Title
Modeling Human Engagement State to Lower Cognitive Burden and Increase User Interaction Responsiveness

Author
Ho, Bo-Jhang

Publication Date
2019

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Modeling Human Engagement State to Lower Cognitive Burden and Increase User Interaction Responsiveness

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Bo-Jhang Ho

2019

ABSTRACT OF THE DISSERTATION

Modeling Human Engagement State to Lower Cognitive Burden and Increase User Interaction Responsiveness

by

Bo-Jhang Ho
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2019
Professor Mani B. Srivastava, Chair

Mobile phones and wearable devices are becoming increasingly ubiquitous, and the relation between mobile sensing devices and human beings is getting more and more intimate. Computation is no longer merely a machine's job, where hardware executes a sequence of operations; human beings are often involved in the process, as in medication intervention, preference configuration, and crisis alert. Product manufacturers and service providers rely on user engagement to make revenue, and users can benefit from these products and maximize their utility only if users are willing to engage with them. We argue that next-generation sensing computation systems should be user-engagement aware, i.e., the systems should treat *user engagement* as part of system resources to make decisions and prioritize tasks.

Although different sensing modalities and optimization techniques have been proposed, user engagement cannot be gauged directly because it is a hidden construct. Users' engagement depends on multiple variables including environmental conditions, physical constraints, psychological status, and self-interests. Unfortunately, the limitations in existing sensing techniques exacerbate the difficulty of engagement measurement. For instance, sensor data noise increases the uncertainty of inferences, battery size constrains sensor coverage both temporally and spatially, and form factor directly impacts users' willingness to carry these devices. Such issues multiply the complexity of modeling user engagement.

In this dissertation, we adapt the performance-based observation approach to describe

user engagement, bridge the gap between engagement measurement with sensing techniques, and seek opportunities to further increase user engagement. We first showcase how wearable sensing systems can increase user engagement in the workout domain by performing opportunistic sensing. This dissertation then discusses how to use sensory data to model user engagement by reinforcement learning algorithms. Finally, we point out concerns specifically in sensing systems that can negatively impact user engagement.

The dissertation of Bo-Jhang Ho is approved.

Bonnie Zima

Santosh Kumar

Yizhou Sun

Mani B. Srivastava, Committee Chair

University of California, Los Angeles

2019

In remembrance of Fu Jun-Song,

an adorable and devoted grandfather.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express an endless amount of gratitude and appreciation to my Ph.D. supervisor and my dissertation committee chair Dr. Mani Srivastava. Mani is open minded in research and always gives us space to explore and to fail, yet he gives us tremendous support whenever we need it. Besides the intellectual influence (and certainly he's the most influential person in my life), he teaches us that one should remain humble and thirsty in learning by setting himself as an example. That is why his words carry so much weight and get a lot of attention within and beyond the community. This dissertation undoubtedly would not have the same quality without your advice. Thank you for giving me a chance to be your student and to learn from you. I hope my achievement does not let you down.

It is a blessing to have such a golden dissertation committee. Dr. Santosh Kumar is a humble, patient, and impactful interdisciplinary researcher. The Center of Excellence for Mobile Sensor Data-to-Knowledge (MD2K) he leads has opened many insightful conversations and been a great testbed for exploring new methodologies. Dr. Bonnie Zima is another role model as a transdisciplinary researcher. She always brings encouragement, energy, and joy to our team meetings. Although clarifications are inevitable during meetings, she never gives up and always has faith that all the problems can be solved. I would like to thank Dr. Yizhou Sun who accepted my dissertation committee request at such short notice, yet still provided invaluable field feedback. Thank you to all my committee members. My research could not have reached such maturity and broadness without your guidance.

NESL is the place that nurtures me and where I complete this accomplishment; it is where to learn, to grow, to broaden one's horizon, and to become mature. To Dr. Bharathan Balaji, you always aim high and never stops learning. In the lab, you are like my second advisor who opens so many incisive discussions. But during leisure time, you are a chill person and we have a lot of activities to do. I will adore those days that we pulled all nighters for paper deadlines as well as those evenings that we played games together. It is really a privilege to work with you. To Dr. Paul Martin, you are a great mentor in NESL. You contribute so much to NESL and you know every detail about lab infrastructure. I appreciate our collaboration

on the air pressure project. It is you who teach me how to be responsible and mature. To Dr. Chenguang Shen, you are an intelligent, hard-working, and considerate co-worker. I enjoyed the year that we spent so much time in the gym to collect workout data. Such a wonderful project not only boosted our research credits, but also improved our physical health. To Dr. Lucas Wanner, I enjoy the time that we spend joking around. Looking back, that is how you made me confident in public speeches. Finally, I would like to thank Dr. Luis Garcia. Although we only overlapped for a couple of months, I admire your passion in research and in education—how many postdocs would like to have 15 one-on-one meetings per week? We don't need to argue that earning a Ph.D. is hard, but thank you for reminding and teaching us to have a balanced life. I'm sure you will be a great professor. There are yet too many episodes and fun memories happening in NESL, and I wish to document them all here. I want to say thank you to Supriyo Chakraborty, Haksoo Choi, Salma Elmalaki, Yasser Shoukry, Fatima Anwar, Moustafa Alzantot, Amr Alanwar, Hsiao-Yun Tseng, Renju Liu, Sandeep Singh, Tianwei Xing, Nathaniel Snyder, Xi Han, Debbie Tsai, Leon Kozinakov, Zhengxu Xia, Eun Sun Lee, Akash Deep, Brian Wang, Ziqi Wang, and Siyou Pei. I could not fight this journey alone without your support.

Special thanks to Dr. Mashfiqui Rabbi. Although we only meet briefly during MD2K annual meetings, you are always curious about what I am working on and give me invaluable feedback. I appreciate your passion and your inspiration.

This Ph.D. would not have happened without the support of my master's thesis advisors, Dr. Hao-Hua Chu and Dr. Polly Huang. Prof. Chu is the person who patiently taught me research skills. As a junior researcher, it is often necessary to have someone's guidance to keep the momentum of the research progress. It's Prof. Chu who has faith in me and spent countless hours supervising me. I thank Polly who saw the value in me. I will not forget the long conversation we had when I was experiencing the most difficult time in the first year of the program. My Ph.D. application couldn't have looked that strong without Dr. Chuang-Wen You's help. I remember so many episodes when we brainstormed together to come out with ideas and solutions, and pushed ourselves to achieve something that seemed impossible. You are a great mentor.

This honor should definitely be dedicated to my mother and my father, who raised me, taught me to be disciplined, and provided whatever I needed. I know it was a big struggle when I told my parents I had to leave them for this program, but they still gave me a chance to explore this world and supported me to the end. I wish I could express how grateful I am! I would also like to thank my adorable brother. Although you never told me, I know how much you care about me and try to encourage me in your own way. Finally, I could not have walked so confidently and conquered all the challenges during my Ph.D. without your company, my best friend and my fiancée, Hsin-Liu Kao. I would not have the courage to face all the difficulties alone; thank you for being my rock.

VITA

2007	International Olympia in Informatics (IOI), Gold Medalist
2007 – 2011	B.S. in Computer Science and Information Engineering (CSIE), National Taiwan University (NTU)
2009 – 2010	Learning Tutors from Dean's List (GPA top 5%)
2009 – 2010	Head of Academic Section, Student Council, Dept. of CSIE, NTU
2007 – 2010	ACM International Collegiate Programming Contest (ICPC), granted admission to World Finals
2011 – 2012	M.S. in Computer Science and Information Engineering, National Taiwan University
2011	Outstanding Teaching Assistant Award, CSIE, NTU
2013 – 2019	Graduate Student Researcher, Computer Science Department, UCLA
2015	Software Engineer Intern, Google, Mountain View, CA, USA
2016	Teaching Assistant, Computer Science Department, UCLA
2017	Software Engineer Intern, Facebook, Menlo Park, CA, USA

PUBLICATIONS

Bo-Jhang Ho, Bharathan Balaji, Mehmet Koseoglu, Sandeep Sandha, Siyou Pei, Hsin-Liu (Cindy) Kao, Mani Srivastava. "Quick Question: Interrupting Users for Microtasks with Reinforcement Learning." PerCom 2019, IEEE International Conference (under review)

Salma Elmalaki, Bo-Jhang Ho, Moustafa Alzantot, Yasser Showery, Mani Srivastava. "VindiCo: Privacy Safeguard Against Adaptation Based Spyware in Human-in-the-Loop IoT." IoTDI 2019, ACM/IEEE International Conference (under review)

Bo-Jhang Ho, Bharathan Balaji, Mehmet Koseoglu, Mani Srivastava. "Nurture: Notifying Users at the Right Time Using Reinforcement Learning." Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers.

Erik S Paschall, F Alethea Marti, Yuri Cheung, Michael P McCreary, James Lee, Kira A Williams, Alpa Patel, Bo-Jhang Ho, Lily Zhang, Bonnie T Zima. "5.9 Opportunities and Challenges in Using a Mobile Health (mHealth) Intervention to Optimize Early Stimulant Treatment in Children With ADHD: Findings From the MH-2^{TM} Pilot." Journal of the American Academy of Child & Adolescent Psychiatry, 2018.

Bo-Jhang Ho, Chenguang Shen, Mani Srivastava. "MiLift: Efficient Smartwatch-Based Workout Tracking Using Automatic Segmentation." IEEE Transactions on Mobile Computing, 2018.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, Kai-Wei Chang. "Generating Natural Language Adversarial Examples." Conference on Empirical Methods in Natural Language Processing, 2018.

Syed Monowar Hossain, Timothy Hnat, Nazir Saleheen, Nusrat Jahan Nasrin, Joseph Noor, Bo-Jhang Ho, Tyson Condie, Mani Srivastava, Santosh Kumar. "mCerebrum: A Mobile Sensing Software Platform for Development and Validation of Digital Biomarkers and Interventions." Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems, 2017.

Bonnie T Zima, F Alethea Marti, Michael McCreary, Nancy Alfaro, Kira Williams, Alpa Patel, Bo-Jhang Ho, Lily Zhang, Lingqi Tang. "6.40 Feasibility of a Web-Based App to Optimize Early Stimulant Medication Treatment." Journal of the American Academy of Child & Adolescent Psychiatry, 2017.

Bo-Jhang Ho, Bharathan Balaji, Nima Nikzad, Mani Srivastava. "Emu: Engagement Modeling for User Studies." Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers.

Bo-Jhang Ho, Renju Liu, Hsiao-Yun Tseng, Mani Srivastava. "MyoBuddy: Detecting Barbell Weight Using Electromyogram Sensors." Proceedings of the 1st Workshop on Digital Biomarkers, 2017.

Amr Alanwar, Moustafa Alzantot, Bo-Jhang Ho, Paul Martin, Mani Srivastava. "SeleCon: Scalable IoT Device Selection and Control Using Hand Gestures." Proceedings of the Second International Conference on Internet-of-Things Design and Implementation, 2017.

Bo-Jhang Ho, Hao Li, Bharathan Balaji, Yue Xin, Paul Martin, Mani Srivastava. "EmbedInsight: Automated Grading of Embedded Systems Assignments." arXiv preprint arXiv:1703.04514, 2017.

Bo-Jhang Ho, Paul Martin, Prashanth Swaminathan, Mani Srivastava. "From Pressure

to Path: Barometer-Based Vehicle Tracking." Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments, 2015. [Best paper nomination]

Paul Martin, Bo-Jhang Ho, Nicholas Grupen, Samuel Munoz, Mani Srivastava. "Demo Abstract: An iBeacon Primer for Indoor Localization." Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings, 2014.

Bo-Jhang Ho, Mani B Srivastava. "Poster: M-Seven: Monitoring Smoking Event by Considering Time Sequence Information via iPhone M7 API." Proceedings of the 12th annual international conference on Mobile systems, applications, and services, 2014

CHAPTER 1

Introduction

User engagement has become an important measure for corporate revenue[1,2]. Internet advertising companies like Google and Facebook rely on users who are willing to use their services to make money[3]. Product recommendation companies such as Amazon and Netflix can operate only if the recommendation results can attract more customers. It is only possible for mobile phone or wearable manufacturers to sell their products if buyers find value in these devices and are willing to use them. Hence, user engagement is the key driver to make the products and services successful.

Engagement is also essential to end users because it augments the utility of these products. Users can only benefit from cloud services, mobile applications, wearables, and Internet of Things (IoT) devices when users are willing to interact with them. For example, patients can receive timely and proper interventions if they are willing to follow instructions from their doctors [HCD09]. Motivated users can acquire accurate and consistent self-monitoring results to track their health status or even identify chronic diseases [MVP15]. In a critical scenario, users can take immediate actions informed by alert messages in an emergency, if the alert systems can grab their attention [SOM10].

User engagement, however, is an invisible mental construct and cannot be measured or sensed directly [OT08]. Although engagement is intangible and there is no standard-

[1]Social jargon: how do you define "engagement" and "influence"? - https://econsultancy.com/social-jargon-how-do-you-define-engagement-and-influence/

[2]App Engagement: The Matrix Reloaded - https://flurrymobile.tumblr.com/post/113379517625/app-engagement-the-matrix-reloaded

[3]Online ad revenues are surging - https://www.businessinsider.com/online-ads-revenues-going-to-google-and-facebook-the-most-2017-4

ized definition of what human engagement is, a common aspect of engagement refers to a certain mental status where one's attention is so absorbed in interacting with physical or conceptual objects [Cha03, CSW99, Kap95, SDN16], that external stimuli are no longer distracting [Csi90]. User engagement can be gauged based on subjective report or behavioral observation. Thanks to advances in sensing technology, the latter opens the opportunity for context-aware systems to model user engagement based on when, where, and what the user is doing, making the sensing systems capable of becoming *engagement-aware*.

As computers have become more and more ubiquitous, computation is no longer just an execution on a sequence of operations. Human beings are often involved in the computation process. For example, a medical intervention will fail if a patient is unavailable or unwilling to follow instructions. In an extreme scenario, a car accident can happen if a driver gets distracted and disengages from steering. Thus, the outcome depends on whether the users can engage in the process. In other words, *user engagement* should be considered as part of the system resources in addition to the hardware utility.

Modeling user engagement is not a trivial task because engagement depends on both physical and mental status. A physical constraint such as being in a meeting prevents a user from answering phone calls. A mental constraint like experiencing stress may increase the probability of ignoring or dismissing incoming notifications. Moreover, engagement relies on whether users can spare their attention for a certain task, but user attention is finite [LFN15] and multi-faceted [Wic02]. All of these properties increase the complexity of engagement modeling.

The goal of this dissertation is to bridge the gap between engagement measurement and the sensing technology. We specifically list the research questions below:

- *How can we use mobile and wearable sensory data to model and infer user engagement?*
- *What are the opportunities to raise user engagement by leveraging mobile sensing techniques?*
- *What are the concerns in the existing sensing techniques that may negatively impact*

user engagement?

To this end, we first formalize the definition of user engagement, point out the system design challenges, and summarize how we address these issues.

1.1 Engagement Measurement

Engagement has been studied in a broad range of domains, such as mental health [ZMM17], behavioral change [RKC18], games [CCC09], sports [DR85], human-computer interaction [WRS02], education [WH97], retailing [AR03], social [PC12], and influence on self-reported data quality [HKK14]. Like other psychological attributes such as emotions, user engagement is a hidden construct. Thus, previous studies developed different techniques to gauge engagement. Exhaustive surveys on the definition of engagement have been conducted in [OT08] and [PBW16]. Prior work can be broadly grouped into three categories: Subjective experience report, biological signal measurement, and performance-based observation. In this section, we present approaches to measuring engagement and explore which is more compatible with modern sensing technologies.

1.1.1 Subjective Experience Report

Engagement can be conceptualized as an experience that draws out one's attention. According to flow theory, when people are engaged, they are *"so involved in an activity that nothing else seems to matter; the experience itself is so enjoyable that people will do it even at great cost, for the sheer sake of doing it"* [Csi90]. Similarly, Laurel defines engagement as a notion of experiencing the willing suspension of irrelevant information [Lau13]. Reaching such a mental status typically requires motivations, both intrinsic and extrinsic. Intrinsic motivation emphasizes that the task is for the sake of pleasure. For instance, Jennings suggests that *"an environment that is designed from an aesthetic perspective will have the features that inherently engage users"* [Jen00]. Rieber points out that playing is voluntary, self-motivating, and external-reward-independent [Rie96]. Unlike intrinsic motivation, ex-

trinsic motivation stresses that people are willing to try a different action which can lead to a separate outcome [RD00] when they believe it will enhance their well-being [HMF15].

Experiences are subjective; researchers rely on participant self report to understand what engagement is. O'Brien and Toms conducted a qualitative study to understand the process of (re)engagement [OT08]. In Schonau-Fog and Bjorner's study, users reported the experience of engagement as being eager to continue playing the game [SB12]. Participants in Henshaw's study describe engagement as being able to concentrate [HMF15]. Lin et al. show that customers are willing to stick on shopping websites when the experience of the navigation is smooth and the content is clear [Lin07]. Moods and emotions can also impact the engagement [BBV13, CCD13]. Besides describing experience in a free-text form, some studies ask participants to rate their experience numerically. The Likert scale has been applied to quantify user perception in different dimensions, including satisfaction [BRP06], agreement [WLR14], understanding [PLJ10], and overall engagement score [HKO15].

Although this approach is arguably the most comprehensive way to measure and model user engagement, the self-report approach entails additional user burden and does not scale in sensing systems.

1.1.2 Biological Measurement

Engagement entails cognitive load. When the cognitive load is high, human beings emit several biological signals unconsciously [ZF15]. Studies have shown when one is experiencing high mental workload, the pupil dilates [IAZ05] and the body dispenses heat [HKF10]. Cognitive load also associates with brain activation. When our brains are activating, the skin surface temperature of our foreheads increases and the temperature of our noses drops due to blood flow [AVD17]. Mathur et al. attempt to model brain activities by electroencephalogram (EEG) directly to gauge engagement [MLK16]. Engagement may cause arousal, which is reflected on galvanic skin response (GSR) [GF17]. Besides these sensing modalities, eye gaze pattern [JCC08], facial expression [WSL14], and respiratory depth [HMT15] are also good indicators of engagement.

Although this approach directly fits the wearable sensing scenario, there are two concerns with this approach: feasibility and social acceptability. Most of the aforementioned bio markers rely on specific hardware and some can only be found in laboratory setups. Thus, these methods are not feasible for everyday end users. Moreover, the form factors of these devices usually do not meet social norms. For example, a user may not find it aesthetically pleasing or comfortable to wear electrodes in public.

1.1.3 Performance-Based Observation

Performance-based measurement has been widely used in behavioral studies and human-computer interaction (HCI). When people are engaged, they have a strong motivation to accomplish certain goals. Hence, their performance can be used to indicate their level of engagement. Performance-based observation techniques have been adapted in healthcare systems to understand adherence and retention, particularly in weight loss [WSC12], alcohol consumption [MWV13, DDG15], clinical visit counts [RGC13, MLZ11], and exercise load [BRP06]. User interface designers and researchers have also defined different metrics to gauge engagement, such as error elimination [HCD12], retention rate [CGF09], browsing time interval [CAZ10], and forum usage count [DBA06]. In game design, developers also derive different instruments such as game scores [BKP07], frequency of system logins [CG12], mouse activities [MKF14], and recovery time from distractions [MCH10] to understand how players interact with games.

The advantage of the performance-based observation approach is that all the measures are well defined and can be easily integrated into sensing pipelines. The caveat, however, is whether these metrics reflect the engagement with high confidence given the potential noise in the sensor readings. Consider the example of encouraging users to keep hydrated. Just monitoring water consumption by number of cups may not be sufficient, because water intake can come from other forms of liquid. Hand movements are one way to track fluid-drinking activities, but the sensing system may confuse drinking events with other activities using similar gestures.

In this dissertation, we adapt the performance-based observation approach to describe user engagement, bridge the gap between engagement measurement with sensing techniques, and seek opportunities to further increase user engagement.

1.2 Challenges

We stand on the edge of an intimate coupling of sensing devices and human beings; computers do not only perform calculation tasks, but also understand user context and track environmental information. Many mobile sensing and persuasive techniques have been developed in the past decades [PM15]. It is time to revisit how we should apply these sensing techniques to engage people. In this section, we identify the challenges and the opportunities of exploiting the mobile sensing technology to augment user engagement.

1.2.1 Can Sensing Automation Increase Engagement?

Recall that the aesthetic theory [Bea70] shows that people are inherently engaged in an environment that is aesthetically designed because the experiences are intrinsically motivated. Similarly, play theory [Ste64] shows that pursuing playfulness can grab users' attention. Both theories suggest that it is possible to establish a scenario that can be highly engaging and immersive if the design integrates aesthetic or playful elements. Under such a scenario, engagement happens naturally and effortlessly.

In this dissertation, we argue that the mobile sensing technique can augment the space of self-motivating elements. In light of Mark Weiser's vision that most technologies should remain peripheral [Wei91], the computation system should keep human beings out of the sensing pipeline when possible but can still assist users. Consequently, sensing systems can save users from external interruptions and preserve cognitive resources.

To achieve such a seamless experience in sensing, the devices have to be available when needed, while being portable with the right form factors. These constraints entail the ultimate bottleneck in battery; the energy consumption is driven by computation, network

transmission, electronic fabrication in sensors, and the modality for notifying users. Continuous sensing can drain the battery in a short time. Hence, opportunistic sensing schedules have to be developed to preserve energy. All of these factors can increase the complexity of the sensing systems.

1.2.2 How Can Sensory Data Model User Engagement?

Although the background process can be optimized, a mobile application has to interact with human beings at some point—otherwise, it is just a background service. The form of interactions can range anywhere from sending a reminder to intervening, delivering an important message, or asking for preference configurations. The process of the interaction usually requires an interruption from the primary task, redirecting the cognitive resources to the secondary task, and resuming back to the primary task [OT08]. Borst et al. develop a process model which points out that the transitions between tasks can cause overhead [BTR15]. The overhead, unfortunately, can be magnified by the dissimilarity between the primary task and the secondary task [MGK08]. Hence, it is important to consider user context when interrupting people. In other words, *interruptibility* plays an important role in keeping users engaged.

Using sensory data to model user engagement has been preliminarily explored. Sarker et al. exploit mobile phone sensor data to identify opportune moments to deliver interventions [SSA14]. Okoshi et al. model engagement by physical activity transitions [OTT17]. Both works only develop a general model and do not consider individual variations. In this thesis, we highlight the requirements of engagement modeling: (i) The engagement model needs to have the capacity to learn user preference; (ii) the engagement model needs to be adaptive because user habit may change; (iii) the learning process should minimize user inputs, i.e., the frequency of acquiring user feedback that indicates the desirability of decisions and interactions made by the sensing systems.

1.2.3 How Does Privacy Impact Engagement?

User engagement relies on the trust of the underlying sensing mechanisms, which are typically invisible to users. Although major mobile platforms adapt to the sensor-level permission protection model—i.e., applications have to justify their intentions and users have to explicitly grant the privilege for these apps to use certain hardware—prior work has suggested that such mechanisms are not sufficiently secure [FHE12, CGH17]. Mobile applications may request permissions to use sensors, seemingly with a legitimate purpose, but sensitive personal data can still be leaked by performing additional inferences without users' awareness. Jim Morris has provided guidelines for data collection, stating that the amount of information collected from the user for an operation to be performed should be balanced, no more and no less than necessary [Wei91]. Thus, it is essential to have a privacy safeguard to detect whether unintentional information can be inferred based on the current sensor set.

1.3 Roadmap and Contribution

In the rest of the thesis, we present our work to address the aforementioned challenges in order.

Part I: Sensing Automation. The use of smartphones and wearables as sensing devices has created innumerable context inference apps, including those that track food intake [TEA15, KS15], emotion [LLL13, STR16], medication adherence [KAL15], indoor localization [LKA15], and more. In this thesis, we pick a workout tracking app as an example to showcase how sensing automation can lower the burden of using the system and further engage users. Workout data generated by mobile tracking apps can assist both users and physicians in achieving better healthcare, rehabilitation, and self-motivation. Previous approaches impose extra burdens on users by requiring them to select types of exercises or to start/stop sessions. In Chapter 2, we introduce ***MiLift*** which is a workout tracking system that performs automatic segmentation to remove user burdens. MiLift uses commercial

off-the-shelf smartwatches to accurately and efficiently track both cardio and weightlifting workouts without manual inputs from users. For weightlifting tracking, MiLift supports both machine-based and free weight exercises, and proposes a lightweight repetition detection algorithm to ensure efficiency. A research study of 22 participants shows that MiLift can achieve above 90% average precision and recall for cardio workout classification, weightlifting session detection, and weightlifting type classification. MiLift can also count repetitions of exercises with an average error of 1.12 reps (out of an average of 9.65). Our empirical app study on a Moto 360 watch suggests that MiLift can extend watch battery lives by up to 8.25x (19.13h) compared with previous approaches. Finally, our participants gave us an average rating of 4.47 out of 5 in the poll.

One limitation in MiLift is that the system cannot track the amount of weight lifted. However, such information is important for users engaged in regular workouts and patients who are undergoing rehabilitation. For example, muscular dystrophy is a group of genetic diseases that cause muscle loss or muscle weakness. A typical treatment for muscular dystrophy patients is routinely performing weightlifting exercises to slow this process. In Chapter 3, we propose a sensor system called ***MyoBuddy***, which aims to help both physical therapists and patients keep track of the weights in workout activities based on electromyography (EMG) sensors embedded in the Myo armband. In our study, we collected 102 sessions of EMG data from barbell bicep curl exercises with a range of weights from 20 to 70 lbs with a 10-lb increment. Both support vector machine and random forest algorithms are explored to classify the weight of barbells lifted. In the end, we achieved an 81.3% classification accuracy on average.

Part II: Engagement Modeling. User interaction is an essential part of many mobile devices such as smartphones and wrist bands, and is strongly associated with engagement. Only by interacting with the user can these devices deliver services, enable proper configurations, and learn user preferences. Push notifications are the primary method used to attract user attention in modern devices. However, these notifications can be ineffective and even irritating if they prompt the user at an inappropriate time. The discontent is exacerbated

by the large number of applications that target limited user attention. In Chapter 4, we propose a reinforcement-learning based personalization technique, called ***Nurture***, which automatically identifies the appropriate time to send notifications for a given user context. Through simulations with the crowdsourcing platform Amazon Mechanical Turk, we show that our approach successfully learns user preferences and significantly improves the rate of notification responses.

Based on the insight of the previous simulation-based study, we develop a real system called ***Quick Question*** to track user engagement and schedule microtasks in the wild based on users' interruptbility, demonstrated in Chapter 5. As we have established previously, human attention is a scarce resource in modern computing. A multitude of microtasks vie for user attention to crowdsource information, perform ecological momentary assessment, personalize services, and execute actions with a single touch. A lot gets done when these tasks take up the invisible free moments of the day. With Quick Question, we model the problem as a Markov decision process and use an Advantage Actor Critic algorithm to identify interruptible moments based on the context and history of user interactions. In our 5-week, 41-participant study, we compare the proposed RL algorithm against supervised learning methods. While the mean number of responses between both methods is commensurate, RL is more effective at avoiding dismissal of notifications and improves user experience over time.

Part III: Privacy Concerns. Pervasive mobile devices have enabled countless context- and location-based applications that facilitate navigation, advertisements, life-logging, and more. Applications and web-services wishing to make use of location-identifying mobile data such as those obtained from GPS, cell towers, and Wi-Fi must explicitly request permission from the user to do so. Less intrusive sensors like accelerometers, gyroscopes, and barometers do not require explicit access permission from the end user for applications on the device itself or even on remote web servers. Though seemingly innocuous, these sensors can be combined and analyzed to make surprising and potentially security- and privacy-breaching inferences. To emphasize the severity of the privacy issue, we conduct two studies to demonstrate how

innocent sensory data can leak undesired information. In Chapter 6, we develop a mobile application called ***GPSI*** which can, without consent and without notifying the user, use pressure data collected from a device's barometer in order to detect with high accuracy likely paths along which the user has recently traveled, compromising both user privacy and security. We further analyze the ability to predict unknown mobile trajectories in terms of the variance in barometer pressure and geographical elevation, demonstrating cases in which more than 70% of paths can be accurately predicted.

Chapter 7 shows how context-aware applications can open a privacy loophole. Context-aware applications adapt their behavior based on contextual information such as user behavior and location. Unfortunately, the fact that context-aware apps adapt to user context opens a side-channel that leaks private information about the user. We showcase that a malicious app can monitor the adaptations triggered by authentic context-aware apps and how much user information can be leaked. We show a concrete instantiation of a new category of spyware apps which we refer to as Context-Aware Adaptation Based Spyware, or ***SpyCon***. Experimental evaluations show that SpyCon applications can predict users' daily behavior with an accuracy of 90.3%.

Here we summarize the contributions of this thesis:

- We showcase that increasing the system utility and eliminating unnecessary machine-initiated queries can increase user engagement. Particularly, we develop different optimization mechanisms in the proposed workout sensing system which can perform opportunistic sensing and allow the battery to last for the entire day.

- We attempt to model user engagement based on user context information to identify the opportune moments to deliver microtasks. Particularly, we explore the feasibility of using reinforcement learning to model user engagement due to the advantages of temporal decision process, online learning, and exploration strategy.

- We highlight the privacy concern that sensory data may unintentionally leak sensitive

personal information which can hinder user engagement. We demonstrate that air pressure information and phone settings can reveal user location.

Part I

Sensing Automation

CHAPTER 2

MiLift: Efficient Smartwatch-based Workout Tracking Using Automatic Segmentation

2.1 Motivation

The emergence of mobile sensing devices such as smartphones and wearables has enabled ubiquitous and continuous context inferences, including various types of health monitoring and workout tracking apps. With rising obesity and cardiovascular diseases linked to physical inactivity [LSL12], such workout tracking can provide quantitative data on users' everyday activities and assist both users and physicians in achieving better health care [PGR08] [CBV15], rehabilitation [CMP14], and self-motivation [RLS13]. Compared with self-reporting, the use of mobile devices for workout tracking can provide more accurate summaries for both *cardio* and *weightlifting* exercises and avoid over- or under-estimations. Visualizing personal workout history from sensor data can help motivate users to maintain or improve workout plans and states of health. In health care and rehabilitation, physicians can also maintain progressive data of patients with the help of mobile sensors [RMB10] [GTV14].

Although there are a variety of approaches for mobile workout tracking, they lack the ability to *automatically* segment workout activities. This missing capability imposes substantial burdens on users, such as requiring users to manually start/stop tracking or to select exercise types. Failure to provide timely inputs may lead to inaccurate tracking results and/or excessive energy consumption. These burdens have made prior approaches less attractive compared with simple self-reporting. For example, smartphone apps such as RunKeeper [run] and Strava [str] can only track specific types of exercises (e.g. running and biking) and require users to manually start and stop tracking. Mobile health monitoring

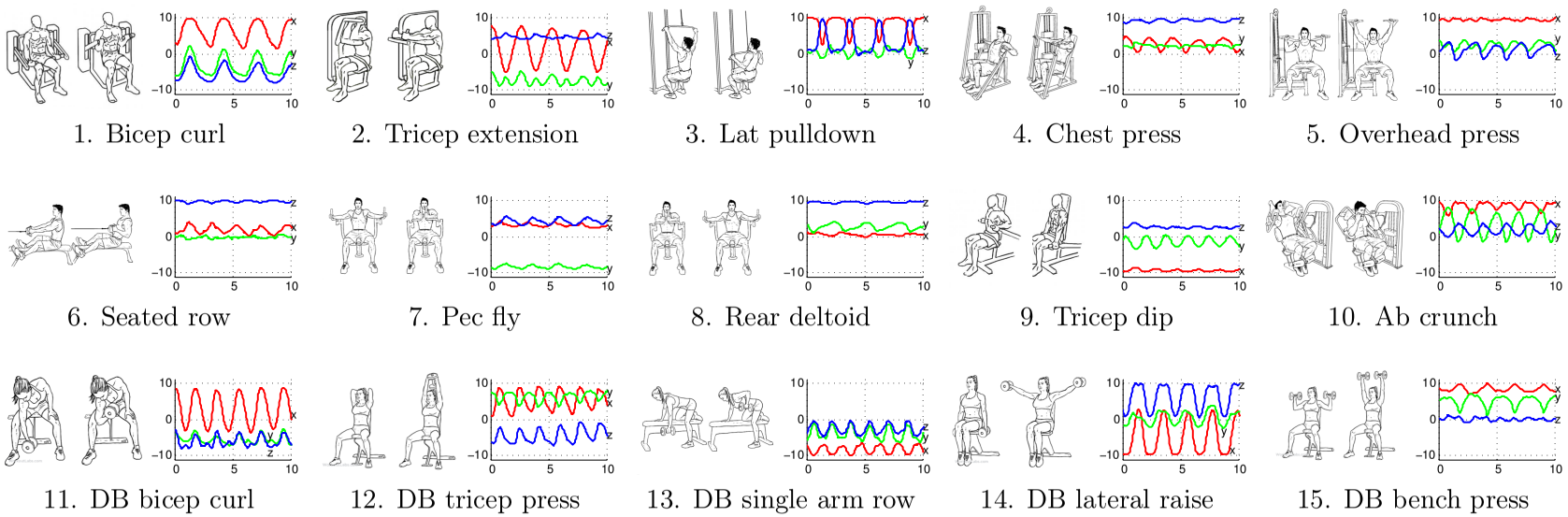


Figure 2.1: Illustration of 15 weightlifting exercises considered in this paper (image sources [wor] [gym]) and repeating patterns in gravity sensor data traces collected from our user study. x-axis shows time in s and y-axis shows 3-axis gravity readings in m/s^2, and x, y, z-channel are encoded in red, green, and blue, respectively (same for the rest of this paper). Type 1-10 are machine exercises, and 11-15 are dumbbell (DB) free weight exercises.

frameworks such as Apple HealthKit [hea] and Google Fit [goo] leverage both smartphones and wearable devices but require users to specify the type of workouts. While most previous work focus on monitoring cardio workouts, a few considered tracking weightlifting exercises [CCC07] [MPA14] [DSY15a] [MLN16]. However, these approaches introduce extra user burdens such as the use of multiple sensors and energy-hungry algorithms while still requiring certain degrees of user inputs for accurate tracking. Recently, there have been initial attempts to perform automatic exercise segmentation, as seen in the Pocket Track feature of RunKeeper [poc], the VimoFit app [vim], and RecoFit [MSG14]. Nevertheless, they focus on limited types of exercises or do not quantify the energy consumption of continuous tracking on commercial mobile and wearable devices.

In this section, we describe the design and implementation of MiLift, a workout tracking system that uses *automatic segmentation* to eliminate the burden on users. MiLift leverages a new generation of Android smartwatches such as the Moto 360 [mota], which benefit from powerful hardware resources, Bluetooth Low Energy radios, and a rich set of sensors. Using a single smartwatch, MiLift can accurately and efficiently track both cardio and weightlifting exercises without requiring inputs from users. Additionally, MiLift applies optimization techniques such as context-aware duty-cycling and lightweight repetition detection to continuously inferring contexts on smartwatches. We highlight the research contributions of MiLift as follows:

First, MiLift can automatically segment both cardio workouts and weightlifting exercises from non-workout activities using a two-stage classification model. It is the first system to apply and fully evaluate such an automated algorithm in workout tracking. Unlike previous tracking approaches, users do not need to provide any manual inputs to MiLift, such as selecting types of exercises or starting/stopping exercise sessions. MiLift runs in the background on a smartwatch and also provides a UI to visualize the workout summary of users for management. From our user study, MiLift's automatic segmentation feature is proven valuable to individuals who regularly perform gym exercises.

Second, MiLift can track both weightlifting machines and free weight (dumbbell) exercises, as shown in Figure 2.1. MiLift meets real-world user requirements during weightlifting

exercises, including (1) automatically detect the start and stop of weightlifting sessions (sets); (2) count repetitions (reps) of exercises; (3) classify the type of exercises. Our evaluation on a dataset of 2528 sets of weightlifting exercises (24408 reps) collected by 22 users shows that MiLift can achieve above 90% average precision and recall for both weightlifting session detection and exercise type classification. The average error of rep counting in MiLift is 1.12 reps (out of an average of 9.65). Weightlifting detection in MiLift requires no model training and is user-independent.

Finally, to achieve efficient resource usage, MiLift employs two techniques that have not been applied to wearable context inferences by prior work: a context-aware duty-cycle optimization and a lightweight revisit-based weightlifting detection algorithm. Our experiments on a Moto 360 smartwatch indicate that watch battery lives can be extended by up to 8.25× (19.13h) by running MiLift instead of unoptimized tracking apps. Even with a continuous execution of MiLift, the watch battery can last for more than a day and therefore will not require extra charging by users.

2.2 Challenges and Design Choices

We discuss several key challenges towards an autonomous and efficient workout tracking system and highlight the design choices made in MiLift.

2.2.1 C1: Single-device Sensing

Workout tracking apps running on smartphones cannot accurately sense user activities when the phone is placed away from the user. Moreover, workout exercises typically involve movements of different body segments such as hands, arms, waists, and legs and cannot be accurately monitored by a single smartphone. Prior tracking algorithms either placed more than one sensing device (e.g. a phone and a watch or multiple wearable sensors) on a user [CCC07, CSG13, MLN16, MLN15] or required instrumentation of weightlifting equipment [DSY15a]. In contrast, smartwatches are less intrusive since most users wear them for the majority of the day. Watches can sense wrist orientations and partial torso movements

whereas smartphones, typically carried in pockets, can only capture body postures. Therefore MiLift uses a single smartwatch to replace smartphones and other sensors previously used for workout tracking.

2.2.2 C2: Automatic Segmentation

Most workout tracking apps require users to manually choose workout types and start/stop the tracking of each session. If a user fails to mark the session end in time or even forgets to do so, the tracking algorithm could overestimate the current session and/or consume excessive energy. Although prior work proposed different classification models to recognize the presence of exercises, to identify exercise types, and/or to quantify the amount of exercises, none of them combined these models in an end-to-end system on commercial devices to automatically segment user activities. To eliminate user burdens, MiLift can detect a user's activity transitions and automatically segment different workout exercises using a two-stage classification model, which is motivated by prior work on hierarchal activity classification [KLL10, ZMN10, XSW11]. Specifically, MiLift first applies a lightweight classifier on low-power inertial sensor data to determine high-level user activities such as non-workout, walking, running, and weightlifting. MiLift then starts the fine-grained weightlifting analysis only upon detection of weightlifting exercises. The multi-stage model can provide detailed information on demand and can avoid unnecessary analysis to save energy.

2.2.3 C3: Weightlifting Exercise Tracking

Medical researchers have shown that weightlifting exercises (or weight training, strength training) can help improve metabolic function and muscle strength [PFB00, CHS05]. Although *free weight exercises*, such as those using dumbbells and barbells, can sometimes lead to greater muscle activities than *machine-based weightlifting exercises* [MF94], both should be combined to maximize training outcome [CDS05, PFB00]. Most workout tracking apps monitor cardio activities such as walking and running, but few can efficiently track weightlifting exercises, including the following metrics:

- Number of *sets*: each set is a workout session that includes several repetitions of the same weightlifting exercise.
- Number of repetitions (*reps*) in a set: each rep is an instance and the basic unit of a particular weightlifting exercise.
- *Type* of the exercise: for example, dumbbell bicep curl.

With noises such as those caused by improper exercise forms or non-workout activities in-between exercises, it remains challenging to accurately and efficiently capture repeating human movements during exercises. MiLift exploits repeating patterns of human arms during weightlifting exercises as demonstrated in Figure 2.1, and performs weightlifting classification include set detection, rep counting, and exercise type classification. MiLift considers 10 types of weightlifting machine exercises (#1 to #10) and 5 types of dumbbell-based free weight exercises (#11 to #15).

2.2.4 C4: Efficient Resource Usage

The limited battery capacity of mobile devices calls for a detailed study and optimization of energy consumption of workout tracking services. We propose that the battery life of a smartwatch should last for at least 16 hours (a full day except sleeping) even with continuous workout tracking, so that users do not need to charge the watch during the day. Previous approaches can rapidly drain out device batteries because of the continuous nature of inference executions and the use of complex algorithms for weightlifting tracking such as Dynamic Time Wrapping (DTW) [PHK13]. MiLift applies two techniques to achieve efficient resource usage on watches: 1) a context-aware duty-cycling optimization, and 2) a lightweight algorithm for weightlifting detection.

2.3 Related Work

Apps on smartphones explore a variety of human contexts including transportation modality [RMB10, HNT13a], social interactions [XLL13, NDA13], and physical and mental health-

Table 2.1: Summary of prior workout tracking approaches and whether they meet our design challenges.

Category	**C1**	**C2**	**C3**	**C4**
Mobile workout tracking apps	•			○
Mobile health frameworks	•	○		•
Weightlifting tracking systems		○	•	
Emerging apps (VimoFit [vim])	•		•	
MiLift (this paper)	•	•	•	•

○ indicates partial fulfillment.

iness [RMM10, LFR12, LLL13, HXZ13]. In addition, researchers have developed radio-based systems to track human activities [CMP12, PGG13, AKK14, WHY15]. Recently, the emergence of Apple Watch [app], Android smartwatches (e.g., the Moto 360 [mota]), and Microsoft Band [msb] also prompts the development of context inferences on smartwatches and wearable devices [MPA14, SBS15, NGG15, LGM15].

We group prior approaches on workout tracking and management using mobile devices into the following categories (summarized in Table 2.1):

Mobile workout tracking apps including RunKeeper [run], Strava [str], and MapMyRun [map] focus on cardio exercise tracking and management using a single smartphone. Due to the limited sensor coverage when using phones, these applications only support specific types of cardio exercises and require users to manually start and stop workout sessions.

