of EEG to the lifelogging system on wearables.
- A series of techniques to integrate EEG electrodes with wearables. We present our signal processing and two phase emotion recognition design, in order to meet the computation and energy constraints of wearable-class hardware. We also discuss the

optimization framework that selects and configures the suitable lifelogging method to best balance the lifelog qualities and energy consumptions.

- The full design and implementation of Memento, an emotion driven lifelogging system on smart glasses. We show the capacities of Memento are efficient in support of emotion tagging and lifelogging workload. The experimental evaluation indicates that Memento is able to provide satisfactory battery life and user experience.

1.3 Organization of This Report

We organize this report as follows: first we present the literature review in Chapter 2. In Chapter 3, we introduce our experience of building PDS, a phantom data usage detection system on smartphones. Next we propose Memento, an emotion driven lifelogging system on wearables in Chapter 4. Finally we conclude the report and introduce our future work in Chapter 5.

Chapter 2

Sensing Enhanced Mobile Computing

In this chapter, we review the literature of sensing enhanced mobile computing on smart devices. A mobile sensing system generally would collect sensing data from various on-board sensors. Then the sensed data are processed and analyzed by sophisticated signal processing approaches and machine learning techniques. In turn the extracted context information can be used to either enhance existing services or provide new innovative systems. Therefore this chapter is organized as follows: first we survey different sensors on smart devices and their potential usage; then we discuss some representative techniques which are widely used for context inference; next we make a survey on typical sensing enhanced mobile computing applications and systems. Finally we review the practical issues when implementing such mobile sensing systems.

We also review the specific techniques and work related to PDS and Memento, which are discussed in detail in §3.5 and §4.5, respectively.

2.1 Sensors on smart devices

Equipped with a rich array of sensors and powerful processing units, a mobile sensing system usually is able to infer the context of the system itself or surrounding environments

by applying sophisticated machine learning algorithms. In turn intelligent actions or optimizations are made based on such context inference [16].

Sensors are the keys to make mobile devices not only communication tools but sensing and computation platforms as well. There is a set of sensors embedded on smartphones. For instance, on the recent LG Nexus 5X, we find a GPS, accelerometer, gyroscope, magnetic sensor, ambient light sensor, proximity sensor, barometer, as well as some components that can be used to sense, such as camera and microphone, etc.

On smartphones, GPS is usually used to obtain the geo-location and time information from GPS satellites. GPS is able to provide accurate geo-locations periodically without external reference. Therefore it is widely used and acts a key role in most of Location Based Services (LBS). However the performance of GPS heavily relies on the satellite signal quality. When devices are in indoor environments where GPS signals might be blocked, GPS works not very well. To address such problems, several indoor localization approaches are proposed based on either RF radio [17, 18] or sensor dead-reckoning [19–21]. By adding a directional antenna, [22] makes attempts to provide the indoor positioning service from direct GPS receiving as well. Energy consumption is another issue for GPS. In [23, 24], authors comprehensively discuss the energy and accuracy trade-off, especially for the continuous positioning scenarios. In [25], a novel rate-adaptive approach is proposed to provide accurate geo-location information while spending minimal energy.

Inertial Measurement Unit (IMU) sensors on mobile devices including accelerometer, digital compass and gyroscope are originally introduced to enhance conventional services such as video playing, photo capturing or localization. For instance, the accelerometer senses the direction of gravity and automatically changes the screen orientation based on that. Not only such IMU sensors are useful in improving user experience, but provides significant possibilities to sense information about people and surrounding environments

as well. By analyzing accelerometer data from smartphones, CenceMe [26] automatically recognizes different activities (e.g. still, walking, running or riding) of phone users. Recently years, more sophisticated algorithms are proposed to recognizes various types of user activities, including transportation classification [27], gesture recognition [28] and falling detection [29], etc. To better leverage IMU sensors, the characteristics of different on-board sensors are researched and sensor fusion algorithms are developed. Acube [30] fuses data from the accelerometer, gyroscope and digital compass, applies a novel calibration algorithm to derive the smartphone attitude.

Proximity sensor and ambient light sensor are introduced to allow smartphones perform the simple context recognitions associated with user experience. The proximity sensor, for instance, detects if the phone is held and touches to the user’s face. In such cases, the screen would be disabled in turn. Making use of such features of the proximity sensor and ambient light sensor, IODetector [31] differentiates the indoor and outdoor environment automatically. In our project Memento, we exploit the ambient light sensor to evaluate the light quality of the current environment and select the suitable lifelogging method based on that.

Barometer senses the atmospheric pressure and infers the corresponding altitude. It is originally introduced to assist GPS positioning. Recently new approaches which leverage the barometer are proposed. In [32], authors presents an activity detection system based on the barometer. Comparing to the inertial sensors, barometer is independent of the device position and attitude, thus it is possible to apply the low sampling rate and simple processing algorithms to further save energy. In [33], barometer is also discussed to detect the people movements among different floors in buildings. Nowadays almost every smartphone is equipped with microphone. By continuously collecting audio, the microphone serves the major input of a plenty of services, e.g. context classification [34] and voice recognition [35–37], etc.

Recent years more new types of sensors are introduced to mobile platforms. On the one hand, various *soft sensors* are being incorporated into mobile systems. *Soft sensor* is not dedicated hardware but a set of API to provide particular sensing functions by fusing several types of other sensors. For instance, gravity sensor of Android [38] measures the acceleration effect of Earth’s gravity by using accelerometers and gyroscopes. On the other hand, new dedicated hardwares are exploited for mobile platforms. In [39], the air quality sensor is embedded on smartphones to study the air pollution with the help of mobile users. Researchers, for example, try to integrate Electromyography (EMG) [40] and Electroencephalography (EEG) [15] with smartphone and even wearable. By sensing EEG, NeuroPhone [41] enables users to use neural signals to control phones. In [42], authors build a diet recognition system by integrate EMG with wearables. In our project Memento, we make attempts to leverage EEG to recognize the user emotions and build a better lifelogging system based on that.

2.2 Techniques of context inference

To further extract the higher-level knowledge from the raw sensed data, machine learning techniques are applied. Generally the most informative features are selected from the data first and then the machine learning model is built according to the phenomenon of interest. For instance, IODetector [31] is built to extract the indoor-outdoor information using the on-board sensors on smartphones, which is the important input for most of location based services. IODetector first applies various band-pass filter on sensed data and then makes use of Hidden Markov Model (HMM) [43] to derive the context of indoor-outdoor. In CrowdSense [44], authors present a framework to classify places into place categories. To understand the semantics of places, authors combine signals based on locations and trajectories, as well as visual and audio hints mined from related images and videos. Multiple context inference features and techniques are discussed,

including HMM, Gaussian Mixture Model (GMM) [45] and Mel-frequency Cepstral Coefficients (MFCC) [46], etc. Combined with Boosting [47], HMM based approaches are also applied in [48] to classify human activities. To recognize or classify context information, Bayesian based methods are often considered. [49] proposes a Bayesian framework to model and predict the human daily mobility. MAQS [50] is an indoor air quality monitoring system, which involves the Bayesian framework to achieve the room localization with fewer Wi-Fi fingerprints. Clustering techniques are also discussed to derive knowledge from, especially, large-scale sensing data. For instance, in our project PDU detection, we make use of K-means clustering to detect the phantom data usage among all traffics of smartphones. Recent years, deep learning techniques are emerging. Researchers try to build powerful mobile sensing systems with the help of deep learning techniques as well. For example, in order to leverage microphones on smartphones to perform common audio sensing tasks, DeepEar [51] proposes a sensing framework built from Deep Neural Network (DNN) [52] to extract human behavior and context.

2.3 Mobile sensing applications and systems

Recent years new types of mobile sensing applications are explored in not only academic laboratories but industrial and startup companies. In this section, we discuss a number of innovative mobile sensing applications, which leverages the features and characteristics of smart devices to serve in many emerging areas.

Smart devices, especially smartphones and wearables are usually carried by users everyday, and are able to access to the user’s personalized information, including web browsing records, application usage history and SNS contacts, etc. Such personalized information make opportunities for researchers and developers to design personal assistant services. For instance, Google Now [53] keeps the user’s information and harvests the related knowledges from Internet which might be needed by the very user. Another

example is MindMeld [54], which is an online video conference software. By collecting and analyzing the conference speech in real-time, MindMeld provides the related knowledge and content that might be useful for the audiences. Such personal assistant services also can be used to enhance existing services. By analyzing the application usage history on smartphones, AppJoy [55] makes attempts to intelligently recommend new apps for the particular user. The application usage history is also used in FALCON [56], which predicts application launches before they occur, so that it pre-launch such apps to dramatically reduce perceived delay. In our project PDS, in order to detect the data usage which does not accord with the user’s perception, we deeply examine the application usage records and the user context information including locations, times and movement states, etc.

The data used for health care usually comes from medical consultations. Nowadays smart devices are becoming another important information source for health caring and well being. Mobile sensing applications collect the in situ data, apply the human behavior model and react according to the sensed context. Such applications have the potential to revolutionize the way how health and wellness are diagnosed as well as therapies and treatments are delivered. By capturing and analyzing motion and audio data, [57] derive various metrics such as users’ physical activities and sociability, which can be widely used for well-being by the medical community. By analyzing communication history and application usage patterns on smartphones, MoodScope [58] tries to infer the mental state of the user and provides mood as an important input for context-awareness computing. Mood also acts a key role to predict if a user is at the risk of depression. Lifelogging systems are also empowered with sensing capacities. For instance, SenseCam [59] is discussed to help patients recover from the Alzheimer’s disease [9].

Today the urban population is experiencing the explosive growth. Issues such as traffic and pollution are becoming serious global problems. With the help of crowd-sourcing,

mobile sensing applications open a new door to solve such problem on a large scale with low cost. [60] installs the application on the smartphones of bus riders to sense the bus travel period. In turn the whole urban traffic condition is predicted. By using data from thousands of smartphones, VTrack [61] are being used to provide the fine-gained traffic estimation. To monitor air pollution in urban, for instance, [62] crowd-sources the air pollution measures from individuals with hand-held devices.

2.4 System implementation issues

Though techniques advance, the computation and energy resources on smartphones are still limited. These practical issues should be well addressed when implementing mobile sensing systems.

To reduce the energy consumption on smartphones, the adaptive sensing is usually considered. In SociableSense [63], authors propose an algorithm to evaluate the current situation so that the sampling rate would be reduced when no certain events are observed. To further leverage multiple types of sensors on mobile devices, researchers consider to active the power-hungry sensors with the ones with low energy profiles when the interesting events occur. This is also the key idea of the Energy Efficient Mobile Sensing System (EEMSS) [64]. Considering smartphones as multi-proposers platforms, there might be several applications running at the same time. Piggyback sensing [65] is discussed to share the sensing data across applications to further save the battery life. Recent years, low power units (LPU) and co-processors are being incorporated. ZOE [66] and LEO [67] make attempts to integrate such LPU to provide affordable continuous sensing on wearables.

To extend the limited processing capacities on mobile, cloud offloading is considered. MAUI [68] proposes to offload the code from mobile devices to remote servers in fine-grained. COMET [69] utilizes the distributed shared memory to build the offloading

approach for Android. In Odessa [70], parallelism optimizations are discussed to achieve both energy efficiency and low network latency. Tango [71] presents a novel offloading mechanism, instead of migrating code, but replicate the program and execute it on both of the mobile and cloud side. To overcome the potential network latency, authors of [72] make use of offloading to Cloudlet [73]. In our project Memento, we consider the offloading as well. Instead of offloading to remote servers, we exploit the local heterogeneous computation resources around wearables, e.g. personal laptop, smartphone, etc.

Chapter 3

Detecting Phantom Data Usage on Smartphones

3.1 Phantom Data Usage Issues

To understand PDU issues in real world, we collect 412 data usage issues from the following forums: (1) popular Android forums including XDA-developers and Android Open Source Project (AOSP) forums; (2) manufacture forums like Microsoft Community, Google Group and Apple support discussions; and (3) service provider forums, e.g., AT&T community. For the issues that are confirmed to be resolved, we study their causes from the official replies and feedbacks. For the open issues whose causes are still unknown in the forums, we download and install the corresponding applications with the certain version. We test and try to reproduce the issues mentioned in the complaints. To dig out more information about the open issues, we make attempts to reach out the related users. We select and reach out ten users from open issues by sending site mails or replying on their posts. We try to understand more details (e.g. the applications usage trace, the system states and the network preferences, etc) when PDU issues occur. We also make attempts to learn the reasons why they regard this data usage as PDU. The feedbacks from interviewees are kept anonymously.

From the survey, we make the following observations:

Figure 3.1: Root case distribution of the data usage issues in the survey.

- Most of the data usage issues is for unknown reasons to users, regarded as PDU. Illustrated in Fig. 3.1, software bugs including client bugs and server bugs contribute about 19% of sampled data usage issues. Indeed only 21% of issues are caused by malware which is generally considered as the main PDU cause by users. More than half of sampled issues are related to PDU. It indicates that PDU is a pervasive problem for smartphone users. For most data usage issues, it is difficult for users to recognize the causes and figure out the solutions.
- PDU issues are reported on even well-designed and popular applications. We download the 61 applications mentioned in the unknown issues. Fig.3.2 shows the majority of the applications are reputed and popular with the high downloads rank, including Facebook, Dropbox and Google Map, etc. These applications are supposed to be well-designed and ought to have no serious problems. The further test verifies our ideas. Among these apps, only 4 are problematic due to client or server bugs, which indicates the causes of most PDU issues are not the exceptions of applications but might be from users themselves.

Figure 3.2: Download rankings of applications in unknown cases.

- The misuse of users is the major cause of PDU. We study the discussions of the unknown issues and find the potential cause is the misuse of applications. Misunderstanding and mis-configuration can result in misusing certain applications. For example, a user switches on the auto-update feature in App Store and the data usage consumed by the auto-update might be regarded as PDU. Note that although the features (like auto-update) are normal and by-designed, the users might take the data consumption of them as PDU.
- The occurrence of PDU issues does not follow generic patterns. We interview 10 users from 10 unknown issues. We discuss with them and try to understand the details when PDU occurs. Different users have different application usage habits and preferences.

Now we give a clear definition of PDU. According to our study, users determine

PDU when the data usage consumption does not accord with their expectations. In this paper, we borrow the concept of Quality of Experience (QoE), which is the difference between perceptions and expectations for a service, to infer PDU. The perceptions and expectations are normally developed based on many factors, e.g., word-of-mouth, past experiences, the intensity of personal needs, etc. We make use of QoE in a simplified way by leveraging the usage habits of individuals as a measure of user expectation. PDU is the data usage consumption that is inconsistent with the usage habits of individuals.

3.2 Context and Data Usage

Figure 3.3: The temporal data usage patterns of two users.

Before presenting the system design, we discuss our observation on context and data usage patterns. Prior works [56, 74, 75] provide the insights of the comparison of the usage time among different applications, i.e., how long the user uses the application.

The data usage, however, is different from the application usage time. It corresponds to the consumed network traffic which is highly related to the user’s operations on the application, i.e., how the user uses the application. Different user operations result in distinctive data usage patterns. For instance, The operations on Dropbox can be divided into browsing and downloading, whose data usage profiles (§3.3.3) are highly different. As our observation reveals in this thesis, the user’s operations on each application are related to the user context. To detect PDU, we have to not only study the data usage patterns of different applications but also investigate the usage patterns of each application in different context settings.

We collect the traces of 18 popular apps in 3 categories which are news, games and SNS from 7 volunteers for three weeks using the information collector (§3.3.1). The time, cellular tower ID and network traffic traces are recorded. The locations are labeled manually by volunteers as home, office, canteen or market, etc. Inspired by the prior work, we select the time, locations and movement status as the factors. Specifically, we record the data usage traces of each app considering the user operations on the app, which are classified using the method introduced in §3.3.3. Based on the experimental study, we make the following observations:

Time. We begin with the analysis of temporal data usage patterns. Our study shows different time of day influences both the overall data usage of different apps and user’s operations on the apps. Fig. 3.3 plots the average amount of data usage of two apps (games and SNS) from two users. As depicted in the Fig. 3.3(top), the data usage of each application is different across different time of the day and different users have different application usage habits. For user A, news apps are most likely used in the morning and SNS apps are preferred in the afternoon while user B spends less time on news app but more time on games. In Fig. 3.3(bottom), the two users’ operations of using the SNS app are plotted in different colors. We see user A and B have different operation combinations on the app and the distribution also changes with time.

Figure 3.4: The spatial data usage patterns for different applications.

Locations. Our study reveals that the user’s locations influence 1) which apps a user prefers to use, and 2) the user operations as well. Fig. 3.4 depicts the data usage of different apps at different locations for the same user. As shown in the figure, the user prefers to launch different apps in different locations. News apps are most likely launched at office and SNS apps are usually used in the classroom. All the three apps have different user operations and their distributions in different locations are also different in different locations.

Movements. We extract the user’s movement status from the mobile phone’s cellular tower ID variation. We use a binary value to present the movements, i.e. static and moving. Our basic idea is that we leverage the cellular tower ID as the signature of an area. We consider the user is static if the cell tower ID which the User Equipment (UE) connects to does not change, moving otherwise. In this thesis, we mainly consider

Figure 3.5: The data usage patterns in different movement statuses.

Macrocell with the coverage range of around several kilometers, since it has been widely deployed. We also notice that in some areas, the cell deployments are dense. UE possibly won't connect with a single static cell tower even the user is moving around within the area. Therefore we propose to use the top-2 strongest cell towers as the signature for a certain area. Finally we use GPS locations as the backup approach if all the cellular designs fail to work. Fig. 3.5 shows the data usage of three apps from 3 users. We see that all the users prefer different apps at different movement statuses. For example, all of them prefer to use news apps when static. But when moving, user A is used to using SNS apps or playing games, but user C prefers playing games mostly.

Though we investigate the three factors separately, their influence on the user's operations and application data usage is not independent from each other. In the following section, we jointly leverage the three factors to build a PDU detection system.

Figure 3.6: Phantom detection system overview

3.3 System Design and Implementation

In this section, we introduce Phantom Detection System (PDS) by describing its overall architecture and core components. PDS is designed to effectively collect and analyze data usage information. PDS leverages *data usage patterns* — a user’s data usage habits in different context settings, which enables PDS to evaluate whether the consuming data usage accords with users’ usage habits to detect PDU. Fig. 3.6 shows the system overview of PDS. PDS comprises of three components. The information collector samples the network traffic, the context information and the related system events. The data usage model (DUM) is triggered by the certain system events including the network status changes and application maintenance messages. Given the input context information, DUM tries to find the record with the most similar context settings from the usage history dataset which contains pairs of context settings and the corresponding data usage profile. Then the decision engine (DE) compares the corresponding data usage profile with the input network traces. DE determines the inconsistent traces as PDU, otherwise updates the usage history dataset. Next we will introduce the key techniques in each component.

Timestamp	UID	In	Out	Src. Addr.	Src. Port	Dst. Addr.	Dst. Port	Len.	Proto.
1378709492618	10074		wlan0	172.22.100.60	49132	111.13.12.89	80	52	TCP
1378709492619	10074	wlan0		111.13.12.89	80	172.22.100.60	49132	40	TCP
1378709492619	10074	wlan0		111.13.12.89	80	172.22.100.60	49132	40	TCP
1378709492620	0		wlan0	172.22.100.60	51220	155.69.24.6	53	59	UDP
1378709492620	10032		wlan0	172.22.100.60	51385	173.194.38.130	443	60	TCP
1378709492621	10032	wlan0		173.194.38.130	443	172.22.100.60	51385	52	TCP

Table 3.1: The trace samples collected in the information collector.

3.3.1 Information Collector

The information collector records the timestamped information including the network traffic, the users’ locations, and the system events. We implement the information collector on Android phones. We leverage IPTable [76] which is an inherent library in Android Open Source Project (AOSP). To execute it on custom ROMs, we build IPTable for different CPU architectures including ARM v5, ARM v7, MIPS and x86 as well. We add logging rules into INPUT and OUTPUT chains of IPTable and capture the network traffic information via monitoring the kernel message (KMSG). As shown in Table 3.1, the detailed network traffic information contains: 1) Time stamp recorded in UNIX format. The time is further quantized into 6 intervals, i.e., 8am - 12pm, 12pm - 4pm, 4pm - 8pm, 8pm - 12am, 12am - 4am and 4am- 8am. 2) UID. UID is the application identifier. Each application is assigned an unique ID when installed. 3) IP address and port number. For each network operation, its source and destination IP address and port numbers are recorded. We collect the package length and its protocol type as well.

To get the locations, we leverage the cellular tower ID instead of GPS for two concerns; to achieve the low sensing overhead since GPS is energy-hungry, while at the same time avoiding using the precise locations for privacy protections. Most of commercial smartphones are able to sense the signals from several nearby cellular towers. In the implementation, we make use of top-3 cellular tower id as the location identity. We evaluate the variance of cellular tower ID to extract the user’s movement status, static or moving.

We mainly collect two types of system events, the network status changes and application maintenance (installation and upgrade) messages. We make use of the *Broadcast* mechanism of Android to receive these event notifications timely and efficiently. Other mobile systems also implement similar mechanisms.

The major requirements of the information collector are the high efficiency and the low overhead. The information we collect is side-product during the normal execution of the system and applications. To protect user privacy, we store the above information in the private storage, which other applications cannot access.

3.3.2 Data Usage Model

The goal of Data Usage Model (DUM) is to find the record containing the most similar context settings with the input context information from the usage history dataset, which we regard as the optimization problem. Given the context information S_r^c when application r is running, it contains three parts: $S_r^c = \{t_r^c, l_r^c, m_r^c\}$ where

- t_r , the quantized time interval. $t_r \in \{T_1, T_2 \dots T_6\}$ which is mentioned in §3.3.1.
- l_r , the locations are presented as the sequences of (n_r^1, n_r^2, n_r^3) in the order of RSS where C is the cellular tower ID.
- m_r , movements status can be extracted from the variance of the cellular tower ID sensed by the mobile phones. Specifically we get two types of movements, steady and riding. $m_r \in \{M_{steady}, M_{riding}\}$.

For each application r , the usage history dataset contains a list L_r^h of $\langle S_r^h, U_r^h \rangle$ where S is the context settings and U is the corresponding data usage profile. To retrieve the most similar context settings with S_r^c from L_r^h , we minimize the object function quantifying the overall bias between two context settings.

$$E_T = \sum_r ||t_r^h - t_r^c|| \quad (\text{Eq. 3.1})$$

$$E_L = \sum_r \sum_i idmatching(n_r^{h,i}, n_r^{r,i}), I \in \{1, 2, 3\} \quad (\text{Eq. 3.2})$$

$$E_M = \sum_r ||m_r^h - m_r^c|| \quad (\text{Eq. 3.3})$$

where the overall bias objective function E comprises of three terms. E_T captures the constraints that the time t_r^h and t_r^c must belong to the same time period. E_L captures the fact the locations should be as closed as possible. In particular, we use the function *idmatching* [77] to compute the matching score of two cellular tower sequence. If $n_r^{h,i}$ is same with $n_r^{r,i}$, they are considered matching, mismatching otherwise. Each match is assigned a positive match score according to its RSS rank (i.e. 1, 2, 3). For each mismatch, a penalty cost (-2) is set. At last E_M minimizes the difference between the movement status. The movement status should be the same. To achieve the objective we regard it as a multiobjective optimization problem, that is $\min(E_T, E_L, E_M)$. Since the search space in this scenario is within the existing significant records in the database, which is limited and discrete. Therefore given the input context information S_r^c , we can optimize to get the record containing the most similar context setting $S_r^{h,\theta}$ and its corresponding data usage profile $U_r^{h,\theta}$.

3.3.3 Decision Engine

The goal of the decision engine is to evaluate the current data usage U_r^c . Based on the observations in §3.2, U_r^c might be PDU if the similarity between U_r^c and $U_r^{h,\theta}$ is low. To compute the similarity, we first profile the current data usage. In this paper, we leverage a set of statistical features of network traffic [6, 7] including: 1) L_{max} , the maximum packet length among the records of U . 2) L_{avg} , the average packet length of the records in U . 3) F , the frequency of records in U , $F = \frac{N}{T}$ where T is the time length of U .

Figure 3.7: The similarity of 20 operations on 6 applications.

We use (L_{max}, L_{avg}, F) as the profile of the given data usage. Then we define a measure of similarity between the profiles of U_r^c and $U_r^{h,\theta}$:

$$s_o(U^c, U^h) = 1 - \sum w_j \cdot \varphi_j(U^c, U^h), j \in \{m, a, f\} \quad (\text{Eq. 3.4})$$

In Equation Eq. 3.4, $w_j, j \in \{m, a, f\}$ is the weight which is pre-defined. $\varphi_j, j \in \{m, a, f\}$ is the similarity function for L_{max}, L_{avg}, F . In our implementation, we make $\varphi_g = ||g_c - g_h||$ where $g \in \{L_{max}, L_{avg}, F\}$.

To evaluate the similarity model, we collect the network traffic traces of 20 different operations on 6 apps. We conduct each operation twice and get 40 traces in totally. As seen in Fig. 3.7.a, the traces of same operations lead to the high similarity. However there are clearly 4 pairs of different operations apart from the diagonal gaining the high similarity as well, hence false positives. The further analysis reveals the combinations of network requests of different operations might result in the similar statistical network traffic features. For instance, operation #6 is downloading multiple documents and operation #13 is browsing the page with lots of images. They obtain the closed L_{avg} and F .

To improve our approach, we introduce the *service grouping* which is based on an observation: different operations on an app trend to access different IP/Port servers. For example, operation #6 accesses the document server and the image server is visited by operation #13. The IP/Port servers can be used to indicate the operations. To apply service grouping, we take the following steps: 1) group the records of network traffic traces according to their IP address and port number. The record group with the same IP address and port number is called a service. 2) Profile the network traffic of each service. 3) Compute s_o for each same service of U^c and U^h . 4) Let the overall similarity s_{ov} as the sum of the similarities of all the same services.

Shown in Fig. 3.7.b, after introducing the service grouping, the false positives are canceled. We determine the default DE similarity threshold to be 0.8 by off-line training. If the current data usage is consistent with that in the usage history, DE updates the usage history dataset. In our implementation, we use the average of statistical features to update the corresponding data usage profiles in the usage history dataset.

We notice that several potential machine learning techniques could help if they are adopted in the decision engine. However since most of machine learning algorithms are heavy, which should be carefully adopted when introduced into the resource limited platforms, especially for the long-term services like PDS. On the other hand, some unsupervised machine learning algorithms, i.e. clustering, require the enough historical data to perform well. How to bootstrap the system is still a practical problem. In this thesis, we present our simple but efficient solution. It can work on the bootstrap and provide the expected performance. We would like to explore to leverage more sophisticated machine learning algorithms to improve the decision engine in the future work.