Mobile health frameworks such as Google Fit [goo], Apple HealthKit [hea], and Microsoft Health [msh] provide APIs for both app developers and data scientists. Each framework also provides an app for users to manage workout tracking. Typically built-in as part of the mobile operating systems, the resource usage of these frameworks are well optimized. However, they mostly focus on cardio exercises and do not support tracking of weightlifting exercises. The front-end apps require manual selection of workout types as well.

Weightlifting tracking systems: The weightlifting classification system in MiLift is motivated by several prior work in this space. Chang et al. [CCC07] performs free weight exercise tracking using two wearable accelerometers, one in a user's glove and one in a waist pocket. Their system can automatically classify types of free weight exercises and count reps. MyoVibe [MLN15] and Burnout [MLN16] embeds wearable sensors in fitness clothing and leverages muscle vibrations to identify muscle activations and to estimate fatigues during exercises. RecoFit [MSG14] provides a model to segment exercises from non-exercise activities and count weightlifting exercises, but does not study the feasibility of running continuous tracking on user devices. NuActiv [CSG13] uses a smartwatch and a smartphone to track exercises and everyday activities by decomposing activities into semantic attributes. It applies zero-shot attribute-based learning for recognizing newly unseen types of exercises. Pernek et al. [PHK13] achieves repetition detection of weightlifting exercises using Dynamic Time Wrapping. FEMO [DSY15a] tracks repeating patterns of free weight exercises by instrumenting dumbbells with RFID sensors and measuring frequency shifts caused by Doppler Effect. myHealthAssistant [SBV11] employs multiple customized sensors on a user's body and uses a smartphone as a hub to track weightlifting exercises. Mortazavi et al. [MPA14] can count reps of weightlifting exercises but call for manual type selection. In contrast, MiLift is the first end-to-end system on commercial smartwatches that uses automatic segmentation to track both cardio and weightlifting exercises without requiring users to start/stop tracking and/or select workout types. MiLift also tackles challenges such as single device sensing and efficient resource usage described in Section 2.2.

Emerging apps and web services including the VimoFit app [vim], the Atlas wristband [atl], and the Microsoft Band API [ban] aim at autonomous workout tracking. Although some of them support rep counting for guided workouts, they still require certain manual inputs from users. Section 2.6.3 shows the high energy consumption of VimoFit and compares it with MiLift. Other services such as JEFIT [jef], WorkoutLabs [wor], and Gymwolf [gym] provide workout management and guidance from self-report workout data.

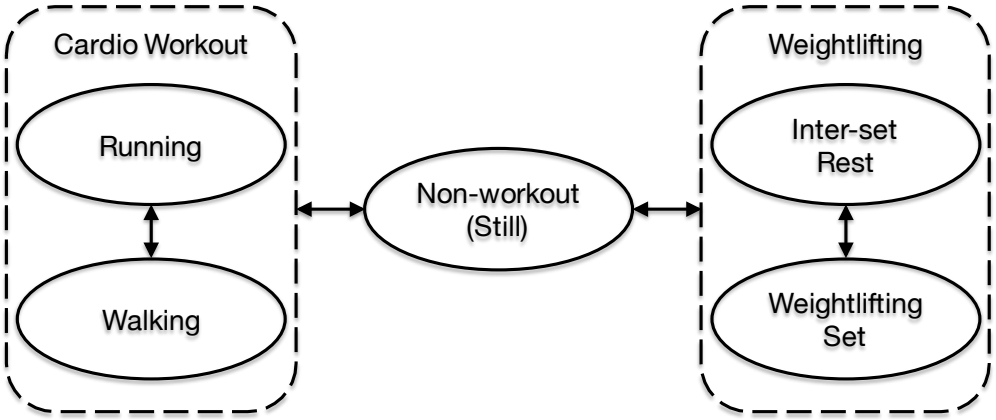


Figure 2.2: State transitions of a user's workout activities.

2.4 System Architecture

In this section, we propose the system architecture of MiLift, describe the implementation of a two-stage classification model, and discuss optimization techniques.

2.4.1 Overview

MiLift aims to track workout activities of a user using only a smartwatch (C1). Exercises of a user can be categorized into three groups: non-workout (still), cardio workouts such as walking and running, and weightlifting. Figure 2.2 describes state transitions of these exercises. A user can start weightlifting or cardio workouts from the non-workout state. However, we assume that a user cannot perform weightlifting right after a cardio workout or vice versa because there has to be a short transition period to non-workout first, for example, taking a rest or preparing for the next exercise. State transitions also take place within each group: The user can switch between walking and running during a cardio session or take rests between weightlifting sets.

Motivated by the state transitions of user activities, MiLift uses a two-stage classification model to accurately and efficiently track workout activities, shown in Figure 2.3. The model contains two stages:

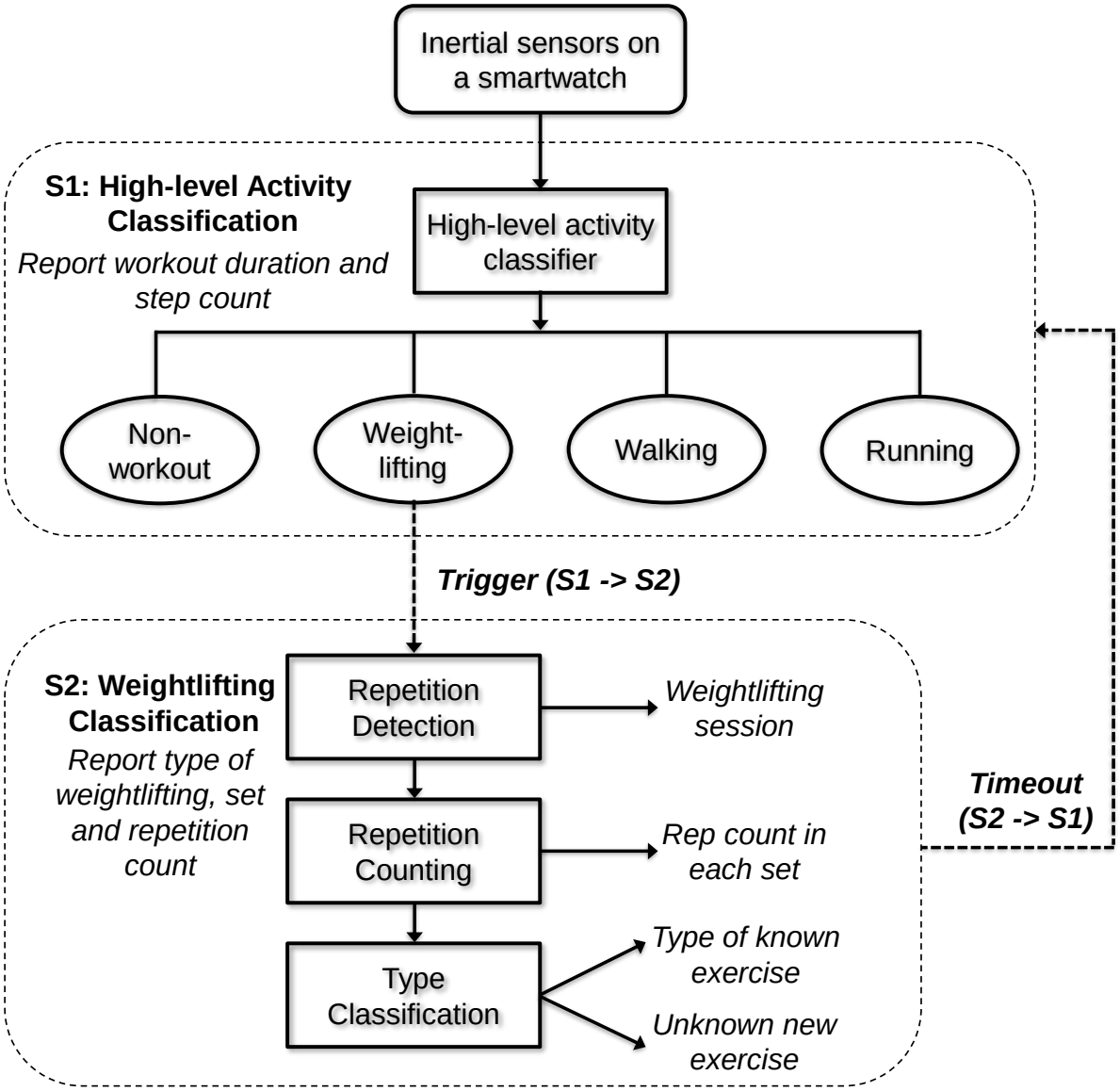


Figure 2.3: MiLift architecture and state transitions.

S1: High-level activity classification: S1 aims at detecting high-level activity state transitions of a user shown in Figure 2.2 and tracking cardio workouts. It implements a lightweight algorithm to label a data window with activities including non-workout, walking, running, and weightlifting. If walking or running is detected, session duration and step counts are logged. Once weightlifting is detected, MiLift wakes up the weightlifting classification module described below which involves more complicated computations.

S2: Weightlifting classification: This module analyzes inertial data and performs detailed weightlifting classification (C3) including set detection, rep counting, and type classification. We have implemented an autocorrelation-based algorithm and a lightweight revisit-based algorithm to achieve efficient resource usage (C4).

To perform automatic segmentation on user activities and eliminate the burden on users to manually start/stop tracking of sessions (C2), MiLift transits between S1 and S2 based on current user contexts. The state transition also helps preserve battery energy of smartwatches (C4).

2.4.2 High-level Activity Classification

In MiLift, inertial data samples from smartwatches are first labeled by a high-level activity classifier. Motivated by prior work on mobile sensing [MSS06] [RMB10] [HNT13a], the classifier takes a window of 3-axis accelerometer data and generates an activity label such as *non-workout, walking, running*, or *weightlifting*. The classification pipeline includes:

Sampling and preprocessing: MiLift uses a 1s classification window on accelerometer data sampled at 50Hz. The choice of 1s window size is also seen in previous activity recognition work [RMB10] [HNT13a] so that the data window contains enough samples for feature functions but also keeps only one type of activity in a single window. Data is buffered for each second and then sent to the next stage for feature extraction.

Feature extraction: This submodule takes an 1s window of accelerometer data and reduces its dimension by applying a set of feature functions in both time and frequency domains. Features considered in our system are mean, variance, range, root mean square (RMS), mean absolute deviation (MAD), magnitude, skewness, kurtosis, quartiles, median, and energy coefficients between 1-5Hz from Discrete Fourier Transform (DFT). For each feature, MiLift fuses three accelerometer axes by computing on each axis separately and then taking an average. To improve performance and reduce feature calculation workload, we apply two feature selection algorithms implemented in scikit-learn [PVG11a] including the univariate statistical test and the tree-based feature ranking. These two selection algorithms rank features based on their significances and select 7 of them for use in MiLift, including mean, standard deviation, MAD, range, the 1st quartile, the 2nd quartile, and DFT at 5Hz. The feature vector is then sent to the next stage for classification.

Classification: The classification module takes a feature vector as input and generates an activity label for each window (i.e., every second). We have implemented two categories of classification models. The first approach uses Conditional Random Fields (CRF) to continuously label each data window represented by a feature vector. CRF is commonly used for sequence labeling tasks such as part-of-speech tagging and image segmentation [SM06]. Prior studies also use CRF for tasks on sensor data such as room-level occupancy inference from motion sensors and human activity recognition from wearable accelerometer readings [YTS14] [LLN11] [VVL07] [NDH10].

In MiLift, CRF is used to exploit the temporal correlation among workout activities. Each state y_t in CRF corresponds to a ground truth activity label at time t, and each accelerometer feature vector is used as an observation x_t. The joint probability of a state sequence $\mathbf{y}$ and an observation sequence $\mathbf{x}$ is modeled as:

$$p_\lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\lambda(\mathbf{x})} \cdot \exp(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i \mathbf{f}_i(\mathbf{y}_j, \mathbf{x}_j))$$

where n is the total time considered, $\mathbf{f}$ is a set of m feature functions used internally by the CRF (not to be confused with our accelerometer features), λ is a weight vector for all feature functions, and Z serves as a normalization term. Because most CRF implementations only accept nominal (string) observations, accelerometer readings cannot be fed into these CRF implementations because they are floating-point numbers. Therefore, we use the k-means algorithm to group accelerometer readings into several clusters and use cluster IDs as the input to the CRF. We have considered different numbers of clusters in k-means, ranging from 4 to 20 to acquire the best CRF training performance.

In addition to the CRF, our second approach for high-level activity classification is to first apply an instance classifier that labels each one-second (1s) window, and then uses a Hidden Markov Model (HMM) to smooth the label series. We consider three popular instance classifiers including Random Forest (RF), Decision Tree (DT), and Support Vector Machine (SVM). Parameters of all models are tuned in the training stage, such as maximum depths for DT, number of trees and size of feature subsets used for each split for RF, and kernel types for SVM. These models label each individual 1s window independently. However, adjacent windows are temporal-correlated as workout activities are continuous and would not transit frequently within a short period. For example, a user may briefly raise his or her arms while sitting in an office but an instance model may classify this action as weightlifting. We apply an HMM classifier to smooth the output activity labels of instance classifiers and filter out unlikely activity transitions for better accuracy. To generate the HMM model, we use ground truth activity labels as hidden states and output labels from instance classifiers as observations.

2.4.3 Weightlifting Classification

The second state in the two-stage classification model of MiLift is a weightlifting classification module. It is waken up by the high-level activity classifier when users are performing weightlifting exercises. This module achieves three tasks: (1) detecting weightlifting sessions and label boundaries of *sets*, (2) counting number of *reps* (repetitions) in each set, and

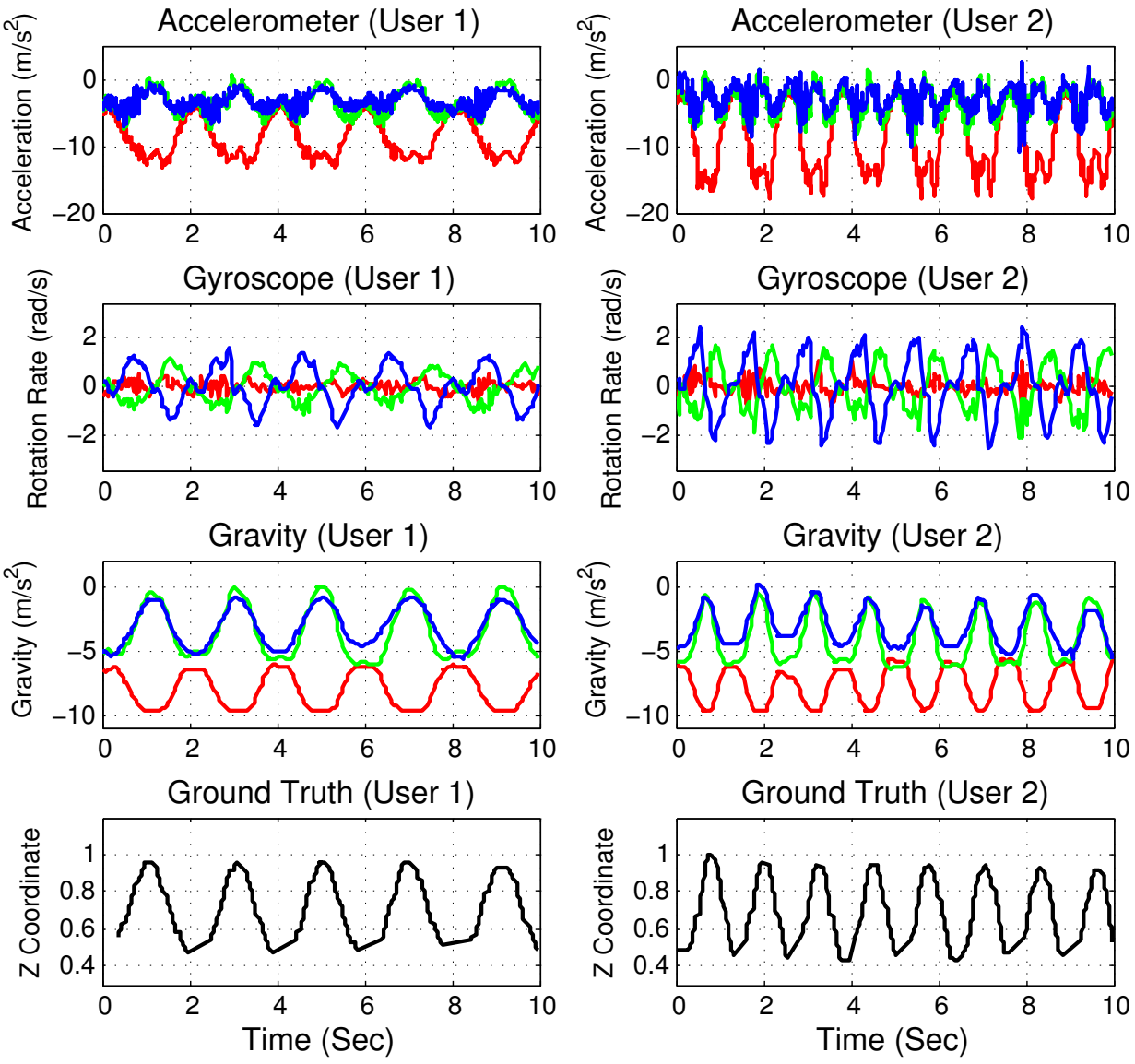


Figure 2.4: Sensor traces comparison of dumbbell single arm row performed by two users, showing that gravity sensor can best demonstrate the repeating patterns.

(3) classifying type of current weightlifting exercise. The weightlifting classification module must conserve battery energy of smartwatches (C4).

2.4.3.1 Sensor Choice

There are several available inertial sensors on a smartwatch that can capture human motions, including but not limit to accelerometers and gyroscopes. A gravity sensor is a low-power software sensor fusing both accelerometer and gyroscope [and]. Gravity sensor data can reflect wrist orientations of users and can be a good indicator of repeating weightlifting exercises with wrist movements. Using the dumbbell single arm row exercise as an example, Figure 2.4 demonstrates the ability of different sensors to capture weightlifting exercises. Sensor data traces collected from two users each performing one set of exercises are compared with ground truth motion traces captured by an OptiTrack Prime 13 tracking system [opt]. For both users, the gravity data is less noisy than the accelerometer and gyroscope data and can better highlight the repeating pattern in signal traces generated by weightlifting exercises. Moreover, the amplitudes of the gravity data across two users are more consistent compared with the other two sensors, indicating that gravity sensor can better identify types of weightlifting exercises across different users. Therefore, MiLift uses the Android gravity sensor for weightlifting detection. However, we acknowledge that single-point sensing has limited coverage on the human body and will discuss this issue in Section 2.7.

Figure 2.5 (a) demonstrates a trace of gravity sensor data during a user's weightlifting exercise session (bicep curl machine). This session can be clearly identified by repeating patterns in the gravity sensor trace. Moreover, since the corresponding gravity data are cyclical, each rep in this session can be identified and the number of reps can be counted. In contrast, we see arbitrary patterns recorded during non-workout periods shown in Figure 2.5 (b). Therefore, MiLift quantifies repeating patterns in gravity data to detect weightlifting sessions and count reps, using two approaches including an autocorrelation-based algorithm and a revisit-based algorithm. MiLift then classifies the type of the detected weightlifting activity.

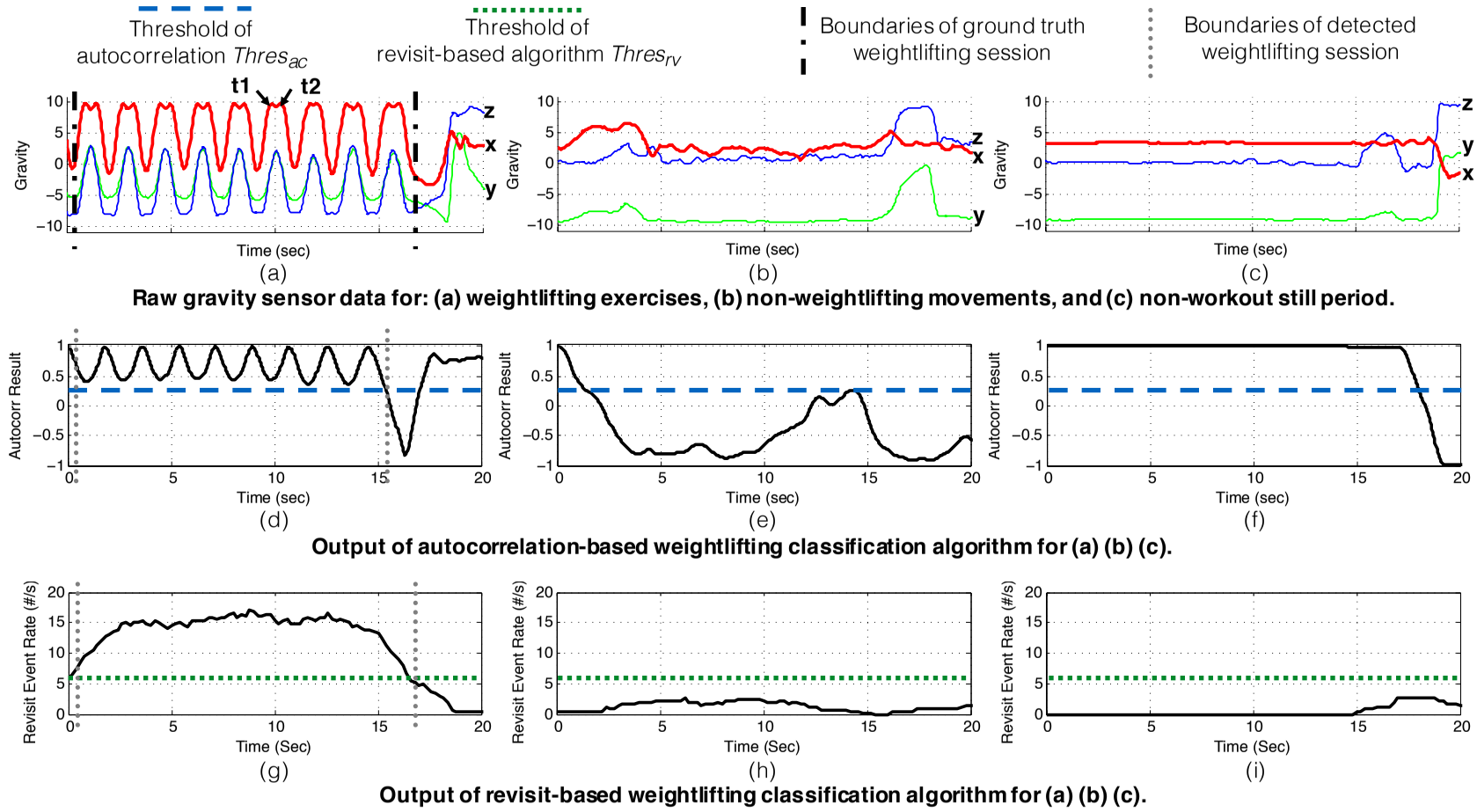


Figure 2.5: Results of applying autocorrelation-based algorithm and revisit-based algorithm on gravity sensor data for three cases: weightlifting (bicep curl), non-weightlifting movements, and non-workout still period.

2.4.3.2 Autocorrelation-based Weightlifting Detection

Weightlifting session detection: Autocorrelation is a technique for examining the periodicity of a signal series. It calculates correlation between a sample window at time t and another same-size window with an offset in time known as the *lag* ℓ. Prior work (e.g., RecoFit [MSG14]) also considers autocorrelation for detecting repeating patterns in sensor signal. The output of an autocorrelation series can be represented as a function of ℓ:

$$\mathbf{R}_\ell = \frac{S_t^{\mathbf{T}} S_{t-\ell}}{|S_t| \cdot |S_{t-\ell}|}$$

where S_t is a $w-$element vector representing a signal window with size w starting from time t. If a sample window and another window with lag ℓ have similar signal patterns, the autocorrelation value $\mathbf{R}_\ell$ will be close to 1 indicating significant similarity between these two windows. We apply a threshold value $Thres_{ac}$ on $\mathbf{R}_\ell$ to identify weightlifting sessions. In order to capture periodicity of weightlifting exercises, we choose a small window size but longer than the typical duration of a rep ($6s$ from our analysis) to cover all possible rep lengths.

For weightlifting session detection, we apply autocorrelation using $1s$ sliding windows on gravity data to capture all possible sessions. Figure 2.5 (d)(e)(f) plot autocorrelation results when applied on three different cases of sensor readings shown in Figure 2.5 (a)(b)(c). The figures also show the threshold $Thres_{ac}$ used for repetition detection as well as both ground truth and detected weightlifting sessions. When autocorrelation is performed on a weightlifting session, the output values are close to 1 and demonstrate spike patterns (shown in (d)). A peak in autocorrelation results is aligned with a valley in raw data because this is the beginning of a new rep and autocorrelation discovers the maximum similarity between current window and the previous one. MiLift applies $Thres_{ac}$ on peaks from autocorrelation results to identify a weightlifting session. In contrast, when the input signal trace indicates no repeating wrist motion, for example when a user is adjusting the weights, the autocorrelation output values show no peaks and are relatively low (shown in (e)). However, one exception exists that can lead to constant high autocorrelation values. When a user's wrist is stationary

during breaks and gravity readings are relatively constant, the output values can remain close to 1 (shown in (f)). Our algorithm filters out such cases by removing sessions with consistently high autocorrelation values but no spike patterns.

Another issue in this algorithm is which axis should be consider for autocorrelation. From the gravity signals shown in Figure 2.1, there is no universally dominant axis with the most obvious peak-valley patterns. In MiLift, we explored three techniques to select the best gravity axis: (1) summing up absolute values of each axis, (2) taking absolute values of sums of each axis, and (3) performing autocorrelation on each axis separately and choosing the one with the best result. Our experiment indicates the last strategy is the most robust against false positives and demonstrates the best accuracy among the three.

Rep counting: After weightlifting sessions are detected and labeled, the rep counting part is achieved by simply performing a naive peak detection on autocorrelation results. Because non-repeating signals are already filtered out, the number of reps can be derived by simply counting peaks.

2.4.3.3 Revisit-based Weightlifting Detection

Although the autocorrelation-based algorithm can effectively detect repetitions in time-series data, it can incur $O(wL)$ overhead upon arrival of each new gravity sample where L is the maximum lag of interest and w is the window size, leading to heavy computations (challenge C4). We propose a lightweight algorithm which quantifies revisit events in gravity sensor data to detect weightlifting sessions and count reps with less computation and better efficiency.

Weightlifting session detection: Figure 2.5 (a) shows that gravity signal demonstrates repeating patterns during weightlifting sessions. This implies that gravity sensor readings (i.e., $(grav_x, grav_y, grav_z)$) with similar values can be found within a short time frame, typically not longer than the period of a rep. For example, the gravity reading marked as $t1$ has a similar value to sample $t2$ about one second later. A necessary condition of a

weightlifting session is that repeating patterns of sensor readings are seen within a short time window whose length is similar to the typical length of one rep (6s from our analysis), indicating that the majority of gravity samples in this window have a similar sample within the same window. However, the strategy of comparing sensor data windows whenever a new sample arrives can cause high overhead, as seen in autocorrelation. Instead, we present a heuristic algorithm to discover repeating patterns in the signal stream, called the *revisit-based approach*:

First, each sample of sensor readings ($grav_x$, $grav_y$, $grav_z$) is discretized at an interval of I giving it a discretized value vector D where:

$$D = \left(\left\lfloor \frac{grav_x}{I} \right\rfloor \cdot I, \left\lfloor \frac{grav_y}{I} \right\rfloor \cdot I, \left\lfloor \frac{grav_z}{I} \right\rfloor \cdot I \right)$$

Therefore two similar samples will share the same D value. Our experiments indicate the best choice of I is 0.6.

Second, we maintain a hash table $\mathcal{H}$ to store incoming samples where D is used as the key and timestamp t of current sample is used as the value:

$$\mathcal{H}[D] = t$$

Third, we define a *revisit event* occurs when a hash collision happens, for instance, when a sample with a key D and a timestamp t arrives, the key D already exists in $\mathcal{H}$ with a value t' ($\mathcal{H}[D] = t'$). The *revisit time frame* $T_{revisit}$ of this event is defined as the time difference between two samples with the same discretized value D:

$$T_{revisit} = t - t'$$

Since we only use revisit events to identify reps, we set a threshold value $Thres_{revisit}$ as 6s to cover the longest possible rep. We only consider revisit events with $T_{revisit} < Thres_{revisit}$. $\mathcal{H}[D]$ is then updated with value t.

Finally, we denote *revisit event generation rate* to represent the number of revisit events generated in the past $1s$ at a given time. A high revisit event generation rate suggests that revisit events within a short time frame are frequently generated. Thus, it indicates the occurrence of repeating patterns, which in turn suggests an ongoing weightlifting session. We apply a threshold value $Thres_{rv}$ on the revisit event generation rate of each second to select significant (high) values.

The above revisit-based algorithm only takes $O(1)$ time to update the revisit event generation rate when a new sensor reading arrives and can significantly reduce computational cost compared with the $O(wL)$ overhead incurred for each new sample by the autocorrelation-based algorithm.

Figure 2.5 (g)(h)(i) plot results of our revisit-based algorithm on three different input traces shown in Figure 2.5 (a)(b)(c). The figures also show the threshold $Thres_{rv}$ used for repetition detection as well as both ground truth and detected weightlifting sessions. If input signals have cyclical patterns due to repeating motions, the revisit event generation rate raises and remains high until the end of current weightlifting session (shown in (g)). In contrast, revisit event generation rates are much lower if there is no repeating pattern in the input trace (shown in (h)). However, whenever a user is stationary and the gravity readings are relatively constant, revisit events will be generated at a high rate equal to the sensor sampling rate. Our algorithm excludes this situation by discarding any sensor reading that has the same discretized value D as its immediate predecessor. As shown in Figure 2.5 (i), this effectively prevents stationary motions from being detected as weightlifting.

Rep counting: After the beginning and end of weightlifting sessions are detected, our revisit-based algorithm performs peak detection on raw gravity sensor data to count reps. However, this is more difficult than the peak detection step involved in the autocorrelation-based algorithm because noises in raw data are not filtered out and therefore we cannot simply count peaks for the number of reps.

There are three issues preventing an accurate rep counting using naive peak detection on

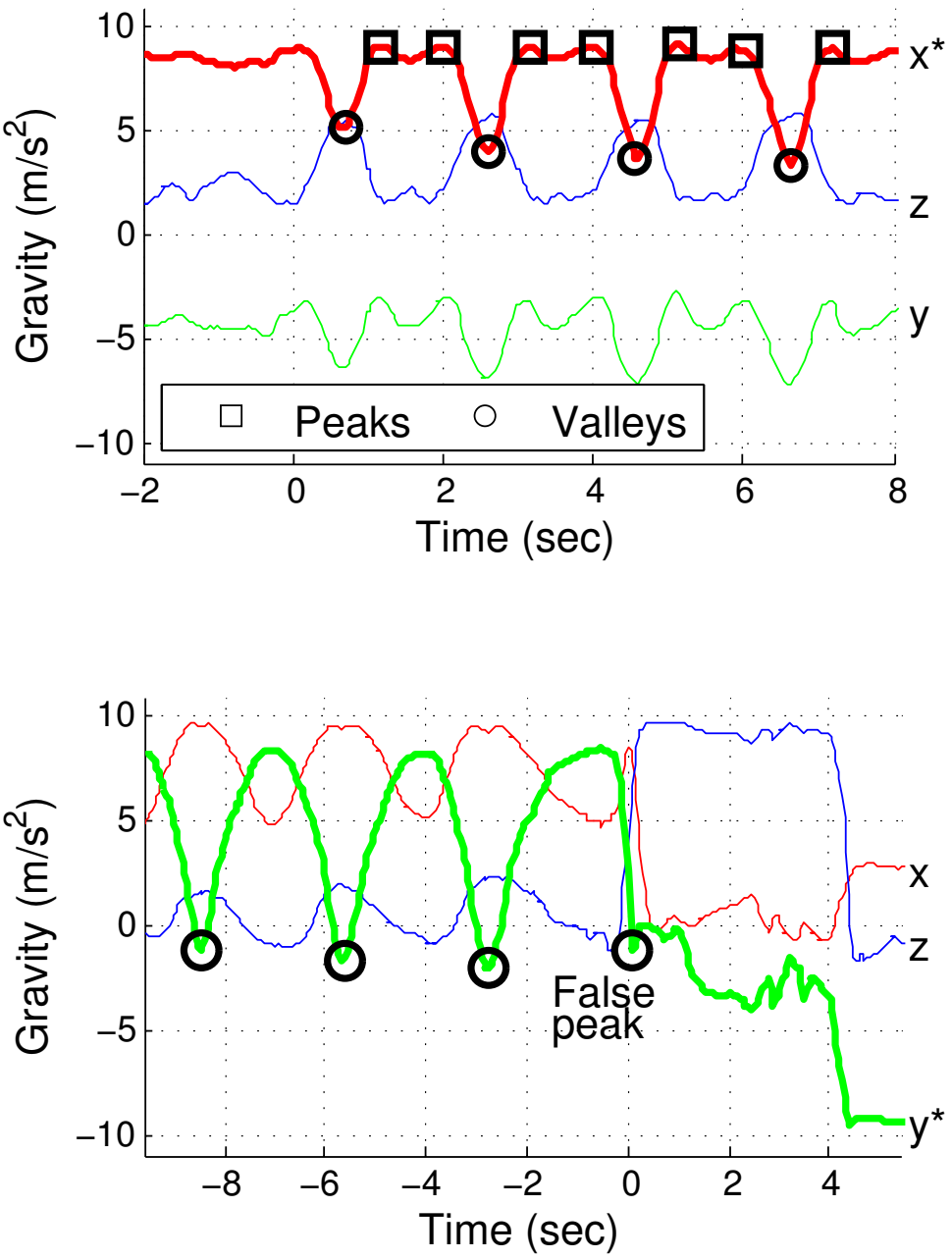


Figure 2.6: Two cases that can lead to failures of naive peak detection. Left: for bicep curl if we only consider upper peaks, each rep will be counted twice. Right: for ab crunch the weightlifting session boundary looks similar to a rep leading to a false count. In both graphs, the dominant axis is marked with stars.

raw gravity data. First, a dominant axis that presents the most distinct repeating patterns out of three gravity axes has to be selected. Second, neither upper peaks nor bottom peaks (valleys) always better indicate repeating patterns. As shown in Figure 2.6 left, simply counting peaks yields double-counting errors because upper parts of bicep curl reps sink in the middle resulting in vertically reversed w-shapes in the signal, i.e. two peaks. Third, the weightlifting session boundaries detected above may not be precise. Figure 2.6 right shows that if we consider bottom peak (valley) values, the valley at $t = 0$ satisfies all constraints even if it is not a rep but the end of this session.

To address the first issue, we select the axis with the largest range between the maximum and minimum value within the session to reduce ambiguities during rep counting. Next, we have to decide whether the number of reps should be derived by counting peaks or valleys. We use vertical displacements within a small delta time (i.e. second derivatives) to capture spikiness of peaks/valleys where a larger displacement indicates a spikier peak/valley. For both peaks and valleys, we compute the average vertical displacement V and count the one with larger V as number of reps to improve the counting accuracy. In the example of Figure 2.6 left, valleys should be considered since they have larger vertical displacements than peaks. Finally, to reduce false rep counts from weightlifting session boundaries, we consider the three axes as a whole when ambiguity occurs at the beginning or end. Taking the example of Figure 2.6 right, the average reading of three gravity axes at $t = 0$ is different from any other detected valleys, allowing us to remove it as a false rep count.

Both the weightlifting detection algorithms require no model training. Because neither algorithm considers any user-specific feature, our proposed weightlifting algorithms are user-independent as well. Note that weightlifting exercises in real life could lead to noises to the sensed gravity signal, such as those caused by tiredness or improper exercise forms, as discussed in Section 2.6.1.2, and therefore lead to errors of both algorithms.

2.4.3.4 Weightlifting Type Classification

In routine weightlifting exercises, a user is not only interested in statistics about sets and reps, but also the type of weightlifting exercises performed in each set. Once weightlifting sessions are detected and numbers of reps are calculated, MiLift starts weightlifting type classification and labels each detected session with an exercise type.

Our intuition for weightlifting type classification is that wrist positions of a user during different weightlifting exercises will lead to unique orientations of a smartwatch. To quantify watch positions, MiLift aggregates 3-axis gravity sensor readings from the current weightlifting session and computes a set of features for each axis, including mean, standard deviation, minimum, maximum, and range. MiLift then applies an SVM classifier to label the type of the current session. We choose SVM here because of the relative simplicity of this classification problem and the ability of SVM to generate confidence scores for each class of types, which can be used to determine if an exercise type is unseen. To enhance the performance of the SVM classifier, we have considered different kernels such as a linear kernel, a Gaussian kernel (with different radius r), and a polynomial kernel (with different degree d).

Another significant aspect of type classification is the ability to determine whether a new sample instance belongs to known exercise types during training or a new type of exercise. To identify new types, MiLift takes advantage of confidence scores reported by the SVM classifier. For each incoming instance, a vector of confidence scores is calculated by the SVM, representing the probability that this instance belongs to each known type class. If the maximum probability in this vector falls below a certain threshold $Thres_{conf}$, MiLift determines this instance belongs to a new type of weightlifting exercise and asks the user for a correct label. This also enables MiLift to perform active learning by re-training the model once a new exercise type is seen. We plan to address this topic in the future (Section 2.7).

2.4.4 Context-aware Optimization

To achieve both automatic segmentation (C2) and efficient resource usage (C4), MiLift takes advantage of available user contexts and applies a context-aware state transition mechanism.

Table 2.2: Summary of data collection in our user study.

Number of participants	22	Time of non-workout	14.88 hours
Number of weightlifting types	15	Time of walking	9.22 hours
Number of weightlifting sets	2528	Time of running	7.95 hours
Number of weightlifting reps	24408	Time of weightlifting	36.15 hours
Total duration: 68.20 hours			

Figure 2.3 shows the transition among two classification states, *S1* and *S2*.

S1: MiLift executes the high-level activity classification in this state. It involves sampling of low-power accelerometer and incurs only lightweight computation and low energy consumption. Because a user typically keeps the same activity for a period of time and will not switch activities frequently within seconds, MiLift duty-cycles the execution of S1 by a 20% schedule. MiLift classifies a 1s window every 5s so that it is able to capture most activity transitions while conserving battery energy. Note we design the 20% duty-cycle schedule as a system parameter and it can be changed based on user preferences and current battery states.

Moreover, to prevent everyday activities from being mis-classified as gym exercises (i.e. reducing false positives), S1 opportunistically leverages coarse location contexts such as those inferred from network accesses or WiFi signatures. For a given user, workout exercises mostly take place near similar locations (e.g. a gym) or WiFi networks (e.g. gym WiFi hotspots). MiLift can automatically learn the typical exercise location of a user after the first few app uses and from WiFi connection histories. If a coarse location is known from recent queries or WiFi scans initiated by the OS or other apps, it can be compared with the user's typical exercise locations. Once weightlifting is detected by S1, only when the two locations match will MiLift transit to S2. The opportunistic use of coarse locations as opposed to querying exact GPS coordinates ensures no additional energy is consumed.

S2: MiLift performs the more sophisticated weightlifting classification in this state. Al-

though S2 is designed to be lightweight and efficient, the complexity of this state is relatively higher than S1 because of the computation incurred by each new sensor reading. Therefore it is triggered by S1 and only starts executing upon detection of weightlifting exercises. When no new weightlifting exercise is detected for a certain timeout (e.g. 5min), MiLift transits back to S1 and performs the simpler high-level activity classification.

Although performing duty-cycled classification will not prevent the watch from being waken up frequently, i.e. the watch will still have to perform sensing and classification in S1 every 5 seconds, Section 2.6.2 and Section 2.6.3 have proven such an optimization to be effective since the watch can last for more than a full day with this two-stage classification model running continuously. This is because most of watch battery energy is spent on data sampling and computation rather than simply remaining power-on. Nevertheless, we acknowledge that the use of low-power co-processors (e.g. DSPs and dedicated sensor hubs) for always-on sensing and computation tasks will further reduce the energy consumption of context inferences, as discussed in Section 2.7.

2.5 Implementation

Hardware device: The commercial off-the-shelf watches we used for data collection include Moto 360 [mota], Moto 360 Sport [motb], LG G Watch R [lgw], and ASUS ZenWatch 2 [asu]. For experiments on energy profiling we used the Moto 360 which has a 1GHz OMAP3 CPU, 512MB RAM, 4G flash storage, and a 3.8V/320mAh Lithium-ion battery. We have implemented a smartwatch data recording app that logs accelerometer and gravity sensors from the watch. Users were also assisted by Google Nexus 5 phones during data collection. Although we used four different watch models for data collection, the recording app implements the same sampling frequency even when running on different models.

User study: Our data collection campaign was performed as part of a user study that involves 22 participants, approved by UCLA IRB. Participants are aged from 19 to 27 and consist of both males and females. All participants regularly engage in gym exercises. We

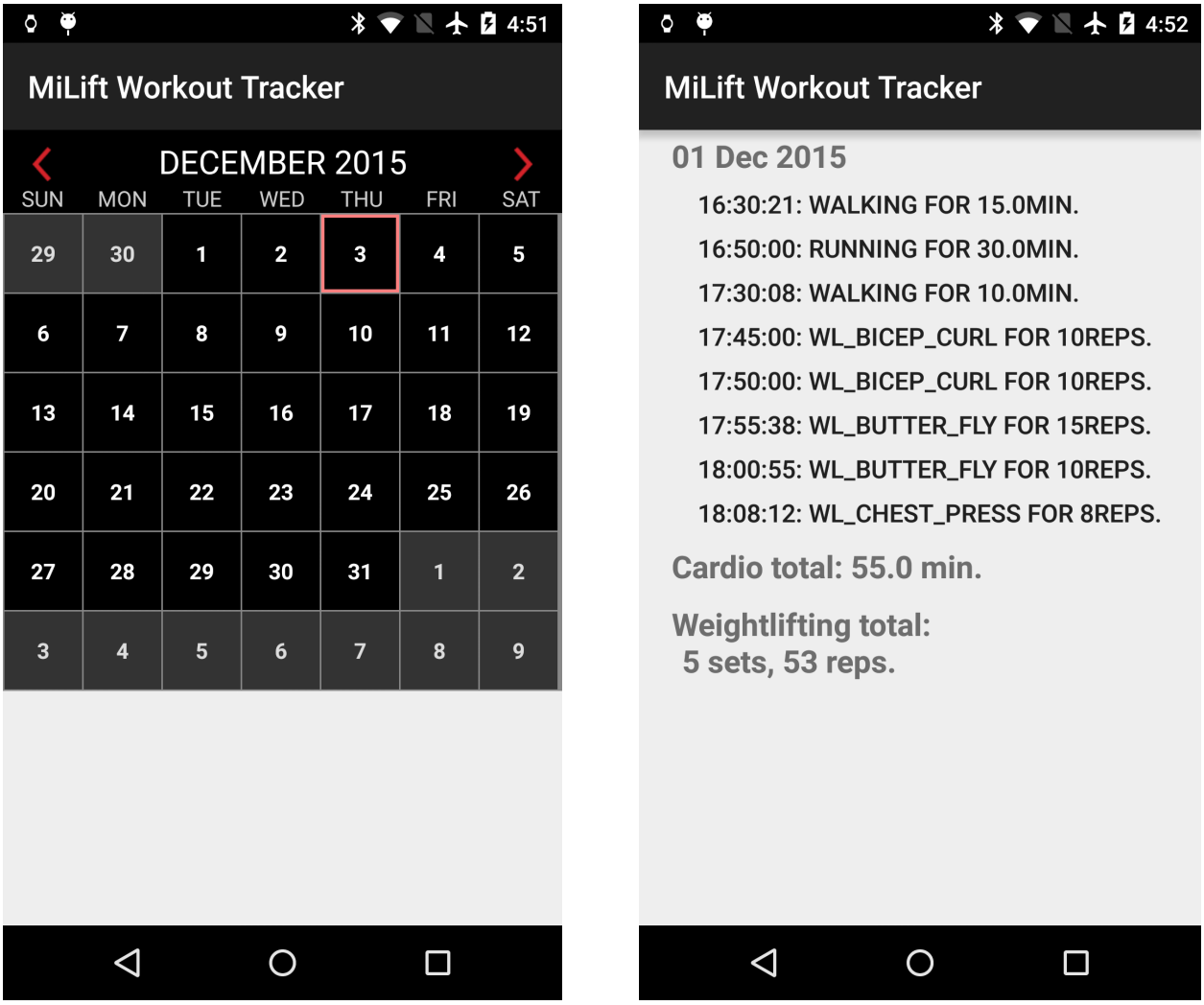


Figure 2.7: Screenshots of the MiLift Android app. Left: a calender for workout management; Right: a workout activity summary of a given date.

have collected 68.2 hours of workout data in total, including 2528 weightlifting sets (24408 reps), as summarized in Table 2.2. For ground truth recording, we used the Moves app [mov] to log cardio activities and let participants manually record set/rep counts in weightlifting exercises. In addition, we asked each participant to finish a post-completion survey on workout tracking (Section 2.6.3).

Offline analysis: With the collected data, we first conducted an offline analysis to train classification models. The high-level activity classifier and the weightlifting type classifier were implemented using the scikit-learn library [PVG11a] in Python. The HMM smoothing of high-level activities and weightlifting classification algorithms including session detection and rep counting were implemented in Matlab.

Android implementation: We implemented MiLift as an Android app using Android 5.1.1 (API 22). MiLift contains two components: a watch app that performs workout tracking and a phone app that provides visualization and assistance for workout management. The watch app executes real-time classifications as a background service and does not require user inputs. Multiple open-source projects were used to port trained classification model to Android, including CRF++ [crf] and jahmm [jah]. Figure 2.7 left shows the MiLift workout calendar where users can choose a particular day to view their progresses. Figure 2.7 right displays workout activities performed in a given day showing durations for cardio workouts and rep counts for weightlifting exercises.

2.6 Evaluation

We first evaluate MiLift using a set of micro-benchmarks including accuracy (Section 2.6.1) and power (Section 2.6.2) profiling of classification models. We then use a macro-benchmark to analyze the effect of MiLift on required user tasks and smartwatch battery lives compared with previous approaches (Section 2.6.3).

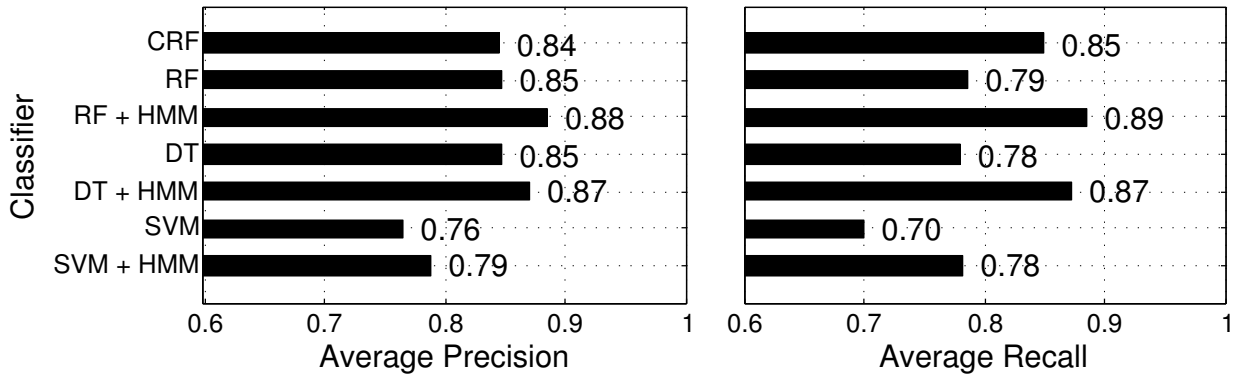


Figure 2.8: Weighted average precision and recall of high-level activity classification (10-fold cross validation).

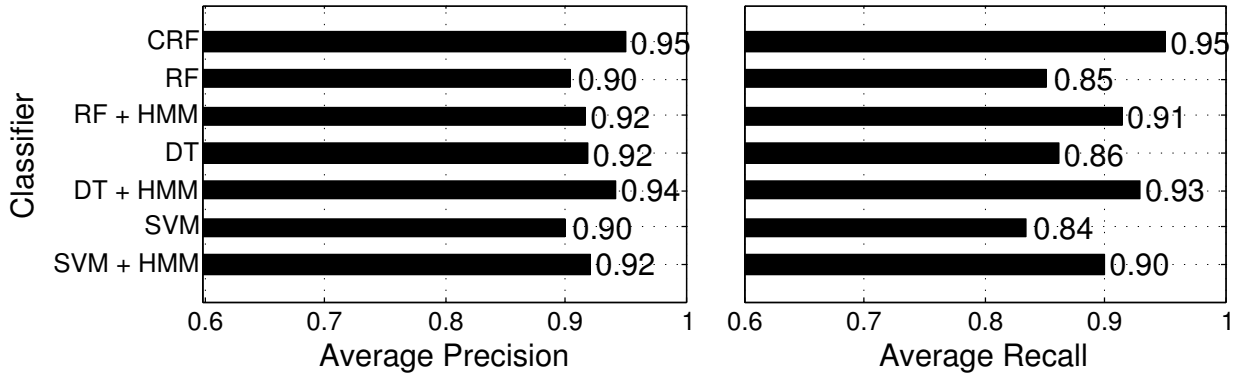


Figure 2.9: Weighted average precision and recall of high-level activity classification (15-hour all-day trace).

2.6.1 MiLift Tracking Accuracy

2.6.1.1 Accuracy of High-level Activity Classification

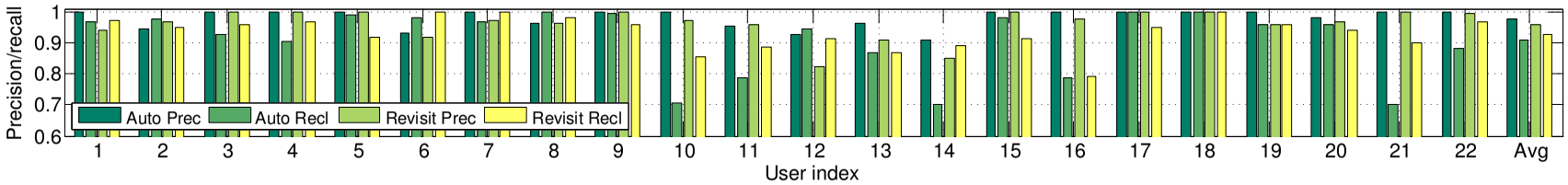
The high-level activity classifier is the first state of the two-stage classification model and aims at detecting high-level activities of users before triggering the weightlifting classifier. To select the best model for this classification task, we first used a 10-fold cross validation with data from all users to examine the performance of the four classifiers, including the

Table 2.3: Memory footprints of high-level classifiers.

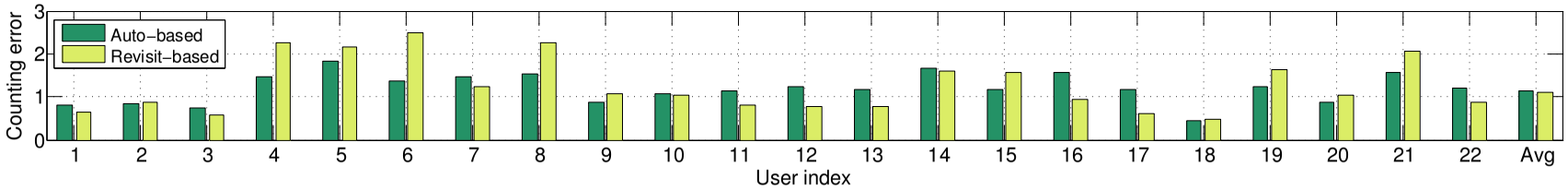
CRF	RF+HMM	DT+HMM	SVM+HMM
1.6 MB	105.1 MB	45 KB	5.6 MB

continuous graph model CRF and three instance classifiers RF+HMM, DT+HMM, and SVM+HMM. The best parameters selected for each classifier include maximum depth of 8 for DT, 64 classifiers and 4 features used at each split for RF, and linear kernel for SVM. For CRF, input feature values are first transformed to integers using cluster centers trained from k-means clustering algorithm, and our parameter tuning has chosen the use of 16 clusters for best CRF performance. Figure 2.8 plots the weighted average precision and recall for all four models from the cross validation, including results before and after HMM smoothing for the three instance classifiers. Among the four models DT+HMM and RF+HMM has the best overall performance (nearly 90%) but CRF only performs slightly worse (around 85%). As discussed in Section 2.4.2, the results suggest that HMM smoothing is crucial in reducing false positives and increasing the overall performance of all three instance classifiers.

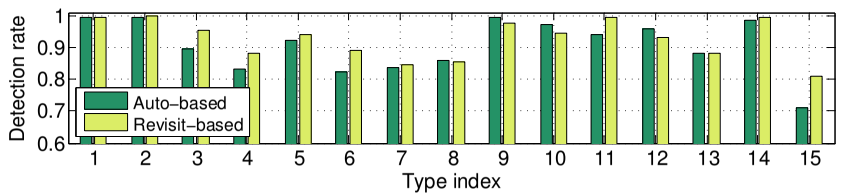
We then used the best model parameters selected from cross validation to showcase real-world performances of the high-level activity classifiers. We had one user collected an all-day data trace consisting of 15 hours of continuous accelerometer data. This trace includes the user walking around school, sitting during classes and study, running in a gym, and performing weightlifting exercises. Our entire dataset described in Section 2.5 is divided into two parts: all but this 15-hour data trace is used to train the high-level classifiers and this trace itself is used as the testing set. Figure 2.9 plots the resulting weighted average precision and recall for classifications on the all-day trace. All four models achieve above 90% average precision and recall. Among them, DT+HMM and CRF have the best overall accuracy but the other two models perform only slightly worse. The results above have proven that the high-level activity classification models in MiLift can successfully separate non-workout activities from gym exercises in real-world user scenarios. Moreover, continuous graphical models (e.g., CRF, HMM) can effectively increase the accuracy of time-series classification



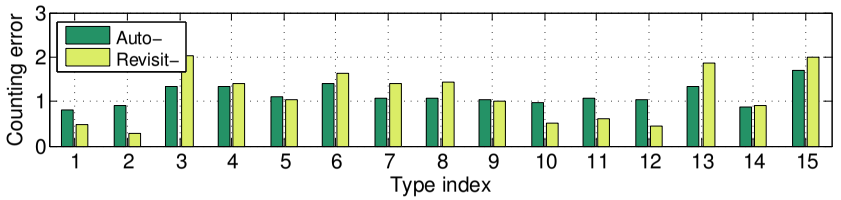
(a) Session (set) detection accuracy based on users.



(b) Rep counting error based on users.



(c) Session (set) detection accuracy by types of exercises.



(d) Rep counting error based on types of exercises.

Figure 2.10: Weightlifting detection performance, reported by users and by types of exercises (as shown in Figure 2.1).

tasks by exploiting temporal correlations in data.

Finally, with RAM becoming another power-hungry component in mobile SoC as RAM power can exceed CPU power in certain workloads [CH10], another factor in choosing classification models is the memory footprint. Table 2.3 compares the size of the four trained models. DT+HMM and CRF have much smaller model sizes compared with the other two because of the relative simplicity of their model structures. Considering both their classification precision/recall performances and the memory footprints, we have chosen to use DT+HMM and CRF for implementation in MiLift.

2.6.1.2 Accuracy of Weightlifting Classification

Session (set) detection and rep counting: We first evaluate the accuracy of weightlifting session detection and rep counting using our two proposed algorithms. Figure 2.10 (a) illustrates the weighted average precision and recall of session (set) detection based on users. In general, both algorithms demonstrate high overall accuracy, as autocorrelation-based approach achieves 97.5% precision and 90.7% recall while revisit-based approach yields 95.7% precision and 92.6% recall. Figure 2.11 shows three common cases seen in our user study that lead to errors in weightlifting set detection: (a) some users tend to take short pauses within a set, possibly because of tiredness; (b) users can sometimes finish the last rep in an incorrect posture; (c) some users may adjust their body postures (including wrist orientations) during a set. All three cases can cause irregular patterns in sampled gravity data and set detection errors from both approaches.

For rep counting, both the autocorrelation-based and the revisit-based approach have similar average errors, shown in Figure 2.10 (b). Here the rep counting error is defined as the difference between the true number of reps and the number of reps counted by our algorithms in each set. On average the autocorrelation-based algorithm and the revisit-based algorithm report errors of 1.125 and 1.122 reps per set, respectively. Some rep counting errors are caused by cases shown in Figure 2.11, because inaccurately detected session (set) boundaries can lead to rep miscounts as well. Other errors are results of unconventional

postures and actions performed by some users during weightlifting exercises. Participants can sometimes incorrectly log ground truth leading to errors as well. Nevertheless, the rep counting errors are insignificant considering a user performs 9.65 reps on average within each set. In addition, the errors are only slightly greater than user expectations from our survey (Table 2.5).

To study the root cause of errors, we report session (set) detection and rep counting results based on the type of weightlifting exercises. Figure 2.10 (c) plots the recall or detection rate of each exercise type shown in Figure 2.1. Note that the precision metric is not applicable here since a workout trace may contain multiple types of exercises and therefore we cannot identify false positive sessions for each exercise type. Both algorithms can nearly perfectly identify all sessions of bicep curl (#1), tricep extension (#2), tricep dip (#9), and dumbbell lateral raise (#14). However, seated row (#6), pec fly (#7), and rear deltoid (#8) are not well captured by either approach. This is because the vertical displacements in gravity data, i.e. the range between peaks and valleys, are small in all three axes for these exercises, making both algorithms susceptible to noises. Seated row does not involve substantial wrist motions and will not affect gravity readings much. Though both pec fly and rear deltoid require large forearm movements, the wrist trajectories of users fall into a horizontal plane and the watch rotates around z-axis in the global frame (perpendicular to the ground). Such rotation will not cause significant changes in gravity readings.

Although free weight exercises can in general cause more complicated body movements than machine-based exercises, MiLift's performances on session detection and rep counting for free weight exercises are similar to those for machine-based exercises. For free weight exercises, MiLift has high errors detecting dumbbell bench press (#15) due to similar reasons stated above. It is also worth noting that keeping body motions stable while performing dumbbell bench press exercises is known to be difficult even for experienced gym users, which adds more noises to the sampled gravity data.

Figure 2.10 (d) reports rep counting errors by exercise types and shows similar error trends as seen in Figure 2.10 (c): type #6−#8 and #15 have higher rep counting errors than others. Lat pulldown (#3) and dumbbell single arm row (#13) also have relatively

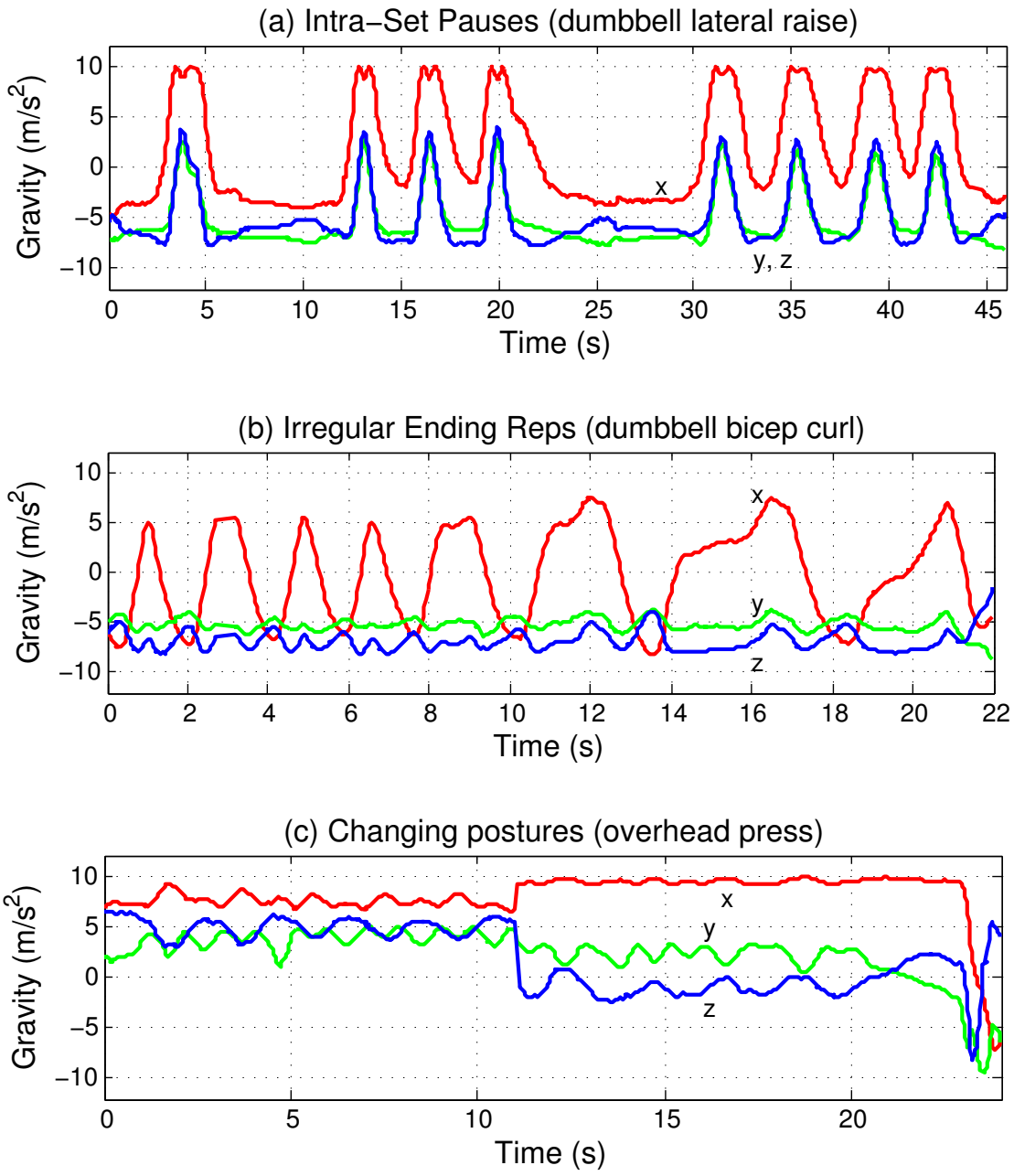


Figure 2.11: Three common error sources of weightlifting session (set) detection seen in our user study.

high errors due to unclear repetition patterns shown in gravity data traces from some users.

Weightlifting Type Classification: We have chosen a Gaussian SVM with $r = 0.05$ for MiLift to determine the type of weightlifting exercises in a session and identify unseen exercise types. We used a 10-fold cross validation on the entire weightlifting dataset to test the performance of this classifier, which shows a precision of 89.71%, a recall of 89.53%, and an F1-score of 89.48% (all weighted average) for all 15 exercise type classes. Figure 2.12 left plots the confusion matrix of all exercise types from cross validation, suggesting that our SVM model has good performance on all classes and does not bias towards certain weightlifting types.

Another important benchmark of weightlifting type classification is the accuracy of identifying newly unseen types. Section 2.4.3.4 proposes our algorithm in MiLift to identify new types by applying a threshold value $Thres_{conf}$ on confidence scores generated by the SVM. We evaluated the performance of this approach by conducting a leave-one-type-out cross validation. For each weightlifting type, we trained an SVM model using data from all other 14 types, and used this model to classify the entire dataset including instances of the 14 known types and the 1 unknown type. Because an instance of an unknown type can be similar to instances of one or more of the known types, the accuracy of identifying unknown types depends on the value of $Thres_{conf}$. A smaller $Thres_{conf}$ will allow more instances of unknown types to be correctly identified but will also lead to more false positives. Figure 2.12 right plots the ROC curve (true positive rate over false positive rate) of identifying unknown types obtained by adjusting $Thres_{conf}$ from 0.1 to 0.8. The results illustrate our SVM model can achieve a true positive rate of 85% with about 10% false positive rate indicating an effective classifier in determining new types.

The micro-benchmark results show that MiLift can accurately track both cardio and weightlifting workouts (C3).

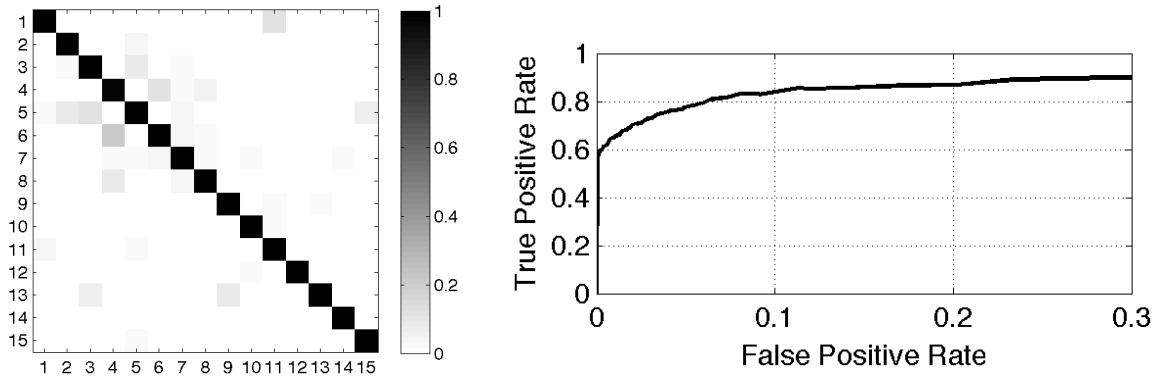


Figure 2.12: Left: confusion matrix of weightlifting type classification. Right: ROC curve of leave-one-type-out experiments.

2.6.2 Energy Efficiency of MiLift Models

Table 2.4 compares the average power consumptions of MiLift at different classification states and the battery life estimations if each state is executed continuously on a Moto 360 smartwatch. We also measured the sleeping power of the watch (S0) for comparison. There are two conclusions we can draw from the results:

First, the two high-level classification models DT+HMM and CRF are energy efficient after the 20% duty-cycle optimization discussed in Section 2.4.4. If only executing these two models, the watch battery can last for more than 16 hours meeting the efficiency challenge (C4). This is achieved even with the duty-cycle executions frequently waking up the watch, indicating that sampling and classification are more energy hungry for the watch than remaining awake. Out of the two models, DT+HMM is about 25.96% more energy efficient than CRF in terms of watch battery life because of DT's lower classification complexity than CRF.

Second, in terms of watch battery life, our revisit-based weightlifting classification algorithm is 41.34% more energy efficient compared with the autocorrelation-based approach, due to less computation incurred by each new sensor sample ($O(1)$ compared with $O(wL)$ as discussed in Section 2.4.3). Note that our micro-benchmarks estimate watch battery lives

Table 2.4: Average power consumption and battery life benchmarks of Moto 360. Showing battery life estimations if executing each state continuously.

State	Description	Power (mW)	Battery Life (h)
S0	Sleep	13.10	97.47
S1-D	High-level activity classification (DT+HMM)	47.08	25.83
S1-C	High-level activity classification (CRF)	58.83	20.67
S2-A	Weightlifting classification (autocorrelation)	159.58	7.62
S2-R	Weightlifting classification (revisit)	112.91	10.77

assuming each state is executed continuously. During real-world executions, weightlifting classification S2-A or S2-R does not need to be executed continuously but will only be triggered by the high-level activity classifier S1-D or S1-C. Section 2.6.3 presents an empirical study showing how our two-stage model improves the battery life of smartwatches.

2.6.3 User Task and Battery Life Analysis

To evaluate MiLift in real-world scenarios, we conducted a macro-benchmark by empirically comparing MiLift with previous approaches using two metrics: (1) number of tasks a user needs to perform for accurate workout tracking, and (2) the battery life of a smartwatch running continuous tracking. We assumed the watch is used only during the day ($16h$), and the user performs weightlifting exercises for 12.5% of the time every day ($2h$). Each approach is described as follows:

Baseline app 1: To continuously track cardio workout activities, app 1 runs an always-on cardio activity classifier during the entire day. However, users have to manually start and stop weightlifting classification before and after each weightlifting session (12.5% of the day). Users need to manually select workout types and manually count sets/reps for weightlifting exercises as well. This app is similar to previous mobile workout tracking apps.

Baseline app 2: To reduce manual input from users, app 2 continuously executes both the cardio activity classifier and the weightlifting classifier all day. Users still have to manually

Table 2.5: Survey questions and answers. For scale questions, 5 stands for strongly agree, and 1 stands for strongly disagree.

Linear Scale Question (1 to 5)	**Score**
Do you find it useful that MiLift can automatically detect ongoing exercises (cardio and weightlifting)?	4.13 ± 0.83
Do you find it useful that MiLift can detect the type of exercises you are performing?	4.47 ± 0.74
Do you find it useful that MiLift can automatically detect and count weightlifting sessions (sets)?	4.67 ± 0.62
Do you find it useful that MiLift can count number of reps (repetitions) in weightlifting exercises?	4.73 ± 0.59
Short Answer Question	**# of Rep**
Max rep counting error you can tolerate.	0.8 ± 0.49

select workout types and manually count sets/reps for weightlifting exercises. This app is similar to existing weightlifting tracking systems.

Baseline app 3: The VimoFit workout tracking app [vim] executes continuously for the entire day. Users need to manually start and stop tracking but the type of exercises and set/rep counts can be automatically determined.

MiLift: MiLift automatically segments exercises from non-workout activities and does not require any manual input from users. It uses a two-stage classification model to duty-cycle the executions for efficient resource usage.

We used power profiles from Section 2.6.2 to estimate watch battery lives for baseline app 1, app 2, and MiLift. For app 3 (VimoFit) we performed a separate experiment to log its battery usage. For app 1, app 2 and MiLift, we used the more efficient DT+HMM approach as the high-level (cardio) activity classifier. App 1 and 2 implemented the more traditional autocorrelation-based algorithm for weightlifting detection, while MiLift considered the revisit-based algorithm.

User tasks: We first present the results of our user survey (Section 2.5) in Table 2.5.

Users find the features of MiLift valuable, including automatic exercise segmentation, exercise type detection, weightlifting set detection, and weightlifting rep counting. This proves that tracking approaches without automatic exercise segmentation can be less attractive to active exercisers. Next, Table 2.6 compares the user tasks required in different approaches, suggesting that all three baseline apps rely on certain user tasks to accurately track exercises. Although baseline app 3 can automatically detect types of weightlifting exercises, the detection accuracy was poor. In our experiments, VimoFit was only able to correctly classify 3 of the 15 exercise types. In contrast, MiLift removes the burden on users and can provide fully autonomous workout tracking for both cardio and weightlifting exercises.

Energy efficiency: MiLift can extend watch battery life by 18.25% (3.31h), 241.56% (15.17h), and 824.57% (19.13h) compared with these three baseline apps, respectively. For baseline app 2 and app 3 the watch battery will run out before the end of the day forcing users to charge the watch. The VimoFit app used for baseline app 3 currently also keeps the watch screen on during executions therefore greatly exacerbating its power consumption. The energy saving of MiLift is achieved by the low-power weightlifting detection algorithm and the context-aware optimization.

The macro-benchmark suggests that MiLift can significantly better preserve battery power of smartwatches than previous approaches (C4) and remove user burdens (C2).

2.7 Discussion

We discuss potential improvements for future work:

Sensing scope and system generalization: Although MiLift can detect a variety of weightlifting exercises including both machine workouts and free weights, exercises which do not involve wrist motions cannot be tracked. This includes both single-arm exercises not performed on the watch-wearing hand and other types of exercises such as leg-based ones.

Table 2.6: User task and watch battery life comparisons of MiLift and baseline approaches.

Approach	**User Tasks**	**Battery Life**
Baseline 1	(1) Manually start/stop weightlifting sessions; (2) Manually select workout types; (3) Count weightlifting sets and reps.	18.14h
Baseline 2	(1) Manually select workout types; (2) Count weightlifting sets and reps.	6.28h
Baseline 3	(1) Manually start/stop weightlifting sessions.	2.32h
MiLift	None	21.45h

We argue that incorporating other non-intrusive sensors such as shoe motion sensors can cover more exercise types. Beyond cardio and weightlifting exercises, algorithms proposed in MiLift can be generalized to detect any exercise or human activity that involves repeating motions, including but not limited to rock climbing, hiking, and racket sports. However, tracking exercises with mild body movements such as yoga may require coordination of multiple sensors.

Energy optimization: We plan to leverage strategies proposed by prior work to further optimize the energy consumption of MiLift, such as offloading sensing and prepossessing from the main processor in mobile SoCs to low-power co-processors [PLL11] [GLR14] [LGQ15], exploiting the coordination of multiple mobile devices and the cloud [MVS15] [SSP15] [CZW15], and exploring the correlation among possible user contexts [Nat12] other than the coarse location used in this paper. These optimization techniques will help save energy budgets of smartwatches for other possible workloads.

Active learning: The weightlifting type classification in MiLift can determine if an instance belongs to an unknown type class. However, currently MiLift does not use such new instances to reinforce its classification model. We plan to implement an active learning system similar to [CSG13] where MiLift would query users for a ground truth label whenever it detects a new type of exercise and then improve the model.

Weight tracking and quality assessment: Prior work has shown that incorrect usages of weightlifting equipment can lead to ineffective training or even injuries. Researchers have proposed several metrics to indicate exercise quality [SBV11] [DSY15a]. Currently MiLift cannot track the amount of weights used in each exercise. However, we observed that users demonstrate different inertial patterns with different weights. This includes changes of peak-to-peak intervals and noise patterns in the accelerometer/gravity data traces. Therefore a similar quality assessment module can be added to MiLift to track weights and to provide exercise feedbacks.

2.8 Summary

In MiLift the combination of a two-stage classification model and a lightweight weightlifting detection algorithm has enabled autonomous and efficient tracking of both cardio and weightlifting workouts. MiLift automatically segments exercises from non-workout activities so that users do not need to manually start/stop tracking or select exercise types. Our evaluations indicate that MiLift can achieve above 90% precision and recall for tracking both cardio workouts and weightlifting exercises. MiLift can also extend the battery life of a Moto 360 watch by up to 8.25× (19.13h) compared with previous approaches. Finally, our user study suggests MiLift's automatic segmentation ability is valuable to individuals actively engaging in gym exercises.

CHAPTER 3

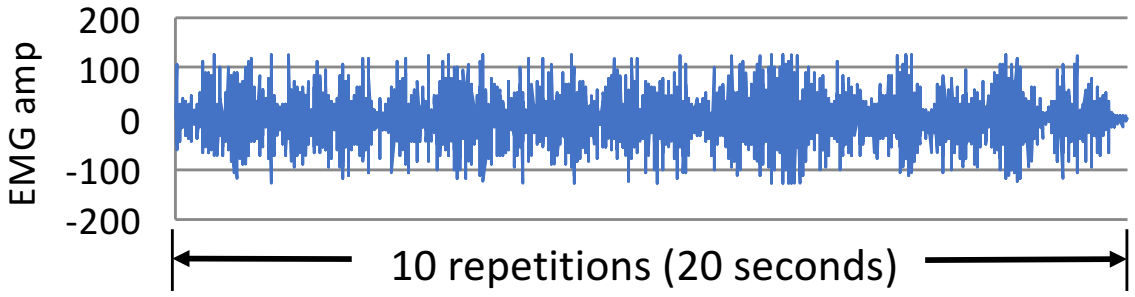
MyoBuddy: Detecting Barbell Weight Using Electromyogram Sensors

3.1 Motivation

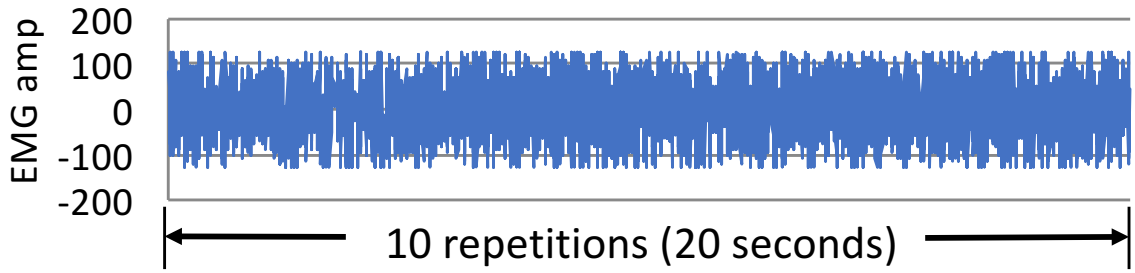
Muscle dystrophy is an umbrella term for genetic diseases whose major symptom is dramatic loss of muscle mass. According to an investigation by the Centers for Disease Control and Prevention, 1 out of $7,250$ males aged 5 to 24 suffer from muscle dystrophy[1]. As of today (April 2017), there is no cure for muscle dystrophy. Steroids are used as major medical treatment for severe patients, however, steroids can disrupt normal hormone production or levels in a patient's body. Daily weightlifting routine is a natural and effective physical therapy to reduce the rate of muscle loss. To help physical therapists monitor the degree of muscle dystrophy, we propose a system called MyoBuddy that monitors and classifies the weights that patients lift during weightlifting exercises.

There are several infrastructure-based solutions for weight detection, for instance, attaching RFID tags on each weight or using computer vision techniques to identify numbers on the weights. However, the deployments of these methods are time consuming and require high installation cost. Moreover, since these techniques rely on external sensing facilities, users have little control how their weightlifting activities are monitored and stored, and hence user privacy becomes a concern. MyoBuddy, on the other hand, seeks for a single-point sensing approach that only requires users to wear a Myo armband [myo]. Myo is composed of 8 electromyography (EMG) sensors, and each EMG sensor measures the change of electrical

[1]The source of the statistics: `https://www.cdc.gov/ncbddd/musculardystrophy/research.html`



(a) 20 lbs barbell curl



(b) 70 lbs barbell curl

Figure 3.1: Two traces of EMG signals measured from 4^{th} channel of Myo armband, each is measured when performing barbell bicep curl with a different weight.

signals from skeletal muscles. These electrical signals allow MyoBuddy to distinguish the degree of muscle activation. Two time series of EMG data are demonstrated in Figure 3.1 when a user lifts two different weights. In both cases, EMG signals show high frequency, but when lifting a heavier weight, the amplitude of EMG sensor signals is larger.

Previous work has shown that detecting the weight lifted is challenging. In Zhou et al's work [ZSC16], only 43% accuracy is achieved where weights are grouped into 4 categories based on participants' body strength. In our work, MyoBuddy aims at achieving a high accuracy in weight detection and targets finer-grained weight recognition. In our experiment, we collected one week of EMG data when performing barbell bicep curl exercise from both authors[2]. By using machine learning models, Support Vector Machine (SVM) and Random Forest, MyoBuddy can train a personal model which achieves up to 80% accuracy with

[2]The data is collected from the authors and thus does not require approval from IRB.

four weight categories and 66% accuracy with six weight categories. We also create user-independent models that achieve 63% accuracy on average.

Our contributions in this work are:

- We develop a system to systematically collect and analyze workout data.
- We collect one week's worth of EMG data from two authors.
- We explore both user-specific and user-independent models and report their performance.

The rest of paper is organized as follows: We first discuss the background of EMG sensors and the Myo armband in Section 2. Then, we talk about related work in Section 3. In Section 4 and 5, we describe the details of our experiments and present an evaluation of our system. We mention the limitations of our work in Section 6. Finally, we provide possible extensions for future work and conclude our work in section 7.

3.2 Background

Electromyography sensors (EMG). Similar to electroencephalogram sensors (EEG), EMG sensors are used to capture the electrical activities of skeletal muscles. EMG sensor is made of electromyograph that can detect the bio-potential of muscle cells. When a person needs to use his muscles, the neurons in the brain release electrical signals to control the movement, causing potential changes in muscle cells. The potential changes can be captured by an EMG sensor.

Myo armband. Myo [myo] armband is one of the cheapest commercial EMG sensors (∼$200), and it features capturing hand gestures through analyzing the EMG and inertial (IMU) signals from a user. Myo armband, as shown in figure 3.2, consists of 8 pods, each pods equips an EMG sensor. The 4^{th} pod additionally contains a Bluetooth Low Energy hardware module and a 9-axis IMU. Each EMG sensor is made by noise filters and one quad

Figure 3.2: Myo Armband is composed of 8 units, each is equipped with an EMG sensor inward of the band. The numbers above the band show the channel identifier.

precision op-amp. The theoretical sampling rate of each EMG sensor is 200 Hz[3].

3.3 Related Work

Weightlifting tracking. FitBit [DKC15] and Nike+ [nik] are commercialized devices that have abilities to track cardio exercise. Several previous researches discussed how the weightlifting activities can be sensed. MyoVibe [MLN15] leverages mechanomyogram (MMG) sensors to sense muscle activation for session segmentation. Repetitions can be counted by processing inertial sensor signals [PHK13, MPA14, vim] or observing the Doppler

[3]The expected sampling rate is clarified in this thread `https://developer.thalmic.com/forums/topic/1945/`

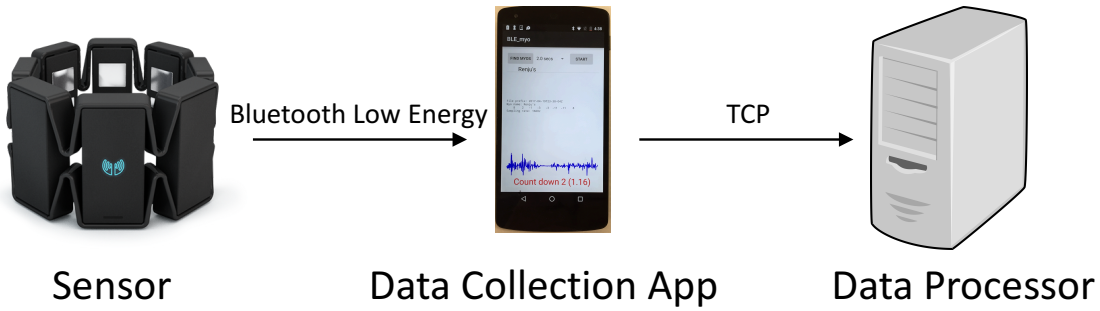


Figure 3.3: Overview of MyoBuddy system.

Effect from RFID sensors [DSY15b]. Prior work also explored exercise type recognition through machine learning algorithms [ZSC16, CSG13]. RecoFit [MSG14] and myHealthAssistant [SBV11] are complete systems which automate the weightlifting tracking process, including repetition counting and type detection. Burnout [MLN16] exploits wearable sensors to quantify muscle fatigue index to prevent injury. Among aforementioned work, only Zhou *et al.* [ZSC16] tackles the weight usage detection. However, the accuracy of distinguishing four groups of weights in their work is only 43%. Our work aims at distinguishing different weights in a finer grained categories with a higher accuracy.