Figure 3.8: Detection results of PDS.

No.	User A	User B	User C	User D	User E
TP	9	7	15	8	6
FN	4	5	5	3	1

Table 3.2: Breakdown of detected PDU cases.

3.4 Evaluation

In this section, we assess the effectiveness and overhead of PDS. We evaluate PDS on real user phones. We recruit 7 Android users from our university campus. The volunteers have 5 different phone models to these volunteers, including 2 LG Nexus 4, Samsung Galaxy X, HTC Sensation XE and HTC Desire. We install PDS on the users' phones. The system traces including network request logs and the system event history as well as the detection results from PDS are collected for three weeks. With the permission of users, the experiment data are uploaded to our back-end server using encrypted links for

the analysis. We anonymize all the data to ensure users' privacy.

In practical, it is hard to reproduce PDU issues or control the occurrences of PDU due to their high dependence on the personalized usage. Therefore we customize PDS that it would report suspicious data usage to the users and ask for the confirmation. We make use of the volunteers' confirmations and feedback as the ground truth. The volunteers are further divided into two groups according to their smartphone usage experience. The first group contains five experienced users. They either have rich experience in using Android devices or are Android developers. As a contrast, the second group contains two users who are fresh to Android phones.

Figure 3.9: The similarity distribution of the PDU cases confirmed by users.

For the first group, we set a low DE threshold to ensure that all the suspicious data usage are reported. As depicted in Figure 3.8, the reported case shows all the suspicious data usage cases; the PDU case is determined by the default threshold of PDS; the

No.	Description	Memory (MB)	Battery Life
1	Idle + Radio	12	21.95
2	Detector + Radio	15	19.74
3	Normal	21	7.74
4	Detector + Normal	25	7.02

Table 3.3: Energy and memory consumptions of PDS in different usage states.

confirmed case shows the cases which the user considers as PDU among the reported cases. In the three weeks, there are 200 suspicious data usage cases reported. Among them 88 cases are detected as PDU by PDS and 63 cases are confirmed as PDU by users.

Table 3.2 shows the further breakdown of the statistics on the results of group #1. TP stands for the case which is detected as PDU by PDS and confirmed by users. FN shows the PDU case which is not detected by PDS but is labeled as PDU by users among all the reported cases. Then the average true positive rate of PDS from these 5 users is 72.2%

Next we present the DE similarity distribution of the PDU cases confirmed by users in group #1 (true positives in Table 3.2). Illustrated in Figure 3.9, more than 80% of PDU cases obtain the DE similarity around 0.8. It is consistent with the default setting of DE similarity threshold. The result also guarantees the high quality of the labels from group #1 users.

For the second group, we record the size of the usage history dataset which not only leads to the storage overhead but also affects the execution time retrieving the context settings. As Figure 3.10 shows, after two weeks the size approximately reaches its maximum level and after three weeks the size approaches its plateau. For both two users, the usage history dataset of an app averagely contains 67.5 records.

We measure the energy consumption on three Android smartphones: HTC G14, Nexus 4 and Sumsang Galaxy S2. We operate the smartphones as usual and use a Monsoon Power Monitor to capture the battery consumption in real time. Figure 3.11

Figure 3.10: The growth of data usage history.

shows the instant battery consumption when the system is in idle state and running PDS. We further measure the PDS energy profiles in different usage states: 1) cellular radio and Wi-Fi on, screen off and phones in idle mode. 2) Cellular radio and Wi-Fi on, screen off and PDS running in background. 3) Cellular radio and Wi-Fi on, screen on, some other applications running normally. 4) Cellular radio and Wi-Fi on, screen on, PDS and some other applications running simultaneously. The battery capacity in our experiment is 3000 mAh. As shown in Table 3.3, in idle state with radio and Wi-Fi open and screen closed. PDS will consume extra 4.5% energy. And for the normal usage cases, battery life will reduce 10.2% due to the overhead of PDS.

We use Android Debug Bridge (ADB) to dump the memory usage in run time. In the case of normal use, PDS occupy 19MB in average. When only logging, memory utilization is lower, about 12MB. When the detection procedure is triggered, memory

usage will reach around 25M. This level of memory occupied belongs to common usage among Android utilization tools.

Figure 3.11: Instant battery consumption of PDS.

3.5 Related Work

In the project PDS, we try to leverage the application usage patterns to infer the abnormal data usage which does not accord with user expectations. Therefore we review the related work in terms of application usage analysis and anomaly detection.

Application usage analysis. A number of studies focus on the application usage prediction. MyExperience [78] captures both objective and subjective in situ data on mobile computing activities, including device usage, application usage and media usage, etc. By analyzing the application usage patterns, a user study of MyExperience shows that users prefer using SMS when they are moving. In [79], authors develop an algorithm

that can be used with mobile based usage data acquired straight from end-users in an established panel study process. The analysis reveal how different mobile services are used under different context, for example, the application usage patterns are highly correlated with the locations. A large-scale study is conducted in [80]. According to that, the application usages and their major context factor are discussed, including the time and locations, etc. It reveals that the usages among different applications are high correlated. Based on the application usage analysis, there are several applications and services as well. By modeling and analyzing the past application usage patterns, PCS [65] makes attempts to predict the certain application use in the future. In turn such opportunities could be leveraged to apply the Piggyback CrowdSensing in order to further save battery life on smartphones. The application usage analysis is also applied to improve the mobile system performance. In [81], a dynamic home screen is proposed based on analyzing the users' applications usage patterns. In such home screen, the most probable application icons are displayed. Furthermore in [56], the most probable application is predicted and pre-loaded in order to provide a fast-loading and seamless user experience. While these studies successfully explore the application usage patterns and how to leverage them, it is more important to understand the fine-grained data usage patterns for detecting PDU.

Anomaly detection. Anomaly detection refers to the problem of finding patterns that do not conform to expected behavior. For example, In [82], authors apply a neural-network based approach to detect fraudulent credit card applications and fraudulent credit card usage in large scale. By monitoring calling activities on mobile phone, several anomaly detection techniques [83] are used to detect the mobile phone fraud. In [84], the authors focus on detecting disease outbreak in a specific area and unusual medical time sequences in the health-care domain. In the previous work, the most related with PDS could be network traffic anomalies detection. In the research and commercial literature, volume based detection schemes are usually considered [85]. However a large class

of anomalies would not cause the obvious and detectable disruptions in traffic volume. Therefore sophisticated features and approaches are explored. For instance, [6] propose to use the distribution of packet features to detect the anomalies in the network traffic, including portscans [86], DOS attack [87] and so on. In [88], authors propose to use the correlation of IP addresses in outgoing traffic at egress routers to detect anomalies. Detecting PDU on smartphones can be treated as an anomaly detection problem. However it could be more complicated due to the PDU characteristics and high diversities among users.

3.6 Discussions

Device-specific. Currently we develop PDS only on Android phones. Although several mechanism of Android make it easier to be implemented, our approach is not limited to any specific platforms. The information we make use of is also available on other platforms including iOS and Windows Phone. For example, *tcpdump* [89] running on iOS can provide all the data usage information. The similar APIs also can be found on other platforms. Further more, PDS can not only serve as an application but also can be integrated into existing data usage management softwares. Currently we design and implement PDS and show the basic performance. Next release and test in public would be one of the future work.

Bootstrap stage. PDS is a history-based detection system. With the growth of the usage database, the detection accuracy is expected to increase as well. However PDS highly depends on both the quantity and the quality of the usage database. In the bootstrap stage, how to build a high-quality usage database in a short time is a practical problem for PDS. Currently PDS provides a simple but efficient threshold-based solution at the bootstrap stage. But it needs to involve the users to ensure the correctness of the usage history. One of our future work is to reduce user interventions. We make

a preliminary observation that the applications of the same categories might share the similar data usage patterns. Based on the observation, we can train models offline for some popular applications, which can be used to cross-refereed when building the usage history database for similar applications. Another of our future work is to explore more sophisticated and appropriate machine learning algorithms to further improve the performance of PDS.

Chapter 4

Building An Emotion-driven Lifelogging System on Wearables

4.1 Lifelogging and Emotions

Lifelogging is generally considered to be the digitalization of personal experience for personal use. In the past decades, several dedicated passive lifelogging devices are proposed for various purposes, e.g. augmenting lives [59], memory aid [8] and health care [9]. With the rapid growth of mobile devices, lifelogging services are implemented on smartphones. Multiple on-board sensors and various sensing techniques enrich the contents of lifelogs, including location information [90], inferred user activities [91] and so on. Social Network Services (SNS) further turns the lifelogging into an emerging lifestyle. With smartphones users love collecting memorial moments and share with friends. Recent years, wearables gain their popularities and show the potential for lifelogging. The question therefore arises that how to architect a better lifelogging system on wearables.

We believe that lifelogging could be improved by capturing the internal experience of users. Rather than continuously logging or requiring manual operations, the system is able to automatically log memorial moments by understanding the user's mental state changes. For example, when the user gets very happy or sad, the happiness or sadness could be recorded in the forms of audios, images or videos automatically. Some

unexpected situations or accidents could be logged as well when the user suffers strong emotional disturbances. We also believe that the lifelogs with emotion information could not only improve the lifelogging system itself but also enhance many other existing services such as SNS, health care, memory aid, etc.

Research on emotions is not new. Generally, the bodily changes follow directly the perception of the exciting fact, and that our feeling of the changes as they occurs is the emotion [92]. The consensus is that emotions are physiological and measurable. Some basic emotions can be found universally across individuals when certain stimulations are given. Ekman's research findings [93] led to classify six emotions as basic: anger, disgust, fear, happiness, sadness and surprise. It is then improved to the bipolar model where arousal dimension (*how energized the experience feels*) and valence dimension (*how negative or positive the experience feels*) are considered.

The emotion changes can be observed via several physiological effects such as heart beats, facial expressions, voices and brain activities. EEG is a measure of brain activities via the brain electric signals. In order to harvest signals, EEG electrodes are attached on the scalps. The emotion changes lead to distinguished patterns in EEG signals from the certain positions. There is an increasing number of EEG based emotion recognition techniques [94–96]. However, most of EEG based emotion recognition algorithms are not designed for the wearable-class platform. To adapt them, the problem we are faced with is the algorithm complexity. The influence of the high complexity is twofold: time constrains and energy overhead. On the one hand, most of the algorithms are proposed for the off-line emotion recognition, even on more powerful platforms. On the other hand, the energy issues are rarely considered in the previous designs.

Figure 4.1: System architecture of Memento

4.2 Memento System Design

To enable lifelogging with wearables, we present our design of an emotion driven lifelogging system, namely Memento. Fig. 4.1 illustrates the system architecture. Memento contains three modules: the signal processing module, the lifelog collector and the off-line emotion recognition module. The signal processing module takes EEG readings as input to detect users’ emotional changes, e.g., emotion events. When an emotion event is detected, the lifelog collector starts to conduct the lifelogging. The camera and microphone on smart glasses serve as the major medias to capture the daily life moments in the form of videos, audios or pictures, according to both the environment conditions and the system states. Finally the recognized emotions are tagged on the respective lifelogs.

The EEG electrodes loosely contact on user’s scalp and the sensed EEG signals can be easily polluted by interfering activities, e.g., eye blink, head movement, etc. To

address this issue, in the signal processing module we propose an effective band-pass filter to remove ambient noises, utilize the kernel-based correlation to exclude non-brain activities, and further leverage motion sensors and the rigid feature of the wearable to score the quality of the perceived EEG signals (§4.2.1). With all high-score EEG signals, the lifelog collector module performs lifelogging according to user’s emotions. However, recognizing emotions from EEG signals is computational intensive, not afforded by the wearable platform directly. To address this issue, we propose a two-phase emotion recognition solution and install them on the smart glasses and the cloud, respectively. On the smart glasses side, without specifying the exact emotion type, Memento merely detects emotion changes by leveraging the intermediate results from the feature extraction phase (§4.2.2). Once a significant change is detected, the lifelogging is launched. To best balance the lifelogging qualities and the energy consumption, we propose the lifelogging engine to automatically select and configure the suitable lifelogging methods, according to the environment contexts and the system states (§4.2.3). Then lifelogging engine would also determine the lifelogging duration dynamically. Finally the collected lifelogs as well as sensed data would be uploaded to the cloud later when the smart glasses are recharging or upon the user requests. On the cloud a sophisticated recognition algorithm asynchronously runs to recognize the exact emotion type, which serves as the emotion tag for the subsequent recorded lifelogs.

4.2.1 Signal preprocessing

In this section, we describe the EEG signal preprocessing techniques applied in Memento. Preprocessing has been discussed in the previous BCI work. However, instead of collected in the controlled environment the signals of Memento are harvested in the uncontrolled and mobility cases, where the EEG electrodes are not fixed on the scalp and more internal and external noises are introduced. Specifically we process the signals as the following steps:

Figure 4.2: Frequency distribution of raw EEG signals.

Band pass filter. EEG frequency normally ranges from 1 to 100Hz, which is further divided into several bands: Alpha (8-14Hz), Beta (12-30Hz), Gamma (>30Hz) and Delta (<4Hz), etc.. On the one hand, as shown in Fig. 4.2 on the whole EEG frequency band, several interferences are usually involved. The noise from electric appliances locates at 50Hz. The interference from the heart activities generally overlap at the low frequency band (1-6 Hz). The muscle activities such as blinks also introduce the noise to EEG signals. On the other hand, Alpha and Beta band contain the significant features to represent mental states, which are widely used in the motion recognition.