Electromyography sensor. Electromyography (EMG) sensor has been studied for more than two decades [De 97], and has been widely applied in medical domain for identifying neuromuscular diseases such as back pain [San10] or muscle fatigue level [MLN16]. In human-computer interaction domain, EMG sensor has been used for gaming [PK11], capturing hand gesture [PK11, myo], idenfication [YWZ16], and an controlling wheelchair [AG06]. Our work further extend the capability of EMG sensors for weight detection.

3.4 Experiment Design

The system overview of MyoBuddy is shown in figure 3.3. A Myo armband measures EMG signals from users. A mobile phone application serves as a proxy to receive EMG data from

Myo via Bluetooth Low Energy, and forwards the data to the server over TCP. All the computation and feedback are done in the server.

3.4.1 Data Collection Application

We implemented an Android application to receive and log EMG data from Myo armband through Bluetooth Low Energy (BLE). Our experimental sampling rate of EMG data in our application ranges between 130 and $170Hz$, which is close to the theoretical sampling rate of $200Hz$ stated by the manufacturer. The fluctuation in the data sampling rate is caused by crowded Bluetooth communication and stability. In fact, several other BLE devices were found in the experimental area.

The packet format of EMG data from Myo is partially open-sourced by ThalmicLab[4]. Each packet contains two consecutive samples, and each sample includes 8 bytes, and each byte represents a EMG sensor reading whose value ranges from -128 to 127. The values indicate the relative bio-potentials, but the unit is not defined.

3.4.2 Experimental Procedure

Each EMG sensor in Myo senses electrical signals corresponding to the muscle groups the EMG sensor is attached to. To reduce ambiguity in data analysis, we described a fixed way to wear the Myo armband, i.e., the orientation and the position of the Myo armband with respect to the forearm is always the same when the device is worn. Figure 3.4 shows the expected way to wear a Myo armband. A user should wear the Myo device on her left forearm close to the elbow, with the logo (printed on 4^{th} EMG sensor) aligned with the center of the fist.

We chose barbell bicep curl as the exercise in our experiment[5]. Figure 3.5 presents the

[4]Myo armband Bluetooth packet format: `https://github.com/thalmiclabs/myo-bluetooth/blob/master/myohw.h`

[5]A standard barbell exercise is described in `https://www.bodybuilding.com/exercises/main/popup/name/barbell-curl`

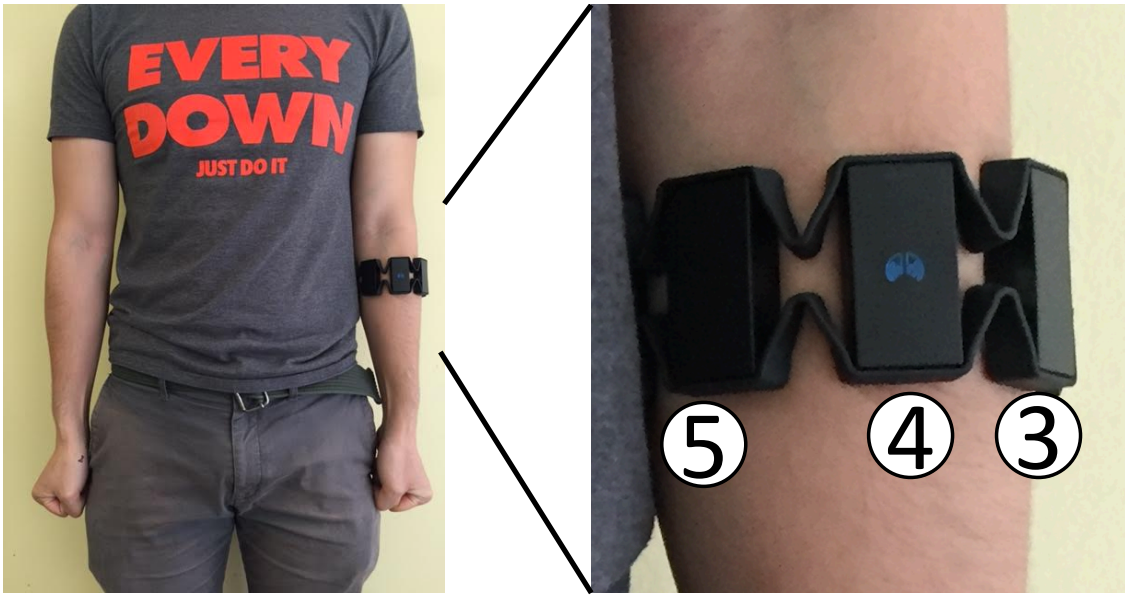


Figure 3.4: The left picture shows that the band should be worn on the middle of a left arm. The right picture shows the direction of the Myo armband. The Myo logo should point outward and align with the left thumb.

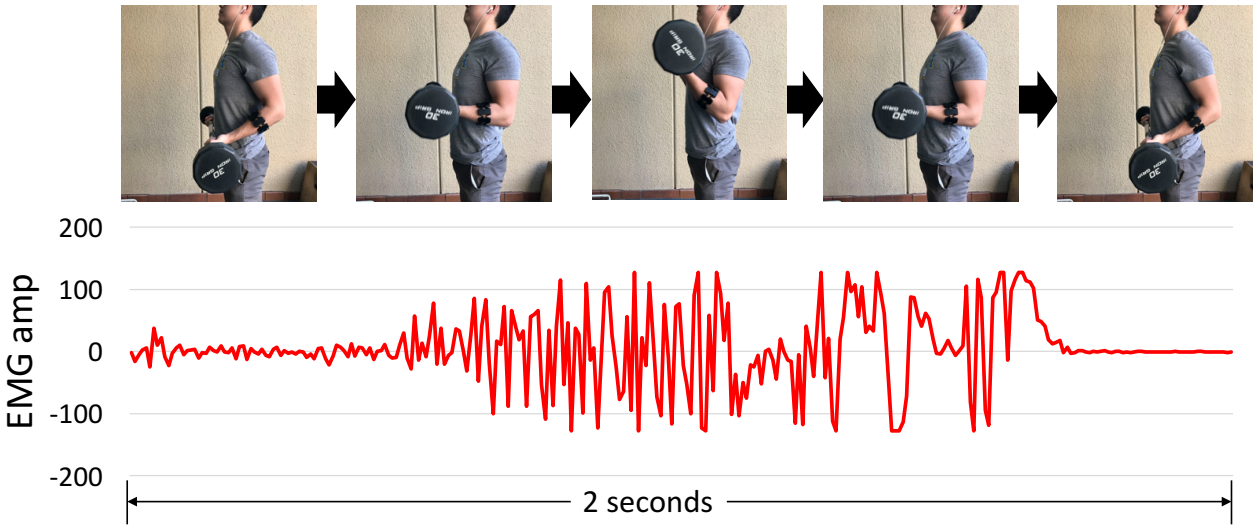


Figure 3.5: The motion of one repetition of the barbell curl exercise and its corresponding EMG data from one EMG sensor.

procedure of a standard barbell curl. Both authors tried their best to complete each barbell curl repetition in 2 seconds: 1 second for lifting the barbell and 1 second for returning to the start position. We define a *repetition* as one cycle of the barbell motion, and a *session* to be several consecutive repetitions. In our experiments, both authors performed a maximum of 10 repetitions per session. To minimize the impact of muscle fatigue in our experimental results, the authors rested for at least 1 minute between sessions.

Because of the variation in each individual's body strength, the maximum weight the first author can lift is 70 lbs, and the second author can lift up to 50 lbs. Both authors performed barbell curls starting from 20 lbs to their respective limits, incrementing by 10 lbs in each experiment.

3.4.3 Weight Classification

The data processing server leverages machine learning algorithms for weight detection. Since EMG signals are time series data, MyoBuddy first segments the time series data and then extracts features in each window. We consider each window to be 2 seconds long because it includes a full-cycle of barbell bicep curl motion. Then, MyoBuddy computes several amplitude-based features in each window including absolute mean, variance, percentiles, and percentage of samples within certain ranges. All features are computed separately for the 8 channels of Myo device. We use Support Vector Machine (SVM) with RBF kernel [CL11] and Random Forest (RF) [Bre01] as our learning classifiers.

3.5 Evaluation

We collected 102 sessions of data over a week. Table 3.1 summarizes the characteristics of our dataset. We conduct several evaluation scenarios based on this dataset to analyze different aspects of our system, which is reported in the following subsections.

Table 3.1: Summary of dataset.

Items	**User 1**	**User 2**
Number of sessions	60	42
Number of repetitions	593	409
Weight range	20-70 lbs	20-50 lbs
Number of different weights	6	4
Total length	40.9 minutes in total	

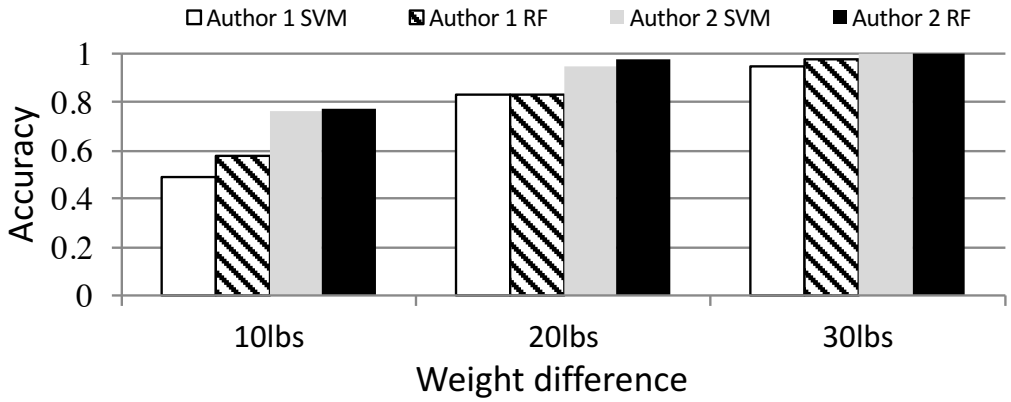


Figure 3.6: Repetition-level weight classification accuracy when applying SVM and RF algorithms

3.5.1 Personal Model

We first train a model for each person separately. To evaluate the sensitivity of our personal models to the weight increments, we selectively choose data points in our dataset based on predefined weight increments. For instance, in the 10-pound increment case, we keep the data points corresponding to 20 lbs, 30 lbs, up to the maximum weight each author can lift. Similarly, in the 20-pound increment case, we keep the data points corresponding to 20 lbs, 40 lbs, and 60 lbs, and discard the rest. For our analysis, we consider 10, 20, and 30-pound increment cases. The resultant datasets corresponding to each case are partitioned into 80% and 20% for training and testing data, respectively.

Figure 3.6 shows the classification accuracy of all repetitions. Generally, the performance of RF is better than SVM. One possible explanation that RF gives a better result is that

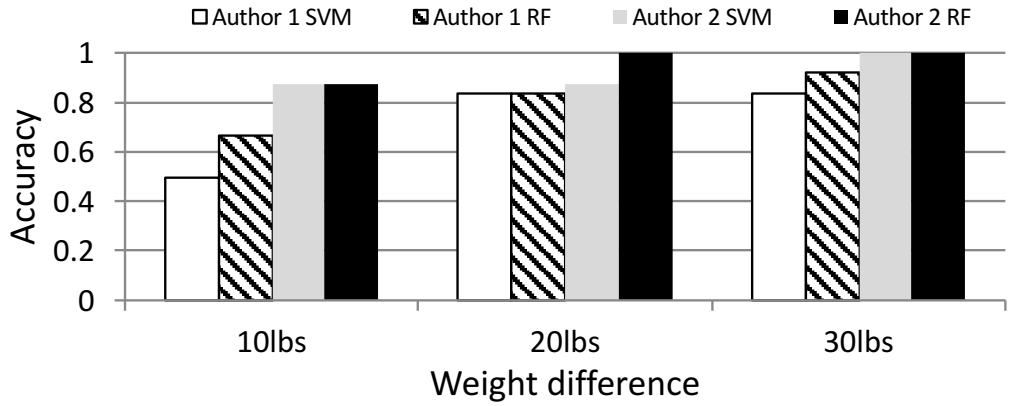


Figure 3.7: Session-level weight classification accuracy when applying SVM and RF algorithms

it employs a better feature selection algorithm and leverages ensemble techniques which are more resilient to over-fitting. Moreover, since the weight range that Author 2 can lift is smaller and hence fewer classification labels, the overall classification accuracy of Author 2 is higher than Author 1.

Our classifier gets a higher accuracy when the weight increment increases. For Author 1, the classifier achieves higher than 80% in both the 20-pound and the 30-pound increment case. The accuracy drops to slightly lower than 60% in the 10-pound increment case. On the other hand, for Author 2, the accuracy of 10-pound increment case is close to 80%, and is higher than 96% in both the 20-pound and the 30-pound increment cases.

We apply majority voting on estimated weights of all repetitions within a session because the weight within a session will not change. This session-level optimization strategy improves the accuracy to 68% for Author 1's 10-pound increment case. Figure 3.7 summaries the session-level weight classification accuracy.

3.5.1.1 Training Data Size

To determine the minimum amount of data to achieve an acceptable accuracy, we increment the training data size and evaluate the classification accuracy. Figure 3.8 summarizes the results. For Author 1, the accuracy does not converge until 9 sessions of all 6 weights are

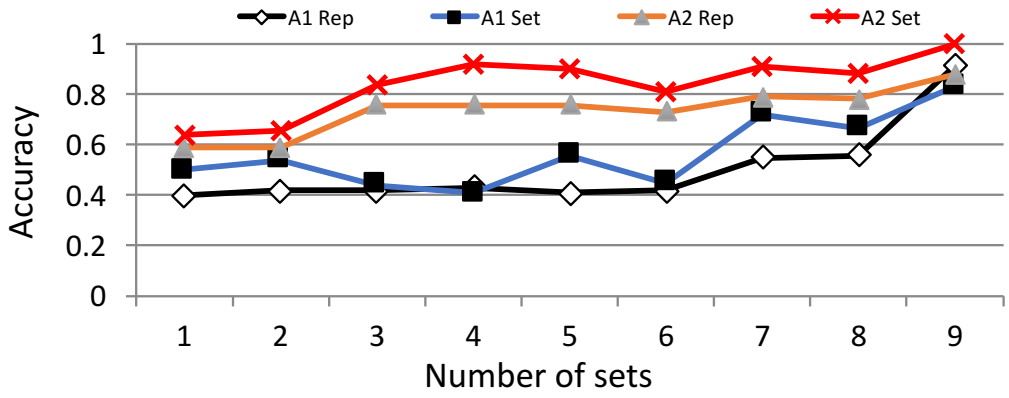


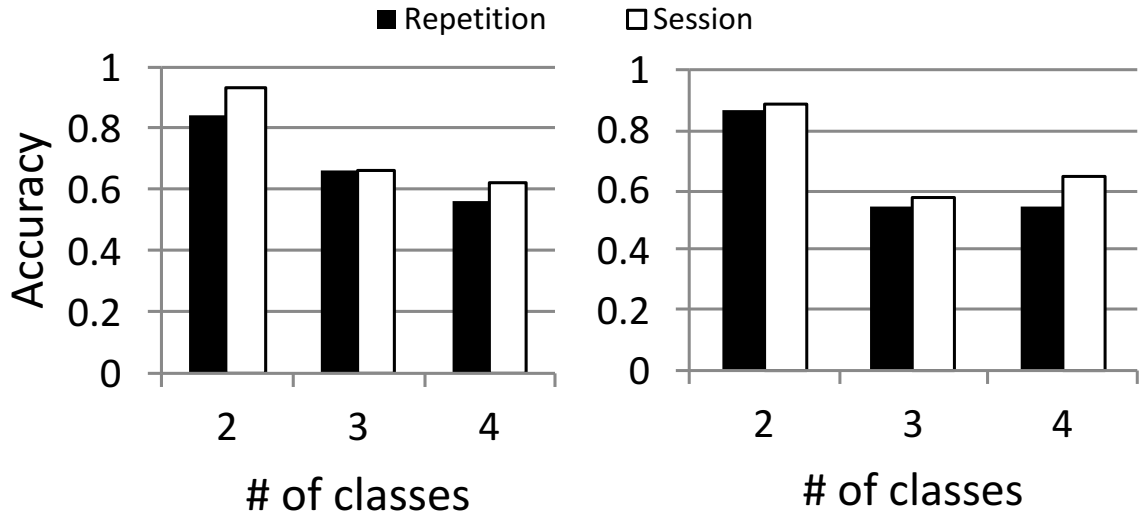
Figure 3.8: The accuracy increases when there are more number of sessions of each weight in the training set. We report both authors' (A1 and A2) repetition (rep) and session (set) accuracies.

included in the training set. The accuracy for Author 2 achieves 82% in just 3 sessions of 4 weights. As a result, our model only needs a few sessions to achieve an acceptable accuracy, and will achieve better performance if more training data are provided.

3.5.2 User-Independent Model

Our application will be much more useful if we can train the classifier from one group of people and apply to other people. We simulate this situation by training on one author's data and applying the model on the other author. The EMG signals can differ a lot for two people lifting the same weight due to the difference in muscle strength. However, EMG signals have similar amplitude when both authors lift a weight close to their strength limits. Based on this observation, the weights are partitioned into 2 to 4 categories. Figure 3.9 demonstrates that when the result is binary (2-category case), both models can achieve an accuracy higher than 85%. However, when there are more weight categories, the accuracy drops to below 65%.

We also utilize Author 1's model without grouping any weight and apply it on Author 2's dataset. The prediction result is interpolated based on their weight limits, i.e., mapping 70 lbs to 50 lbs. The average predicted weight error is 5.2 lbs.



(a) Author 1 → Author 2

(b) Author 2 → Author 1

Figure 3.9: Accuracy of user independent model. X → Y means we take X's data as the training set and apply on Y's data. Both repetition and session accuracy are reported.

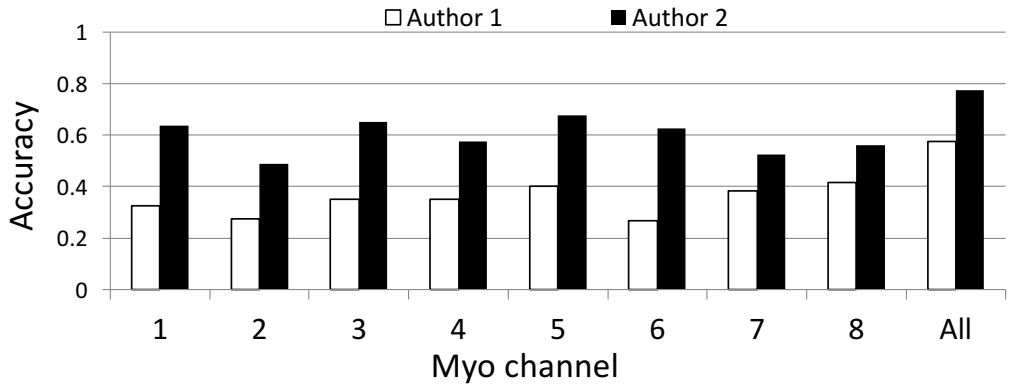


Figure 3.10: The repetition accuracies if we only consider one particular EMG channel from Myo.

3.5.3 Muscle Groups Matter

To evaluate the necessity of the 8 EMG sensors in Myo, we simulate the situation by keeping the data from one EMG channel at a time for classification. Figure 3.10 shows the results that only one EMG sensor data is used. As we can see, the 5^{th} EMG sensor is the most

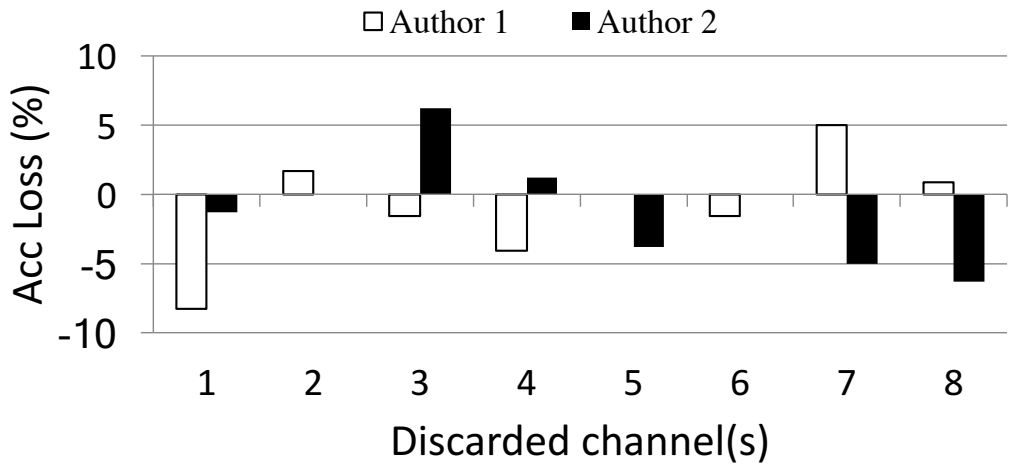


Figure 3.11: The result of the leaving-one-out experiment for each EMG channel. The y-axis presents the accuracy loss compared to the original personal models.

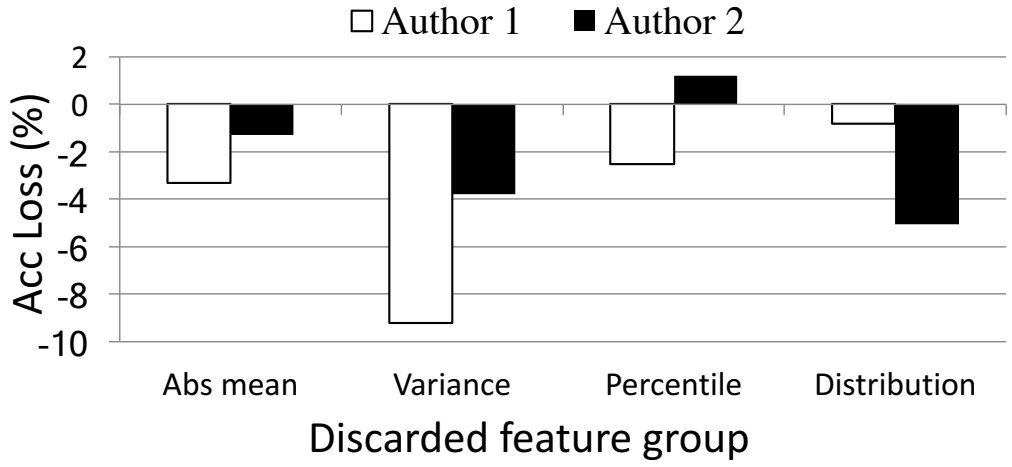


Figure 3.12: The result of the leaving-one-out experiment for features including absolute mean, variance, percentile, and percentage of samples within certain ranges (distribution). The y-axis presents the accuracy loss compared to the original personal models.

significant channel because the greatest amount of force is exerted where the sensor resides. In contrast, the 2^{nd} EMG sensor gives the lowest accuracy because the muscles under it do not play a key role in bicep exercises. However, the accuracy of each EMG sensor does not drop dramatically compared to when all the 8 EMG channels are used, suggesting all the EMG sensors can sense activated muscles.

To examine the importance of each channel, we conducted a leave-one-out experiment for every channel. Figure 3.11 presents the accuracy loss when the data of one EMG channel is absent. The results show that none of the channels significantly affect the classification performance. In addition, the accuracy increases after removing the 1^{st} and 4^{th} channels because they cause noise in the model. After removing these two noisy channels, the accuracy can improve by 2.5% and 1.3% for Author 1 and Author 2, respectively.

3.5.4 Feature Analysis

Similar to the previous experiment, we conducted a leave-one-out experiment for the following feature groups: absolute mean, variance, percentiles, and percentages of samples within certain ranges, as mentioned in Section 3.4.3. Figure 3.12 shows that the accuracy increases the most when the variance feature is removed. The final accuracy increases to 66% and 80% for Author 1 and Author 2, respectively.

3.6 Discussions and Limitations

Orientation of Myo. One limitation of our experiments is that the Myo armband must be worn in a specific orientation on the forearm. However, this orientation can be automatically acquired using the built-in inertial sensors in Myo. Once the device orientation is measured, the angular difference between the measured versus expected orientation can be detected, and EMG data can be calibrated by shifting the EMG channels.

Arm motion pace. EMG measurements capture the electrical signals sent out from muscles, which is an indicator of how much force is being exerted. However, a person may use a different amount of force when lifting the same weight because of the pace of the motion. Generally, one needs to use more force to complete a workout session more quickly. Our experiment avoids this complication by constraining the duration of each bicep curl rep. However, this restriction could be removed by measuring the duration of each repetition

from inertial sensors [PHK13, MPA14].

The sensing scope of workout. In our study, we choose barbell bicep curl for our application evaluation. However, the methodology can be generalized to other workout activities as well, such as all upper body exercises because arms are always needed to coordinate for upper body exercises. In the future, the weight detection model should consider the workout type as a prior [ZSC16, CSG13] because the muscle activation can vary among different exercises.

3.7 Summary and Future Work

We designed a practical system called MyoBuddy using EMG sensors to estimate the weights of the barbell exercise. Our results show that MyoBuddy can distinguish weights with a 10-pound increment with 73.4% repetition-level and 77.1% session-level accuracy.

Although we only collected one week of data from two authors, in the future, we plan to recruit more volunteers and to test our system in different weightlifting exercises such as tricep ropes, bench dips, etc. We further aim to migrate the data processing program to smartphone so that users can obtain real-time feedback and see the performance improvement. Our system has potential to enable different interesting applications, such as a tool for bodybuilders to track their weight limits and adjust their training plan accordingly.

Part II

Engagement Modeling

CHAPTER 4

Nurture: Notifying Users at the Right Time Using Reinforcement Learning

4.1 Motivation

Edge devices depend upon human interaction for numerous functionalities. Crowd-sourcing apps such as Waze[1] and Yelp[2] bank on user input to deliver services; intervention-based apps such as Apple Watch Activity app[3] prompt users to exercise; and apps such as Google Now[4], Medisafe[5] send useful context-based reminders. User engagement with edge devices is leveraged to learn preferences, label activities and configure devices.

The usefulness of edge devices diminishes significantly if users do not want to interact with them [MCD18]. For instance, a medical therapy may not work if the patients do not read the intervention messages. We can imagine another case where the quality and strength of a survey research study can suffer if participants do not consistently respond to the in-situ survey questions. In a more critical example, users may not have the time to react if they ignore warnings and alerts preceding a natural disaster. footnoteNotifications can also lead to negative effects if they distract the user at the wrong time. For example, interacting with a handheld device while driving increases the crash risk 3.6 times [DGL16].

Most edge devices currently employ a simplistic interaction model that assumes the user

[1]Waze - https://www.waze.com/

[2]Yelp - https://www.yelp.com/

[3]Use the Activity app on your Apple Watch - https://support.apple.com/en-us/HT204517

[4]Google Now - https://en.wikipedia.org/wiki/Google_Now

[5]https://medisafe.com/

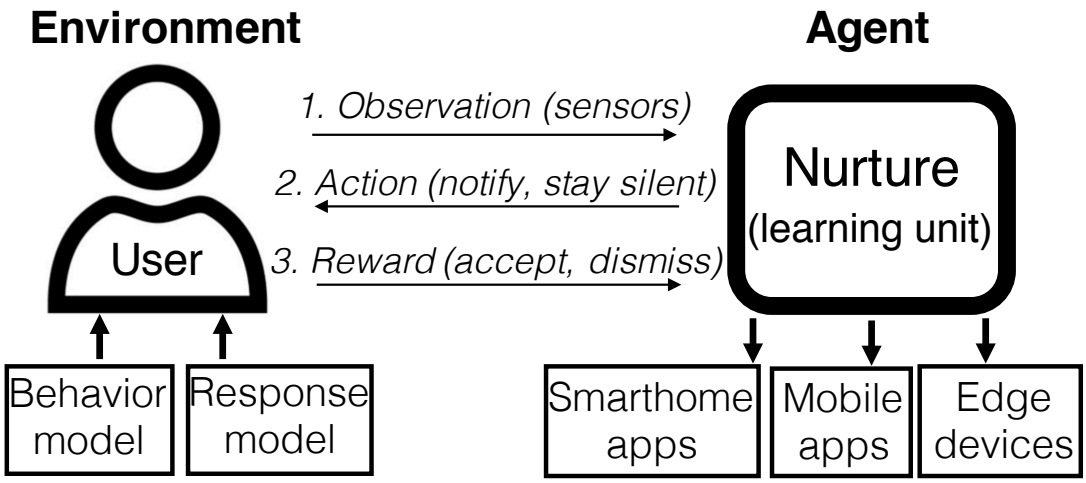


Figure 4.1: The proposed reinforcement learning agent interacts with the user to learn the right time to send notifications. Nurture aims to empower different applications. In this paper, the user is simulated by underlying models.

is always available to engage with the device. In reality, user attention is a limited cognitive resource [LFN15]. Users get over 60 notifications a day [PCD14], so it is natural to skip notifications when feeling overwhelmed and to dismiss all communications when disturbances are undesirable[6]. Nevertheless, there is no systematic way to infer user availability yet, and identifying the ideal time for human interaction remains a challenging problem as user engagement depends on a wide range of variables such as context [ORN15], environment [WVK16], hardware status [OTT17], and content of messages [MMH15].

Here we approach notification timing as a reinforcement learning problem, and propose a notification time selection technique, *Nurture*, which learns user preferences through interacting with users over time without any prior knowledge. Figure 4.1 illustrates the learning flow. Nurture obtains the user state via sensors, decides if it should notify the user, and observes user reaction after sending the notification. By considering accepted notifications as positive signals and dismissed notifications as negative signals, the agent learns the appropriate times to notify the user. Previous studies on intelligent notification systems mostly approached the

[6]https://www.wired.com/story/turn-off-your-push-notifications/

problem as a supervised learning problem. We evaluated Nurture with both a synthetic and an online interactive crowdsourcing-based simulation. Our results show that reinforcement learning is able to capture notification preferences from real users and significantly improves user response rate compared against previous supervised learning methods.

4.2 Related Work

For mobile applications, push notifications are a convenient way to interact with users. However, it has been shown that push notifications can reduce work performance and increase stress especially if they arrive at inappropriate times. Hence, there is a growing interest on how to make notification systems more intelligent [MM17a].

There are two main approaches to identify opportune moments for notifying users. The first category is to show notifications when the user is transitioning between activities. An activity transition usually indicates that the former activity is completed, hence, it is a good time to interrupt users before they start another task. Oasis [IB10] is the first system to exploit breakpoints between computer tasks to interrupt users. Attelia [ORN15] detects physical activity transitions using both smartphones and wristbands, and identifies which transitions indicate user availability. The downside of this approach is that the changes in sensor readings do not always reflect activity transitions, hence, high false positive rate becomes a concern.

The second category is to infer user interruptibility from context. Sarker et al. [SSA14] showed that factors such as location, activity type, stress, time, and day of the week affect user participation in medical interventions. PrefMiner [MHM16] analyzes the content of notifications and learns to filter out those notifications that a user will not pay attention to based on her preferences. Goyal et al. [GF17] show that users are likely to pay attention to the notifications which are shown at times of increasing arousal detected from the electro-dermal activity. Aminikhanghahi et al. [AFS17a] proposed a combined activity recognition and intelligent notification framework based on supervised learning which improves notification response rate.

Prior works have primarily focused on supervised learning and manual labeling to understand the relationship between the user state and notification response. Here we approach the problem as a reinforcement learning/contextual bandit problem where the agent learns when to notify the user merely by interacting with the user and observing the context of the interaction. The strength of our approach comes from the fact that we do not have an explicit training phase and gradually improve notification timing over time. In the reinforcement learning setting, we treat the problem as a sequential decision problem, which implicitly takes the daily routine of the user into account. We do not require users to provide explicit feedback and can adapt to changing user habits.

4.3 Design and Implementation

4.3.1 Problem Setup

In a typical reinforcement learning (RL) setting, an *agent* interacts with an *environment* to achieve certain goals. At each step, the environment is in a certain *state* and the agent takes an *action*. The environment reacts to the action by transitioning into another state and returns a *reward* to the agent. The strategy with which the agent selects its actions is referred to as its *policy*. To this end, the RL agent tries to find an optimum policy by trying different actions which maximizes the cumulative reward over successive iterations.

In our setup, **Nurture** is the RL *agent* which empowers applications on smartphones or other edge devices, and the *environment* is the **user**. At every pre-defined time interval (e.g. 10 minutes), Nurture senses the user *state* based on **sensor readings**. The user context we have considered are time, location, physical activity, and last notification usage, summarized in Table 4.1. Nurture performs one of the following *actions* - **remain silent** or **send a notification** to the user. The *reward* is based on the reaction of the user: A positive reward is received if the user responds to the notification, a negative reward is received if the notification is dismissed, and zero reward is received if the user delays the notification to respond later.

4.3.2 Learning Strategies

4.3.2.1 Contextual Bandit

We first approach our problem as a contextual bandit problem which is a special case of reinforcement learning where the agent does not keep track of sequence of states visited. In a bandit problem, the agent can take a set of actions with unknown rewards. Similar to reinforcement learning, the agent maximizes cumulative reward over successive iterations. The bandit agent explores the rewards associated with the each action (i.e, arm) and aims to learn the best action. Contextual bandit is a type of bandit problem where the agent receives observations as *context*, and the context is used to learn the state in which the environment is in.

In our setup, the two arms of the bandit problem are (i) to send a notification, or (ii) to remain silent. The context is the information about the user sensed by the application. The reward from the first arm is stochastic and depends on the response from the user. The reward from the second arm is always zero as no feedback is received when a notification is not sent.

4.3.2.2 Reinforcement Learning

We also consider our problem as a Markov Decision Process (MDP) which we solve using reinforcement learning. Different from the contextual bandit problem, reinforcement learning approach implicitly learns transitions between states. Similar to the contextual bandit problem, the aim is to maximize the long term reward.

The expected long term cumulative reward for taking an action from a given state under a specific policy is referred to as *Q-value*. We use the standard Q-learning algorithm [WD92] where the agent updates the Q-value as it visits different states using a random exploration policy. Across iterations, the Q-value converges to its optimum value, and we get the final policy by picking the actions with the highest Q-value at each state.

Table 4.1: The state categorical values. Each state is presented by the combination of these five categories, so there are 144 different states in total.

Category	**Values**
Time of the day	*morning, afternoon, evening*
Day of the week	*weekday, weekend*
Location	*home, work, others*
Motion activity	*stationary, walking, running, driving*
Last notification	*within / beyond 1 hour*

4.3.3 Experimental Setup

We conducted a simulation-based experiment using crowdsourced data as a proof-of-concept study. Our experimental procedure is approved by UCLA IRB. The simulation process is depicted in Figure 4.1. In our simulator, we imitate a user in her daily routine. The mobile application senses the user context through her mobile phone and wearables, and Nurture decides whether to send a notification or to remain silent. Once our agent sends a notification, the simulated user can respond to the notification, explicitly dismiss the notification, or perform no action to skip it. Nurture then obtains the reward according to the user's reaction and adjusts the strategy.

A simulated user is driven by a *behavior model* and a *response model*. The behavior model reflects the daily routine of the user, i.e., the location and the activity of the user during a week. We derived the behavior model from the ExtraSensory dataset [VEL17], which includes daily traces of 60 participants for improving context recognition in-the-wild. The dataset consists of 29 locations and 15 activity contextual labels, and the participants marked the applicable labels at a granularity of one minute. We group these labels to fit our simulation settings and pick four user traces which have distinct lifestyles. Important statistics regarding the weekly routines of these users are summarized in Table 4.2. The response model simulates how a user *responds* to a notification at a given context, and the user context is determined by the behavior model.

Each question in the mTurk survey describes a scenario that specifies the time of the

day, day of the week, location and activity of the user, and when the last notification was responded to. We provide a sample survey question below:

> *It is 10:30 AM on Saturday. You're sitting in a shopping mall. You responded to a notification 1 hours 15 minutes ago. Now you receive a notification from our app to complete a 10-second task. What action will you take?*

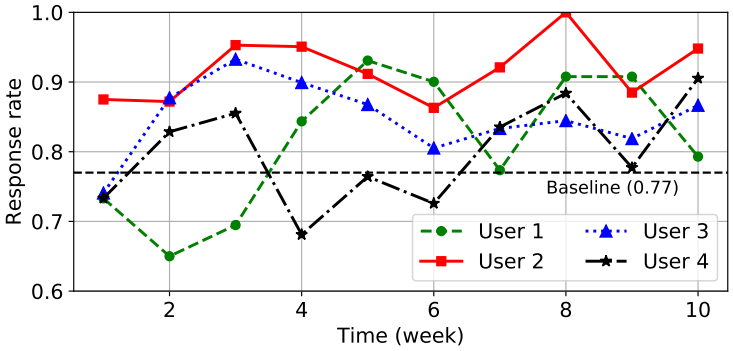
The responders pick an answer from the following:

> *a) Dismiss this notification*
> *b) Leave the notification and answer it later*
> *c) Take ten seconds to respond the notification*

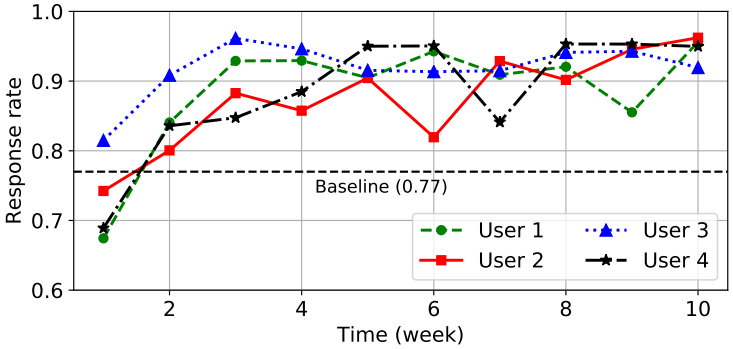
We performed two different types of simulations to imitate human responses: a synthetic simulation and an online interactive simulation. In the synthetic simulation, we deployed a survey on the Amazon Mechanical Turk[7] (MTurk) prior to running Nurture to collect data on how users will respond to notifications given different context. The crowdsourced data is used to *synthesize* the response of a single user. Specifically, the user response is determined by looking up the survey response describing the same situation.

In contrast, in the online interactive simulation case, we do not collect data a priori. Nurture interactively sends notifications in the form of an mTurk survey to model a "pseudo-user". Depending on the responses collected in an iteration, Nurture updates the notification scheduling policy either using the contextual bandit or the Q-learning algorithm. The updated policy is used to generate a new mTurk survey so as to maximize the likelihood of positive response from the workers. We create two pseudo-user profiles where each user has a different daily routine, and we collect each of their responses from a different region, e.g., the U.S. vs India.

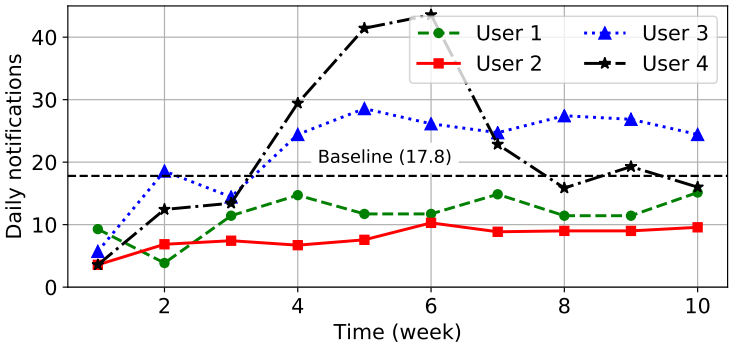
[7]Amazon Mechanical Turk - https://www.mturk.com/



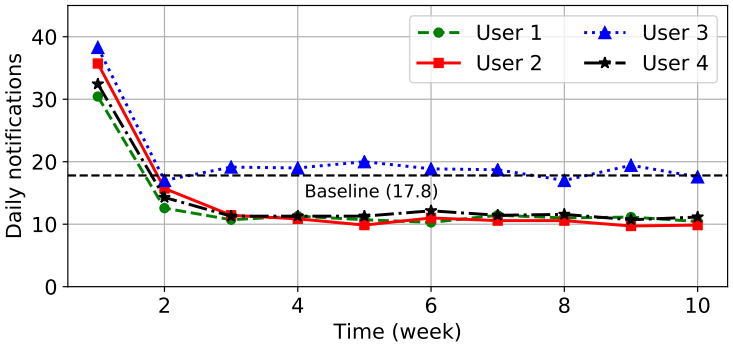
(a) Response rate of contextual bandit



(b) Response rate of Q-learning



(c) Number of daily notifications of contextual bandit



(d) Number of daily notifications of Q-learning

Figure 4.2: The performance of contextual bandit and Q-learning algorithms over weeks, compared with SVM as our baseline.

Table 4.2: Statistics for four users, showing the number of hours each user spends at each location and activity

Attributes/Routings	**User1**	**User2**	**User3**	**User4**
Staying workplace	2.81	2.96	9.74	3.89
Staying outdoor	2.76	7.29	3.80	6.96
Walking+running	1.32	0.54	1.57	1.34
Driving	0.37	1.20	0.43	2.05

4.4 Evaluation

We first performed synthetic simulation in which the human responses are approximated. This gives us the advantage of repeatedly and systematically iterating over our algorithms. We then tested our algorithm on online interactive simulation where the user responses are collected in the wild over a crowdsourced platform. This gives us insight on whether our agents can adapt to user preferences over time.

4.4.1 Synthetic Simulation Results

We collected 3,019 survey responses from MTurk across 123 workers. 44.1% notifications were accepted, 26.6% were dismissed, and the rest were marked as 'answer it later'.