Therefore we apply a 7-31 Hz band pass filter. Some research points out where the EEG frequency bands begin/end varies depending on the input sources and the EEG frequency boundary usually varies a few Herz, we set a fixed band pass filter in our design. We do not focus the specific signal characteristics on each band. The goal of the band pass filter is to get the most significant partial of signals, while filtering out the

Figure 4.3: Detecting blinks from raw EEG signals using kernel correlation

external and internal noises.

Non-brain activity removal. Non-brain activities including chews, blinks and other muscle activities, produce electric signals and might be harvested by EEG electrodes. Frequent non-brain activities involve lots of interference to EEG signals. In EEG study electrodes are usually attached on certain positions such as TP9, TP10, AF7 and AF8 (see Fig. 4.7.c as a reference) according to 10-20 system [97]. On this area, the most significant non-brain activities are blinks. In order to detect then remove the blink effects, we examine the following methods:

We notice that EEG amplitude changes on AF7 and AF8 channels when blinking, as shown in Fig. 4.3. We decompose signals into frames with the fixed length and apply a threshold based method to detect if a frame contains blinks. However due to the tail effect of the blinks and other brain activities, high false positive rate is obtained. The frequency characteristic might be distinguished between signals from the brain activities and blinks,

Figure 4.4: Band power correlation across EEG electrodes when drifting.

but continuous time-frequency transformation could involve a so heavy workload that we do not consider it.

In our design, we apply a kernel correlation based approach on time domain. We empirically set a frame with the size of 400 ms which equals to a typical blink duration. We train the kernel of blinks off-line. Fig. 4.3 illustrates the results when we apply the trained kernel on EEG signals directly. In each frame, correlation between the signals and the kernel is computed to detect if the blink effects are contained. After that, for every frames containing blink effects we apply a modified algorithm from [98] to remove the effect of blinks.

Signal quality estimation. To further understand the signal quality, we apply the signal quality estimation. We evaluate the signal quality in three aspects:

- First we make use of the reference electrode to estimate the contact status. The reference electrode emits the constant current. The energy harvested on the reference

Figure 4.5: Resultant accelerometer readings under different usage cases.

electrode indicates the contact status, if the contact status is good the power loss should be limited, otherwise only less power could be harvested on the reference electrode. We apply a binary filter to determine the contact status and use it as the trigger of the whole processing procedure.

- Second, we use the power correlation across electrodes to infer the electrode drift. When EEG electrodes drift we have the observation that since the wearable is a rigid-body, all the electrodes should be influenced at the same time and the drift impacts the signal power harvested in a short period. Fig. 4.4 illustrates the signal power on AF7 and AF8. When the drift occurs the correlation of the signal power on these two channels is observed.
- Third, we notice that the signal quality is very likely to get worse when the user is in the vigorous movements. Therefore the motion sensor is utilized. Fig. 4.5 shows

the resultant acceleration in different movement states. We use the variance of the resultant acceleration to infer the movements and use it as another indicator of the signal quality.

Finally we linearly combine the two indicators, the power correlation and the acceleration variance with the even weights as the signal quality score.

4.2.2 Two phase emotion recognition

Many emotion recognition algorithms are too heavy to adapt them on wearables directly, especially considering the near real-time processing constrain and the computation capacities. To best our knowledge, most of algorithms are designed for recognizing emotions off-line.

Emotion recognition algorithms are based on different theories, but the procedures of these algorithms share the similar patterns: they can be decomposed into the feature extraction phase and the classification phase. We observe that the overall execution time is too heavy for wearable-class devices though, the classification phase is the major contributor and usually the feature extraction phase is light enough. It is because that most of feature extraction approaches leverage near linear algorithms, while a plenty of iterations are involved in the classification phase. For instance, in [95], the lifting based wavelet transformation is used to extract features, which is acceptable yet, but in the classification phase, the Fuzzy C-Means clustering is applied, introducing a vast number of computation. Based on this observation, we make attempt to separate the emotion recognition algorithm into the feature extraction phase and the classification phase, then install them on the wearable and the cloud, respectively.

EEG segmentation. Before introducing the feature extraction phase on Memento, we first describe our data segmentation strategy. Algorithm 1 shows the pseudo-code of our segment strategy. The design goal is to balance the data redundancy and the

processing efficiency. We empirically apply a sliding window whose size is 10 seconds containing 2200 samples. We set 50% overlap initially. With the larger window, the emotion recognition algorithm could be feed by more comprehensive data and might provide more accurate estimation, but consumes much more time to process and lead to the greater delay between the certain event occurs and the lifelogging process is triggered. Since Memento is designed as near a real-time lifelogging system, we initially set a relatively small window and dynamically adjust the overlap size.

Leveraging the results of signal quality estimation (§4.2.1), we determine to keep the current segment or drop it. We use an adaptive threshold instead of the fixed threshold. A fixed threshold which can perform well in most possible circumstances is hardly determined. For instance, an absolute high threshold might lead to the data fragmentation while a low threshold might introduce more noisy segments.

In this paper we average the signal quality score of all validated segments, then a certain tolerance score is subtracted to ensure the diversity (line 20-26). We use it as the adaptive threshold. The invalidated segments are dropped. If the consecutive segments are invalidated, it indicates the current situation gets worse. We reduce the overlap 10% for each consecutive dropped segment. We always reset the overlap size to 50% when a segment is accepted (line 11-18). Finally we store the validated segments in the circular buffer. The buffer follows the FIFO rule (line 4-6). If there is no space for the coming segment, the existing item would be replaced.

Feature extraction. Our system fetches the EEG segments from the buffer continuously and then extracts features. Since the exact emotions would not be recognized at this phase, we consider to use the intermediate results from features to trigger lifelogging. Some features that only describe the statistic characteristics are inappropriate. In Memento we use the fractal dimension (FD) based approach. FD is an index for characterizing fractal sets by quantifying their complexity. FD calculation has been used in

Algorithm 1: Segmentation Strategy

```
1 segmentation(data);  
   Input : The EEG data after preprocessing data  
   Output: Circular buffer  
2 currentSegment  $\leftarrow$  capture data from startLoc with windowSize;  
3 isValid  $\leftarrow$  valid(currentSegment);  
4 if isValid then  
5 |   index = valid segment nums / bufferSize;  
6 |   Buffer[index]  $\leftarrow$  currentSegment;  
7 end  
8 startLoc  $\leftarrow$  advance(startLoc, isValid);  
9 return Buffer;  
10  
11 advance(startLoc, isValid);  
   Input : Last segment start index startLoc and whether it is valid isValid  
   Output: The start index for the next segment  
12 if isValid then  
13 |   consecutiveDrop  $\leftarrow$  0;  
14 else  
15 |   consecutiveDrop  $\leftarrow$  consecutiveDrop + 1;  
16 end  
17 startLoc = startLoc + windowSize * (1 - max(0, overlap - consecutiveDrop * 0.1));  
18 return startLoc;  
19  
20 valid(segment);  
   Input : The current segment segment  
   Output: If the current segment is accepted isValid  
21 qualityScore  $\leftarrow$  the signal quality score;  
22 if qualityScore  $\geq$  threshold - tolerance then  
23 |   threshold  $\leftarrow$  average score of all valid segment;  
24 |   return True;  
25 else  
26 |   return False;  
27 end  
28
```

the EEG analysis [96] and is considered to positively correlate with how energized the user is, hence the arousal level. Particularly we make use of Katz's FD calculation [99]

and apply it directly on the waveforms. We obtain the FD of a curve using

$$D = \frac{\log n}{\log \frac{d}{L} + \log n}, \quad (\text{Eq. 4.1})$$

where L is the total length of the curve and d is the diameter that can be estimated as the farthest distance with the beginning point. n is the number of the steps in the curve, which is adjusted by the granularity. Particularly we define $n = L/a$ where a is the average distance among the successive points.

Larger FD values are associated with higher arousal levels. To tolerate the absolute arousal differences among individuals, we do not use the absolute arousal level as the threshold to trigger the lifelogging procedure. We propose to use the trend of emotional changes, which is calculate as the derivation of the FD values. The FD trends represents the mental change direction, positive or negative and the intensity as well. We further use the sliding average to smooth the trend curve. Fig. 4.6 illustrates when the stimulation is applied the positive trend occurs, and the lifelogging process is triggered.

Two phase emotion classification. Finally if Memento determine to launch the lifelogging, the suitable lifelogging methods (video, audio or image) is selected and properly configured by adjusting the settings such as FPS and resolution, according to the environment conditions and the system states (§4.2.3). The lifelogging duration for video and audio capturing is determined as well in the lifelogging engine. The lifelogs and the features of EEG are compressed and saved in the storage of the smart glasses, as well as the related sensor readings including motion sensor readings, GPS and so on. We notice that although the computation and energy resources are so limited on wearables, the storage is satisfied with the lifelogs and sensing data. The data is uploaded to the private cloud (PC or smartphone) later when the smart glasses are recharging or upon the user requests. On the cloud, we run the sophisticated emotion recognition algorithm

Figure 4.6: Using the arousal trends as the trigger to perform lifelogging

to recognize the emotions. Particularly we adapt the emotion recognition algorithms from [96].

4.2.3 Lifelogging Engine

Memento automatically launches the lifelogging process once the emotion event is detected. Generally the camera and microphone on wearables can serve as the major media and lifelogs can be recorded in the forms of video, audio or pictures. In ideal cases the video clips contain almost all the details of the audios and pictures. However, in practical the environment conditions impact the lifelogging qualities. For instance, the poor light condition lead to low qualities of captured videos. The vigorous motions might make the camera go out of focus, in turn lead to blurred pictures. On the other hand, each lifelogging method with specific settings, e.g. resolution, FPS and sample rate, has the different energy profile. Table 4.1 shows the power draw and the estimated battery

life for audio recording, image capturing and video capturing on Google Glass. We notice that the energy consumption of video capturing is almost 5 times of the audio recording. Meanwhile increasing the settings makes the device consume more energy [100] as well. Therefore it is crucial to balance the lifelogging qualities and the energy consumption.

To tackle this problem, we design the lifelogging engine, which automatically selects and configures the suitable lifelogging method with the analysis of the environment conditions and the system states. By sensing the current environment conditions including light conditions, movements and noise levels, we infer the expected qualities of each lifelogging method and formulate it as *Utility*. We then formulate the *Cost* of each lifelogging method in terms of the energy consumption and the resource usage. We also consider the device surface temperature to ensure the user-comfort on wearables. Finally the lifelogging engine selects the proper lifelogging method by minimizing the *Cost* and maximizing the *Utility*. Then the lifelogging process is launched by using the selected lifelogging method. Next we describe the design in details.

Lifelogging Methods	Power Draw	Battery Life
Idle	18 mW	116 hours
Audio recording	642 mW	3.2 hours
Image capturing	2842 mW	0.8 hours
Video capturing	3008 mW	0.7 hours

Table 4.1: Energy profiles of different lifelogging methods on Google Glass

Utility. We design the utility function to represent the expected quality of a lifelog. The function assess the utility based on two criteria: how poor the environment conditions are (u_e) and what lifelogging configurations are used (u_c). We apply a linear combination of u_e and the reciprocal of u_e with the predefined weights:

$$u^m = w_e^m (u_e^m)^{-1} + w_c^m u_c^m, \quad (\text{Eq. 4.2})$$

where $m \in \{\text{video}, \text{audio}, \text{photo}\}$ representing each lifelogging method, respectively.

To obtain u_e^m we consider three types of environmental factors: the light condition, the movement level and the noise level. We observe each lifelogging methods is able to tolerate one or few negative impacts from the environment. For instance, The light condition cannot impact the audio quality and the noise level cannot influence the photo quality. We thus formulate u_e^m for each lifelogging method as the followings:

$$\begin{aligned} u_e^v &= w_l^v \theta_l + w_m^v \theta_m + w_n^v \theta_n, \\ u_e^a &= w_n^a \theta_n, \\ u_e^p &= w_l^p \theta_l + w_m^p \theta_m, \end{aligned}$$

where $\theta_i, i \in \{\underline{light}, \underline{movement}, \underline{noise}\}$ stands for the current environment conditions and respective w_i^m is the assigned weight. We sense the light condition θ_l via the on-board ambient light sensor, which provides the light intensity in luminance.

We extract the movement level θ_m from the accelerometer readings. We borrow the results from the movement estimator in §4.2.1. Then we adapt and simplify the algorithm from [48] to identify the movement state into one of {stationary, walk, ride}. To obtain the acoustic noise level, a one-second acoustic sample is collected. From the sample we extract the sound pressure level (SPL) as θ_n by using the build-in API.

To obtain u_e^m , we consider four types of configurations: resolution, frame rate, sampling rate and acoustic channel. We get the top configurations the device can support. Then we get u_e^m by using the ratio of the current configurations to the top configurations:

$$u_c^m = \text{current_settings} / \text{top_settings} \quad (\text{Eq. 4.3})$$

We notice the *Utility* is so device-specific and task-specific. On the one hand, the quality of sensed lifelogs relies on the performance of camera and microphone on the

device, where some optimization approaches might be applied to tolerate the poor environment. On the other hand, the quality requirements for lifelog might be very specific according to particular tasks, i.e. object recognition, speech recognition, etc.. Therefore the weights in the formulation above should be conservatively adjusted on different devices, for particular tasks. Here we demonstrate to find the weights w on the Google Glass by performing a face detection task on video lifelogs, which is also conducted in the later evaluation (4.4.3). We use the F1-score [101] of the face detection tasks as *Utility* train set and apply the regression to find the corresponding weights.

Cost and system Capacities. We assess *Cost* in three aspects: the energy consumption c_e^m , the resource usage c_r^m and the temperature increment c_t^m :

$$c^m = w_e c_e^m + w_r c_r^m + w_t c_t^m, \quad (\text{Eq. 4.4})$$

where $m \in \{\underline{v}ideo, \underline{a}udio, \underline{p}hoto\}$ standing for the lifelogging method. w_e , w_r and w_t is the particular weights for each cost components. In the current implementation, we make use of the equal weights.

We define the energy consumption c_e^m as the function related to the lifelogging duration t linearly $c_e^m = p^m t$, where p^m is the power of lifelogging. We experimentally get p^m via the off-line tests on wearables. The results on Google Glass are shown in Table 4.1. For the photo capturing, we set p^p as the average power when taking photos consecutively. The capturing interval is fixed and set to 5 seconds. With the lifelogging settings changing, we scale the power p^m proportionally.

We formulate the resource usage c_r^m by considering the CPU and memory utilization of each lifelogging method. We set them as the constants C_{cpu} and C_{memory} and learn them from the off-line tests on wearables.

Usually the smart glasses directly touches the skin of users. The high device temperature leads to the degraded user-comfort. Thus we also consider the device temperature

of wearables and ensure it will not exceed the certain constraint. Since CPU is the major heat contributor, we apply a simple linear model related to the CPU load to predict the temperature increment over the duration t .

Finally we also access *Capacity* in terms of battery budge cap_e , the CPU and memory utilization cap_u and the device temperature cap_t :

$$\begin{aligned} cap_b &= \theta_b Budget, \\ cap_u &= \theta_u Memory + \theta_u CPU, \\ cap_t &= T_{max}, \end{aligned} \tag{Eq. 4.5}$$

where we experimentally set the energy budget constraint θ_b as 90% of the full battery life and resource utilization constraint θ_u as 95% of the memory quota. We set the temperature constraint T_{max} as 48°C to ensure the user-comfort according to [102]. We inquire the current battery life, the available resource quota and the CPU temperature by the system APIs.

Optimization framework. We define *Utility* and *Cost*, as well as *Capacity*. In order to select the suitable lifelogging method, we try to maximize the utilities and minimize the costs. We regard it as a combinatorial optimization problem, specifically, the 0-1 knapsack problem [103]. Each lifelogging method with certain settings has the estimated utility and the costs in terms of the energy consumption, the resource usage and the heat increment. The *knapsacks* represent the current capacities, which are the battery budget, the available CPU and memory quota and the device temperature. The optimization objective is to find the one that maximizes the utility (Equation (Eq. 4.2)) while making the cost (Equation (Eq. 4.4)) does not exceed the energy, resource and temperature capacities (Equation (Eq. 4.5)). After solving this optimization problem, we get a suitable lifelogging method with the specific settings as well as the lifelogging duration for video and audio capturing. To further reduce the optimization search space,

the fixed and predefined configurations are used. We list them in Table 4.2. We also provide a set of predefined lifelogging duration to speed up the optimization process, which is $t = 30 + i \cdot 60$ seconds, for i we set the range from 0 to 9.

Method	Item	Settings		
Video	Resolution	320*180	640*360	1280*720
	Frame Rate	15 FPS	30 FPS	30 FPS
Audio	Sampling Rate	16K Hz	22K Hz	44K Hz
	Channel	Mono	Stereo	Stereo
Image	Resolution	640*360	1280*720	1920*1080

Table 4.2: Pre-defined lifelogging configurations in Memento

4.2.4 Computation offloading

We describe the design of Memento on smart glasses. Now we elaborate on the optimization that aims at maximizing the battery life of wearables. We make use of the concept of offloading. Instead of leveraging the cloud, we make attempts to offload tasks to the user’s personal smartphone. We make such a design due to the following considerations: first, we think Memento should be used in various circumstances, where the Internet connections to the cloud might be not always reliable. In such cases the benefits of offloading cannot be guaranteed. Furthermore the lifelogs and the related sensed data are private. Offloading them to the third-party servers might raise potential privacy issues. Second, we observe that most wearable users also have smartphones. Comparing to the wearables, smartphones have more energy budget and stronger computing power.

Inspired by such heterogeneous computation capacities around users, we propose to offload tasks to the user’s personal smartphone. Particularly we install the same programs and algorithms of the signal processing, emotion event detection. Once the smartphone is in use, all the sensed data including EEG data and motion sensor readings are processed and analyzed on the phone until the certain emotion event is detected. Then the smartphone in turn notifies the smart glasses to trigger the lifelogging engine.

We switch the computation between the smart glasses and the smartphone according to three criteria:

- 1) When the CPU on the smartphone is awake for a period of time, we offload tasks to the smartphone due to the Piggyback effect [65]. We try to avoid waking the smartphone from the idle state to perform tasks, because the tasks are simple though, the CPU and many related subsystems must be activated, which consume lots of energy. We piggyback the tasks on the smartphone in order to maximize the CPU utilization and save the battery life.
- 2) When the battery budget on the smart glasses is lower than a threshold, we offload tasks to the smartphone since we try to use to record as many lifelog as possible before the battery get died. Memento allows the user to set the battery threshold and by default we set it as 15% of the full battery life.
- 3) When the system of the smartphone is busy, we give back the tasks to the smart glasses in order to ensure the responsiveness. We monitor the circular buffer mentioned in §4.2.2. The time difference between the oldest and newest segment in the buffer indicates if the data is processed in time. We check the time difference when fetching the segment and send tasks back to the smart glasses if it is longer than 240 seconds in our implementation.

4.3 Implementation

We close the description of Memento by introducing the specific implementations of current prototype.

Hardware. As illustrated in Fig. 4.7.a, the current prototype is comprised of two commercial devices: a Muse EEG headband [104] and a Google Glass Explorer Edition.

4.7.a: hardwares used in the prototype. 4.7.b: UI of Memento on smartphones. 4.7.c: electrode locations in 10-20 system.

Figure 4.7: The hardware and software implementation of Memento prototype.

Muse headband is designed as a personal meditation assistant, which helps user for meditation exercises. Four channels are supported on the Muse headband and they are illustrated by 10-20 system [97] in Fig. 4.7.c. Besides two pairs of channel electrodes and one reference electrode, the Muse headband is also equipped with an accelerometer. A PIC24 MCU is used to conduct simple processing operations e.g. low-pass filtering and band filtering. The headband has a separated battery with the size of 250mAh. Only Bluetooth can be used for the data communication on the Muse headband.

We use a Google Glass as the smart glasses platform, since it is one of the most completed smart glasses we can find on the market. Android OS on the Google Glass also provides the openness and easy development to developers. The core of the Google Glass is the OMAP4430 system-on-chip (Soc), which owns a dual-core ARM Cortex-A9 CPU, 2BG RAM and 16GB flash storage. Its rich sensors satisfy the requirements of lifelogging and sensing, including microphone, camera, accelerometer, gyroscope, ambient light sensor and so on. The power supply comes from a 570mAh liPo battery. Both Wi-Fi and Bluetooth are supported.

In our prototype, raw EEG reading are preprocessed by the band-pass filter on the

Muse headband. Since the headband is not programmable, the rest of the processing operations are done on the Google Glass. EEG data is pushed from the headband to the glasses via Bluetooth. We demonstrate the smartphone services on a LG Nexus 5 and a Samsung Galaxy. The back-end service runs on our experiment workstation with a Intel Xeon processor, 16GB RAM and 1T disk.

Software. We implement the Memento functionalities in three software packages and deploy them on the smart glasses, the smartphone and the back-end server, respectively.

- We implement the preprocessing and lifelogging functions as an Android Wear application. Most of the code is written in Java. For the common utility algorithms used in the data cleaning and feature extracting e.g. Goertzel algorithm, Katz's FD calculation, etc., we implement them as the executed libraries in native code.
- Memento service on the smartphone. The service on the smartphone includes two major features. First, we build the offloading host as a standard Android service. The service keeps in sleep until the certain offloading requests are received through the notification mechanism. Since the smartphone runs Android as well, the service share the same set of native libraries and the sub-set of Java code with the Memento application on the Google Glass. We use Wi-Fi Ad hoc network to setup the connection between the smartphone and the glasses. Second, we implement the application with GUI to browse the lifelogs in the private cloud and the Google Glass. The application is also used to share the lifelogs to the SNS easily. Fig. 4.7.b shows the current user interface on Android.
- Back-end server on the private cloud. We implement the full version of the emotion recognition on the back-end server in Java. We setup a web service on the server. Lifelogs along with related sensing data uploaded through the encrypted links. The back-end sever also provides the features of querying and browsing lifelogs.

Lifelog Store. Lifelogs are distributed both on wearables and the private cloud. On the Google Glass, media lifelogs are stored in the raw format. Before storing the EEG features, we compress them to reduce the size. We use SQLite database to maintain the index. An entry in the index represents a lifelogging event, which contains the corresponding media lifelogs and EEG data. Currently we have four attributions in each entry, which are the timestamp, the location obtained from GPS on the smartphone, the file pointers of the media lifelogs and the EEG features. The lifelogs on the Google Glass are uploaded to the private cloud when the glasses are in charge or upon the user’s requests. On the private cloud, lifelogs are stored in the similar structure with on the glasses, but the emotion tags are added. Each entry is further assigned one type of basic emotions, along with its arousal and valence levels.

4.4 Evaluation

In this section, we present our evaluation on Memento. We first introduce the dataset and the experiment settings. Then we evaluate Memento in terms of the system performance and the energy consumption.

4.4.1 Dataset

DEAP as the ground truth. We are given the access to DEAP [105], which is an open dataset for the analysis of human affective states. In DEAP EEG signals of 32 participants are recorded by the dedicated equipments with 32 electrode channels, as each watches 40 one-minute videos. The self-assessment tool [106] is used to let participants rate each video in terms of the level (1 to 9) of arousal, valence, dominance and so on. In total the size of DEAP is about 2.7GB and the video used can be assessed via Internet. The collected signals of DEAP are from the dedicated equipment and the results of the

emotion assessments are obtained by the widely used self-assessment tool. Therefore we make use of DEAP as the ground truth.

EEG data collected in the lab. Using one-minute videos in DEAP mentioned above as the stimulations, we collect the EEG readings by Memento. The raw EEG readings are recorded. This experiment is conducted in the lab, the participant is not allowed to move around. In total 40 pieces of raw EEG signals are recorded corresponding to each one-minute video in DEAP.

Lifelogs, EEG and sensor data collected by customized Memento. We collect the EEG readings, motion sensor readings and lifelogs by Memento under different scenarios, including the laboratory, the office, the street and the park. We build a customized Memento for the data collection. Instead of automatically launching the lifelogging process, the customized Memento continuously records lifelogs in the form of video. When the significant emotion changes occur and Memento is triggered to performance lifelogging, the event is marked. We add an external battery to supply power to the Google Glass in order to make it continuously capture videos at the best quality. Totally we get about 6 hours videos and sensor data traces.

Energy consumption trace. We use Monsoon power monitor to collect the real time energy consumption traces of the Google Glass and the smartphones.

4.4.2 Emotion Recognition Performance

To evaluate the emotion recognition accuracy, we use Manhattan distance on the Arousal-Valence model to measure the differences between emotion samples.

First we run our full emotion recognition algorithm on DEAP EEG data and compare the results with the self-assessment value in DEAP. For each one-minute video, we have a sequence of EEG segments. Every segment is processed to extract the recognized emotion, which is represented by a pair of values on the Arousal-Valence model. We compare

4.8.a: emotion recognition on DEAP data V.S. self-assessment in DEAP

4.8.b: emotion recognition on the data collected by Memento V.S. the self-assessment in DEAP

4.8.c: emotion recognition on the data of DEAP V.S. Memento

Figure 4.8: Emotion recognition on the data of DEAP and collected by Memento.

every point with the corresponding self-assessment value and calculate the distances. Fig. 4.8.a illustrates the distance distribution of all 40 test cases. For each test case, we calculate Root Mean Square (RMS) between recognized emotions and self-assessment value. Overall the average RMS is 2.42 and the maximal and minimal RMS is 4.25 and 0.90, respectively. Among them, 60% of cases is less than 2.5 and the RMS of over 80% test cases are less than 3.29. We use this result as the base line.

In the similar way, we run the emotion recognition algorithm on the data collected by Memento. We show the details in Fig. 4.8.b. Overall the average RMS is 2.76 and

the maximal and minimal RMS is 4.22 and 1.5, respectively. In all the test cases, the RMS of more than 80% cases is less than 3.66. Comparing to the base line, the emotion recognition performance reduction is around 12%.

We further compare the recognition results between on the DEAP data and on the data collected by Memento. We align the recognized values in the time sequence and calculate the distance between each two values who are closed in time. Illustrated in 4.8.c, the average RMS 2.12 is obtain and RMS from more than 80% cases is less than 2.7.