We use the *response rate* and *notification volume* as performance metrics. Response rate is defined as the fraction of accepted notifications over notifications sent without considering those marked as skipped. High response rate implies our agent can accurately identify when to approach users. However, an agent may increase the response rate by avoiding interaction with users. Thus, we use number of notifications to balance this effect. A well-behaved agent should keep a high response rate while maintain a high notification volume.

We consider the following two supervised algorithms as our baseline: Support Vector Machine (SVM) with Radial Basis Function kernel, and 2-layer Neural Network (NN) with 32 neurons in each layer. The models apply one-hot encoding to represent contextual user data (i.e. the user state shown in Table 4.1) as a feature vector, and an anticipated action

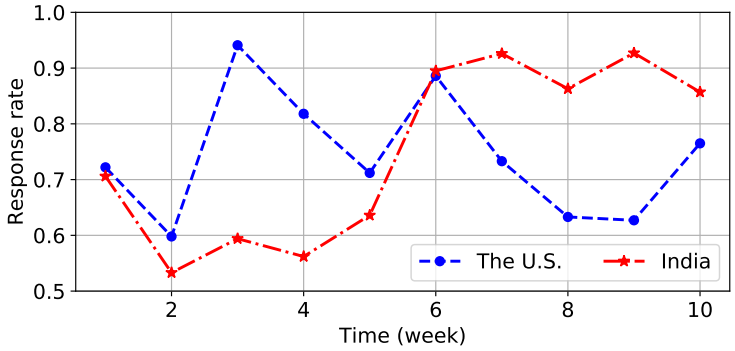
(i.e. user response) as the predicted label. We created training and testing schedules that last 4 weeks and 10 weeks, respectively. During the training phase, each algorithm randomly sends 15 notifications a day, and obtains the responses from MTurk as the training dataset. During the testing phase, both algorithms predict an action every 10 minutes to determine whether to send a notification or not. We apply the baseline algorithms to the four users weekly routines as described in Section 4.3.3. SVM and NN achieve a mean response rate of 77.4% and 77.1% each, and send an average of 17.8 and 6.7 notifications per day, respectively. We choose SVM as our baseline for the rest of the paper.

Figure 4.2 shows the performance of Nurture. We apply contextual bandit and Q-learning algorithms on the four different weekly routings. We found that these two algorithms take opposite strategies: Contextual bandit starts conservatively, whereas Q-learning sends more notifications in the first two weeks. Both algorithms are able to achieve higher response rate over time, and converge at 89.6% and 93.8% response rate respectively. Therefore, both algorithms demonstrate that they can achieve better response rate compared against the baseline in less than 4 weeks; Q-learning only takes two weeks to achieve the response rate over 80%. Q-learning is superior because it considers the state transition, whereas contextual bandit does not consider pursuing future rewards.

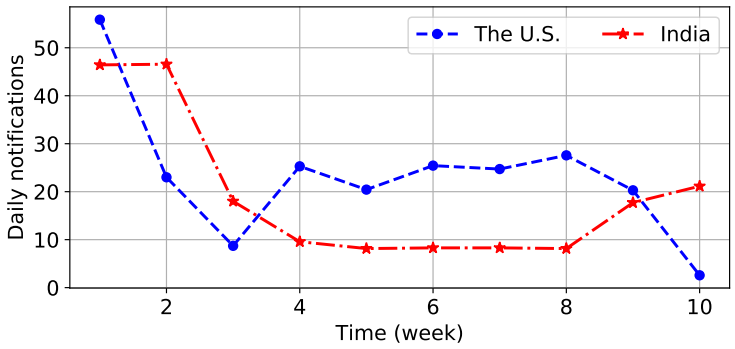
4.4.2 Online Interactive Simulation Results

To demonstrate Nurture can interact with real users and adapt to their preferences, we update our model online based on the crowdsourced responses from MTurk. We choose Q-learning algorithm because it performed better in the offline case. We started two simulations which obtain notifications from the U.S. and India. Each simulation is assigned to a different behavior model. We removed the responses from abusive workers that take unreasonably short amount of time to finish the survey, or those who enter a single option for more than 95% of their total responses. At the end, we recruited 223 workers from the U.S. and 67 workers from India that collectively completed this experiment.

The simulation result is shown in Figure 4.3. In the first two weeks, Q-learning aggres-



(a) Response rate



(b) Number of daily notifications

Figure 4.3: Performance of interacting with MTurk workers using Q-learning.

sively sends notifications to explore user preferences under different contexts. The development of the two simulations deviate there onward. In the simulation of India, Nurture still cannot find the opportune moments to interrupt users, hence, it becomes conservative while continuing to learn, and the response rate improves after week 6. In contrast, in the simulation of the U.S., Nurture converged to a high response rate on week 3, and reaches out to the user more often. The agent then realizes the user starts to show disagreement with the notification schedule, and adjusts the strategy to carefully choose when to approach the user. As we can see, the response rate increases in the end after learning from its mistakes.

4.5 Limitations and Future Work

In our experiments, we have used crowdsourced data to emulate user response to approximate what may happen in a real deployment. Although we have only considered user context that can be inferred from mobile phone sensors, our proposal can make use of other sensing modalities. For example, acoustic sensors such as Alexa can be a rich channel to infer user emotion, or electro-dermal activity from a watch can be used to detect arousal or stress [GF17], which are closely related to user availability. Additionally, instead of treating all notifications uniformly, Nurture can consider the importance of the message by manipulating the reward associated with each notification, e.g., a large reward for a high priority message. Building upon these preliminary experiments, we plan to do a human subject study to evaluate the performance of the proposed algorithms.

4.6 Summary

We have designed a reinforcement learning based algorithm to improve notification response rate, which in turn increases the quality of interactions from edge devices and reduces disturbance. Our algorithm takes several factors into account such as the activity of the user, location, and time of the day to optimize user engagement via push notifications. We have conducted experiments using crowdsourced data to evaluate the performance of our algorithm. The experimental results show that our algorithm improves the notification response rate significantly with respect to a supervised learning benchmark, while balancing the amount of notifications. Therefore, our proposed approach improves the quality of interaction between the user and the applications running on edge devices by providing more timely notifications.

CHAPTER 5

Quick Question: Interrupting Users for Microtasks with Reinforcement Learning

5.1 Motivation

Human computer interaction has evolved over the years from desktop-only machines to wearables that interface at a glance. Modern services in navigation, commute, local business discovery, crowdsourcing, [ASS10], participatory medicine [LPS10] depend upon such on-demand interaction, where the user can access the services wherever they go. Push notifications exploit this interaction to proactively seek user attention. Notifications are used to check mail, remind users, nudge behavior [NHS15] get feedback[1], label datasets[2], etc. However, human attention is a limited resource [LFN15], and serving content irrelevant to the context leads to annoyance, reduces productivity [BK06] and diminishes engagement [MPV16].

User interruptibility has been extensively studied in literature [MM17b]. We categorize prior works into two approaches - rule-based and data-based policies. The rule-based policy relies on human behavior analysis and identifies the moments that people are likely available. Proposed policies include identifying breakpoints between two tasks [IB10] and using events such as unlocking the phone as a heuristic [VWC14]. As the policy is fixed, it does not fit users who have different preferences. Data-based approach leverages machine learning [PM14, MMH15, SSA14, PDP15]. Prior works used supervised learning that learns the non-linear

[1]Yelp review - https://www.yelp.com/

[2]Google Maps: Question About a Place - https://goo.gl/Jf9mTq

relationships between user context and availability using dataset collected from an existing policy. Thus, the data-based policy learns preferences for each user based on observed behavior.

While prior works use supervised learning (SL) methods, we propose using reinforcement learning (RL) to identify user interruptibility. We identify the following advantages of RL:
(i) Sequential decision process: SL assumes data samples are independent from each other, whereas RL models each sample a function of previous samples. Users can get annoyed if they get too many notifications, and RL will capture this effect.
(ii) Exploration: SL methods passively collect data based on an existing policy, while RL algorithms actively explore the problem space to learn policies that are robust.
(iii) Online learning: SL methods need a training dataset to learn whereas RL is designed for online learning.

We focus on identifying interruptibility for microtasks [CTI15], where we ask the user a "quick question" that can be answered in a few seconds. Microtasks have several use cases - crowdsourcing, personalization [OTD14], labeling datasets [GNW14], ecological momentary assessment [PHM17]. We seek to identify appropriate moments of the day to maximize microtask responses. We collect user context using a smartphone and periodically send the information to the cloud. Our web server uses SL and RL to determine whether to send a microtask to user. User interactions over time are used to train the learning models.

We conducted a 5-week, 41-participant user study to compare SL and RL methods. Our results indicate the microtask responses vary dramatically from person to person and both data-based methods capture the individual preferences. We penalized notification dismissals with a negative reward for RL, and it effectively learned to avoid dismissals while ensuring number of responses commensurate to SL. Users indicated they were available to answer quick questions when the RL agent interrupted them 73% of the time compared to 56% for SL. Users expressed improved experience over time with RL and data indicates that RL adapts to changing preferences within a few days.

The following are the contributions of this work:

- This is the first work to exploit reinforcement learning technique to identify user interruptibility.
- We implemented a cloud service that collects user context from a smartphone app and determines interruptibility using both supervised learning and reinforcement learning.
- We conducted a 5-week user study and recruited 41 participants to compare supervised-learning and reinforcement-learning based microtask scheduling policy in the wild.

5.2 Related Work

5.2.1 Microtask

A *microtask* typically refers to a simple task that can be done within seconds [CTI15]. The microtask technique is widely used in crowdsourcing context: This technique aims to lower the mental burden [KCS08] and to improve response quality [CTI15]. Microtasks have also been applied to solve big, complex tasks by partitioning them into multiple independent microtasks [KSK11]. A more sophisticated approach is to automatically decompose a task based on domain ontology [LSN14]. Microtask techniques have been successfully applied to high-complexity tasks such as domain-specific language annotation [GNW14], article writing [KSK11], and software development [LTA14]. In this paper, we address an orthogonal issue how to schedule microtasks to increase user responses.

5.2.2 Interruptibility Modeling

Machine-to-human interruptibility has been extensively studied in Human Computer Interaction (HCI). One major interruption source from mobile and wearable devices is push notifications; prior studies have shown that scheduling notifications at an improper time increases anxiety [PCD14] and reduces productivity [BK06]. We broadly categorize interruptibility modeling techniques into rule-based and data-based. Rule-based techniques rely on prior knowledge to estimate opportune moments of interacting with people. For example,

opportune moments can be identified based on mobile phone usage pattern such as after phones are unlocked [VWC14], after phone calls or text messages are finished [FGB11], or when a user *reviews* an application [BBM14]. Scheduling microtasks at the task boundaries (i.e. *breakpoints*) can reduce mental effort [AB04], and this technique has been applied to the desktop domain [IB10] and mobile platforms [ORN15]. Goyal et al. [GF17] observes that it would be the most effective to schedule a notification when electrodermal activity (EDA) reading increases under a high-stress context. All these techniques rely on a fixed policy and cannot be generalized to all the users or adapted for changes in user preference.

A more sophisticated approach is to derive a classification model based on the user context, which is sensed by mobile or wearable devices. Besides the common mobile phone sensor data such as time, location, and motion activity, Sarker et al. [SSA14] further considers stress level and social engagement and uses SVM to detect when a user is available. InterruptMe [PM14] takes emotion as an additional feature to infer if sending an instant message is appropriate at the moment. Mehrotra et al. [MMH15] leverages the content of notifications to infer how likely the notifications will be responded. Pielot et al. [PDP15] delivers news feeds when a user gets bored by training a random forest classifier. PrefMiner [MHM16] mines the notification usage patterns and users can pick some of those patterns to effectively filter out undesired notifications. Thyme [AFS17b] shares the same goal with us to maximize user responses to microtasks. They use SVM to identify interruptibility. Our work differs from these works by applying reinforcement learning techniques to address the interruptibility problem. As opposed to supervised learning, reinforcement learning is an online learning process and it learns user preference from interacting with users without a separate training phase.

5.2.3 Human-in-the-Loop Reinforcement Learning

Reinforcement learning (RL) has recently achieved several accomplishments in artificial intelligence. For example, RL agents are able to master more than 50 Atari games without prior training [MKS15]. In 2017, DeepMind developed a Go play system called AlphaGo which

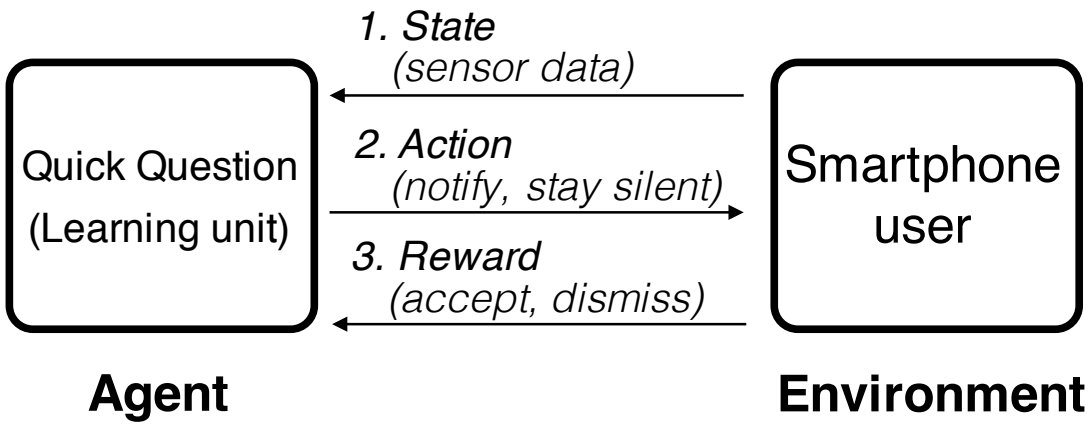


Figure 5.1: Reinforcement learning Setup.

defeated the world champion [SSS17]. These breakthroughs demonstrate the capability of reinforcement learning. Besides playing games, there are several works that apply RL to help humans. Sentio [ETS18] uses a variant of Q-learning to prompt forward collision warnings in cars. Rafferty et al. [RBG16] develops a tutor system based on Partially Observable Markov Decision Process (POMDP). Greenewald et al. [GTM17] exploits contextual bandit to enhance a mobile health system. Silver et al. [SNB13] uses RL to maximize an objective of a company (e.g., revenue) by performing actions to customers (e.g., offering a discount). Our work aligns with these works and uses RL to optimize notification response performance.

5.3 Background

In a typical RL setting, there is an *agent* and an *environment* whose relationship is depicted in Figure 5.1. At each step, the agent first makes an *observation* to obtain a representation of the environment called *state*. The agent then takes an *action* based on its *policy*. As a result of the action, the environment moves to a new state and returns a *reward*. The agent maximizes the discounted sum of future rewards accumulated over successive steps.

The policy π indicates which action a_t should be performed given the current state s_t. Let the discounted sum of future rewards $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ where r_i is the reward received

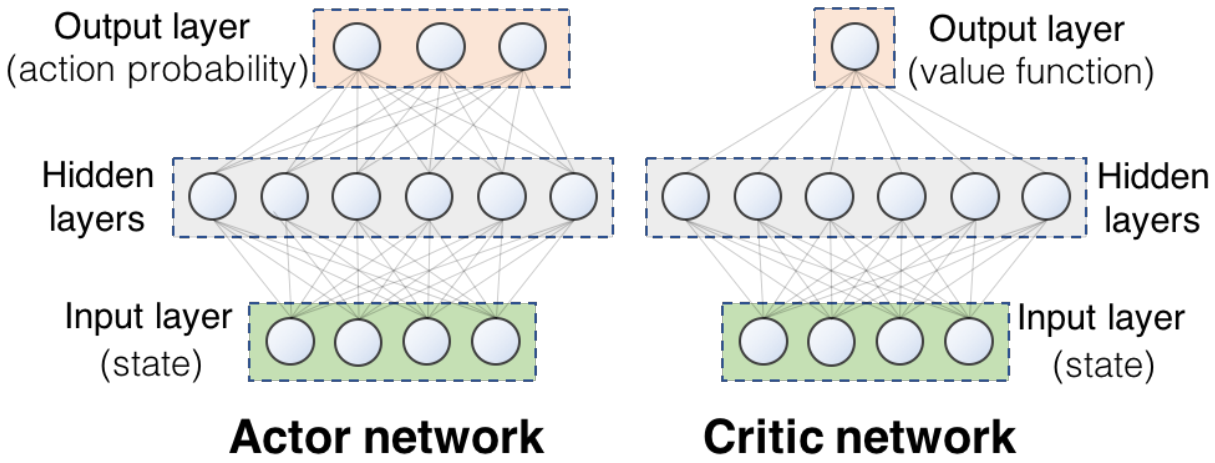


Figure 5.2: Neural network structure of A2C.

at the i^{th} step and γ is the *discount factor* between 0 to 1. The state-value function $V^{\pi}(s)$ is the expected value of discounted future rewards from a given state s following a policy π:

$$V^{\pi}(s) = \mathbb{E}[R_t | s_t = s]. \tag{5.1}$$

Similarly, the action-value function is defined as the expected value of taking an action a at a state s following a policy π:

$$Q^{\pi}(s, a) = \mathbb{E}[R_t | s_t = s, a_t = a]. \tag{5.2}$$

Then, the advantage function which is defined as

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s) \tag{5.3}$$

indicates how advantageous it is to take an action at a given state compared to other actions.

Quick Question uses the Advantage Actor-Critic (A2C) [MBM16] as the RL algorithm. A2C uses two neural networks - an actor network and a critic network. The actor network generates actions by approximating the policy $\pi(a|s, \theta)$ with parameters θ, while the critic network learns the value function to assess the benefit of an action. The network structure

is depicted in Figure 5.2. In policy gradients methods, θ is updated in the direction of $\Delta_\theta \log \pi(a_t|s_t;\theta)R_t$ where R_t is the accumulated reward after a policy run. To reduce the variance of updates, an unbiased baseline is subtracted from the accumulated reward as $\Delta_\theta \log \pi(a_t|s_t;\theta)(R_t - b(t))$. In A2C, the baseline is the state-value function: $b(t) = V^\pi(s)$. Hence, the estimate of the value function as given by the critic network is used in computing the gradient.

In our framework, the *agent* is our system **Quick Question**, and the *environment* is the smartphone user. The agent observes the **user context** as a representation of the user *state*, and takes an *action*: either **to send a notification** or **to keep silent**. The agent gets a positive reward when the notification is answered, and a negative reward when the notification is dismissed.

5.4 System Design and Implementation

Figure 5.3 shows our server-client architecture with: a phone client app and a web server. Our phone app senses user context data and sends it to the server every minute. Our server determines if the user is interruptible at the moment based on the current user context along with the past user daily routing and response history, and returns the binary decision back to the client app, indicating *to prompt a microtask* (i.e., a short question) or *to keep silent*. Once the app displays the microtask, the app tracks how the user responds to the question and sends the response by piggybacking on the next request.

5.4.1 Client App

Our app is composed of three components: A *sensing module* to monitor user context, a *microtask pool* to store a list of short questions, and a *user interface module*.

Sensing Module It collects time of the day, day of the week, location, motion status, screen status, ringer mode and the elapsed time since last prompt (see Table 5.1). To minimize battery impact, we use Android's `AlarmManager` to schedule sensing tasks, use

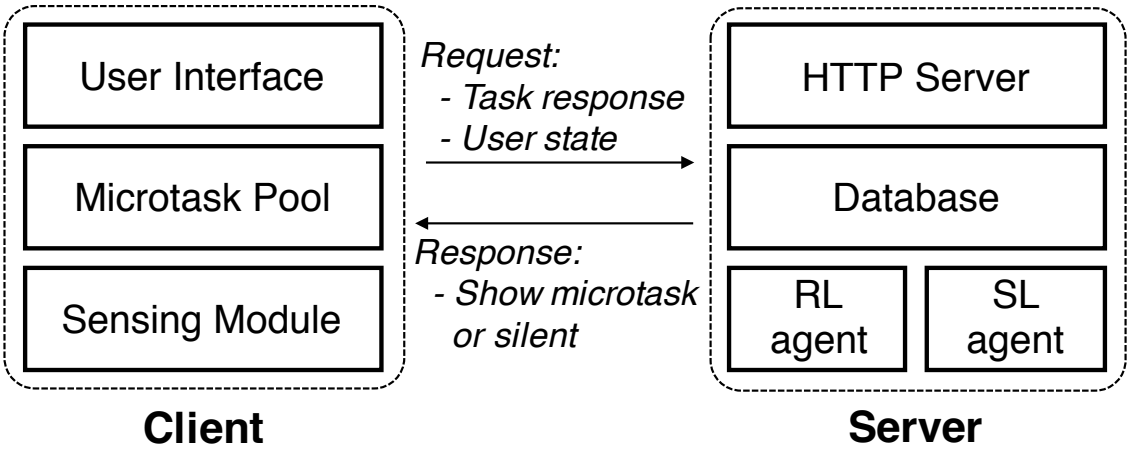


Figure 5.3: Quick Question system overview.

Table 5.1: Features considered in Quick Question as user context.

Sensing modality	Category	Values
Time of the day	Continuous	00:00 to 23:59
Day of the week	Continuous	Sunday (0) to Saturday (6)
Location	Discrete	Home, Work, Others
Motion	Discrete	Stationary, Walking, Running, Biking, Driving
Ringtone mode	Discrete	Silent, Vibration, Normal
Screen mode	Discrete	On, Off
Notification elapsed time*	Continuous	0 to 120 (minutes)

*Defined as how many minutes has elapsed since last notification.

`ActivityRecognition` API for motion activities and the `Geofencing` API for location.

Microtask Pool We use ecological momentary assessment (EMA) questions, commonly used in human behavioral studies [SS94]. All the questions are designed in the micro-EMA style [PHM17] in multiple choice form and can be answered within a few seconds.

The questions can be partitioned into: (i) Self-monitoring questions that track user's mental and physical status such as stress and diet; (ii) Participatory sensing questions collect the environmental information, e.g., noise level at the current location; (iii) The crowdsourcing questions, e.g., image ground truth labeling. We have nine question types listed in Table 5.2. For some questions, correctness can be verified.

Table 5.2: Quick Question client app includes 9 types of microtasks. The examples of each microtask are provided below.

Category	**Microtask**	**Example question statement *(options)***	**Number of questions**	**With gold standard**
Self-monitoring	Availability	Are you available at the moment? *(Yes, No)*	1	No
	Emotion	Which describe your current emotion? *(Stressed, Neutral, Relaxed)*	4	No
	Hydro Diary	How long ago did you drink water? *(Within 1 hour, Within 2 hours, longer)*	1	No
	Diet Tracker [a]	Do you regularly eat wholegrain cereals, with no added sugar? *(Yes, No)*	30	No
	Planning	Where are you going after you leave here? *(Home, Work, Others)*	3	(Yes)[c]
Participatory sensing	Noise level	How loud is it at your location? *(Loud, Moderate, Quiet)*	1	No
	Crowdedness	How many people are there around you? *(0-5, 6-20, >20)*	1	No
Crowd-sourcing	Image labeling [b]	What is the object in the image? *(Bear, Bird, Butterfly)*	1,100	Yes
	Arithmetic	What is answer of 23×33? *(259, 759, 1259)*	1,000	Yes

[a] Source: How healthy is your diet questionnaire. https://tinyurl.com/y9hssxz7

[b] Source: ImageNet dataset [DDS09].

[c] This question is not considered for analyzing user response accuracy due to potential sensor error.

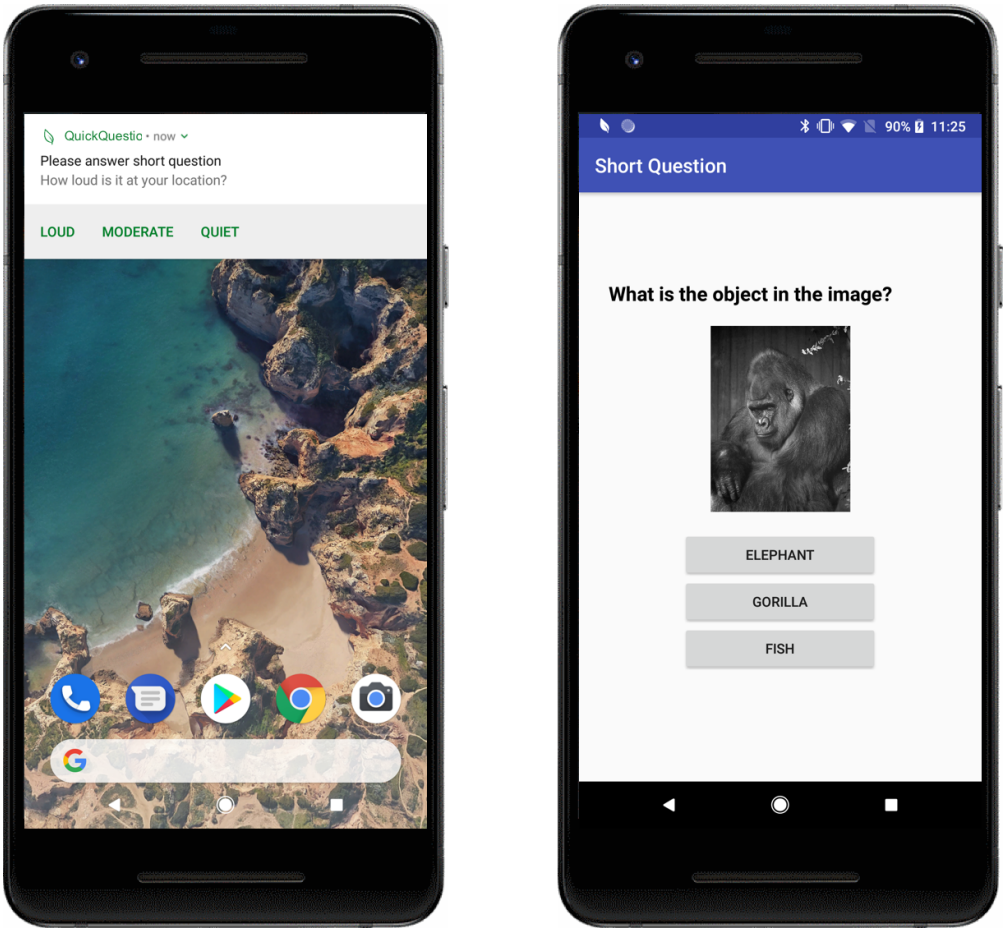


Figure 5.4: The microtask answering interface in the client app, through a notification (left) or in the app (right).

Notification Interface We embed the questions into the push notification (Figure 5.4). The notification is displayed *heads-up* style with the possible options right below (with Android 7.0 or later version). Alternatively, users can also answer the questions by manually launching our app and selecting a choice in the task view. A microtask is timed out after an hour, or when a new microtask is scheduled.

5.4.2 Server

Our server is composed of a standard *web server*, a *database system*, and several *learning agents*. When the web server receives a request (i.e., the user reaction of the previous action

and the current user context), it restores the learning agent by retrieving the *learning policy* from the database. The agent updates the policy by considering the user reaction if necessary. The agent then makes an inference of user interruptibility and decides if it is appropriate to prompt a task at the moment based on the revised policy. The policy is dumped back to the database.

HTTPS Server It serves as the frontend for our mobile client app to query user interruptibility through the RESTful API. We implement a dashboard to identify if the data is not collected as anticipated due to a connection loss from the user side. We use Django[3] to develop our web application, which follows the standard model-view-controller (MVC) design pattern.

Database The database stores logs including the interruptibility request records, task response time and results, and the sensor data. These logs are for data analysis and are not part of the interruptibility inference. The *policy* of each learning agent is also kept in the database.

Supervised Learning (SL) Agent The SL agent converts the user context into a feature vector and the user response as a classification label. To convert a user context into a vector, the agent normalizes sensing modalities which output a continuous value (e.g., time of the day) into a number between 0 and 1, and uses one-hot encoding to represent sensors with discrete values (see Table 5.1). As a result, the user context can be represented by a 15-dimensional vector. To produce a classification label, the supervised learning agent creates a positive label if the notification is answered, and a negative label otherwise.

There are two stages in the supervised learning: A training phase for the data collection and a testing phase to finalize the notification scheduler. In the training phase, the agent first receives a *sample frequency* as a hyper parameter. For example, if the sample frequency is configured as 30 minutes per notification, the agent will schedule a notification every 30

[3]Django Web Framework: https://www.djangoproject.com/

minutes but the actual prompt time will be randomized. The training phase lasts for three weeks[4]. The agent trains a classifier before moving into the testing phase. We use Random Forest as our supervised learning algorithm because it outperforms Support Vector Machine and Neural Network in our empirical study. Our implementation is based on Scikit-Learn library [PVG11b].

Reinforcement Learning (RL) Agent Similar to SL, the RL agent uses the same feature representation to encode user context. Different from SL, RL maps the user response to different reward values: A positive reward that decays exponentially based on response time to encourage scheduling a microtask that can get an immediate response; a strong negative reward if a user dismisses the notification to avoid negative user experience; a small penalty if the user ignores the notification (i.e., does not answer it within one hour because the user overlooks it or forgets to reply). To achieve this design principle, we define the reward function as:

$$reward = \begin{cases} 1 \times t^{0.9}, & \text{if answered,} \\ -0.1, & \text{if ignored,} \\ -5, & \text{if dismissed} \end{cases}$$

where t is the notification response time in minutes (i.e., the time difference between the prompt and when the answer is received). Although the hand-picked reward values work well in our study, the reward function can be fine-tuned by Inverse Reinforcement Learning [BWJ17].

Our RL agent implementation is built upon Coach [CLN17], a reinforcement learning library. Integrating an RL algorithm into a web framework is not a trivial task due to the sequential nature of RL, but connections in web applications are stateless. To this end, we enforce a breakpoint after an action is determined.

[4]Prior work on training personal models using SL demonstrates that the classification accuracy converges in two weeks, e.g., [PM14, MMH15]

Our RL algorithm is selected based on an empirical study. The procedure is similar to [nur], but we additionally tested other neural-network based RL algorithms including Deep Q-learning Network (DQN) [MKS15], Advantage Actor-Critic (A2C) [MBM16], and Proximal Policy Optimization (PPO) [SWD17]. A2C achieves the best performance among these algorithms and converges in the shortest time. Hence, we choose A2C for the real user study. We employ a fully-connected neural network with one hidden layer (256 units). We set the discount factor $\gamma = 0.99$. The algorithm uses categorical exploration strategy which performs a stochastic action based on the probability distribution of both actions.

5.5 User Study

We conducted a user study to evaluate Quick Question. We were guided by the following inquiries:

- What was the relative *notification response amount, rate, and accuracy* (for notifications with correct answers) collected from the reinforcement learning (RL) method and how is it compared to the supervised learning (SL) method?
- How did the *user experience* resulting from the RL method compare to the SL method?

To make a meaningful comparison, this comparison study emphasizes the aspect of machine learning algorithms, i.e., the SL method vs. the RL method, while maintaining consistency in other aspects, such as the same user study procedure, the same qualification criteria for selecting participants, and the same analysis method on the collected notifications.

5.5.1 Participants

This study was approved by University IRB. In total, we recruited 41 participants (24 females, 17 males) from a major research university. Among these participants, 38 were students and 3 were staff. Ages ranged from 17 to 29 (mean=21.1). The inclusion criteria of our study are active Android users with OS version 7 or higher. Participant phone models included Samsung (N=19), OnePlus (N=7), Google Pixel/Nexus (N=7), Sony (N=2), HTC

(N=2), LG (N=2), Motorola (N=1), and Xiaomi (N=1). Additionally, Android OS 7, 8, 9 accounted for 12, 25, and 4 participants respectively. The participants received gratuity of $50 for each completed week, and an additional $50 if they complete the entire study.

5.5.2 Procedure

15 participants took part in the reinforcement learning method, and 26 in the supervised learning method. One user dropped out after 3 weeks since her phone constantly killed our app and prevented the app from working in the background. The procedure consists of two phases: (1) a screening phase to select qualified participants, and (2) an experimental phase. Participants were recruited via university mailing lists and snowball sampling.

In the screening phase, interested candidates completed a questionnaire regarding the phone model they were using, its OS version, and whether they would have network reception during the entire study even if WiFi is not available. After they passed the screening phase, candidates were asked to fill out a pre-study questionnaire with their personal information. Finally, qualified participants were asked to attend an orientation on how to use the Quick Question app. During the orientation, we emphasized that (1) our study app will send no more than 150 notifications[5] in each day between *10am* to *10pm*, and each question can be answered within a few seconds, and (2) participants were asked to not change the way they respond to notifications, hence, answering all questions is not necessary. We then helped the participants to install our app and complete the location configuration (i.e., user's home and work location) for the classifier.

The experimental phase ran for 5 weeks, during which participants went about their everyday activities with their app-installed phone. A weekly survey was sent out at the end of each week to gauge user perception towards the notification schedule on a 1-5 point Likert scale. At the end of 5 weeks, a post-study survey consisting of open-ended questions was conducted to gather participant feedback on the overall user experience.

[5]To test the limit of how many notifications one can handle, we choose a number twice larger than the number of daily notifications (53.8) [PVP18].

5.6 Evaluation

In total, we collected 24,029 hours of data with 1,441,772 data points. Among these data, our system sent out 70,933 notifications. 22,569 (31.8%) notifications were answered, 2,541 (3.6%) were dismissed, and 45,823 (64.6%) were ignored. We compare the performance of reinforcement learning (RL) and supervised learning (SL) from different dimensions. In SL, we report the results in the training phase (SL-train) and the testing phase (SL-test) separately.

5.6.1 Task Response

Table 5.3 compares the task response performance of both algorithms. We list five metrics in the table: the number of notifications answered in a week, the number of notifications dismissed in a week, the ratio of answered notifications, the ratio of dismissed notifications, and the weekly accumulative rewards.

RL makes more microtasks answered, but SL achieves better answer rate. On average, RL is able to get 155.4 microtasks answered per week, which is slightly higher than SL with 143.7 microtasks per week. To break down, RL outperforms SL-train (130.5/week) but slightly lower than SL-test (163.4/week). SL achieves a higher answer rate (41% in SL-train and 33% in SL-test) than RL (27%). The reward function of RL incentivizes answering microtasks and discourages dismissals with a heavy penalty. However, the penalty for an ignored microtask is low. In addition, we only show one micro-task at a time, the prior micro-task notifications are removed before sending a new one. Given this design, RL agent learned to send lot of microtasks based on the observed that users ignored most notifications but dismissed very few of them. Hence, RL agent gets a lot of questions answered, but its response rate is low.

RL can effectively suppress dismissed notifications. The above hypothesis is confirmed by the fact that RL keeps the task dismiss rate low (3%) and participants only dismissed 15.3 notifications per week. In contrast, both SL-train and SL-test exhibit higher numbers of dismissed notifications (18.8/week) and dismiss rate (6%). Note that RL agent's

Table 5.3: The comparison of the response performance of short questions between a2c and supervised learning algorithms.

RL algorithm (15 participants)						*SL algorithm (26 participants)*					
	NA	**ND**	**AR**	**DR**	**REWD**		**NA**	**ND**	**AR**	**DR**	**REWD**
Week 1	224.0	13.0	0.41	0.03	145.7	**Week 1**	128.4	11.5	0.47	0.06	38.5
Week 2	181.4	27.1	0.28	0.04	-26.1	**Week 2**	134.4	17.9	0.38	0.07	3.5
Week 3	140.3	20.3	0.23	0.06	-34.4	**Week 3**	128.7	20.5	0.37	0.05	-48.8
						Avg (SL-train)	130.5	16.6	0.41	0.06	-2.3
Week 4	121.0	8.4	0.22	0.02	69.1	**Week 4**	159.8	19.4	0.35	0.04	33.8
Week 5	110.3	7.9	0.20	0.02	32.6	**Week 5**	167.0	24.6	0.31	0.07	12.6
Avg (RL)	155.4	15.3	0.27	0.03	37.4	**Avg (SL-test)**	163.4	22.0	0.33	0.06	23.2

*NA = number of accepted notifications, ND = number of dismissed notifications, AR = answer rate, DR = dismiss rate, REWD = total reward.

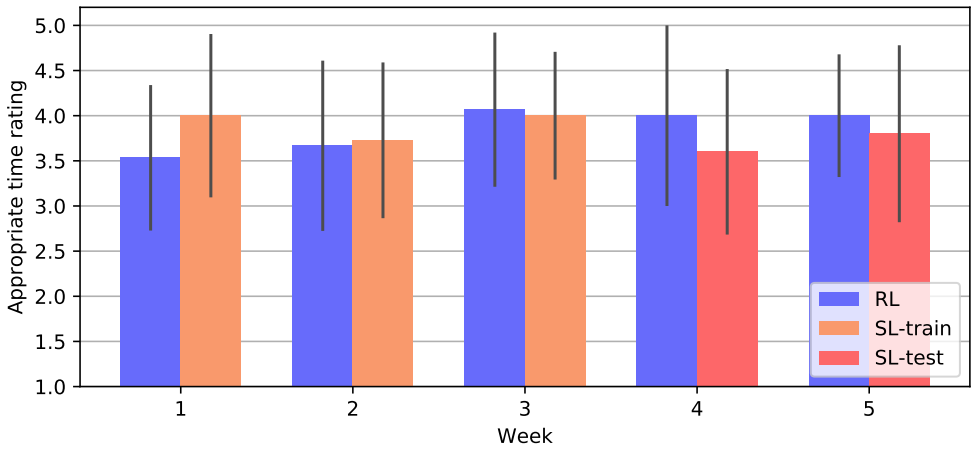


Figure 5.5: Weekly rating of both algorithms.

dismiss rate is low despite the fact that it sends larger number of notifications. RL has been incentivized to avoid dismisses has a sequential decision making process, whereas the loss function in vanilla SL algorithms pick actions based on probability distribution of past data and do not learn the impact of their actions on the user state. SL sends 50% more notifications in testing compared to the training phase.

RL improves by maximizing the reward. The improvement of RL is reflected in weekly reward. Users are highly engaged at the beginning of the study resulting in higher rewards. They lose interest in the 2^{nd} and 3^{rd} weeks hence we observe lower rewards. However, the RL agent learns from user interaction and can distinguish when is a better time to reach out to people, hence returns positive rewards in the last two weeks.

5.6.2 User Experience of Interruptibility

User experience in RL trends towards improvement, but is inconsistent in SL. Figure 5.5 shows the weekly survey result in which we ask participants to rate the appropriateness of the timing of the prompted tasks with a 5 point Likert scale. The result shows that SL starts with a high rating (3.9 ± 0.8 in SL-train), and the rating remains relatively flat in the testing phase (3.7 ± 1.0 in SL-test). RL starts with a low rating (3.6 ± 0.9 in the

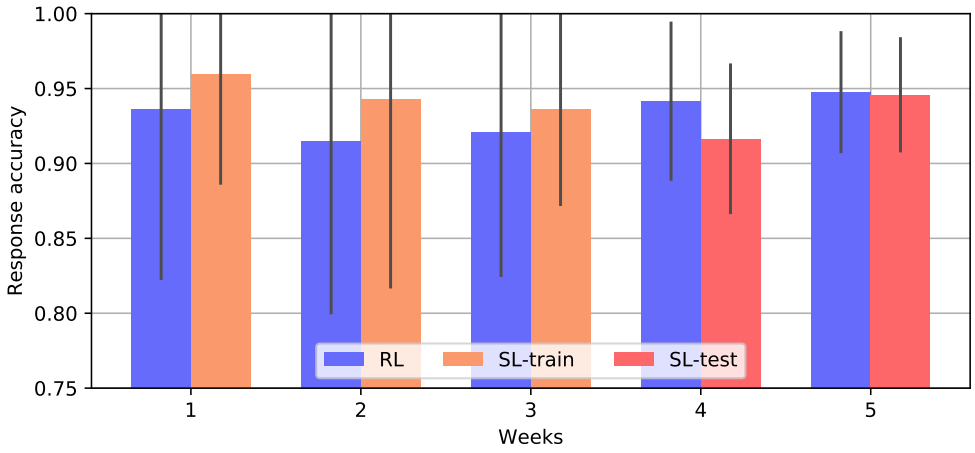


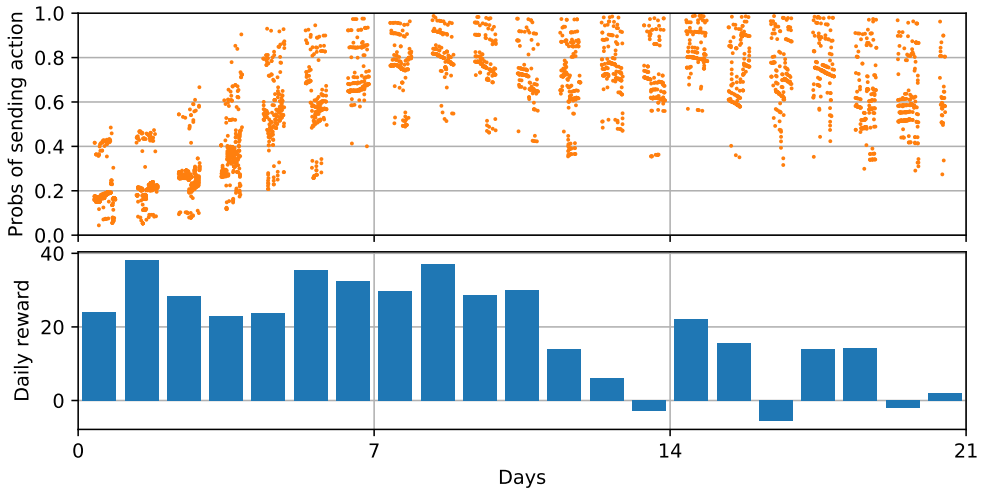
Figure 5.6: The task response accuracy over weeks.

first two weeks). This is likely due to the fact that RL sends more notifications to explore the problem space, and this causes disturbance. However, the rating in RL improves over weeks (4.0 ± 0.7 in the last week) and outperforms SL starting week 3. One-way ANOVA between the first week and the last week of RL gives *p=0.12*, where *p=1* means they are equivalent.

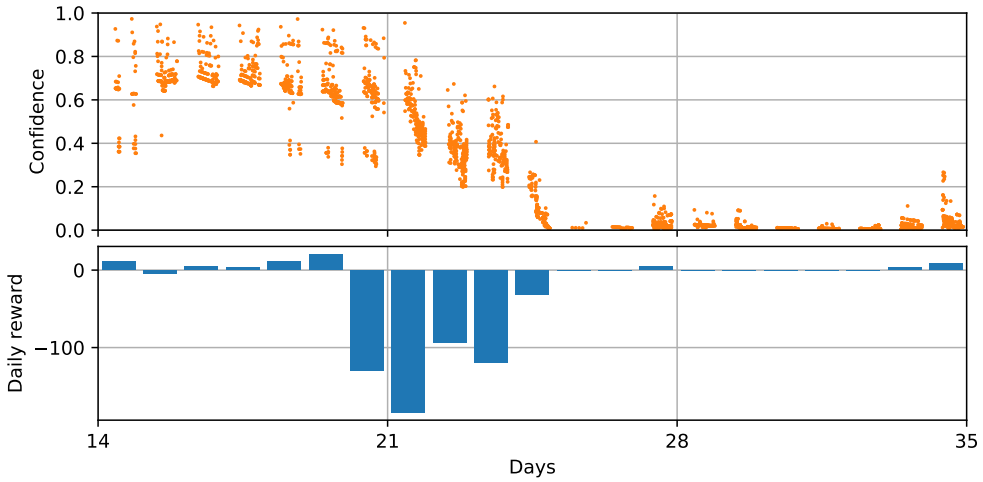
5.6.3 Microtask Response Analysis

RL trends toward higher response accuracy Figure 5.6 shows the response correctness of microtasks. We define *response accuracy* as number of correctly answered microtasks over number of tasks with correct answers (marked in Table 5.2). Both algorithms achieve over 90% of response accuracy of all the 5 weeks. SL starts with a high response accuracy in the training phase $94.6\% \pm 8.8\%$. When moving to the testing phase, the response accuracy remains about the same at $93.1\% \pm 4.4\%$. On the other hand, RL starts with a lower rating ($92.6\% \pm 11.4\%$ in the first two weeks), but we can see that the variance steadily decreases over weeks ($94.6\% \pm 3.8\%$ at week 5).

RL can better identify available moments. Prior work [PCK17] has identified that users respond to microtasks even when their perception is that they are not available. Our



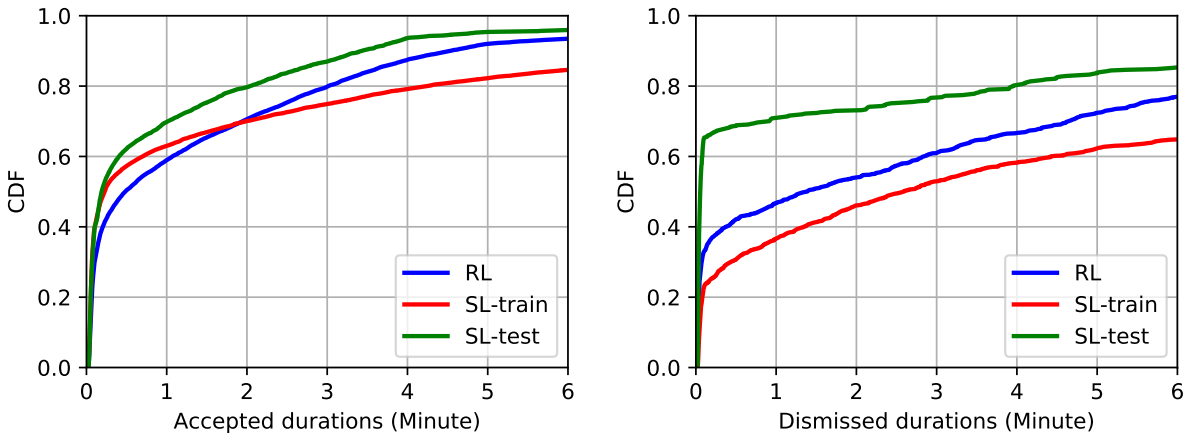
(a) The RL agent takes one week to converge.



(b) The RL agent can quickly adapt to user preference change.

Figure 5.7: Probability of sending a microtask across time by the reinforcement learning (RL) agent for two different users.

result shows that 73% of the responses from RL participants are *yes* which indicate the users were available when they answered the questions. However, only 54% of the responses in SL indicates users were available, suggesting that RL achieves a better job of finding



(a) (b)

Figure 5.8: CDF of time intervals for (a) answering and (b) dismissing notifications.

interruptible moments.

People dismiss notifications much faster in SL-test. Another measure of interruptibility can be to observe how long users take to respond to microtasks. Figure 5.8a displays the time intervals that users take to answer a notification since prompted. The result shows that at least 58% of the microtasks are answered within one minute in both algorithms. SL-test achieves the shortest answer time but the answer time in both algorithms do not differ too much. Figure 5.8b shows that 34%, 24%, and 65% of the notifications are instantly dismissed in RL, SL-train, and SL-test (i.e., within 5 seconds) right after they are scheduled.

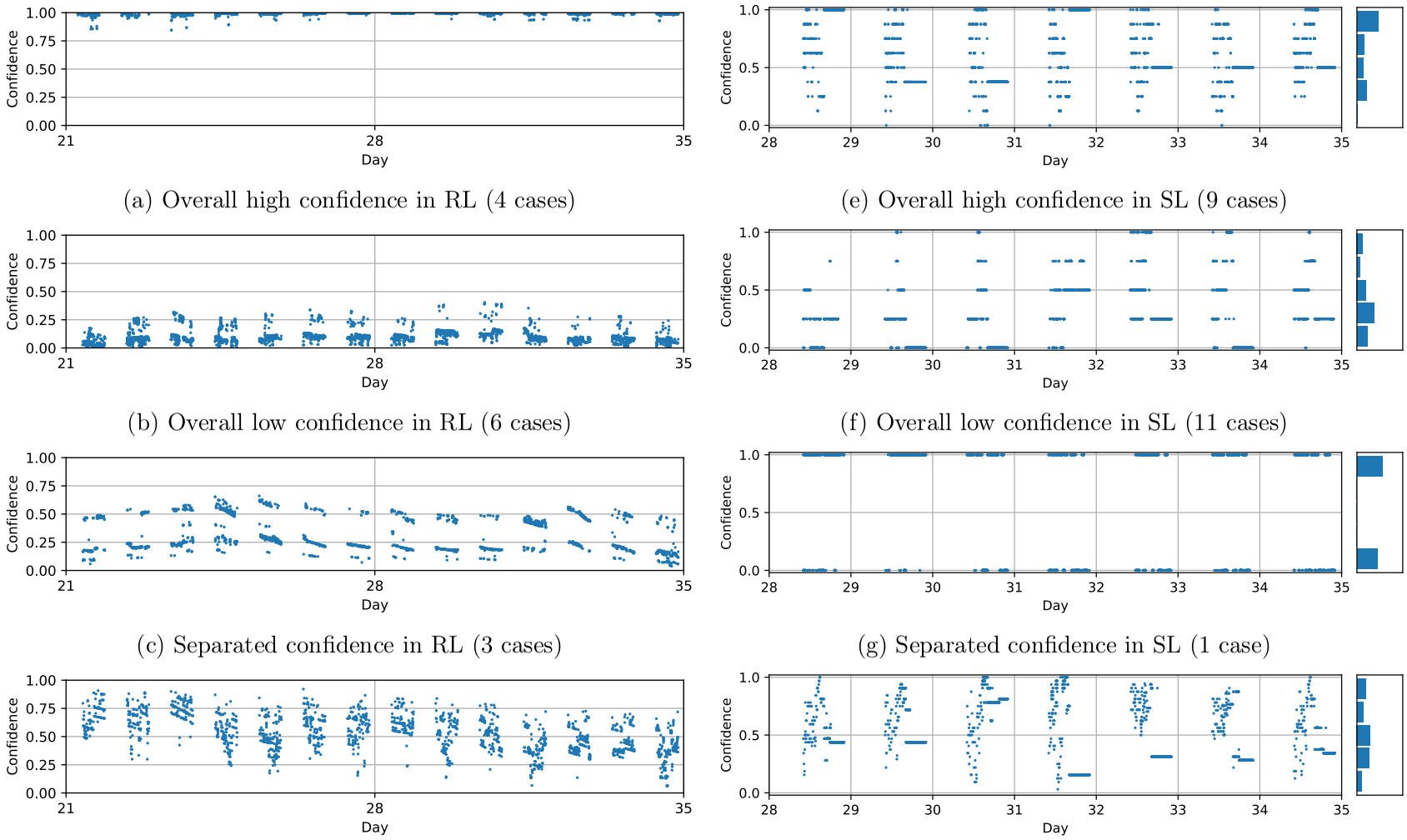
5.6.4 Learning Algorithm Analysis

A2C converges in a week. To understand when the RL agent starts to learn something meaningful, we pick one user as an example and plot the confidence scores of all the interruptibility queries in Figure 5.7a. The *confidence* is defined as the likelihood for the learning algorithm to send a microtask, which is part of the output of A2C. We provide daily reward on the bottom for comparison. Since the agent receives bigger rewards for the first 7 days possibly due to high user interest in the beginning, the agent gets more confident in prompting notifications. As the user behavior changes around 10^{th} day, the daily reward

and the confidence drops. It can be explained that as time progresses, the agent adapts to the changing user behavior.

RL can adapt to user preference change and capture the weekly pattern. Figure 5.7b presents another user who actively dismissed notifications in the middle of the study. The amount of dismissed notifications significantly increased after day 20. The confidence drops when RL starts receiving negative reward which in turn suppresses the microtasks to this user. However, the confidence raises on day 28 and day 35 which are Sundays. We ask the user about this pattern after the study and they confirmed that they were only available during weekends.

Users can be categorized into four coherent groups. We observed that response behavior varied widely between users. We analyzed the learned behaviors of both RL and SL agents for each user. For RL, we use the probability of sending a notification as given by the policy network. For SL, we use the Random Forest confidence, which is the number of votes it gets for sending a notification from the tree ensemble. Figure 5.9 depicts typical user from the four groups we have identified. The first group follows a high-confidence pattern because of the high answer rate, and the second follows a low-confidence pattern as users dismiss or ignore majority of the microtasks. The third pattern shows the points are vertically separated into two clouds. We observed that both RL and SL increase the confidence of sending a microtask when screen is turned on. The final pattern shows the confidence varies significantly within a day. We found that it is because there are several factors that impact the decision. For example, we found that the agent becomes more confident when either the screen is on or ringtone mode is adjusted to normal in Figure 5.9d, and the confidence increases when screen is on or non-stationary motion is detected in Figure 5.9h. The combination of multiple variables cause different confidence levels. We checked the learned patterns with each of the users response patterns and they aligned well.



(a) Overall high confidence in RL (4 cases)

(e) Overall high confidence in SL (9 cases)

(b) Overall low confidence in RL (6 cases)

(f) Overall low confidence in SL (11 cases)

(c) Separated confidence in RL (3 cases)

(g) Separated confidence in SL (1 case)

(d) Diffused confidence in RL (2 cases)

(h) Diffused confidence in SL (5 cases)

Figure 5.9: Examples of confidence change in RL (left column) and SL (right column). Four common patterns are found in both algorithms. For SL, we provide the confidence distribution on the side to assist visualization.

5.6.5 System Performance

Client App Battery Impact The battery consumption was measured on a new Pixel 2 phone with Android 8.0 with a sim card equipped. We factory reset the phone to minimize measurement noise. We measured the battery consumption with and without our app installed separately. With a fully charged phone, our results show that the battery level drops to 84% after 12 hours without our app installed, and to 77% when our app is running in the background. Hence, our app only increases 7% battery overhead during a day.

Server Request Handling Our server is hosted on a desktop with a 4-core Intel CPU @ 3.5 GHz and 32 GB DDR3 memory. We benchmarked the overhead of both algorithms when processing an interruptibility query. Supervised learning consumes 163 MB and takes 1.27 ± 0.07 seconds to complete a query, and RL consumes 243 MB and takes 2.28 ± 0.16 seconds. The major cause of the time overhead is for loading and initializing the agents in both algorithms. Our results suggest RL introduces higher overhead.

5.6.6 Post-Study Survey

We collected 26 effective post-study surveys, 14 from the RL group and 12 from the SL group. We first present whether participants perceived any difference during the 5-week duration of the study. 13 participants in RL group and 11 in SL observed difference. These users were subsequently asked to rate the change of the task schedule with a 5-point Likert scale where 1 (5) means noticeably worse (better). The rating is 4.23 ± 0.58 and 3.45 ± 1.44 in RL and SL, respectively, implying participant perception that RL can more effectively learn the opportune moment to engage with users. Participants in RL express *"started getting more/fewer notifications during specific times of the day"*. Participants mentioned receiving less undesired notifications during work (N=3)[6], studying (N=1), in the morning and evening (N=1), or receiving more notifications at opportune times such as when they are "sitting down" (N=1). A few SL participants observed a polarizing change when transitioning

[6]We use N=? to denote number of people.

to the testing phase: one participant received significantly more tasks, while two participants received significantly less amount of tasks (N=2). Two users indicated the app gave more notifications when they were studying (undesirable, N=2) while one user experienced less notifications when at work (N=1).

Participants were asked to identify the reasons behind certain notifications being perceived as disruptive. We observe that RL and SL users have different concerns. RL participants indicated that the tasks were sent out far too often than they expected (N=6). Some also reported that the app is unaware when they are engaged with other phone activities such as watching videos or playing games (N=5). On the other hand, the major concern of SL participants is that microtasks were delivered at inopportune moments in which they could not answer (e.g., driving, at work) (N=5). The frequency of notifications also heightens the disturbance (N=4).

In both algorithms, prompting at an inopportune moment is the major reason for dismissed notifications (N=8). However, participants sometimes dismissed notifications when they found the microtasks to be too challenging (N=4). 16 users reported arithmetic questions to be more difficult than other questions (N=16). However, one user chose to randomly select answers instead of dismissing notifications (N=1).

5.7 Discussion, Limitation, and Future Work

The goal of an RL agent is to maximize the long term reward, and the reward function is designed to achieve the desired outcome. In Quick Question, we investigate a simple objective which optimizes for the number of completed microtasks while minimizing the number of dismissed notifications to reduce disturbance. However, we do not claim such a reward mechanism is universal and should be redesigned based on the requirement of the target application. For example, the reward function can be augmented to discourage when a high-priority notification is missed. Also, our reward function can potentially incorporate the response rate and the response accuracy. In Quick Question, we hand picked the reward ratio of answering and dismissing a notification to be 1 to 5, but a better reward mechanism can be

explored based on behavioral models, or automatically optimized by Inverse Reinforcement Learning algorithm [BWJ17]. Designing a reward function that can generalize to all types of notifications is challenging, we will explore this in future work.

Although we keep the microtasks as homogeneous as possible in our study (i.e., length and task style), some questions do cause bias for certain users. For example, one user reported that he did not want to answer the diet questions because they made him feel self conscious. This bias, however, can be explicitly modeled in RL by augmenting the action space, i.e., the agent can decide which question to prompt based on user preference [GTM17]. Another direction to be explored in the future is to consider a different workload based on the intensity of interruptibility [YGL17]. For example, a system can prompt more than one microtasks in a row [CIT16] when a user is more available.

One limitation in our user study is that we only focus on Android population. Since the ecosystem of Android and iOS platforms are different (for example, swiping gesture means to dismiss a notification in Android but to engage it in iOS), a separate study has to be conducted to confirm that the results can be generalized to iOS users. Quick Question, however, can interface with an iOS app if the implementation is ready. Our system currently computes interruptibility on the server side and encounter two seconds of overhead. In the future, the learning pipeline can be optimized to reduce the computation time and the memory overhead.

5.8 Summary

We presented our system Quick Question which is a reinforcement learning based microtask scheduling method. To understand the trade-off between supervised learning and reinforcement learning for identifying user interruptibility, we conducted a 5-week user study with 41 participants recruited to collect user interactions with notifications in the wild. Our results suggest that RL is able to get more microtasks completed and effectively reduce the dismissed notifications. RL can achieve better user experience and more accurately identify interruptible moments. Moreover, RL can smoothly adapt based on user preference and

lower the burden for users to handle microtasks. Thus, reinforcement learning is a compelling technique to model user interruptibility.

Part III

Privacy Issues

CHAPTER 6

GPSI - From Pressure to Path: Barometer-based Vehicle Tracking[1]

6.1 Motivation

The increasing ubiquity of mobile computing devices has been accompanied by new sensing modalities and data fusion methods to sense or infer a wide array of physical phenomena and stimuli. The result of this is a distributed mobile sensor network whose sensors can yield information about users' behaviors and surrounding environments. Inferences made from these mobile sensors often provide valuable services such as accelerometer-based motion tracking, camera-based heart rate monitors, and life-logging or quantified-self services. These services have also proven to be integral components for environmental monitoring [ATP12], traffic analyses [MPR08, ME12], event discovery [OST14], and population-level analytics in general in what are increasingly referred to as *smart* cities.

Recent mobile devices have introduced yet another sensing modality in the form of barometric pressure sensors. These sensors can already be seen on mobile devices such as the Apple iPhone 6, Google Nexus 5, and Nexus 6. Barometric pressure sensors introduce a level of geographical dependency unseen in previous sensors, including magnetometers. While a magnetic compass might behave differently in different geographical locations, a barometric sensor is *designed* to differentiate between different pressures and thus, to a large degree, different elevations. In this paper, we show that the high correlation between pressure sensors and elevation allow for accurate tracking of driving patterns based on pressure from mobile

[1]The idea behind the project name is to infer location data (GPS) from air pressure information (PSI, an air pressure unit), hence GPSI.

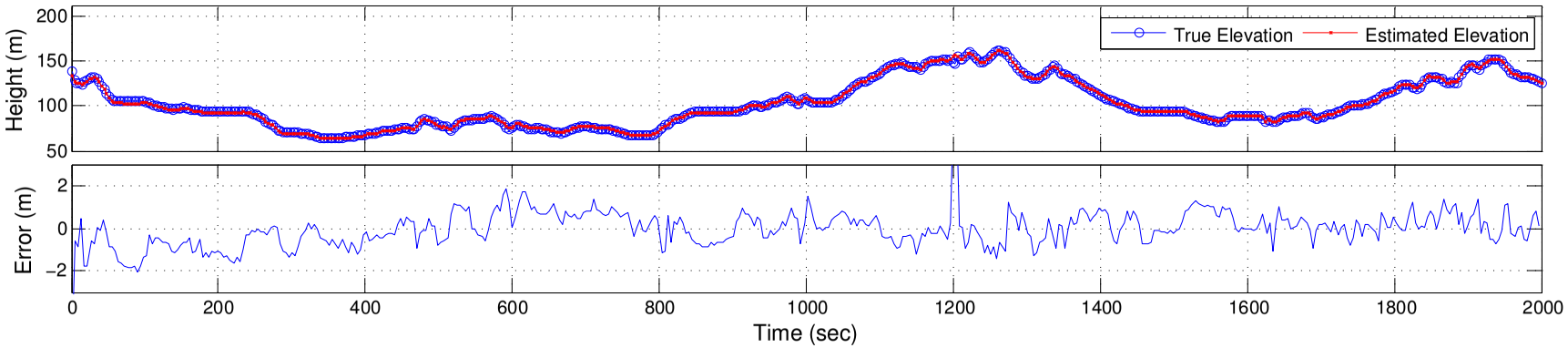


Figure 6.1: Elevation estimation from pressure with corresponding error, using simple linear model.

devices collected by a centralized computer. Tracking vehicles in this manner provides a low power method for analyzing a user's driving behavior over long periods of time, while the power consumption associated with GPS, cellular, or WiFi positioning methods may prove prohibitively high.

In addition, both Apple's iOS and Google's Android OS treat the barometer as a non-private sensor such as an accelerometer or gyroscope. In other words, an application that wishes to read a mobile device's barometer can do so without alerting the device user. We demonstrate how this *public* data access model, while easing data collection for city-level traffic analytics, can compromise a user's privacy and security, allowing malicious third parties to estimate a user's location over time, undetected by the user. There have been a number of similar efforts that have demonstrated such nefarious inferences made from mobile sensory data. Several notable works in this vein include accelerometer touch-type keystroke identification [MVB12a, OHD12a] and gyrophone-based microphones [MBN14a] for identifying spoken digits, such as credit card or social security numbers. These privacy and security threats are only made worse by the increasing tendency towards wearable and pervasive sensing [RGK11a]. In the face of these unceasing efforts to record and analyze personal user data, mobile computing and users thereof can no longer remain agnostic to the security ramifications of this data deluge.

In this work, mobile devices are treated as distributed sensors collecting pressure data and storing it locally. These sensors then opportunistically transmit their pressure data to a remote server whenever power and connectivity (e.g. WiFi) permit. This pressure data is then analyzed by a more capable and less power-constrained server to reveal the traffic patterns of each user. To do this, a user's coarse-grained location estimate is obtained by associating the device's IP Address with an ISP's geolocation. From there, the user's time-series pressure data is compared against a database of possible road segments and their corresponding elevation signatures. Under certain conditions regarding the uniqueness of the observed pressure data with respect to the elevation of the underlying map, this allows us to obtain a series of 'ranked' paths, ordered by descending likelihood. When a high confidence path can be obtained, the estimated path can be treated as an approximation of

the user's driving route. This provides mobile devices with a low power path logging utility for scenarios where power consumption is an important consideration.

Though many factors affect the pressure reported by barometer sensors (not the least of which are weather, air movement, and sensor drift), mobile barometric sensors can still be correlated with elevation *changes* with surprising precision. Using a simple linear model (the details of which are discussed in Section 6.3), height can be predicted to within an error of several meters. An example of this correlation is shown for 30 minutes of driving data in Figure 6.1, where the error rarely exceeds ± 2 m. This correlation, however, is made difficult due to several important factors: first, the conversion from pressure to elevation is time-varying and unknown *a priori*. Second, the user's vehicle is traveling at an unknown speed, essentially 'sampling' elevation points at variable and unknown rates. This makes it difficult to directly correlate pressure data to the elevation of a given road segment. Third and most importantly, the search space of possible paths against which to compare the user's collected pressure data is immense, even given coarse grained location estimates such as those obtained from IP Address geolocation.

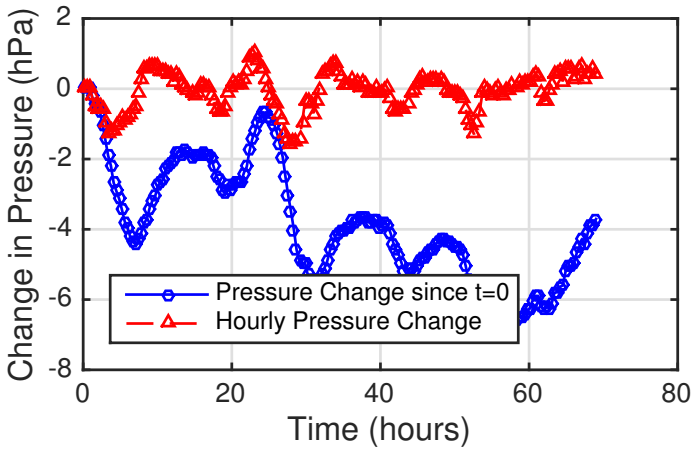
6.1.1 Contributions

In order to elucidate the degree to which pressure can be used to determine a user's driving path in the face of the difficulties mentioned above, we make the following contributions:

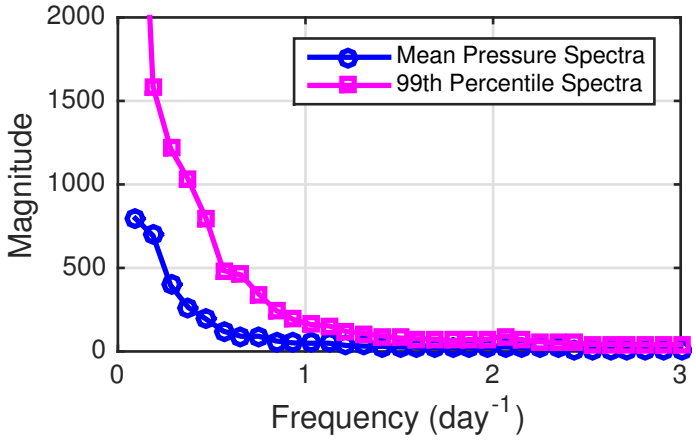
- We evaluate the accuracy with which pressure data can be used to predict the correct path from a *fixed* database of candidate paths using dynamic time warping (DTW). We also describe an algorithm for pressure-based path prediction using dynamic programming and DTW to find a jointly minimal cost path through an *arbitrary* graph of road segments.

- We evaluate the performance of our path estimation algorithms over a number of real test cases totaling 150 km and 4.6 hours of driving data.

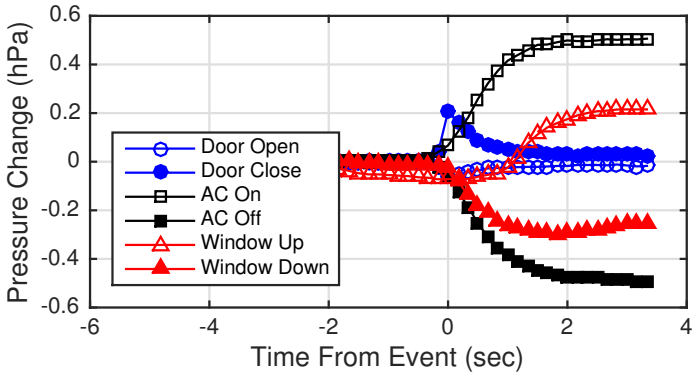
- By modeling errors in barometer sensors and elevation estimates, we simulate path



(a) Pressure change across multiple days.



(b) Pressure spectra from 2,309 U.S. cities provided by NOAA.



(c) Pressure changes for various driving events.

Figure 6.2: Noise and variations in barometric pressure measurements.

estimation results for random driving paths selected across a large number of cities with varying geographical landscapes.

- We evaluate the accuracy of our path prediction algorithms in terms of the distinctness of the pressure data with respect to the surrounding landscape, offering insights into the conditions under which driving paths can be accurately predicted.

6.2 Related Work

Related work in this area can roughly be divided into two categories: low power GPS, cellular, & WiFi positioning methods, and location- or transportation-specific inference mechanisms using low power sensor data.

Low Power GPS, Cell, & WiFi Positioning: Considerable work has gone into decreasing the power consumption of global (GPS) and regional (Cell & WiFi) positioning schemes. This includes the numerous assisted GPS (A-GPS) techniques [DT09] and cloud-offloaded (CO) post-processing of raw GPS [LPH12]. These technologies range from power consumption in the hundreds of milliwats (traditional GPS) to milliwatts (A-GPS) and even hundreds of microwatts (CO-GPS), while the latter requires highly customized GPS receiver hardware. By comparison, barometer power consumption is typically less than 10 μW [bmp], requiring no specialized hardware and providing a very practical tradeoff between traditional high power, high accuracy positioning techniques and ultra-low power mobile path *estimation* and analytics, where highly robust positioning may not be a hard requirement. As a comparison, a GPS module like the GlobalSat EM-506 would consume 170 mW even during a *hot start* measurement (acquisition recently acquired) and requiring up to 1 second to achieve a new fix [gps]. Duty cycling such a GPS receiver to achieve 10 μW average power consumption would allow for just 1 reading every 4 hours.

Inferring Location and Transportation: In the realm of location discovery techniques, recent work has demonstrated trajectory identification through inertial navigation

(dead-reckoning) for both pedestrians [PPC12] and vehicles [GPL10]. Han *et al.* further demonstrate in [HON12a] that accelerometer time-series data can be used to constrain a user's driving path to a subset of possible candidate paths within a given map. Hemminki *et al.* [HNT13b] demonstrate methods for inferring transportation modes using mobile accelerometer data, and Sankaran *et al.* demonstrates methods for inferring transportation modes using barometer data [SZG14a]. Finally, Zhou *et al.* demonstrate in [ZDH13a] that app usage statistics, network address-resolution, and speaking detection can be used to infer user identity, coarse geo-location, and even whether or not a person has a certain disease. The methods presented in this paper demonstrate how pressure data collected from mobile barometers can be used to predict driving paths. Unlike methods like those presented in [GPL10] and [HON12a], the proposed methods allow for accurate *absolute* path predictions, benefiting from the high correlation between barometer and elevation, as detailed in [MKM14].

6.3 Estimating Elevation

Though barometric sensors are strong indicators of geographic elevation, they are sensitive to a host of other pressure changes as well, making the conversion from pressure to elevation non-trivial. As elevation changes, changes in air density due to Earth's gravitational pull and many other factors cause a pressure gradient dictated by the *barometric formula*. Ignoring the effects of temperature change as a function of altitude, this formula is given in [usa76] as

$$P = P_0 \cdot \exp\left[\frac{-gM(h-h_0)}{RT_0}\right] \tag{6.1}$$

where P is the pressure in hecto-pascals (hPa) at height h meters above reference level h_0, g is the gravitational acceleration constant, M is the molar mass of Earth's air, and R is the universal gas constant for air. P_0 is the pressure measured at the reference height h_0 with temperature T_0, all of which can be measured beforehand. From (6.1), we can derive the equation $(h - h_0) = \frac{\log(P_0)RT_0}{gM} - \frac{RT_0}{gM}\log(P)$. For relatively small changes in elevation (hundreds of meters), this can be approximated by a simple linear function in P. Because

the reference pressure P_0 is in general a function of time, height h can be more generally approximated as

$$h(t) \approx \alpha(t) - \beta P \tag{6.2}$$

for scalar β and time-varying offset $\alpha(t)$. Over short periods of time (less than 1 hour), this prediction can be quite accurate. In fact, though the Bosch BMP280 pressure sensor used in both the iPhone 6 and Nexus 5 specifies a vertical pressure resolution of about 1 m [bmp], we have experimentally validated relative pressure sensitivity closer to 10-20 cm. Despite this strong correlation, determining the parameters $\alpha(t)$ and β can be challenging, especially given the time-varying aspect of the offset $\alpha(t)$ due to, among other things, weather.

6.3.1 Elevation Model Estimation

The model parameters $\alpha(t)$ and β in (6.2) dictate the accuracy of *absolute* elevation prediction. Thankfully, the scaling term β can be considered constant over very large ranges in elevation, due to the relative flatness of earth's surface with respect to its diameter. The offset $\alpha(t)$, on the other hand, varies wildly with time and coarse location. This can be seen in Figure 6.2(a), where pressure collected at a static location over a 70-hour period exhibits pressure changes nearing 7 hPa or nearly 60 m estimation error. Muralidharan *et al.* describe this problem in detail in [MKM14]. Note, however, if we look at the pressure change over 1 hour periods, the change rarely exceeds ± 1 hPa (or roughly 8.3 m). Furthermore, a survey of hourly pressure data from 2,309 cities in the U.S. provided by the National Oceanic and Atmospheric Administration (NOAA) Climatic Data Center [noa] shows that, the pressure change in 1-hour periods is below 1 hPa over 99% of the time. Additionally, analyzing these changes in the frequency domain indicates that the vast majority of pressure changes happen at the scale of 1 or more days, rather than hourly. This is shown in Figure 6.2(b). Given the slow dynamics of weather, pressure data provided by weather stations can be used to calibrate the offset term $\alpha(t)$ to within 1 hPa error—the typical resolution of pressure reported by weather stations. If such a station does not exist in close proximity to a user's coarse location, in some scenarios *relative* elevation can still be used to estimate a

user's driving path with some reduction in estimation accuracy. This is discussed in more detail in Section 6.4.

6.3.2 Pressure Events & Noise Sources

In addition to model dynamics caused by weather, mobile barometers experience 'noise' from a number of events causing changes in air flow. This includes opening and closing doors and windows as well as changing air conditioning. Examples of this are shown in Figure 6.2(c). The largest magnitude pressure change is 0.5 hPa (or 4.2 m error) when air conditioning is turned completely on or off. Our path detection algorithm must be resilient to these slight perturbations.

Finally, barometer sensors themselves are not perfect and typically exhibit (small) drift over time. This error is in general non-gaussian, exhibiting temporary drifts from the true pressure while periodically returning to accurate estimates. We propose modeling this error using an Ohnstein-Uhlenbeck diffusion process, which is similar to a low-pass-filtered white noise process [Enr08]. This error model will be used to generate realistic barometric pressure traces for simulations, as outlined in Section 6.5.2.

6.4 System Overview

Our path prediction system is composed of two main components: (1) a low power mobile app that continually monitors barometer data, periodically sending the data back to the second component: (2) a centralized analytics server that maintains road maps and elevation data and uses the collected sensory data together with the map and elevation database to estimate likely paths. This is shown in more detail in Figure 6.3.

Each server for a given city contains the corresponding road maps and elevation data from publicly available online databases. Specifically, the server downloads and manipulates data from (i) Open Street Map (OSM) [Ope15], providing road topologies including segments and intersections, and (ii) the Google Elevation API [Goo15], which provides a database from

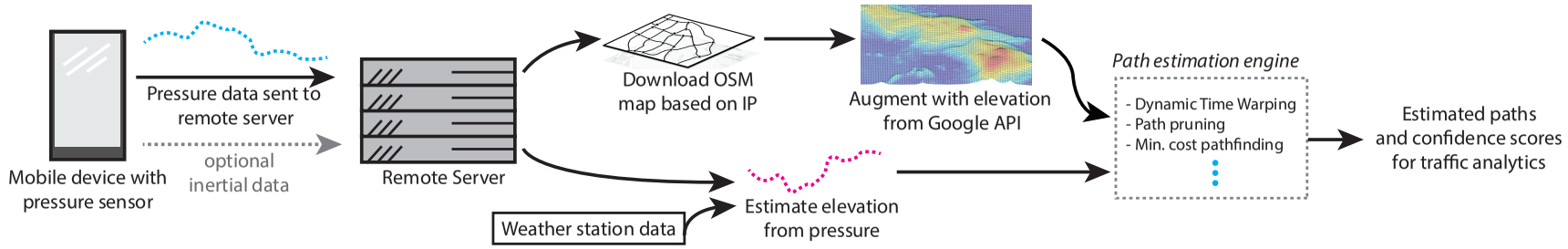


Figure 6.3: System overview, from pressure data collection to path estimation and confidence score.

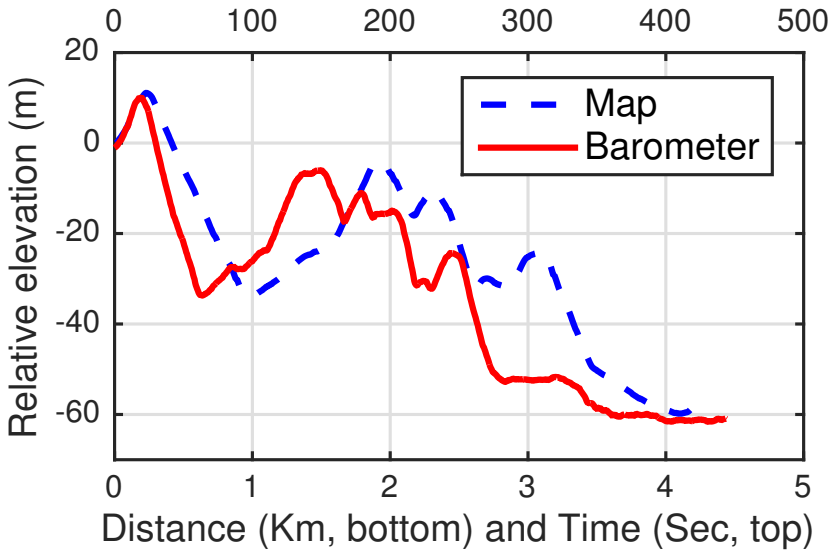
which to query the elevation of individual latitude and longitude points. Upon receiving the barometer data from a mobile device, the server's elevation conversion module converts the pressure readings to an estimate of absolute elevation values. If the sensor data is collected in a region close to a weather station, the pressure from that station can be used to calculate the offset $\alpha(t)$ in (6.2) and *absolute* pressure can be obtained. Otherwise, the system reduces to comparing *relative* elevation changes instead. Finally, we perform a number of pattern recognition routines including dynamic time warping (DTW) to perform path matching. These estimation routines are the main contribution of this work and the subject of the following sections.

6.4.1 Elevation Map Generation

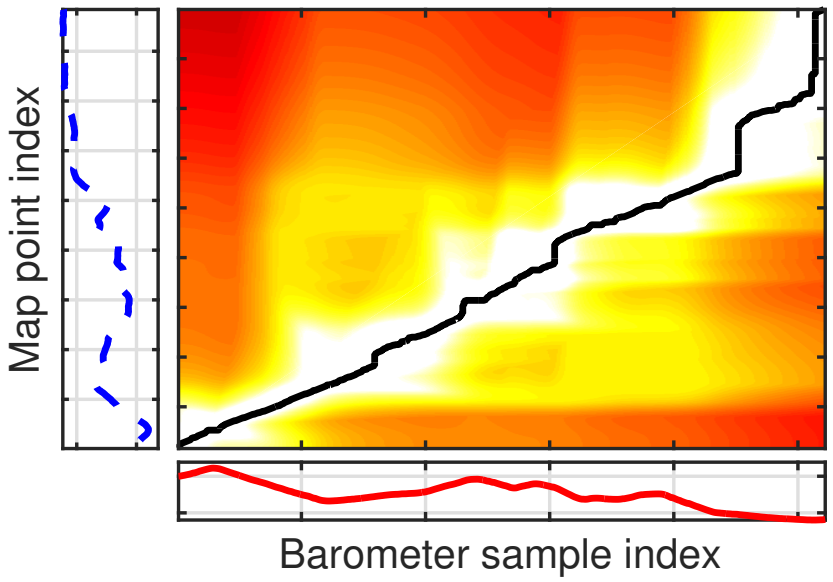
In order to estimate the path along which a user has driven, the server must first generate a database of possible road segments and their corresponding elevations. To do this, we combine the OSM road topologies with Google's publicly available Elevation API. Road maps are downloaded from OSM in an XML format composed of 64-bit unique *node* identifiers and their corresponding latitude and longitude values. These 'nodes' are connected by elements termed *ways*—ordered lists of connected road nodes. In general, either endpoint of a *way* aligns with road intersections. Ways composed of more than 2 nodes are often used to better represent road curvatures between intersections. From the OSM road nodes and ways, we construct a graph $G = (V, E)$ where V is the set of road nodes such as intersections and dead ends on the map, and each edge in E represents a road segment. The elevation of each segment is then queried every 10 meters, creating new *internal* nodes belonging to set N and extending the original graph to $G' = (V, E, N)$. All nodes $n \in N$ are augmented with latitude, longitude, and elevation attributes.

6.4.2 Dynamic Time Warping

Because the user is traveling at an unknown and variable speed, the corresponding pressure data behaves as a sampled version of the true elevation with variable sampling rate. In other



(a) Map elevation vs. barometer-based elevation.



(b) DTW matrix D with min-cost path.

Figure 6.4: An illustration of path elevation matching using dynamic time warping.

words, the collected pressure may be of a short duration, long duration, or it may contain pauses (when the vehicle is not in motion) or increased speeds. To compute the similarity of two signals which could potentially be scaled by time, we use dynamic time warping (DTW) algorithms like those developed for speech recognition (where the same word may be spoken with variable durations) [Vin68]. DTW is a time-series alignment algorithm in which two signals are compared against each other by means of a cost matrix. If the two series are denoted by column vectors $\mathbf{s}^p = \{s_i^p\} \in \mathbb{R}^{N_p}$, representing the elevation corresponding to a candidate $\underline{\mathbf{p}}$ath in G', and $\mathbf{s}^b = \{s_j^b\} \in \mathbb{R}^{N_b}$, corresponding to the $\underline{\mathbf{b}}$arometer-based elevation estimate, the cost matrix C contains $N_p \times N_b$ elements where element $c_{i,j} = f(s_i^p, s_j^b)$. The function $f(\cdot)$ serves as a distance function to represent the difference between the two signals at given indices and is typically defined by an ℓ_2-norm. The goal of DTW is to find the minimum-cost path through cost matrix C starting at $c_{0,0}$ and ending at c_{N_p,N_b} The details of the DTW algorithm are given in Algorithm 1. For each element $d_{i,j}$ in a DP matrix D, we store the minimum cost of all possible paths from $c_{0,0}$ to $c_{i,j}$. Each element $\psi_{i,j}$ in a traceback matrix Ψ records the last transition which leads to the minimum cost of $d_{i,j}$. Thus, the final similarity between $\mathbf{s}^p$ and $\mathbf{s}^b$ is embedded in d_{N_p,N_b}. If $\mathbf{s}_p$ and $\mathbf{s}_b$ are similar, their corresponding *DTW* score will be low, and if they are dissimilar their cost will be high, regardless of variable lengths or sampling rates. An example of the DTW procedure is illustrated in Figure 6.4(a) for example map path- and barometer-based elevation data collected from Los Angeles, CA, with the corresponding DP matrix D in Figure 6.4(b). Following Algorithm 1, the complexity of the DTW algorithm is $\mathcal{O}(N_p \cdot N_b)$.