Second we evaluate the performance of our preprocessing and arousal extracting algorithms. We use the performance obtained in the previous experiment as the baseline and compare against three modified processes that exclude either the non-brain removal, the signal quality estimation or both of them. In this experiment, we run tests on the data collected on Memento and calculate the average RMS with the self-assessment value on the whole 40 test cases. In Table 4.3 we list the results. From the table we observe that 1) the full preprocessing design provides more than 32% better performance comparing to the straightforward design (band-pass filter only). 2) Among the components of the preprocessing algorithm, the non-brain removal brings about 24% improvement. We think it is attributed to the fact that the non-brain noise is universal. 3) About 9% is brought by the signal quality estimation. The reason is that we collect the data in a controlled lab environment. The signal quality trends staying stable.

Process	Averaged RMS
Full design	2.76
Band pass filter (BPF)	3.63
BPF + Non-brain removal	2.92
BPF + Signal quality estimation	3.32

Table 4.3: Breakdown emotion recognition performance.

4.4.3 Lifelogging performance

Next we evaluate Memento by a user study to answer the question: how the emotion-driven lifelogging approach meet the users' expectations.

Comparing to the previous continuous lifelogging or manual lifelogging, it is necessary to learn if the emotion driven lifelogs meet the users' expectations. We conduct a simple user study. From the continuous lifelogs collected in the experiment, we ask the volunteers to manually select the memorized moments. For each lifelog, the volunteer labels the corresponding start/end time. In order to completely derive the memorized moments, the volunteers are asked to select clips in multiple rounds. In every round, the clips selected in the previous rounds are skipped. We apply three selection rounds and use these user selected clips as the base line.

We compare the lifelogs marked by Memento with the base line. We particularly categorize the comparison result into the following buckets: (i) For the video lifelog, we evaluate the overlap between the clips marked by Memento and the selected video clips. If the overlap rate is more than 50%, we label it as *fit*. If the overlap is ranged from 30%-50%, then we label it as *semi-fit*. We mark the video lifelog as *none-fit* if the overlap is less than 30%. (ii) We use the similar way to categorize the comparison results of the audio lifelogs but set stricter conditions due to the less information contained in the audio lifelogs. We set *fit*, *semi-fit*, *none-fit* when the overlap rate more than 65%, 45%-65% and less than 45%, respectively (iii) For the photo lifelogs, we labels it as *fit* if it appears in the selected videos, *none-fit* otherwise.

From the 6 hours lifeog traces, Memento marks 16 pieces of lifelogs in total. Fig. 4.9 shows the comparison results. We observe that more than 40% are labeled as *fit* and more than 75% are marked as *semi-fit* and upper.

We also evaluate the coverage of the selected videos against the lifelogs collected by Memento. We define the selected video is not covered when the overlap with the Memento

Figure 4.9: How the emotion-driven lifelogs fit the users' expectations.

Figure 4.10: The coverage ratio of the emotion-driven lifelogs against the clips selected by users.

lifelogs in any forms is less than 30%, covered otherwise. From the 6 hours lifelogging traces, the volunteers manually select 24 clips in total. Among them more than 70% lifelogs are covered. When we narrow down the test case space into the clips only from the selection first round, Memento achieves about 83% coverage.

Memento automatically selects the suitable lifelogging method according to the environment and system status. The gains mainly comes from two aspects: Memento selects appropriate lifelogging method which tolerates the current poor environment conditions, and adjusts settings to balance the energy consumption and the lifelog qualities. To understand the quality of emotion-driven lifelogs, we conduct a face detection experiment. We envision that the lifelogs would be used in some machine learning tasks, for instance object recognition, semantic extraction and so on. Face detection is a common task in Computer Vision (CV). We leverage the performance of the face detection to evaluate the qualities of lifelogs.

Using the lifelogs collected by the customized Memento, we run the tests on two cases. 1) We find the lifelogs marked as lifelogging in video. We clip out the video lifelogs at the corresponding time slots as the base line. We further scale the resolution, FPS and so on

according to the settings determined by the lifelogging engine. We regard them as the video lifelogs degraded by Memento. 2) We get the lifelogs marked as lifelogging in audios or pictures. We clip out the videos from the continuous videos at the certain time slots. We regard them as the video lifelogs collected without Memento. We use an open-sourced face detection library¹ and just apply the default parameters. We select 6 one-minutes lifelogs from the first case and 4 lifelogs in the second case. We manually label and count the faces appeared in these 16 lifelogs as the ground truth. We run the face detection algorithm on each frames of the video. Fig. 4.11 details the detection results. In the first case, to balance the energy consumption the lifelogging engine degrades the lifelogging settings. The performance reduction of face detection, however is very limited comparing to the ground truth. In the second case, a simple continuous lifelogging leads to the poor performance. From the results we also get the insights that compared with the lifelogging settings, the environment influence the lifelog quality more. Therefore it is essential to select the suitable lifelogging methods according to the current environment conditions.

4.4.4 Energy consumption

Next we evaluate the energy consumption of Memento. The EEG sensing overhead mainly comes from the EEG electrodes and the MCU on the EEG headband. Since the APIs provided by the Muse EEG headband are limited, we are not allowed to test in more comprehensive settings. The power consumption of the headband is stable during our experiments. The average battery life of the Muse EEG headband is about 4.2 hours with the battery size of 250mAh.

On the smart glasses, Monsoon Power monitor is used to collect the real time energy consumption traces under the following circumstances: (i) Memento continuously receives the EEG and motion data. We regard it as the sensing overhead. (ii) Memento cleans

¹https://github.com/ageitgey/face_recognition

Figure 4.11: Lifelog quality evaluation using face count.

the data, extracts the FD values and computes the arousal trends. We label it as the processing overhead. (iii) When the certain arousal changes are detected, the lifelog collector is triggered to perform lifelogging in the form of video. We regard it as the lifelogging overhead.

Fig. 4.12 illustrates the real time energy profiles of these three circumstances. We have the following observations: (i) Compared with the sensing overhead (589mW), the additional overhead brought by the processing is less than 200mW. (ii) The energy consumption of the lifelogging, especially video capturing is a burden, which is about 1200mW. Currently we cannot avoid it due to the existing architecture of the camera and OS. However, since we apply an emotion driven approach, the lifelogging process is only triggered opportunistically. (iii) Receiving data also brings the overhead in our implementation. Basically it is closed to the lowest energy to hold the CPU wake-lock. We think it could be largely reduced or even removed if EEG electrodes could be

Figure 4.12: Energy consumption of each component.

physically connected with the smart glasses and Low Power Unit (LPU) could be used. We regard it as our future work.

4.5 Related Work

Memento touches several different research areas. In this section, we survey the prior work which is the most related to Memento mainly in the following two aspects:

Emotion recognition and its applications. Emotion is one of the human natures and can be inferred in several ways [107]. Existing work exploits the recognition of emotions by leveraging the acoustic [108] or the visual signals [109–111], where the facial or spoken expressions are carefully analyzed. Comparing to the voice and video, the physiological signals are more direct indicators of how we are feeling. The heart beats, breath rates and blood pressures are used to extract affective states in [112, 113].

There are a number of EEG based emotion recognition algorithms proposed in recent

years. For example, to classify basic emotions, support vector machine is used in [94]. In [95], wavelet based methods are applied to extract features, fuzzy k-means and fuzzy c-means clustering are considered to do the classification. Besides the statistic based methods and wavelet transform based methods, more approaches are considered. In [114], higher order crossing analysis (HOC) is adopted. Authors in [96] make use of the fractal dimension based algorithm to quantify the basic emotions. However, these algorithms mainly focus the emotion recognition accuracy. The time constraints and the energy consumptions are rarely considered. Most of them are proposed for offline recognition process. [96] builds a real time emotion recognition system but it is not affordable for the wearable-class hardware in terms of both the computation capacities and the battery life.

Recent years a devise of sensing algorithms on smartphones are proposed to offer a rich set of user signals, including affective activities. Some indirect approaches are explored. EmotionSense [115] builds an audio based emotion recognition on smartphones. In [116, 117], the relationship between the phone usage pattern and the personalities is discussed. The authors of MoodScope [118] propose their findings that by analyzing the communication history and application usage patterns, users' daily mood can be inferred accurately. Comparing to these systems, Memento leverages EEG directly and interprets the users' affective activities in the perspective of physiological properties.

Lifelogging system. The origin goal of the lifelogging system is to digitize the users' activities and to provide the query services. In MyLifeBits [119], various activities are collected, including emails, photos, presentation slides, home videos and audio recordings. It is considered one of the first bite to the comprehensive lifelogging system. In the past decades, many lifelogging systems are proposed. We categorized them into two groups: dedicated equipments for certain purposes and lifelogging systems on personal mobile devices, smartphones specifically. SenseCam [8] is a body mounted camera, which

passively captures photos through a wide-angle lens. SenseCam is proposed for the memory aid. Extending it, Footprint Tracker [10] studies the effects of multiple memory cues. In [9], a ubiquitous lifelogging system is also designed for the episodic memory impairment. Recently wearables gain the popularities. Some wearable cameras [13, 14, 120] can be used to log daily lives. However, most of these lifelogging devices are designed to passively log or request users' intervention. Memento aims to provide a automatic, affordable lifelogging solution on COTS smart glasses.

Lifelogging systems on smartphones are also emerging. Leveraging the rich on-board sensors and sensing algorithms, some context-aware approaches [121, 122] are proposed and can be used to improve lifelogging. Experience Explorer [123] not only senses and captures contextual lifelogs but also provides social network features. Lifelogger [124] collects a diverse of sensor data and focuses on providing the robust lifelog query services. In UbiqLoq [91], authors propose a lightweight framework allowing developers easily create lifelogging application based on it. Recent ZOE [66] leverages the advanced system-on-chip(Soc) techniques. It supports continuously sensing a number of user activities(physical, personal and social) and provides the dialog based user interaction. These lifelogging systems are developed to record the external environment. Meanwhile the internal experience of people is important and helpful for lifelogging as well [125]. Memento makes attempts to introduce the user emotions to improve the quality of lifelogs.

4.6 Discussions

In this section, we briefly discuss the issues related to the current design and implementation of Memento.

LPU utilization. One of the problems Memento tackles is the energy bottlenecks. Heterogeneous computation could significantly reduce the energy consumption. In our implementation, the Muse headband uses a micro control unit to apply the preprocessing

operations on the EEG signals such as smoothing and band-pass filtering. On the Google Glass, currently the co-processor or LPU is not utilized, since Android does not open the related API. In this paper, we focus on showing the proof-of-concept of the emotion-driven lifelogging system, and make it affordable on COTS wearables. For the future work, we plan to design the custom hardware to integrate EEG sensors physically, and utilize the LPU to improve the performance.

User study. The previous work [125] has pointed out the importance of the affective experience for lifelogging systems. The expected lifelogging systems on wearables, especially on smart glasses are discussed in [126]. Instead of applying a wide range of user study urgently, in this thesis we propose a series of designs and show the possibility of the emotion driven lifelogging. Via Memento, we tackle the problems mentioned in §1.2, which we believe are essential to design such a system. In future, we plan to explore some interesting user studies including how the user evaluate the lifelogs collected by Memento, how the lifelogs with emotion enhance the SNS, health-care and other services. We also plan to make use of the user feedback to improve our system in the future work.

In this paper, we present the design and implementation of Memento, an emotion driven lifelogging system on wearables. Memento senses the emotional changes of users and automatically launches the lifelogging based on that. To the best of our knowledge, Memento is the first of its kind lifelogging system. Through a series of techniques, Memento integrates EEG and proposes the two phase emotion recognition that makes it efficient and affordable on wearables. Finally Memento outputs lifelogs tagged with emotion information which we believe could enhance many existing services.

Chapter 5

Conclusion and Future Work

In this report, we introduce our experience of building mobile sensing system on smartphones and wearables.

We first propose PDS, a phantom data usage detection system. PDS makes the advances towards automatically detecting phantom data usage on smartphones. To achieve the goal, we present a characterization study of PDU based on about 400 data usage issues in real world. We also make the observation that the context affects how the users use applications and each user preserves specific data usage patterns under particular context. Based on it, we present PDS, a system tool to automatically detect PDU. The evaluation results shows PDS achieves about 72% of PDU cases can be successfully detected by PDS and confirmed by users.

Then we present Memento, which is an emotion driven lifelogging system on wearables. Memento senses the emotional changes of users and automatically launches the lifelogging based on that. To the best of our knowledge, Memento is the first of its kind lifelogging system. Through a series of techniques, Memento integrates EEG and proposes the two phase emotion recognition that makes it efficient and affordable on wearables. Finally Memento outputs lifelogs tagged with emotion information which we believe could enhance many existing services.

In the future, we would still focus on building and continuously improving the sensing enhanced mobile computing systems on smart devices. We see some potential opportunities and directions which could help us to achieve this goal:

Sophisticated machine learning techniques. Data science and machine learning techniques rapidly advance in recent years, especially for artificial neural network and deep learning techniques. Deep learning techniques can be used in mobile sensing systems to extract knowledge from massive sensing data. For example, MobileDeepPill [127] proposes to use a multi-CNN model to recognize the medical pill by the pictures on smartphones. DeepEar [51] leverages the deep learning techniques to build a common audio sensing framework. Though various machine learning techniques are promising, there are still several critical problems before such techniques can benefit mobile sensing systems. In the perspective of smart devices, the existing algorithms are usually too heavy to be adapted directly onto mobile platforms. How to adjust them to fully leverage their power while considering the practical limited resource and the user experience requirement is challenging. On the other hand, in the perspective of data science, introducing human beings into the machine learning loop is promising. As a sense of human being, we see more opportunities of sensing techniques on smart devices. How to intelligently adjust and integrate the mobile sensing systems into the distributed learning systems in large scale is truly interesting. Therefore this is one of our future directions.