DTW can correctly identify paths using barometer measurements provided that the errors in the barometer sensor and Model (6.2) are sufficiently smaller than the variance in the traversed path elevation—i.e. when the path elevation is sufficiently distinct in the presence of noise. The result of an initial experiment performed over 29 road segments using DTW to compare measured pressure to "candidate" path elevations is shown in Figure 6.5. Here the true path is correctly identified by the minimum DTW score for all test cases, and the runner-up paths have anywhere from 10× to 10000× higher cost.

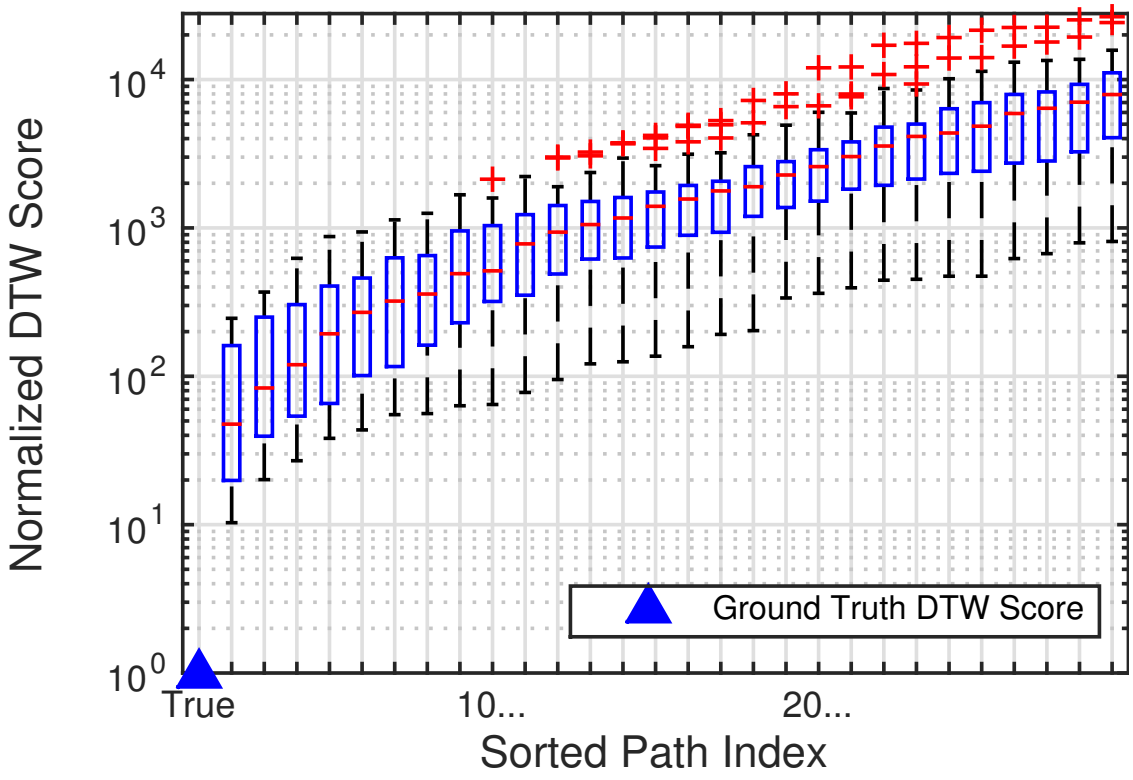


Figure 6.5: Normalized DTW scores for 29 barometer traces collected while driving. False paths have between 10 and 10000× higher DTW scores.

6.4.3 Candidate Path Generation

DTW provides the means for comparing measured pressure data against candidate path elevations: the server must search for a path $\hat{p}$ through G' such that the elevation along $\hat{p}$, denoted by $\mathbf{s}^{\hat{p}}$, and and the elevation converted from barometer data, denoted by $\mathbf{s}^b$, are similar, i.e.,

$$\hat{p} = \arg\min_{p} DTW(\mathbf{s}^p, \mathbf{s}^b)$$

Unfortunately, the search space of all candidate paths can be quite large. In fact, if we allow for path loops, the search space can be infinite—i.e., there are infinite combinations of paths in G'. Because of this, the server must perform DTW while traversing G' in an efficient manner. In the following sections we present two potential methods for doing so: (i) a naive

Algorithm 1: Dynamic Time Warping via Dynamic Programming with traceback.

Data: Signals $\mathbf{s}^p \in \mathbb{R}^{N_p}$, $\mathbf{s}^b \in \mathbb{R}^{N_b}$
Result: similarity score R_{score} and traceback vectors $\mathbf{v}^p \in \mathbb{Z}_+^{N_p}$ and $\mathbf{v}^b \in \mathbb{Z}_+^{N_b}$

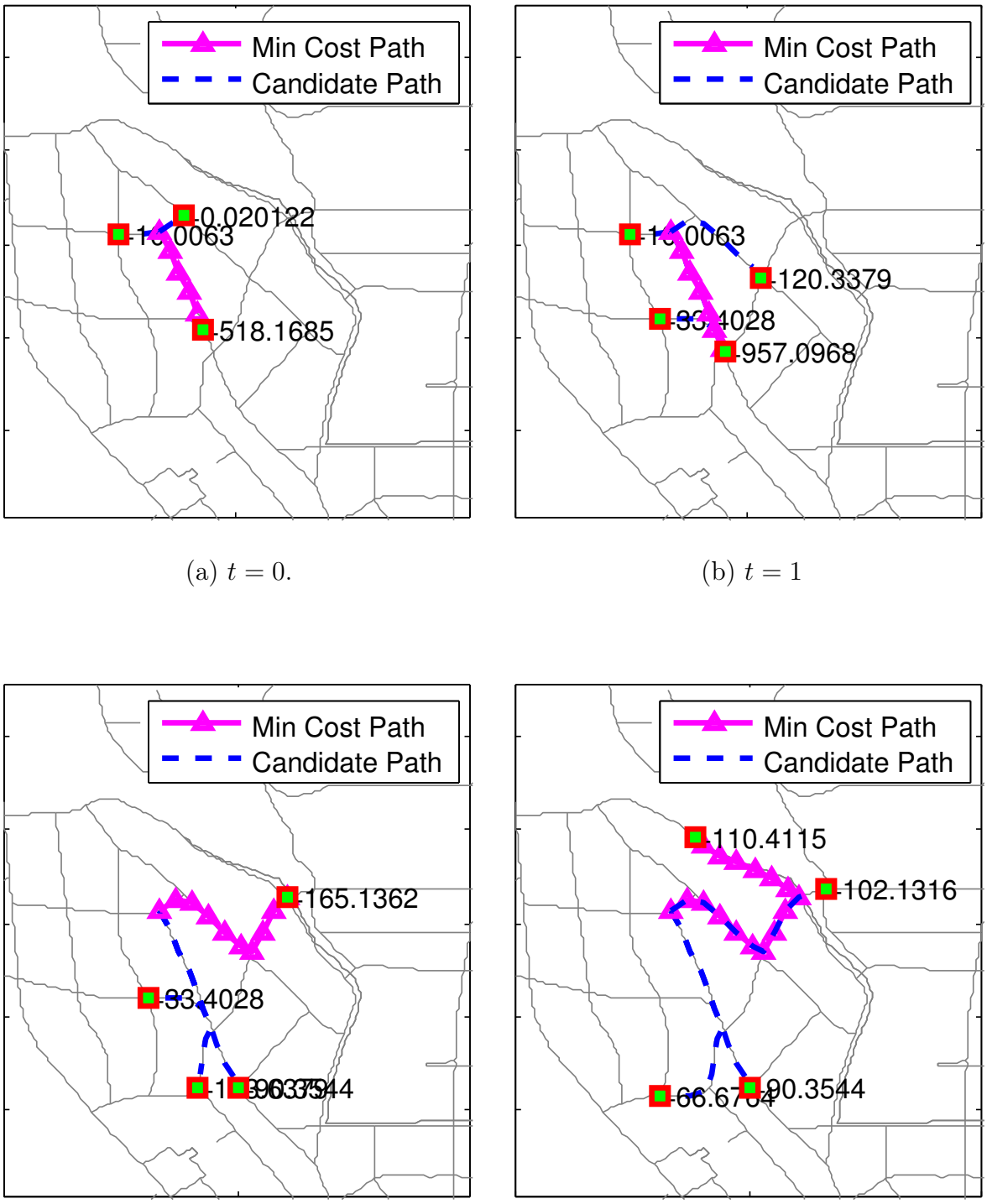
Cost matrix: $C = \{c_{i,j}\} = |s_i^p - s_j^b|^2$;
DP matrix: $D = \{d_{i,j}\} = 0$;
$d_{0,j} \leftarrow \infty$, $\forall j = 1 \dots N_b$;
$d_{i,0} \leftarrow \infty$, $\forall i = 1 \dots N_p$;
Traceback matrix: $\Psi = \{\psi_{i,j}\} \leftarrow \phi$;
for r *in* $1 \dots N_p$ **do**
    for c *in* $1 \dots N_b$ **do**
        $d_{r,c} \leftarrow \min_{t=(\Delta r, \Delta c)} (d_{r-\Delta r, c-\Delta c}) + c_{r,c}$;
        $\psi_{r,c} \leftarrow \arg\min_{t=(\Delta r, \Delta c)} (d_{r-\Delta r, c-\Delta c})$;
    end
end
$R_{score} \leftarrow d_{N_p, N_b}$;
traceback: $r \leftarrow N_p$, $c \leftarrow N_b$;
while $r > 1$ *or* $c > 1$ **do**
    $v_r^p \leftarrow c$; $v_c^b \leftarrow r$;
    $(\Delta r, \Delta c) \leftarrow \psi_{r,c}$;
    $r \leftarrow r - \Delta r$; $c \leftarrow c - \Delta c$;
end

breadth-first graph traversal method with pruning heuristics, and (ii) a joint optimization approach using dynamic programming.

6.4.3.1 Greedy Path Finding

In order to determine which of all possible candidate paths would most likely generate an observed series of pressure data, we can use an agent-based approach in which each agent traverses G' beginning at one specific node in V. Because G' is not necessarily (and in general is not) free of cycles, a breadth-first traversal will quickly escalate into an exponential problem. To combat this, each agent ensures that no path it explores creates a loop of length less than a threshold Γ_{loop}. If Γ_{loop} is large enough, this allows for reasonable driving trajectories while greatly limiting the search space.

At each iteration, agents perform an exploration phase and a pruning phase. Pruning occurs in three stages: (1) after all agents have finished exploring new nodes, the solver



(a) $t = 0$.

(b) $t = 1$

(c) $t = 2$

(d) $t = 3$

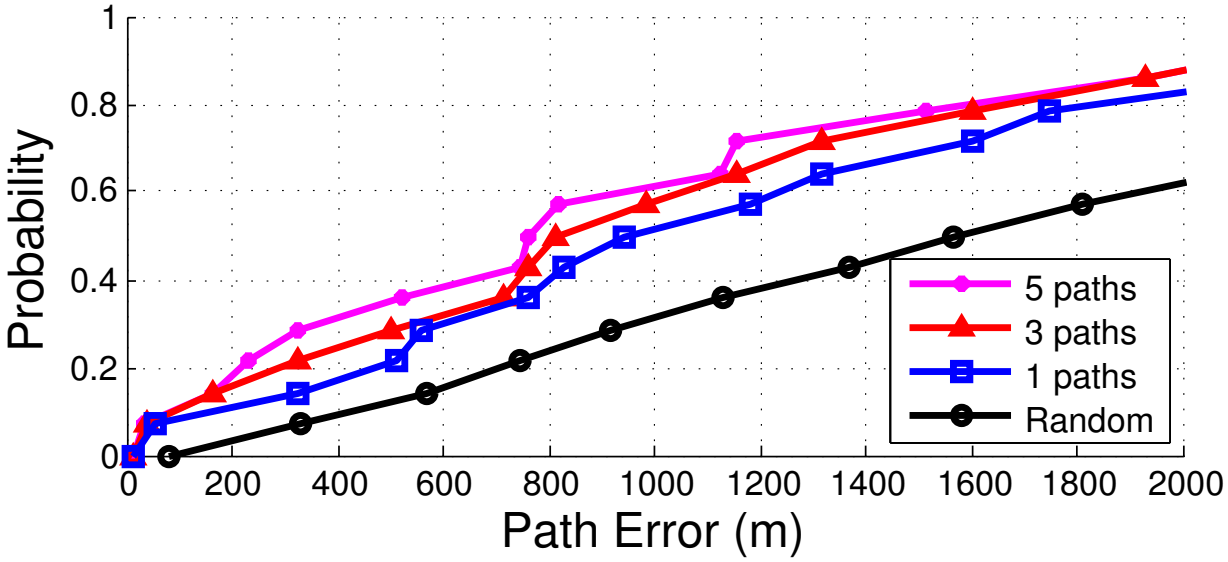
Figure 6.6: Snapshots of an agent from greedy pathfinding, exploring 4 possible paths.

calculates all candidate path scores using a path-length-normalized version of Algorithm 1. These scores are sorted and a threshold T_{score} is computed so that (2) any path whose DTW score exceeds T_{score} is pruned, and finally (3) all agents are instructed to prune their worst paths until they contain no more than $\Gamma_{maxpath}$ paths. This allows for diversity in possible path starting locations and reduces the complexity of the search algorithm to polynomial time. The solver terminates if the minimum cost of all candidate paths does not change by more than ϵ (1%) across a recent period of W (10) iterations. Upon terminating, the greedy solver returns the top ranked (lowest cost) paths and their corresponding scores. A series of snapshots from the operation of a single agent in the greedy solver is shown in Figure 6.6. Here we have set $\Gamma_{maxpath}$ to 4, so that at each iteration only 4 paths are being considered (labeled by green squares). For each iteration, the minimum cost path is displayed by a string of magenta triangles, while other candidate paths are displayed with a dashed blue line. At each iteration, there are at most $|V| \cdot \Gamma_{maxpath} \cdot d_{out}^{max}$ paths, where d_{out}^{max} is the maximum out-degree of any node in V. As a result, the time complexity can be bounded by $\mathcal{O}(|V|) \cdot \mathcal{O}(DTW) = \mathcal{O}(|V| \cdot N_p \cdot N_b)$. In practice, we truncate the map elevation segment to at most length N_b, giving a final complexity of $\mathcal{O}(|V| \cdot {N_b}^2)$. These calculations are performed over a number of iterations, but by setting a maximum number of iterations (30 in our experiments), the computational complexity remains unchanged.

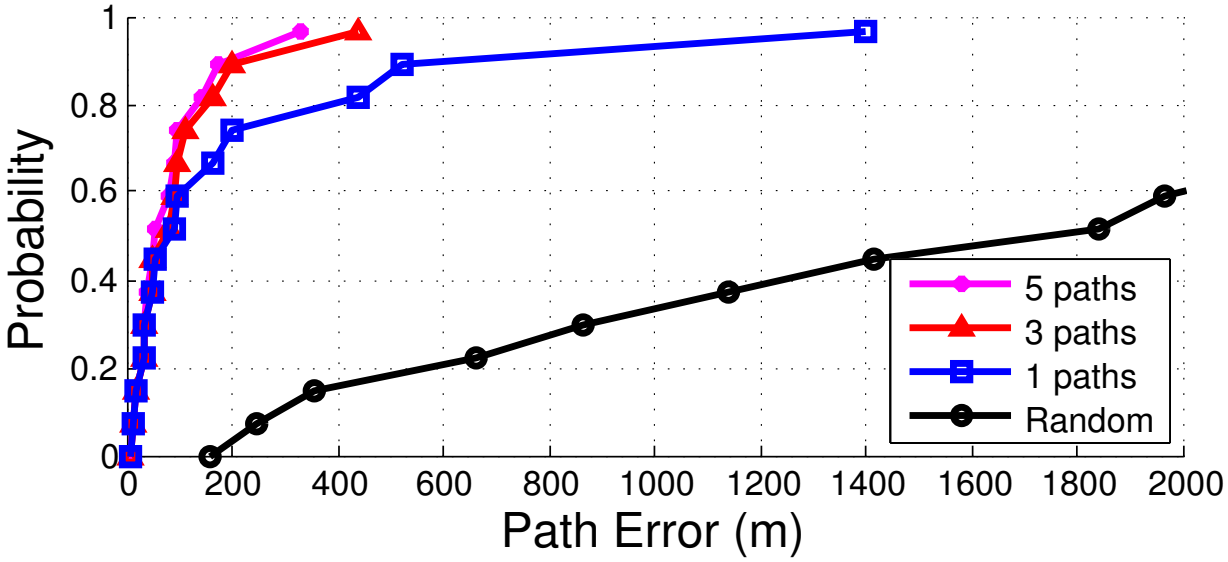
6.4.3.2 DP-based Path Finding

The greedy search algorithm explained in the previous section is intuitive and can search efficiently over large maps. However, the pruning heuristics and search space reduction can easily lead to local minima and non-optimal path prediction. Additionally, the greedy search heuristics do not consider path timing information, reducing estimation accuracy once more. To overcome these drawbacks, we instead consider an approach inspired by Dijkstra's shortest path search algorithm, whose underlying algorithm is again solved via dynamic programming. We call this the *DP-based* path finding algorithm.

Intuitively, we would like to use a shortest path algorithm such as Dijkstra's, where the

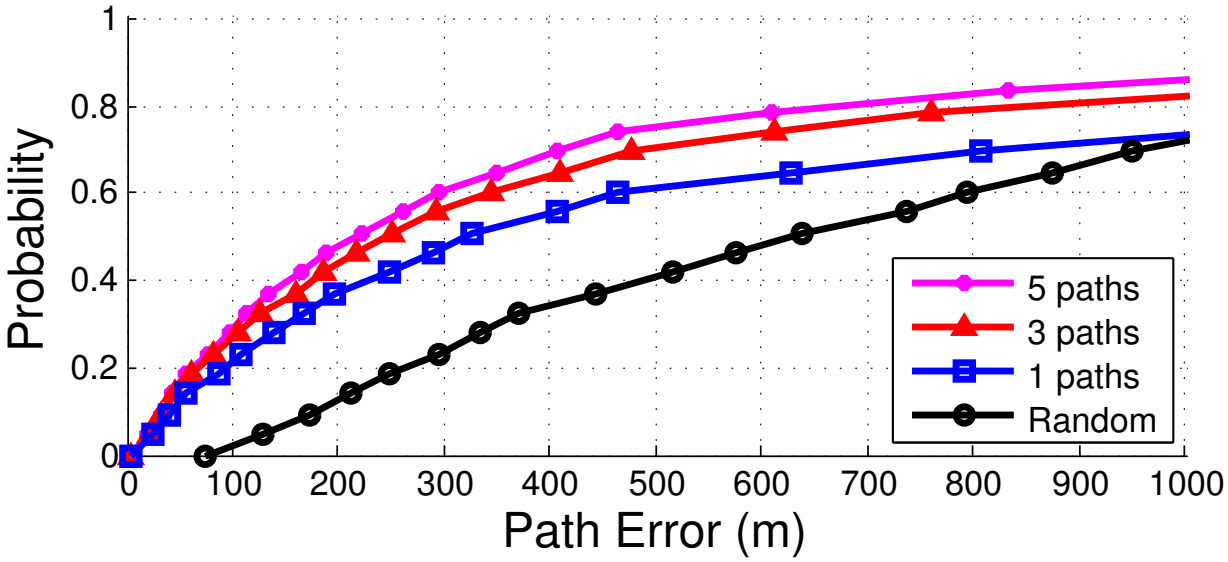


(a) Greedy solver

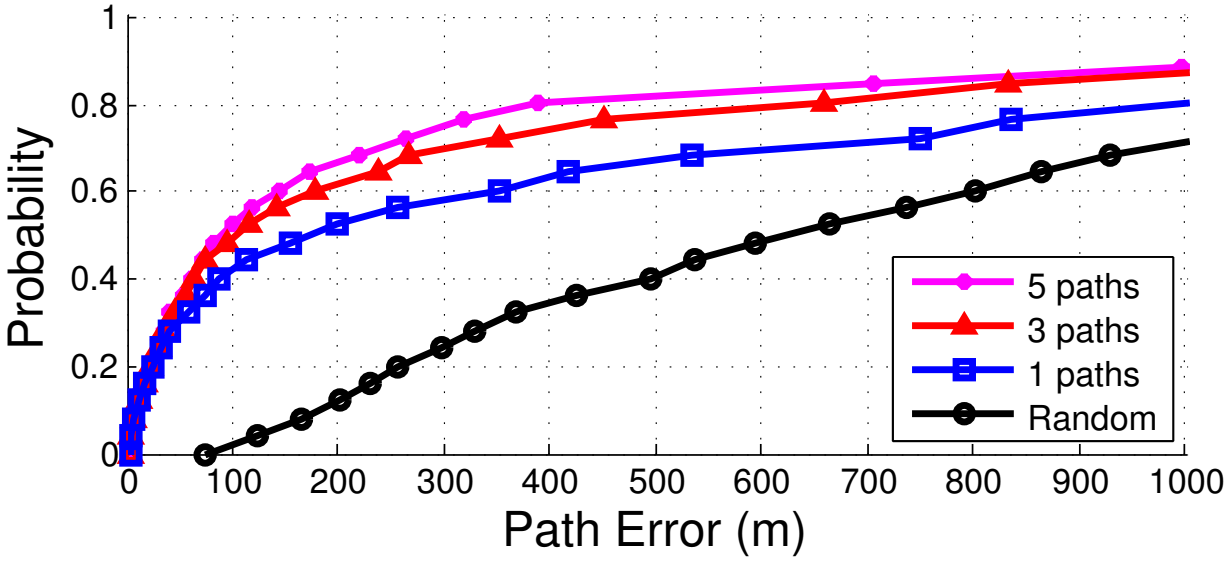


(b) DP-based solver

Figure 6.7: Path prediction errors for real driving data



(a) Greedy solver



(b) DP-based solver

Figure 6.8: Path prediction errors for simulated driving data

cost of each edge is calculated by DTW. However, there are two obstacles in doing this: first, the cost of each edge cannot be determined statically, as the cost of traversing a road segment depends on what segments were crossed previously. For example, road segment A might be a poor match for the *start* of our collected pressure data, but it may be a very close match if the user passed through segment B first on their way to A. Second, we must have some notion of time—e.g., for each node $v_i \in V$, what is the minimum cost of visiting that node given arrival time t_i? This adds an additional dimension to our dynamic cost shortest path algorithm.

To provide a notion of time, we consider a path to be not just a series of locations but also a series of corresponding timestamps. Thus, we redefine path p to encompass a series of *states* $[q_0, q_1, \ldots, q_k]$ where each state q_i is a 2-tuple (v_i, t_i) indicating that the user has reached intersection v_i at time t_i. Rather than define cost in terms of the DTW score between two entire paths, we can now define the marginal cost in transitioning from state q_i to q_j. Specifically, if we have already discovered a portion of the true path, say $\mathbf{q}' = [(v_0, t_0), (v_1, t_1), \ldots, (v_i, t_i)]$ whose matching cost thus far is δ_{q_i}, then the cost of transitioning to $q_j = (v_j, t_j)$ is defined as $cost(q_i, q_j)$, so that $\delta_{q_j} = \delta_{q_i} + cost(q_i, q_j)$.

Thus, for each state $q = (v, t)$, the optimal δ_q is defined as a partial path ending at vertex v at time t and can be determined by the following recursive function:

$$\delta_q = \min_{t_- < t, \langle v_-, v\rangle \in E} (\delta_{q_-} + DTW(\mathbf{s}^p, \mathbf{s}^b)) \tag{6.3}$$

$$\delta_{q=(v,0)} = 0, \forall\ v \in V$$

where t_- and v_- specify the time and node corresponding to the previous state q_-, $\mathbf{s}^p = Elev(\langle v_-, v\rangle)$ is the elevation along the edge $\langle v_-, v\rangle$, and $\mathbf{s}^b = [s^b_{t_-}, \ldots, s^b_t]$ is the barometer-based elevation estimate from time t_- to t. This recursive definition successfully reduces the exponential number of candidate paths to a polynomial time search algorithm: the complexity is decided by (1) the table size of δ, (2) the number of state transitions, and (3) the complexity of DTW, leading to the final complexity of $\mathcal{O}(|N|^2 \cdot {N_b}^3)$. This does not scale to large maps as well as the greedy approach discussed in Section 6.4.3.1, but the

jointly minimal solution provided by dynamic programming provides a drastically increased accuracy in path prediction.

6.4.3.3 Improving DP-based Search Complexity

The DP-based path finding algorithm suffers from high complexity due to many redundant calculations, both across time and location (i.e., vertex). For example, state $\delta_{q=(v,t)}$ is updated by any prior state $q_{\text{-}}$ with edge $\langle v_{\text{-}}, v\rangle \in E$. As described in (6.3), DTW is performed for each possible transition to q. It is possible to amortize this DTW cost by flattening out this recursive relation.

The root cause of this redundancy is that we treat each node $v \in V$ as a 'checkpoint' representing a temporary path. Traveling from v_a to v_b requires enumerating possible arrival times t_b. If, however, we consider *intermediate* nodes $n \in N$ and treat adjacent nodes as 'micro' edges, the cost of computing each edge cost by DTW is reduced since each edge length is always 1. Furthermore, we can assume that it takes at least one time unit to travel to any adjacent node if the barometer sampling rate is sufficiently low. This removes our dependency on time, and the number of possible state transitions reduces to

$$\delta_q = \delta_{q_{\text{-}}} + \min_{\langle n, n_{\text{-}}\rangle} |s_n^p - s_t^b|^2 \tag{6.4}$$

which is bounded by the constant $\mathcal{O}(d_{out}^{max})$, yielding a final complexity of $\mathcal{O}(|N| \cdot N_b)$. The runtime of the DP-based prediction algorithm written in MATLAB and running on an Intel i7 laptop takes roughly 7-15 minutes for a 92 km^2 map running on a single thread, while the greedy algorithm typically completes in 5-10 minutes.

6.4.3.4 Additional Pruning Metrics

In addition to pressure data, a number of other inertial sensors can be used to further improve path prediction accuracy and prune improbable paths to increase runtime efficiency. Most notably, mobile accelerometers, gyroscopes, and magnetomers can be combined to give

accurate turn estimation as described in [HON12a]. These additional metrics can be easily integrated into both the greedy and DP-based path discovery algorithms at the cost of increased power consumption on the mobile devices.

6.4.3.5 Elevation estimation robustness

As described in Section 6.3.1, it is not always possible to achieve a high accuracy *absolute* elevation estimate. Our path search routines remain robust to errors in elevation estimation in two ways: first, the path search algorithm can be operated in *relative* elevation mode, in which only relative pressure changes are considered. Additionally, the DP-based search routine can be instructed to search over a range of possible elevation offset values, $\alpha(t)$. In doing so, the minimum score of all paths from all offset values is reported. This inevitably reduces the accuracy of the prediction algorithms, but it allows for some error margin in Eq. (6.2).

6.5 Evaluation

In order to evaluate the performance of our path prediction algorithms, we collected real driving data and performed extensive simulations over a wide range of geographical landscapes.

6.5.1 Tests on Real Driving Data

We collected real driving data across 150 km, totaling 4.6 hours of driving time and covering a range of different map topologies. Data was collected using a Nexus 5 smartphone with barometer pressure data sampled at 30 Hz and GPS sampled at 1 Hz for ground truth analysis.

The results of the two path prediction algorithms over all driving data are shown in Figure 6.7. For each, we plot the CDF of prediction root-mean-square errors (RMSE) for a number of ranked paths versus the average error induced by a random walk. More specifically, for

each point on the predicted path we calculate the squared distance to the corresponding point on the true path, averaging the squared errors and taking the square root to give us our final RMSE value. The result from '5 paths' represents the best result from the 5 lowest cost paths as estimated by the solver, the result from '1 path' represents the single lowest cost path from the solver, and so forth. The random results represent the minimum error from 5 random walks of G'. Figure 6.7(a) shows that the Greedy solver demonstrates a median of around 800 m error, versus the random walk's median error of 1600 m—only a marginal improvement over a random guess. On the other hand, the median error reduces to less than 60 m by using the dynamic programming algorithm, as shown in Figure 6.7(b). Additionally, in 80% of the cases the lowest cost path has an average error of just 200 m—roughly the length of a standard city block—and in 90% of cases one of the 5 lowest cost paths is correct to within 200 m.

6.5.2 Simulation

In lieu of collecting driving data across multiple cities for many hours, we conducted a series of simulated test cases. To begin, we downloaded 92 km^2 regions of road data and corresponding elevation data for 26 high-population cities in the U.S with, on average, 1046 km of roads per map. For each city, we conducted a series of random walks of variable lengths and speeds and with barometer noise modeled using an Ohnstein-Uhlenbeck diffusion process as stated in Section 6.3.2. This random process simulates a signal with periodic deviations from a mean μ (true pressure). The frequency and magnitude of these deviations are dictated by a volatility constant σ and reversion time θ describing how quickly disturbances return to the mean. For barometric sensors, we determined empirically that values of $\sigma = 0.04$ and $\theta = 150$ accurately simulate the errors observed in our collected data sets. These simulated barometer pressures and map/elevation databases were passed to our estimation algorithms in an identical manner to solving the real driving cases. The results of path estimations over these simulated data are summarized in Figure 6.8. Here, the greedy solver shows an improvement over the real driving data, but the DP-based algorithm shows a reduced estimation performance. On average over more than 500 simulated test cases, the greedy

solver can predict paths to within about 200 m with 50% probability while the DP-based algorithm can predict paths to within 100 m with 50% probability and to within 300 m with around 80% probability.

6.5.3 Analysis of Parameters

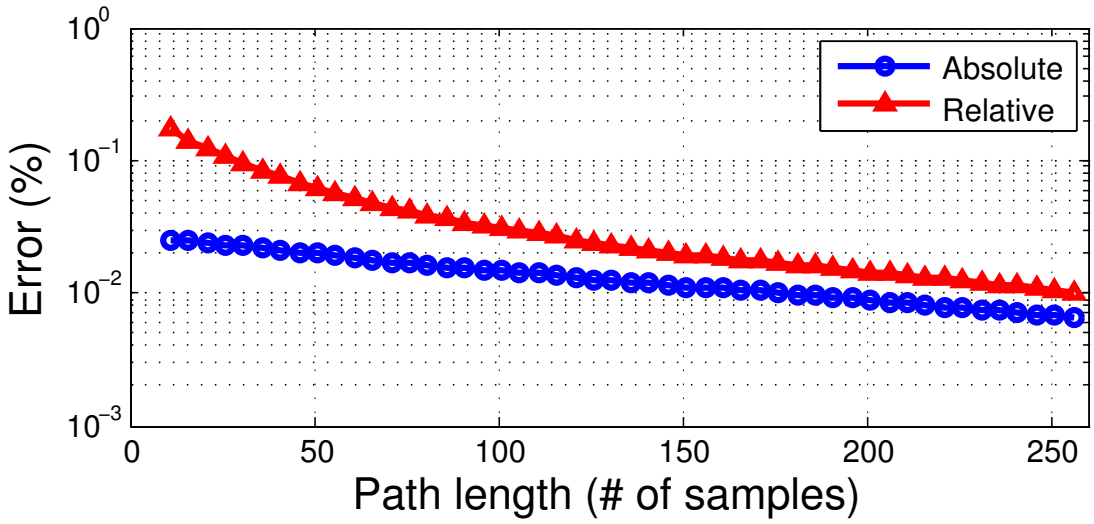
The results from real and simulated experiments demonstrated in the previous section indicate that under certain circumstances, driving paths can be quite accurately predicted from pressure alone. In this section, we provide some intuition into factors that affect this prediction accuracy.

6.5.3.1 Path Length

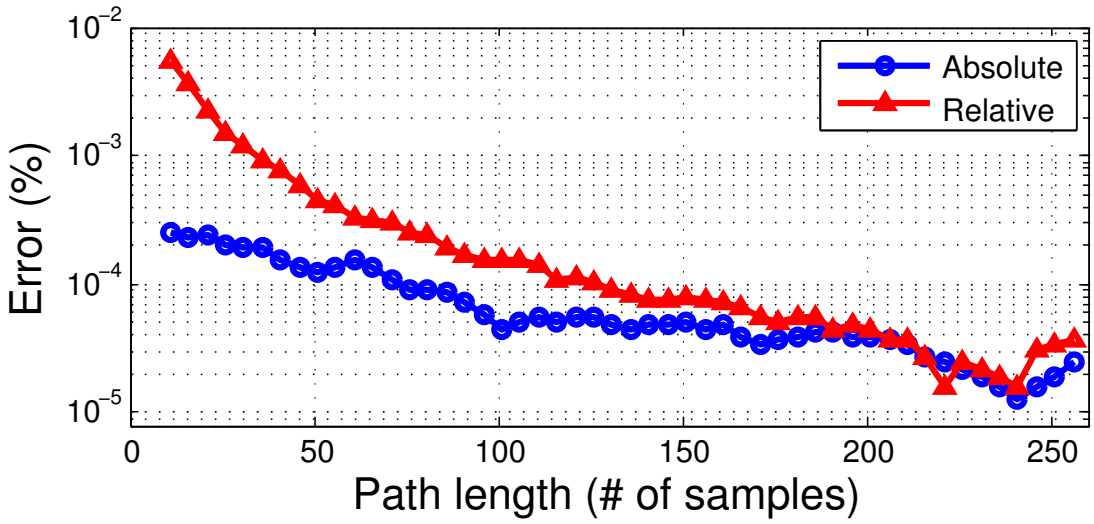
As more barometer data is collected, the probability of distinguishing the correct path from the set of all candidate paths increases. In other words, increased path length typically (though not always) leads to increased path uniqueness. This can be seen in Figure 6.9 for a city with low elevation variation (a particular 9.5 $km \times 9.5$ km block in Chicago) and one with high variation (a 9.5 $km \times 9.5$ km block in Seattle). For each iteration, we generate two random paths p_a and p_b with the same length. Path $\tilde{p}_a$ is generated by adding modeled barometer noise over p_a. A *confusion error* is defined when DTW fails to distinguish the correct, noisy path $\tilde{p}_a$ from the incorrect path p_b. Over multiple iterations of simulation, we observe that an increase in length decreases this confusion error.

6.5.3.2 Map Size

Surprisingly, there does not seem to be a high correlation between map size and path error, as shown in Figure 6.10. This is most likely due to variations in the underlying map's elevation—if absolute elevation estimates can be accurately made, increasing the map size is unlikely to add potential paths whose starting points are of a similar elevation *and* who exhibit similar relative elevation signatures.



(a) Chicago 9.5km x 9.5km



(b) Seattle 9.5km x 9.5km

Figure 6.9: Path length v.s. DTW Confusion Error

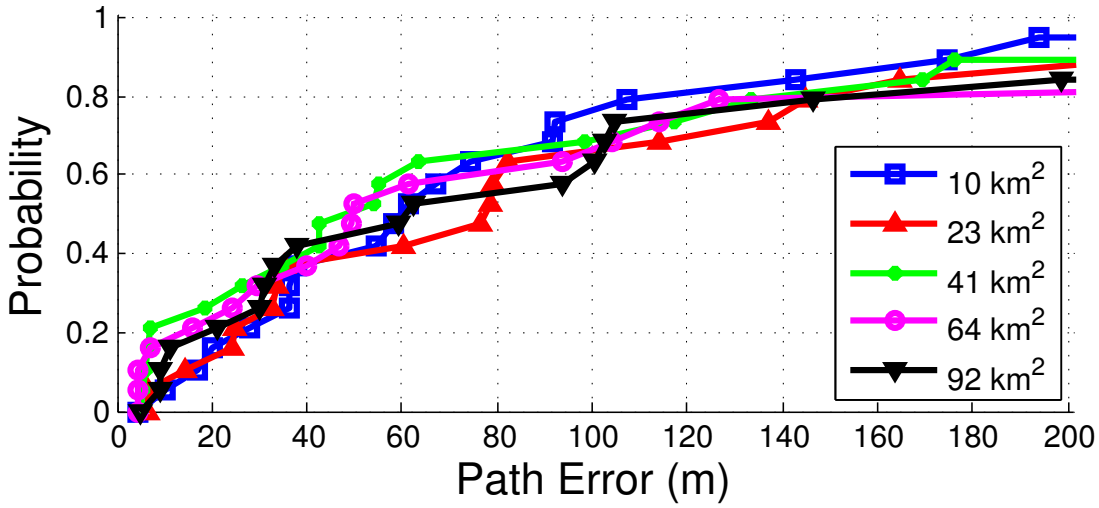


Figure 6.10: Path prediction errors vs. map size.

6.5.3.3 Geographical Landscape

The elevation variation of the underlying map also plays a significant role in the ability to accurately predict paths based on pressure. The variations in elevation for the 26 city maps tested in this work are shown sorted in Figure 6.11. Revisiting Figure 6.9, we see that high variation cities like Seattle have a much lower % Error than low variation cities like Chicago. This trend was also observed in general over the 26 cities studied in this paper. For example, Figure 6.12 shows the probability of confusing a given random path with any other path in a particular map. As the elevation variation of the underlying map increases (cross-listing again with Figure 6.11), there is an increased chance for path confusion, i.e. an increased probability that any given path may exhibit non-unique elevation signatures.

6.6 Discussion

We have shown through extensive tests in real driving experiments and simulated test-cases that it is often possible to predict a user's driving path with high accuracy from a time-series of barometer data. This prediction, however, is not without its limitations, as discussed be-

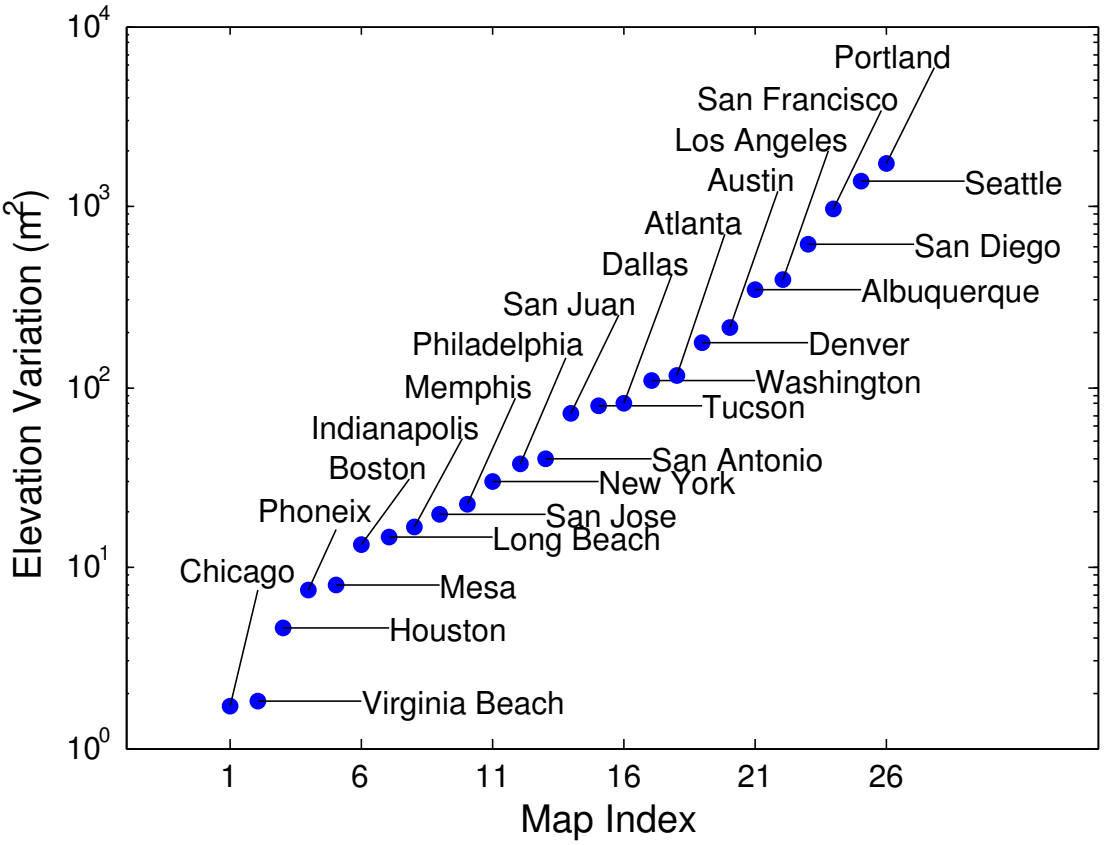


Figure 6.11: Elevation variations for sampled city maps.

low.