Heterogeneous computations on smart device. Battery life is always a serious concern for mobile systems. The problem would get worse when more computation is required for machine learning tasks on smart devices. Though some new battery technologies are proposed, we still believe the battery issues would still be there, considering the tiny size of devices and the high energy consumptions of applications. To cope with this issue, one of the possible solutions is to leverage heterogeneous computing on smart devices, including main CPU, GPU, co-processor, various micro-controllers (MCU). These processing units

have their own different capacities and energy profiles. It would benefit a lot if we can integrate them in an appropriate way for a certain goal. For instance, LEO [67] tries to schedule different sensing tasks onto different computing units in order to save the battery life, by performing both frequent and joint optimizations. In [128], authors propose to offload the convolutional layers to mobile GPU and accelerate the processing so that deep learning algorithms could be run purely on mobile platforms. Besides the heterogeneous computing units on mobile, there are also more computation resources around smart devices, including personal PCs, personal workstations, Cloudlets [73] and Clouds, etc. Designing the proper offloading scheme among these resources and fully leveraging them would be interesting. Furthermore we also see some unique constraints for mobile systems, especially for wearables, e.g. the responsiveness (near real-time processing) and the user-comfort (heat control), etc. We make the very first attempt in Memento and in the future exploiting heterogeneous computations on/around smart devices is still one of our future directions.

Beyond sensors on smart devices. Recently more and more types of sensors are exploited to improve mobile sensing systems and provides new innovative applications. On the other hand, smartphones and mobile technologies are transformative to the deployment of biomolecular diagnostics as well. In Memento, we make attempts to integrate EEG into lifelogging systems on wearables. The gains we expect are mainly two aspects: first we could provide emotion information for each lifelog and second mental changes act as an efficient trigger for the lifelogging systems. Beyond existing sensors on smart devices, we also see some potential sensors which could provide opportunities for us to in-depth understand human activities. For example, Electromyography (EMG) is a measure of muscle activities. In [129], authors integrates EMG with smart glasses to analyze chewing activities and in turn detect the intake of various food. Electrocardiogram (ECG) is the process of recordings the electric activities of heart. [130] considers ECG as

an important input to infer the moods of users. Furthermore with the growth of Internet of Things (IoT), more and more tiny IoT devices are deployed in the environment. Such devices are not only equipped with network and computation capacities, but also can possible provide sensing interfaces to achieve better understandings of environment context for us. Thus they could be the extra sensors around our mobile devices. Making use of human-centric approach and leveraging more types of sensors and inputs to improve the mobile sensing system would be one of our future directions.

Appendix A

Author's Publications

- (i) **Shiqi Jiang**, Pengfei Zhou, and Mo Li. 2016. "Detecting Phantom Data Usage on smartphones with analysis of contextual information", in International Journal of Distributed Sensor Networks, Volume 2015, January 2015, Article 3.
- (ii) Pengfei Zhou, **Shiqi Jiang** and Mo Li, "Urban Traffic Monitoring with the Help of Bus Riders", in 2015 IEEE 35th International Conference on Distributed Computing Systems, Columbus, OH, 2015, pp. 21-30.
- (iii) Zhidan Liu, **Shiqi Jiang**, Pengfei Zhou and Mo Li, "A Participatory Urban Traffic Monitoring System: The Power of Bus Riders", in IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 10, pp. 2851-2864, Oct. 2017.
- (iv) **Shiqi Jiang**, Pengfei Zhou, Zhenjiang Li and Mo Li, "Memento: An Emotion Driven Lifelogging System with Wearables", in 2017 IEEE 26th International Conference on Computer Communications and Networks, Vancouver, Canada, 2017.
- (v) **Shiqi Jiang**, Pengfei Zhou, Zhenjiang Li and Mo Li, "An Emotion Driven Lifelogging System with Wearables", submitted to IEEE Transactions on Mobile Computing.

References

- [1] “Number of mobile phone users worldwide from 2013 to 2019,” <https://goo.gl/6N9vDs>.
- [2] C. S. Inc., “Cisco visual networking index: Global mobile data traffic forecast update, 2016-2021,” <http://goo.gl/Z6U9nI>, 2017.
- [3] “More than thirty percent of users believe their mobile data is stolen,” <http://goo.gl/eINHMz>, 2013.
- [4] C. Starrett, “U.s. iphone users reporting phantom data usage,” <http://goo.gl/jO9XVu>, 2010.
- [5] “My data manager,” <http://goo.gl/vZWmL0>.
- [6] A. Lakhina, M. Crovella, and C. Diot, “Mining anomalies using traffic feature distributions,” *SIGCOMM Comput. Commun. Rev.*, 2005.
- [7] T. Qin, X. Guan, W. Li, P. Wang, and Q. Huang, “Monitoring abnormal network traffic based on blind source separation approach,” *J. Netw. Comput. Appl.*, 2011.
- [8] S. Hodges, L. Williams, E. Berry, S. Izadi, J. Srinivasan, A. Butler, G. Smyth, N. Kapur, and K. Wood, “Sensecam: A retrospective memory aid,” in *Proceedings of the 8th International Conference on Ubiquitous Computing*, ser. UbiComp’06, 2006, pp. 177–193.

- [9] M. L. Lee and A. K. Dey, “Lifelogging memory appliance for people with episodic memory impairment,” in *Proceedings of the 10th International Conference on Ubiquitous Computing*, ser. UbiComp ’08, 2008, pp. 44–53.
- [10] R. Gouveia and E. Karapanos, “Footprint tracker: Supporting diary studies with lifelogging,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’13. New York, NY, USA: ACM, 2013, pp. 2921–2930.
- [11] “Everyday,” <http://everyday-app.com/>.
- [12] “Instant,” <http://instantapp.today/>.
- [13] “Narrative clip,” <http://getnarrative.com/>.
- [14] “Vicon,” <https://www.vicon.com/>.
- [15] “Electroencephalography,” <https://goo.gl/okX9pC>.
- [16] R. Rosen, “Anticipatory systems,” *Anticipatory systems*, pp. 313–370, 2012.
- [17] Z. Yang, C. Wu, and Y. Liu, “Locating in fingerprint space: Wireless indoor localization with little human intervention,” in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom ’12, 2012.
- [18] J. Xiong, K. Sundaresan, and K. Jamieson, “Tonetrack: Leveraging frequency-agile radios for time-based indoor wireless localization,” in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ser. MobiCom ’15, 2015.
- [19] I. Constandache, R. R. Choudhury, and I. Rhee, “Towards mobile phone localization without war-driving,” in *Proceedings of the 29th Conference on Information Communications*, ser. INFOCOM’10, 2010.

- [20] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao, “Travi-navi: Self-deployable indoor navigation system,” in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom ’14, 2014.
- [21] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao, “A reliable and accurate indoor localization method using phone inertial sensors,” in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp ’12, 2012.
- [22] S. Nirjon, J. Liu, G. DeJean, B. Priyantha, Y. Jin, and T. Hart, “Coin-gps: Indoor localization from direct gps receiving,” in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’14, 2014.
- [23] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær, “Entracked: Energy-efficient robust position tracking for mobile devices,” in *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’09, 2009.
- [24] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao, “Energy-accuracy trade-off for continuous mobile device location,” in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’10, 2010.
- [25] J. Paek, J. Kim, and R. Govindan, “Energy-efficient rate-adaptive gps-based positioning for smartphones,” in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’10, 2010.
- [26] E. Miluzzo, N. D. Lane, S. B. Eisenman, and A. T. Campbell, “Cenceme: Injecting sensing presence into social networking applications,” in *Proceedings of the 2Nd European Conference on Smart Sensing and Context*, ser. EuroSSC’07, 2007.
- [27] S. Hemminki, P. Nurmi, and S. Tarkoma, “Accelerometer-based transportation mode detection on smartphones,” in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys ’13, 2013.

- [28] H. Wen, J. Ramos Rojas, and A. K. Dey, “Serendipity: Finger gesture recognition using an off-the-shelf smartwatch,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI ’16, 2016.
- [29] B. Aguiar, T. Rocha, J. Silva, and I. Sousa, “Accelerometer-based fall detection for smartphones,” in *2014 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, 2014.
- [30] P. Zhou, M. Li, and G. Shen, “Use it free: Instantly knowing your phone attitude,” in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom ’14, 2014.
- [31] M. Li, P. Zhou, Y. Zheng, Z. Li, and G. Shen, “Iodetector: A generic service for indoor/outdoor detection,” *ACM Trans. Sen. Netw.*, vol. 11, no. 2, pp. 28:1–28:29, Dec. 2014.
- [32] K. Sankaran, M. Zhu, X. F. Guo, A. L. Ananda, M. C. Chan, and L.-S. Peh, “Using mobile phone barometer for low-power transportation context detection,” in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys ’14, 2014.
- [33] R. Sen, Y. Lee, K. Jayarajah, A. Misra, and R. K. Balan, “Grumon: Fast and accurate group monitoring for heterogeneous urban spaces,” in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys ’14, 2014.
- [34] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, “Soundsense: Scalable sound sensing for people-centric applications on mobile phones,” in *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’09, 2009.
- [35] “Apple siri,” <https://www.apple.com/ios/siri/>.
- [36] “Google assistant,” <https://assistant.google.com/>.

- [37] “Microsoft cortana,” <https://www.microsoft.com/en-us/windows/cortana>.
- [38] “Motion sensors on android.”
- [39] R. Honicky, E. A. Brewer, E. Paulos, and R. White, “N-smarts: Networked suite of mobile atmospheric real-time sensors,” in *Proceedings of the Second ACM SIGCOMM Workshop on Networked Systems for Developing Regions*, ser. NSDR '08, 2008.
- [40] “Electromyography,” <https://en.wikipedia.org/wiki/Electromyography>.
- [41] A. Campbell, T. Choudhury, S. Hu, H. Lu, M. K. Mukerjee, M. Rabbi, and R. D. Raizada, “Neurophone: Brain-mobile phone interface using a wireless eeg headset,” in *Proceedings of the Second ACM SIGCOMM Workshop on Networking, Systems, and Applications on Mobile Handhelds*, ser. MobiHeld '10, 2010, pp. 3–8.
- [42] R. Zhang, S. Bernhart, and O. Amft, “Diet eyeglasses: Recognising food chewing using emg and smart eyeglasses,” in *2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, 2016.
- [43] L. E. Baum and T. Petrie, “Statistical inference for probabilistic functions of finite state markov chains,” *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.
- [44] Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao, “Automatically characterizing places with opportunistic crowdsensing using smartphones,” in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp '12, 2012.
- [45] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
- [46] Z. Fang, Z. Guoliang, and S. Zhanjiang, “Comparison of different implementations of mfcc,” *J. Comput. Sci. Technol.*, vol. 16, no. 6, pp. 582–589, Nov. 2001.

- [47] Y. Freund, R. E. Schapire *et al.*, “Experiments with a new boosting algorithm,” in *icml*, vol. 96, 1996, pp. 148–156.
- [48] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford, “A hybrid discriminative/generative approach for modeling human activities,” in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, ser. IJCAI’05, 2005, pp. 766–772.
- [49] J. Mcinerney, S. Stein, A. Rogers, and N. R. Jennings, “Breaking the habit: Measuring and predicting departures from routine in individual human mobility,” *Pervasive Mob. Comput.*, vol. 9, no. 6, Dec. 2013.
- [50] Y. Jiang, K. Li, L. Tian, R. Piedrahita, X. Yun, O. Mansata, Q. Lv, R. P. Dick, M. Hannigan, and L. Shang, “Maqs: A personalized mobile sensing system for indoor air quality monitoring,” in *Proceedings of the 13th International Conference on Ubiquitous Computing*, ser. UbiComp ’11, 2011.
- [51] N. D. Lane, P. Georgiev, and L. Qendro, “Deepear: Robust smartphone audio sensing in unconstrained acoustic environments using deep learning,” in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp ’15, 2015.
- [52] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.
- [53] “Google now,” <https://goo.gl/Wx2t2n>.
- [54] “Mindmeld,” <https://www.mindmeld.com/>.
- [55] B. Yan and G. Chen, “Appjoy: Personalized mobile application discovery,” in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’11, 2011.

- [56] T. Yan, D. Chu, D. Ganesan, A. Kansal, and J. Liu, “Fast app launching for mobile devices using predictive user context,” ser. MobiSys ’12, 2012.
- [57] M. Rabbi, S. Ali, T. Choudhury, and E. Berke, “Passive and in-situ assessment of mental and physical well-being using mobile sensors,” in *Proceedings of the 13th International Conference on Ubiquitous Computing*, ser. UbiComp ’11, 2011.
- [58] R. LiKamWa, Y. Liu, N. D. Lane, and L. Zhong, “Moodscope: Building a mood sensor from smartphone usage patterns,” in *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’13, 2013, pp. 389–402.
- [59] D. H. Nguyen, G. Marcu, G. R. Hayes, K. N. Truong, J. Scott, M. Langheinrich, and C. Roduner, “Encountering sensecam: Personal recording technologies in everyday life,” in *Proceedings of the 11th International Conference on Ubiquitous Computing*, ser. UbiComp ’09, 2009, pp. 165–174.
- [60] P. Zhou, S. Jiang, and M. Li, “Urban traffic monitoring with the help of bus riders,” in *2015 IEEE 35th International Conference on Distributed Computing Systems*, 2015.
- [61] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, “Vtrack: Accurate, energy-aware road traffic delay estimation using mobile phones,” in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys ’09, 2009.
- [62] P. Dutta, P. M. Aoki, N. Kumar, A. Mainwaring, C. Myers, W. Willett, and A. Woodruff, “Common sense: Participatory urban sensing using a network of handheld air quality monitors,” in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys ’09, 2009.
- [63] K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow, “Sociablesense: Exploring the trade-offs of adaptive sampling and computation offloading for social

- sensing,” in *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom ’11, 2011.
- [64] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh, “A framework of energy efficient mobile sensing for automatic user state recognition,” in *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’09, 2009.
- [65] N. D. Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha, “Piggyback crowdsensing (pcs): Energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities,” ser. Proceedings of SenSys ’13, 2013.
- [66] N. D. Lane, P. Georgiev, C. Mascolo, and Y. Gao, “Zoe: A cloud-less dialog-enabled continuous sensing wearable exploiting heterogeneous computation,” in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’15, 2015, pp. 273–286.
- [67] P. Georgiev, N. D. Lane, K. K. Rachuri, and C. Mascolo, “Leo: Scheduling sensor inference algorithms across heterogeneous mobile processors and network resources,” in *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking*, ser. MobiCom ’16. New York, NY, USA: ACM, 2016, pp. 320–333.
- [68] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, “Maui: Making smartphones last longer with code offload,” in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’10, 2010.
- [69] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen, “Comet: Code offload by migrating execution transparently,” in *Presented as part of the 10th*

USENIX Symposium on Operating Systems Design and Implementation (OSDI 12), 2012.

- [70] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, “Odessa: Enabling interactive perception applications on mobile devices,” in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’11, 2011.
- [71] M. S. Gordon, D. K. Hong, P. M. Chen, J. Flinn, S. Mahlke, and Z. M. Mao, “Accelerating mobile applications through flip-flop replication,” in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’15, 2015.
- [72] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, “Towards wearable cognitive assistance,” in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’14, 2014.
- [73] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The case for vm-based cloudlets in mobile computing,” *IEEE Pervasive Computing*, 2009.
- [74] Y. Xu, M. Lin, H. Lu, G. Cardone, N. Lane, Z. Chen, A. Campbell, and T. Choudhury, “Preference, context and communities: A multi-faceted approach to predicting smartphone app usage patterns,” ser. ISWC ’13, 2013.
- [75] T. M. T. Do, J. Blom, and D. Gatica-Perez, “Smartphone usage in the wild: A large-scale analysis of applications and context,” ser. ICMI ’11, 2011.
- [76] “Iptable,” <https://en.wikipedia.org/wiki/Iptables>.
- [77] P. Zhou, Y. Zheng, and M. Li, “How long to wait?: Predicting bus arrival time with mobile phone based participatory sensing,” ser. MobiSys ’12, 2012.

- [78] J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay, “Myexperience: A system for in situ tracing and capturing of user feedback on mobile phones,” ser. MobiSys ’07, 2007.
- [79] H. Verkasalo, “Contextual patterns in mobile service usage,” *Personal Ubiquitous Comput.*, 2009.
- [80] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer, “Falling asleep with angry birds, facebook and kindle: A large scale study on mobile application usage,” ser. MobileHCI ’11, 2011.
- [81] C. Shin, J.-H. Hong, and A. K. Dey, “Understanding and prediction of mobile application usage for smart phones,” ser. UbiComp ’12, 2012.
- [82] S. Ghosh and D. Reilly, “Credit card fraud detection with a neural-network,” in *HICSS’ 94*, Jan 1994.
- [83] C. Aggarwal, *On Abnormality Detection in Spuriously Populated Data Streams*, ch. 8, pp. 80–91.
- [84] J. Lin, E. Keogh, A. Fu, and H. Van Herle, “Approximations to magic: Finding unusual medical time series,” ser. CBMS ’05, 2005.
- [85] A. Lakhina, M. Crovella, and C. Diot, “Diagnosing network-wide traffic anomalies,” in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM ’04, 2004, pp. 219–230.
- [86] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, “Fast portscan detection using sequential hypothesis testing,” in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, May 2004, pp. 211–225.
- [87] A. Hussain, J. Heidemann, and C. Papadopoulos, “A framework for classifying denial of service attacks,” in *Proceedings of the 2003 Conference on Applications,*

Technologies, Architectures, and Protocols for Computer Communications, ser. SIGCOMM '03, 2003, pp. 99–110.

- [88] S. S. Kim, A. L. N. Reddy, and M. Vannucci, *Detecting Traffic Anomalies through Aggregate Analysis of Packet Header Data*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 1047–1059.
- [89] “Tcpdump,” <http://www.tcpdump.org/>.
- [90] V. Kalnikaite, A. Sellen, S. Whittaker, and D. Kirk, “Now let me see where i was: Understanding how lifelogs mediate memory,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 2045–2054.
- [91] R. Rawassizadeh, M. Tomitsch, K. Wac, and A. M. Tjoa, “Ubiqlog: A generic mobile phone-based life-log framework,” *Personal Ubiquitous Comput.*, vol. 17, no. 4, pp. 621–637, Apr. 2013.
- [92] W. James, “What is an emotion?” *Mind*, vol. 9, no. 34, pp. 188–205, 1884.
- [93] P. Ekman, “An argument for basic emotions,” *Cognition & emotion*, vol. 6, no. 3-4, pp. 169–200, 1992.
- [94] R. Horlings, D. Datcu, and L. J. M. Rothkrantz, “Emotion recognition using brain activity,” in *Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, ser. CompSysTech '08. New York, NY, USA: ACM, 2008, pp. 6:II.1–6:1.
- [95] M. Murugappan, M. Rizon, R. Nagarajan, S. Yaacob, I. Zunaidi, and D. Hazry, “Lifting scheme for human emotion recognition using eeg,” in *2008 International Symposium on Information Technology*, vol. 2, Aug 2008, pp. 1–7.

- [96] Y. Liu, O. Sourina, and M. K. Nguyen, “Real-time eeg-based emotion recognition and its applications,” in *Transactions on computational science XII*, 2011, pp. 256–277.
- [97] “10-20 system,” <https://goo.gl/8Yzwv5>.
- [98] Y. Li, Z. Ma, W. Lu, and Y. Li, “Automatic removal of the eye blink artifact from eeg using an ica-based template matching approach,” *Physiological Measurement*, vol. 27, no. 4, p. 425.
- [99] M. J. Katz, “Fractals and the analysis of waveforms,” *Computers in Biology and Medicine*, vol. 18, no. 3, pp. 145 – 156, 1988.
- [100] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl, “Energy characterization and optimization of image sensing toward continuous mobile vision,” in *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’13. New York, NY, USA: ACM, 2013, pp. 69–82. [Online]. Available: <http://doi.acm.org/10.1145/2462456.2464448>
- [101] C. M. Bishop, “Pattern recognition,” *Machine Learning*, vol. 128, pp. 1–58, 2006.
- [102] R. LiKamWa, Z. Wang, A. Carroll, F. X. Lin, and L. Zhong, “Draining our glass: An energy and heat characterization of google glass,” in *Proceedings of 5th Asia-Pacific Workshop on Systems*, ser. APSys ’14, 2014, pp. 10:1–10:7.
- [103] K. W. Ross and D. Tsang, “The stochastic knapsack problem,” in *Proceedings of the 27th IEEE Conference on Decision and Control*, Dec 1988, pp. 632–633 vol.1.
- [104] “Muse eeg headband,” <http://www.choosemuse.com/>.
- [105] S. Koelstra, C. Muhl, M. Soleymani, J. S. Lee, A. Yazdani, T. Ebrahimi, T. Pun, A. Nijholt, and I. Patras, “Deap: A database for emotion analysis ;using physiological signals,” *IEEE Transactions on Affective Computing*, vol. 3, no. 1, pp. 18–31, Jan 2012.

- [106] M. M. Bradley and P. J. Lang, “Measuring emotion: The self-assessment manikin and the semantic differential,” *Journal of Behavior Therapy and Experimental Psychiatry*, vol. 25, no. 1, pp. 49 – 59, 1994.
- [107] Z. Zeng, M. Pantic, G. I. Roisman, and T. S. Huang, “A survey of affect recognition methods: Audio, visual and spontaneous expressions,” in *Proceedings of the 9th International Conference on Multimodal Interfaces*, ser. ICMI '07. New York, NY, USA: ACM, 2007, pp. 126–133.
- [108] C. M. Lee and S. S. Narayanan, “Toward detecting emotions in spoken dialogs,” *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 2, pp. 293–303, March 2005.
- [109] M. Pantic and L. J. M. Rothkrantz, “Automatic analysis of facial expressions: The state of the art,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 12, pp. 1424–1445, Dec. 2000.
- [110] J. F. Cohn, “Facial expression and emotion,” in *Proceedings of the 8th International Conference on Multimodal Interfaces*, ser. ICMI '06. New York, NY, USA: ACM, 2006, pp. 233–238.
- [111] “Microsoft cognitive services,” <https://www.microsoft.com/cognitive-services/en-us/emotion-api>.
- [112] A. Gluhak, M. Presser, L. Zhu, S. Esfandiyari, and S. Kupschick, “Towards mood based mobile services and applications,” in *Proceedings of the 2Nd European Conference on Smart Sensing and Context*, ser. EuroSSC'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 159–174.
- [113] M. Zhao, F. Adib, and D. Katabi, “Emotion recognition using wireless signals,” in *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '16. New York, NY, USA: ACM, 2016, pp. 95–108.

- [114] P. C. Petrantonakis and L. J. Hadjileontiadis, “Emotion recognition from eeg using higher order crossings,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 2, pp. 186–197, March 2010.
- [115] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas, “Emotionsense: A mobile phones based adaptive platform for experimental social psychology research,” in *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, ser. UbiComp ’10. New York, NY, USA: ACM, 2010, pp. 281–290.
- [116] H. Lee, Y. S. Choi, S. Lee, and I. P. Park, “Towards unobtrusive emotion recognition for affective social communication,” in *2012 IEEE Consumer Communications and Networking Conference (CCNC)*, Jan 2012, pp. 260–264.
- [117] G. Chittaranjan, J. Blom, and D. Gatica-Perez, “Who’s who with big-five: Analyzing and classifying personality traits with smartphones,” in *Proceedings of the 2011 15th Annual International Symposium on Wearable Computers*, ser. ISWC ’11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 29–36.
- [118] R. LiKamWa, Y. Liu, N. D. Lane, and L. Zhong, “Moodscope: Building a mood sensor from smartphone usage patterns,” in *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’13. New York, NY, USA: ACM, 2013, pp. 465–466.
- [119] J. Gemmell, G. Bell, and R. Lueder, “Mylifebits: A personal database for everything,” *Commun. ACM*, vol. 49, no. 1, pp. 88–95, Jan. 2006.
- [120] “Gopro,” <https://gopro.com/>.
- [121] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, “The jigsaw continuous sensing engine for mobile phone applications,” in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys ’10. New York, NY, USA: ACM, 2010, pp. 71–84.

- [122] S. Consolvo, D. W. McDonald, T. Toscos, M. Y. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, I. Smith, and J. A. Landay, “Activity sensing in the wild: A field trial of ubifit garden,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 1797–1806.
- [123] P. Belimpasakis, K. Roimela, and Y. You, “Experience explorer: A life-logging platform based on mobile context collection,” in *2009 Third International Conference on Next Generation Mobile Applications, Services and Technologies*, Sept 2009, pp. 77–82.
- [124] S. Chennuru, P.-W. Chen, J. Zhu, and J. Y. Zhang, “Mobile lifelogger–recording, indexing, and understanding a mobile users life,” in *International Conference on Mobile Computing, Applications, and Services*. Springer, 2010, pp. 263–281.
- [125] L. Ivonin, H.-M. Chang, W. Chen, and M. Rauterberg, “Unconscious emotions: Quantifying and logging something we are not aware of,” *Personal Ubiquitous Comput.*, vol. 17, no. 4, pp. 663–673, Apr. 2013.
- [126] N. Caprani, N. E. O’Connor, and C. Gurrin, “Experiencing sensecam: A case study interview exploring seven years living with a wearable camera,” in *Proceedings of the 4th International SenseCam & Pervasive Imaging Conference*, ser. SenseCam '13. New York, NY, USA: ACM, 2013, pp. 52–59.
- [127] X. Zeng, K. Cao, and M. Zhang, “Mobiledeeppill: A small-footprint mobile deep learning system for recognizing unconstrained pill images,” in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '17, 2017, pp. 56–67.
- [128] L. N. Huynh, Y. Lee, and R. K. Balan, “Deepmon: Mobile gpu-based deep learning framework for continuous vision applications,” in *Proceedings of the 15th Annual*

- International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '17, 2017, pp. 82–95.
- [129] R. Zhang and O. Amft, “Bite glasses: Measuring chewing using emg and bone vibration in smart eyeglasses,” in *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, ser. ISWC '16, 2016, pp. 50–52.
- [130] A. Exler, A. Schankin, C. Klebsattel, and M. Beigl, “A wearable system for mood assessment considering smartphone features and data from mobile ecgs,” in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, ser. UbiComp '16, 2016, pp. 1153–1161.