6.6.1 Prediction Robustness

As discussed in Section 6.3, the process of converting pressure to elevation depends largely on determining the pressure offset $\alpha(t)$. When this cannot be determined by nearby weather stations, the accuracy will be greatly decreased. This can be counteracted by including additional sensor data such as turn-detection using accelerometers, gyroscopes, and magnetometers. For example, the greedy path estimation algorithm can operate on relative elevation rather than absolute, obviating the need for $\alpha(t)$ entirely. If in addition to using relative elevation estimates we use information from mobile inertial sensors such as accelerometers

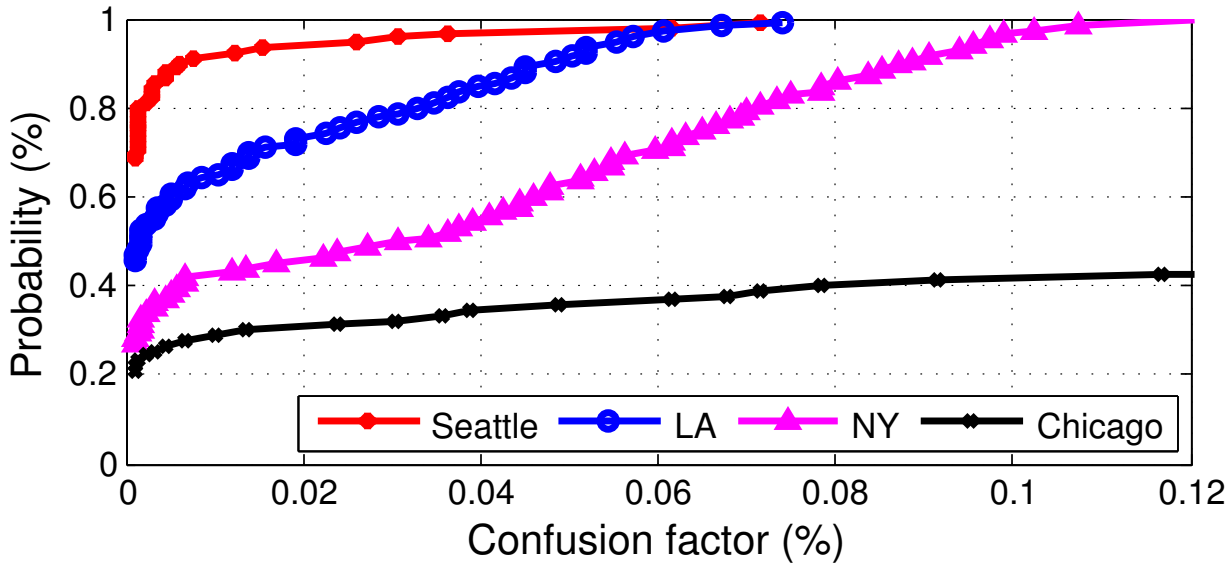


Figure 6.12: CDF of path confusion factors for cities of varying elevation variation.

and gyroscopes, the solver is still able to predict 50% of paths to within 500 m RMSE. This may be further improved by considering metrics such as driving speed estimation, driving mobility models, high traffic/highly probable routes, etc.

6.6.2 Privacy Implications

This work demonstrates real driving data in which 80% of tested paths can be predicted to within 200 m RMSE using barometer data alone. Additionally, this accuracy considers only single prediction instances—by combining data across multiple days it is likely that commonly traveled routes can be predicted with a much higher accuracy. With increasingly tight integration of social media applications in mobile devices, the potential privacy risks escalate from associating an *anonymous* user with an estimated driving path to associating a *specific*, personally identified user with a given driving path. When user anonymity, location, and behavior are compromised, the potential for breaches in security and privacy are all-the-more impressive. Additionally, mitigating privacy leaks through innocuous sensors like barometers may not be as simple as implementing stricter access controls—balancing

usability with utility is a non-trivial task, both technically and philosophically.

6.6.3 Future Work

This work demonstrates methods for accurately predicting driving paths based on barometric pressure data, resulting in a very low power method for large scale traffic analysis in emerging smart cities. The high correlation between pressure and elevation and the inclusion of these sensors on modern mobile devices raises a number of additional research questions. For example, can barometer pressure be leveraged to improve location-services in real-time to aid in spotty GPS coverage or to further reduce power consumption of location services? In addition, can similar methods to those discussed in this work be used to predict pedestrian paths in an unconstrained environment, such as for hikers? Finally, leveraging results describing pressure changes as a function of vertical motion indoors (i.e. elevators, escalators, and stairs) [MKM14], is it possible to infer which building or subset of buildings a user may be walking through based on unique patterns of floor changes, enhancing path estimation and occupancy detection algorithms? In future work, we plan to explore these questions in an attempt to further evaluate the benefits of a city-wide, distributed network of pressure sensors.

6.7 Summary

We have demonstrated methods by which barometric pressure data collected on mobile phones can be used to infer driving paths with surprisingly high accuracy. Specifically, we described both a greedy graph traversal approach and a dynamic-programming approach to estimating likely driving paths given pressure-based elevation estimates and a map of potential road segments. These methods leverage results from dynamic time warping literature to calculate a rate-independent similarity score between estimated and candidate path elevation signatures. Pressure data collected over a total of 4.6 hours and 150 km demonstrates that these algorithms can predict upwards of 90% of paths with less than 100 m error. Additionally, we illustrated the accuracy of these prediction methods for more than 500 simulated

test cases across 26 cities. The results of these simulations show that across all cities more than 70% of paths can be predicted to within an error of 200 m. We further evaluated the ability to estimate a user's driving path as a function of several variables, including length of barometer data, map size, and the elevation variance of the underlying map.

The results of the methods described in this paper serve to emphasize the importance of distributed networks of smart devices in emerging smart cities as well as the growing problem of personal data privacy, lending credence to research efforts focused on treating data in a privacy-preserving and security-aware manner. Finally, all sensor data and software described in this paper is open source and available at https://github.com/nesl/mercury.

CHAPTER 7

SpyCon: Context-Aware Adaptation Based Spyware

7.1 Motivation

Context-awareness is the capability of software systems to sense and adapt to the surrounding environment. Many contemporary mobile applications provide adaptations to users in different ways based on contexts such as locations [SSH15], connectivity states [SJP14], energy resources [EGG14], and proximity to other users and devices [JGC13], to name a few. As we stand on the edge of an explosion of data from these sensory devices, there has been a corresponding increase in applications [UMP14] and dedicated sensing frameworks [LYL10, JLY12, EWS15] targeting the integrity between contexts (sensing) and corresponding adaptations (actions).

Unfortunately, the same act of adapting to user context often leads to systems where increased sophistication comes at the expense of more privacy weaknesses. At the heart of mobile privacy is the notion that information collected from the physical world through sensors poses a significant privacy risk on *inferring* user sensitive information like behavior and location [ASB12, MVB12b, SZG14b, HMS15]. While there exists a recent body of work on identifying malicious apps which if granted access to sensory data can perform unwanted *inferences*, the question of whether a malicious app can still perform inferences *without having direct access to sensory data* remains unanswered. In other words, we ask the following questions: (1) Do context-based actions taken by authentic context-aware applications—which are granted access to sensory data—open side channels for malicious apps which do not have direct access to such data? That is, by monitoring actions triggered by authentic context-aware apps, can a malicious app still be able to perform unwanted inferences about

user sensitive information like behavior and location. (2) What software mechanisms can be deployed to detect and mitigate such privacy leak? Answering these questions becomes of critical importance in assessing the privacy of these context-aware applications.

In this paper, we introduce a new type of spyware that exploits the privacy leaks in context-aware adaptation which we call **SpyCon**.

7.1.1 Related Work

Context Monitoring Malware on Mobile Platforms: While mobile users benefit from sensing technologies, there are increasing privacy and security concerns. The permission systems on both Android and iOS become the first line of defense to protect users from leaking sensitive information. However, the traditional grant-all-or-none policy allows third-party apps to have all permissions [HHJ11]. Even worse, most users have trouble with realizing the potential privacy hazards after granting such permissions. For example, though seems innocuous, `ACCESS_WIFI_STATE` becomes a heavily privacy intrusive permission since local MAC address can serve as a unique device identifier [ACR14]. Felt et al. [FHE12] shows that as little as 17% of users pay attention to the permissions during app installation phase.

Different side-channel attacks have been proposed, for example, using inertial sensors and touch screen to infer user input such as passwords [HON12b, MVB12b, OHD12b, MVC11]. Besides, we witnessed how to exploit cellular signal strengths or air pressure for locations [MSV15, HMS15], gyroscope for eavesdropping conversations [MBN14b], system-level aggregate statistics for user's real world identity [ZDH13b], and the state of shared memory for foreground apps, and even, `activity` transition sequences [NYY15]. There is a trend that malicious apps are adapting to wearable devices [RGK11b]. For example, MoLe [WLR15] exploits the wrist motion derived from smartwatches to infer keystroke inputs. So far we've provided many examples showing "Your apps are watching you" [KT10] in a broad spectrum which a majority of users will never realize, and for sure "These aren't the droid you're looking for" [HHJ11].

Contrary to the aforementioned side-channel attacks, we consider a spyware *does not* have

access to sensor information like inertial or gyroscope sensors. Instead, a spyware can only monitor actions made by other apps, and these actions are triggered based on the changes in sensory data. Similar to the spyware considered in this work, [ZDH13b] demonstrates some information leaking channels in Android (e.g. phone speaker status) that a malicious app can monitor without any permission to acquire sensitive information about the user.

7.1.2 Paper Contribution

The primary innovation of this paper is to identify a new category of spyware apps. Specifically, we list the contributions of this paper below:

- We exploit a new side channel attack vector arising from monitoring changes of phone adaptations by context-aware applications. We call this new set of attacks a *context-aware adaptation based spyware*, or in short, SpyCon.
- We show a concrete instantiation of a SpyCon which can maliciously infer user's behavior by monitoring context-based adaptions. We assess the performance of the developed SpyCon through a one-month user study.

7.2 System Overview

This section introduces the Context-aware Adaptation based Spyware (SpyCon) and illustrates how it works and its potential negative outcomes. To this end, we show an exemplar of SpyCon app that stealthily learns user locations inferred by those adaptions made by other context-aware apps. Though we use location as an example, the same concept can be generalized and applied to collecting other types of sensitive user data.

Table 7.1: Phone settings (PS) considered in our apps.

PS	Description	PS	Description
R	Ringer mode	P	Wallpaper
H	Touch sound	D	Dialpad sound
W	Enable WiFi	A	Alarm volume
I	Ringer volume	M	Media volume
T	Display timeout	B	Screen brightness
V	Vibration on touch	L	Screen locking sound

7.2.1 Popular Phone Manager Apps

Location-based phone setting management is one of the most popular context-aware applications[1]. Due to their capability to adapt to user contexts, apps like Llama [lla], Tasker [tas], and Locale [loc] have gained more than one million downloads from Google Play Store. These context-aware apps can change *phone settings* such as ringer volume or screen brightness based on the current GPS location. Traditional configurations in these apps are, for instance, muting the ringer volume when a user is in a class or a conference room or enabling WiFi whenever the user enters home. Motivated by the popularity of these location-based context-aware apps, we choose user location as the sensitive data for which our SpyCon leaks.

7.2.2 Spyware Description

As described before, we are interested in designing a SpyCon that monitors changes in phone settings—which are triggered by a location-based context aware app—and uses these changes to leak user's location. We start by making the following two important remarks:

No user permissions: Many phone settings can be monitored without seeking user permissions. For example, SpyCon can easily get current screen brightness or alarm volume without user consent.

Ambiguity on setting changes: Manual adjustment can make changes in phone settings through physical buttons. Although SpyCon can not discriminate *a priori* between the

[1]By the time this paper was written, context-aware phone settings management applications ranked 3^{rd} in the Productivity category in the Android Developer Challenge [tas].

settings' change that are triggered by location change and by the changes that are manually performed, machine learning algorithms can be handy in discovering repetitive patterns in the data.

The operation of the designed SpyCon is divided into two phases as follows:

Logging: SpyCon monitors all the changes in phone settings and records a timestamped value upon a change is detected. A list of phone settings that we consider in our SpyCon is given in Table 7.1.

Data Mining: Once enough data is collected, SpyCon analyzes these data to discover repeated patterns and hence infers user's daily behavior. More details about the implemented data mining algorithm are given in Section 7.2.4 after we discuss the user study setup.

7.2.3 SpyCon User Study

We developed two applications, an authentic context-aware app and a SpyCon. Both apps are developed on Android 5.0.1 and run on Nexus 4 and Nexus 5. Totally we recruited 7 participants including 4 males and 3 females. We asked all participants to carry a phone with them for four weeks on which our apps are installed. Based on the data we collected during the user study, we perform clustering algorithm and explore what information we can mine maliciously.

7.2.3.1 Context-Based Adaptation Application

We developed an application that resembles popular context-based adaptation apps such as Tasker [tas] and Locale [loc]. Our app provides a friendly UI for users to define their profiles. A *profile* contains a trigger condition and a set of actions. A condition is specified by a fixed-radius circular geofence. The corresponding actions are media settings, such as adjusting screen brightness to 60% or setting ringer mode to vibration. The full phone

settings we considered are listed in Table 7.1. When a user enters a registered region, our app changes the phone settings accordingly. Our app maintains the golden output from our users by passively sensing their location. The *golden output* is packed as records; each record contains a timespan and a profile. The golden output is used for the evaluation of SpyCon.

7.2.3.2 SpyCon Application

We developed a SpyCon whose only task is to stay in the background and log phone settings *without any interaction* with all the other apps, including the previous one[2]. All of the settings collected by the SpyCon app can be accessed *without permissions* on Android OS.

However, we should mention that any SpyCon app may require the knowledge of the installed applications to know whether the phone settings—or any other context-based adaptations—are triggered as a consequence of context changes. For example, the SpyCon we describe here can not reason that the change in phone settings come from a change in user's location unless it knows a priori that this phone has a location-based application that adapts the settings based on location. This knowledge can be done in two steps. First, a malicious app needs to know what other apps are running on the same phone. This information can be easily retrieved by Android API *getInstalledApplications* which requires no permission. Second, the malicious app needs to know the nature of the context adaptions that are triggered by such apps. Typically, this can be known from the context-aware application store ("Google Play") page. For example Tasker [tas] and Locale [loc] apps specify that phone settings are adapted based on user location.

7.2.4 Experiment 1: Data Mining by Clustering

Revealing the semantics of user location, or equivalently, active profile sequences from phone settings is challenging since the profile and the phone settings do not have a one-to-one mapping. This lack of one-to-one correspondence is due to the following factors (1) users

[2]In the real world, this SpyCon can provide some functionality but collect data stealthily, which is a typical way a spyware hides its true intention.

can manually change some phone settings regardless of their location, (2) users usually set only a subset of the 12 settings (listed in Table 7.1), and (3) two different profiles may update different sets of phone settings, which means some settings from the first action may still exist after the second action is applied. Thus, we use a clustering technique to approach the user data mining problem, and in particular, we use k-means algorithm.

Deciding the number of clusters in k-means algorithm is known to be hard in general and is usually application dependent. Since our SpyCon does not know how many profiles are defined by users, we brute-forcedly set k to be any value between 2 through 7 (selected based on the maximum number of profiles defined by our participants). Our algorithm returns the clustering result with the highest silhouette score.

7.2.4.1 Critical Phone Settings

Inspired by how most unsupervised machine learning algorithms work, we implement a greedy algorithm to find dominant phone settings. The algorithm procedures are provided below:

1. Initialize the selected feature set $S = \phi$.
2. We examine every other setting f not in S by performing k-means with feature set $S \cup \{f\}$. The silhouette score h_f is computed accordingly.
3. Denote $\hat{h}$ as the maximum h_f from the previous step. If $\hat{h}$ is larger than previous silhouette score, then $S = S \cup \{f\}$ and go back to step 2. Otherwise, the algorithm terminates.

7.2.4.2 Privacy Implications

The clustering result of one participant in our study is demonstrated in Figure 7.1. Figure 7.1a shows the actual user profile changes across the day (the golden output as explained in Section 7.2.3.1). Figure 7.1b shows the k-means clustering result (using an adaptive number of clusters) and demonstrates similar patterns with the golden output in Figure 7.1a. Our

Table 7.2: Clustering accuracy of all users compared to the baseline accuracy by applying k-means using the settings from Table 7.1.

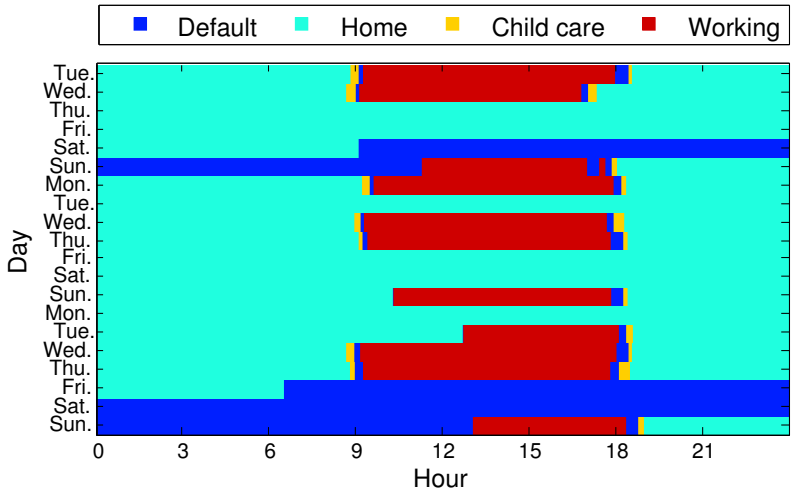
UID	# clusters using all features				Dominant features	
	base	2	3	2-7		
1	0.752	+0.189	+0.229	+0.191	+0.218	W,R,V
2	0.560	+0.172	+0.240	+0.183	+0.241	R,B,W
3	0.805	+0.129	+0.144	+0.136	+0.167	R
4	0.456	+0.373	+0.342	+0.356	+0.359	W,R,L
5	0.420	+0.240	+0.352	+0.240	+0.418	T,R,A
6	0.579	+0.044	+0.360	+0.044	+0.407	A,R,B,W
7	0.780	+0.150	+0.155	+0.150	+0.156	R,O
Avg.	**0.622**	**+0.185**	**+0.258**	**+0.182**	**+0.281**	

algorithm is able to capture subtle events, for example, learning that the user regularly went to a certain place (which turns out to be the child care) after leaving or before returning home, despite the portion of time this user spent in child care is short. Clustering result derived by dominant features from our feature selection algorithm is shown in Figure 7.1c. Figure 7.1b and Figure 7.1c indicate the ability of the developed SpyCon to reconstruct user context (switching profiles in this case) by just monitoring its side effect (changes in phone settings).

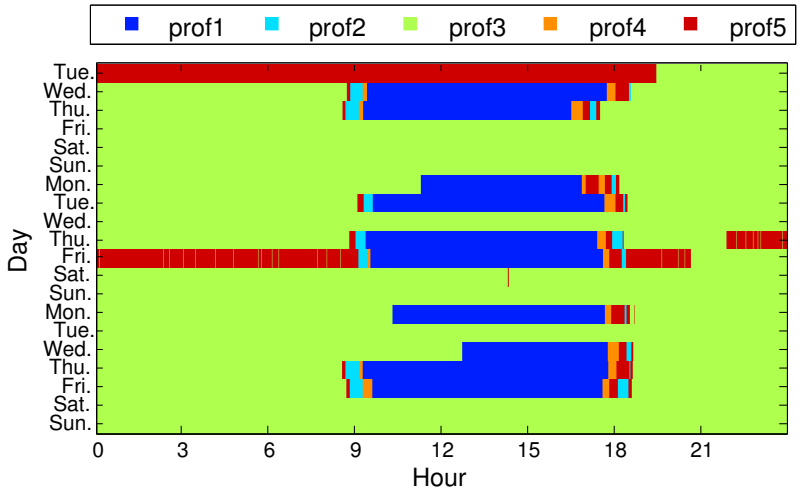
The overall accuracy of our clustering algorithm is reported in Table 7.2. We report the baseline accuracy which is the accuracy that the SpyCon can have without making any inference based on random guesses, such as assuming that the user is at home. The results in the rest of the columns are the additional information (the increase in accuracy) the SpyCon gains over the baseline accuracy if an inference is used using different number of clusters. The accuracy derived from dominant features is slightly higher because the feature selection algorithm excludes noisy features leading to a better result. We report dominant features for each user in the last column of Table 7.2. We observed that the ringer mode is a dominant feature in all the users' data.

In summary, this study shows that the designed SpyCon can estimate and learn with an average accuracy of 90.3% the user behavior, in particular:

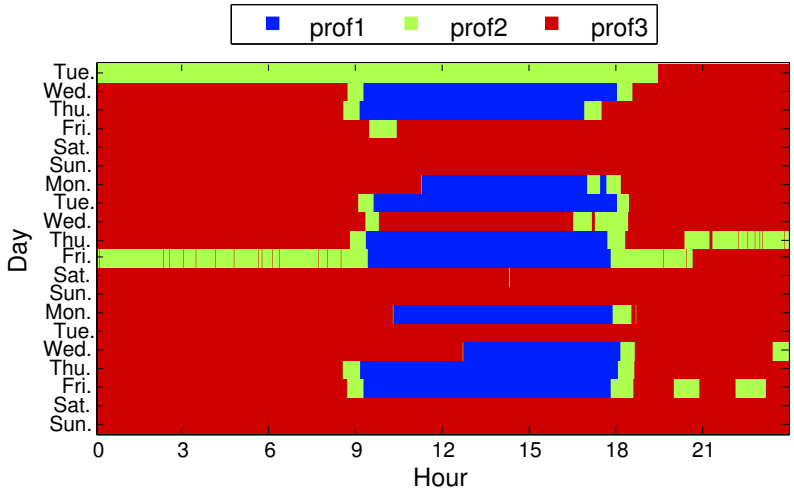
1. Average commuting time between home and work.



(a) Golden output.



(b) Clustered by all features.



(c) Clustered by dominant features.

Figure 7.1: One example of profile timeline from user #2.

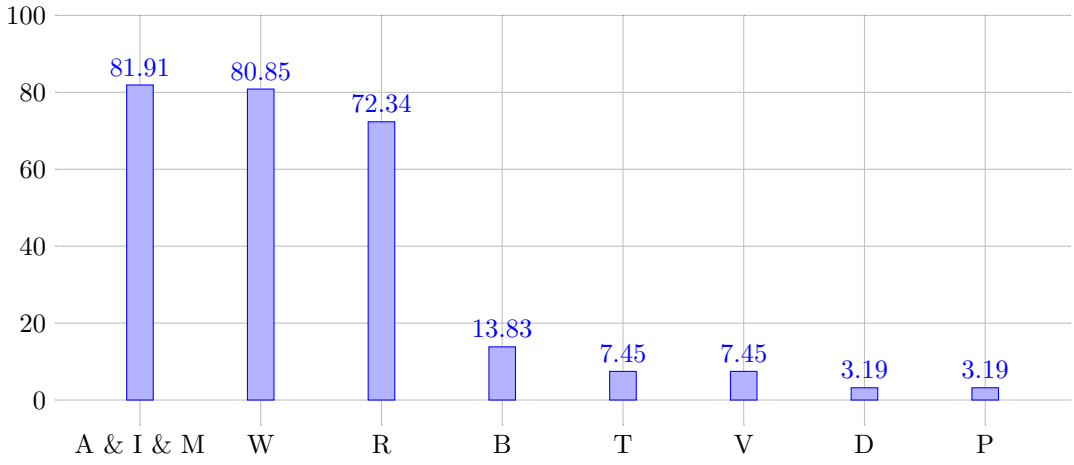


Figure 7.2: The percentage of apps from top 100 downloaded free Android apps that use the APIs in Table 7.1.

2. Average time spent at work and at home.
3. Weekend behavior, such as if a specific place is frequently visited on Sundays and average time spent at home.

7.2.5 Experiment 2: Significance of SpyCon

We performed static analysis on the top 100 downloaded free apps from the Google Play store to see if they *can* exploit this side channel. Our analysis shows that around 60% of the sampled applications use 4 or more APIs from Table 7.1. In particular, 80% of the apps checks if WiFi is turned on using the same API in the proposed phone-setting SpyCon. Similarly, 82% of the apps get the current audio volume. The results of the overall analysis are shown in Figure 7.2.

We examined the information possessed by these real applications, and due to the space limitation, we report the details of 5 of them in Table 7.3. Our results show that any of these 100 apps has information that is enough to exploit this side channel and act as an undetected SpyCon.

To conclude, in this section, we showed a full instantiation of a SpyCon that leaks infor-

Table 7.3: Clustering accuracy by information possessed by real apps on user's behavior whenever apps like Locale [loc] and Tasker [tas] are installed on the same phone.

App name	APIs used	Accuracy
Don't Tap the White Tile	RWAIMB	85.4%
Hulu Plus	RWAIM	86.1%
Skype	RHWAIM	83.0%
Power Battery	RWTBV	85.0%
Yahoo Mail	RW	81.9%

mation by monitoring an authentic context-aware application.

7.3 Discussion

7.3.1 Beyond phone settings

While the study in Section 7.2.4 shows that the proposed SpyCon is practical, SpyCon is not limited only to the phone settings listed in Table 7.1. In general, SpyCon can take advantage of any pair of `get` and `set` methods. More specifically, Android framework APIs has many examples of this kind of get-set pairs existing in `PackageManager`, `ClipboardManager`, and others.

7.4 Summary

We examined a new class of privacy threatening spyware that are designed to snoop around adaptations made by context-aware apps which we called SpyCon. We showed through user study that by monitoring the context-based adaptations triggered by authentic context-aware apps, a malicious app could infer information about the current user behavior and location. To exacerbate the situation, our experiments show that this new spyware is undetectable using off-the-shelf antivirus and moreover many of the top 100 downloadable free Android apps have access to enough information to extract sensitive information about user.

CHAPTER 8

Conclusion and Future Work

8.1 Conclusion

This dissertation attempted to bridge the gap between user engagement and existing sensing techniques. User engagement is equally essential for both product providers and users themselves. The utility of smartphones and wearables is maximized when people use them; users can enjoy the services provided by mobile devices only if the users are willing to explore the features and functionalities offered. Having a notion of user engagement in modern sensing computational systems helps prioritize tasks, allocate resources, and reach out to users at opportune moments. To this end, we exploited the performance-based observation approach to gauge user engagement. Following this definition, we explored how the advances of sensing techniques can increase user engagement and how user engagement can be modeled by sensor data from edge devices, and raised the privacy concerns that can hinder the trust between users and technologies, which consequently discourages user engagement.

We used a workout application to demonstrate how sensing techniques can optimize service utility, which in turn encourages people to engage with exercise activities. The workout app is composed of a ***MiLift*** system and a ***MyoBuddy*** system. MiLift features a single-device, automatic, and hardware-resource efficient weightlifting workout sensing technique. MyoBuddy offers the weight measurement algorithm, which completes the entire weightlifting exercise tracking pipeline. Our users rated the proposed workout tracking system, which is capable of accurately identifying exercise sessions and counting repetitions while minimizing battery drain, with an average overall score of 4.47 out of 5.

We studied how to model user engagement from sensor data by first conducting a crowd-

sourced user study called ***Nurture***. In the study, we developed both a generative simulator and an online interactive simulation to evaluate the capability and feasibility of using reinforcement learning to model interruptibility. In this pilot study, reinforcement learning demonstrated the adaptiveness and increased user responsiveness with an acceptable model converging time. Based on these insights, we developed a real system called ***Quick Question*** that aims to identify opportune moments to deliver microtasks in a natural, interfere-less, and in-the-wild setup. We compared the performance of the reinforcement learning approach with the supervised learning approach, and our findings suggest that while the average number of responses between both methods is commensurate, the reinforcement learning is more effective at avoiding dismissal of notifications and improves user experiences over time.

It is natural to assume that acquiring more sensory data leads to higher accuracy in modeling engagement, which in turn increases the user engagement. However, in the last part of this dissertation, we showcased that although some sensors seem harmless, they can still open side channels that allow attackers to access sensitive personal information and create great privacy concerns. We built ***GPSI*** and ***SpyCon*** systems to demonstrate that sensor-based side-channel attacks are practical. Such a privacy breach can significantly decrease user engagement when the trust between the sensing technology and human beings vanishes.

The implementations of the projects presented in this thesis are open source and available from the repositories listed below:

- MiLift: `https://github.com/nesl/WorkoutTracking`
- MyoBuddy: `https://github.com/nesl/MyoBuddy`
- Nurture: `https://github.com/nesl/Nurture-UbiTtention18`
- Quick Question: `https://github.com/nesl/EngagementService`
- GPSI: `https://github.com/nesl/mercury`

8.2 Future work

Below, I outline opportunities for further investigation on increasing user engagement and user experience enhancement.

8.2.1 Mobile-Application Initiated User Engagement Enhancement

User engagement can be modeled by considering user context, as extensively discussed in this dissertation. But application developers are always eager to engage their users. Such a contribution from the developer side is hindered by the huge gap between high-level application logic and low-level sensor data, i.e., implementing user context classifiers from scratch in order to infer the appropriateness of interrupting a user at a certain time is challenging for most app developers. As a result, most applications fall back on a simple periodic scheduling strategy to interact with users. We hereby point out a future research direction: How can we place application developers into the equation for user engagement modeling, making it not only consider user context, but also *application context*?

Ultimately, a high-level context-aware language to express when to reach out to people and what to expect from users should be offered to ease development efforts. Figure 8.1 attempts to illustrate the vision. We believe that to accomplish this goal, there are two challenges to be addressed. First, a programming language that bridges the semantic context and the raw sensory data has to be developed and also has to be sufficiently expressive. Second, a conflict resolver between application requirements and user preferences has to be studied. For example, sending a medication reminder is a legitimate motivation, but users may decline to follow the instruction. In contrast, applications that excessively deliver low-priority messages can reduce productivity, which is not desirable. We have drafted a complex-event based language, called ***Emu*** [HBN17], which aims to concisely and precisely describe when to engage users (see the example in Figure 8.2), but additional improvements are necessary to reach this vision.

Figure 8.1: Examples of specifying context triggers and action compliance in different applications from different domains. We envision that the system should provide a high-level context language to express the context triggers and the compliances, symbolized as the black icons.

8.2.2 On-Device Engagement Computation

Since the benefit of having an "engagement meter" is universal across all the mobile applications, we foresee the need for integrating user engagement measurement as part of the system services in mobile operating systems. A service offered at the system level requires high availability. Although this dissertation has explored a server-client architecture to gauge user engagement as demonstrated in the Quick Question system, it is for easing the prototyping effort and does not optimize for availability. For example, our client app does not receive engagement information during network outages. We argue that migrating the computation logic from the server side to local mobile phones is one of the future directions to explore,

```
Task taskBP = TaskBuilder.create()
  .repeat('every day')
  .when('walking for 1 hours
        or running for 15 minutes'
  .then('nearBPMachine within 1 hours')
  .notify('Measure blood pressure',
          PRIORITY_MEDIUM,
          'snooze 2 times', 'every 15 mins')
  .launch(bpActivity)
  .report(bpCallBack, 'timeout 2 hours')
  .startTask(BP_TASK_ID);
```

Figure 8.2: Example code of registering a task in Emu. The bold font are reserved keywords in Emu.

given the computational performance and hardware accelerators equipped in modern mobile phones (e.g., hardware-supported neural network computation[1,2]). We believe that computing user engagement locally can potentially reduce measurement processing time and battery consumption because network communication is no longer required. Additionally, it reduces the chance of personal information leaks because the data does not leave the phones. Consequently, it reduces privacy concerns and can positively affect user engagement. However, local computation increases memory usage. Hence, there will need to be further analysis in terms of overall system performance.

8.2.3 Toward a Seamless Engagement Modeling Experience

One challenge in user engagement modeling is that the process of modeling can decrease engagement. For example, in Quick Question, the learning agent learns when interrupting users will annoy them. This usually happens in the beginning of the learning process, e.g., within the first couple of days, since the learning agent starts to interact with users. We

[1]Neural Engine in iPhones - https://www.apple.com/iphone-xs/a12-bionic/

[2]Edge TPU by Google: https://cloud.google.com/edge-tpu/

observe that a model with low complexity can usually boost the modeling accuracy in a short span of time, but it saturates at a low accuracy after the warm-up period. In contrast, a high-complexity model achieves better accuracy, but it takes more time to converge. We think that a possible approach is to develop a hybrid model that benefits from the advantages of both models, and we believe that shortening the learning time necessary to ramp up the agent and reach a reasonable performance is yet another direction to explore.

REFERENCES

[AB04] Piotr D Adamczyk and Brian P Bailey. "If not now, when?: the effects of interruption at different moments within task execution." In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 271–278. ACM, 2004.

[ACR14] Jagdish Prasad Achara, Mathieu Cunche, Vincent Roca, and Aurélien Francillon. "Short paper: Wifileaks: Underestimated privacy implications of the ACCESS_WIFI_STATE Android permission." In *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*, pp. 231–236. ACM, 2014.

[AFS17a] S. Aminikhanghahi, R. Fallahzadeh, M. Sawyer, D. J. Cook, and L. B. Holder. "Thyme: Improving Smartphone Prompt Timing Through Activity Awareness." In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 315–322, Dec 2017.

[AFS17b] Samaneh Aminikhanghahi, Ramin Fallahzadeh, Matthew Sawyer, Diane J Cook, and Lawrence B Holder. "Thyme: improving smartphone prompt timing through activity awareness." In *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*, pp. 315–322. IEEE, 2017.

[AG06] Dinal Andreasen and Darren Gabbert. "Electromyographic switch navigation of power wheelchairs." In *Annual conference of the rehabilitation engineering and assistive technology society of North America*, 2006.

[AKK14] Fadel Adib, Zach Kabelac, Dina Katabi, and Robert C Miller. "3d tracking via body radio reflections." In *Proc. 11th USENIX NSDI*, 2014.

[and] "Composite sensor type summary - Android Developer." `https://source.android.com/devices/sensors/sensor-types.html#composite_sensor_type_summary`.

[app] "Apple - Apple Watch." `https://www.apple.com/watch/`.

[AR03] Mark J Arnold and Kristy E Reynolds. "Hedonic shopping motivations." *Journal of retailing*, **79**(2):77–95, 2003.

[ASB12] Adam J Aviv, Benjamin Sapp, Matt Blaze, and Jonathan M Smith. "Practicality of accelerometer side channels on smartphones." In *Proceedings of the 28th Annual Computer Security Applications Conference*, pp. 41–50. ACM, 2012.

[ASS10] Florian Alt, Alireza Sahami Shirazi, Albrecht Schmidt, Urs Kramer, and Zahid Nawaz. "Location-based crowdsourcing: extending crowdsourcing to the real world." In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, pp. 13–22. ACM, 2010.

[asu] "ASUS ZenWatch 2." https://www.asus.com/us/ZenWatch/ASUS_ZenWatch_2_WI501Q/.

[atl] "Atlas Wristband." https://www.atlaswearables.com/.

[ATP12] S. Aram, A. Troiano, and E. Pasero. "Environment sensing using smartphone." In *Sensors Applications Symposium (SAS), 2012 IEEE*, pp. 1–4, Feb 2012.

[AVD17] Yomna Abdelrahman, Eduardo Velloso, Tilman Dingler, Albrecht Schmidt, and Frank Vetere. "Cognitive heat: exploring the usage of thermal imaging to unobtrusively estimate cognitive load." *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **1**(3):33, 2017.

[ban] "Guided Workout: Microsoft Band." https://www.microsoft.com/microsoft-band/en-us/support/health-and-exercise/guided-workouts/.

[BBM14] Nikola Banovic, Christina Brant, Jennifer Mankoff, and Anind Dey. "ProactiveTasks: the short of mobile device use sessions." In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*, pp. 243–252. ACM, 2014.

[BBV13] Daniël Bossen, Michelle Buskermolen, Cindy Veenhof, Dinny de Bakker, and Joost Dekker. "Adherence to a web-based physical activity intervention for patients with knee and/or hip osteoarthritis: a mixed method study." *Journal of medical Internet research*, **15**(10), 2013.

[Bea70] Monroe C Beardsley. "The aesthetic point of view." *Metaphilosophy*, **1**(1):39–58, 1970.

[BK06] Brian P Bailey and Joseph A Konstan. "On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state." *Computers in human behavior*, **22**(4):685–708, 2006.

[BKP07] Nadia Bianchi-Berthouze, Whan Woong Kim, and Darshak Patel. "Does body movement engage you more in digital game play? and why?" In *International conference on affective computing and intelligent interaction*, pp. 102–113. Springer, 2007.

[bmp] "BMP280 Digital Pressure Sensor." https://ae-bst.resource.bosch.com/media/products/dokumente/bmp280/BST-BMP280-DS001-10.pdf. Accessed: 2015-03-04.

[Bre01] Leo Breiman. "Random forests." *Machine learning*, **45**(1):5–32, 2001.

[BRP06] MH Van den Berg, HK Ronday, AJ Peeters, EM Voogt-van Der Harst, M Munneke, FC Breedveld, and TPM Vliet Vlieland. "Engagement and satisfaction with an Internet-based physical activity intervention in patients with rheumatoid arthritis." *Rheumatology*, **46**(3):545–552, 2006.

[BTR15] Jelmer P Borst, Niels A Taatgen, and Hedderik van Rijn. "What makes interruptions disruptive?: A process-model account of the effects of the problem state bottleneck on task interruption and resumption." In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pp. 2971–2980. ACM, 2015.

[BWJ17] Nikola Banovic, Anqi Wang, Yanfeng Jin, Christie Chang, Julian Ramos, Anind Dey, and Jennifer Mankoff. "Leveraging Human Routine Models to Detect and Generate Human Behaviors." In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 6683–6694. ACM, 2017.

[CAZ10] Mick P Couper, Gwen L Alexander, Nanhua Zhang, Roderick JA Little, Noel Maddy, Michael A Nowak, Jennifer B McClure, Josephine J Calvi, Sharon J Rolnick, Melanie A Stopponi, et al. "Engagement and retention: measuring breadth and depth of participant use of an online intervention." *Journal of medical Internet research*, **12**(4), 2010.

[CBV15] Meredith A Case, Holland A Burwick, Kevin G Volpp, and Mitesh S Patel. "Accuracy of smartphone applications and wearable devices for tracking physical activity data." *JAMA*, **313**(6):625–626, 2015.

[CCC07] Keng-Hao Chang, Mike Y. Chen, and John Canny. "Tracking Free-weight Exercises." In *Proceedings of the 9th International Conference on Ubiquitous Computing*, Proc. ACM UbiComp, 2007.

[CCC09] Meng-Chieh Chiu, Shih-Ping Chang, Yu-Chen Chang, Hao-Hua Chu, Cheryl Chia-Hui Chen, Fei-Hsiu Hsiao, and Ju-Chun Ko. "Playful bottle: a mobile social persuasion system to motivate healthy water intake." In *Proceedings of the 11th international conference on Ubiquitous computing*, pp. 185–194. ACM, 2009.

[CCD13] Paul Cairns, Anna L Cox, Matthew Day, Hayley Martin, and Thomas Perryman. "Who but not where: The effect of social play on immersion in digital games." *International Journal of Human-Computer Studies*, **71**(11):1069–1077, 2013.

[CDS05] Michael L Cotterman, Lynn A Darby, and William A Skelly. "Comparison of muscle force production using the Smith machine and free weights for bench press and squat exercises." *The Journal of Strength & Conditioning Research*, **19**(1):169–176, 2005.

[CG12] Jaeyong Chung and Henry J Gardner. "Temporal presence variation in immersive computer games." *International Journal of Human-Computer Interaction*, **28**(8):511–529, 2012.

[CGF09] Helen Christensen, Kathleen M Griffiths, and Louise Farrer. "Adherence in internet interventions for anxiety and depression: systematic review." *Journal of medical Internet research*, **11**(2), 2009.

[CGH17] Saksham Chitkara, Nishad Gothoskar, Suhas Harish, Jason I Hong, and Yuvraj Agarwal. "Does this App Really Need My Location?: Context-Aware Privacy Management for Smartphones." *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **1**(3):42, 2017.

[CH10] Aaron Carroll and Gernot Heiser. "An Analysis of Power Consumption in a Smartphone." In *Proc. USENIX ATC*, 2010.

[Cha03] Erik Champion. "Applying game design theory to virtual heritage environments." In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pp. 273–274. ACM, 2003.

[CHS05] Edmund Cauza, Ursula Hanusch-Enserer, Barbara Strasser, Bernhard Ludvik, Sylvia Metz-Schimmerl, Giovanni Pacini, Oswald Wagner, Petra Georg, Rudolf Prager, Karam Kostner, et al. "The relative benefits of endurance and strength training on the metabolic factors and muscle function of people with type 2 diabetes mellitus." *Archives of physical medicine and rehabilitation*, **86**(8):1527–1533, 2005.

[CIT16] Carrie J Cai, Shamsi T Iqbal, and Jaime Teevan. "Chain reactions: The impact of order on microtask chains." In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 3143–3154. ACM, 2016.

[CL11] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: a library for support vector machines." *ACM Transactions on Intelligent Systems and Technology (TIST)*, **2**(3):27, 2011.

[CLN17] Itai Caspi, Gal Leibovich, Gal Novik, and Shadi Endrawis. "Reinforcement Learning Coach.", December 2017.

[CMP12] Gabe Cohn, Daniel Morris, Shwetak Patel, and Desney Tan. "Humantenna: using the body as an antenna for real-time whole-body interaction." In *Proc. ACM CHI*, 2012.

[CMP14] Gianluca Castelnuovo, Gian Mauro Manzoni, Giada Pietrabissa, Stefania Corti, Emanuele Maria Giusti, Enrico Molinari, and Susan Simpson. "Obesity and outpatient rehabilitation using mobile technologies: the potential mHealth approach." *Frontiers in psychology*, **5**, 2014.

[crf] "CRF++: Yet Another CRF toolkit." `https://taku910.github.io/crfpp/`.

[CSG13] Heng-Tze Cheng, Feng-Tso Sun, Martin Griss, Paul Davis, Jianguo Li, and Di You. "Nuactiv: Recognizing unseen new activities using semantic attribute-based learning." In *Proc. 11th ACM MobiSys*, 2013.

[Csi90] Mihaly Csikszentmihalyi. "Flow: The psychology of optimal performance.", 1990.

[CSW99] Peter Chapman, Sanjeebhan Selvarajah, and Jane Webster. "Engagement in multimedia training systems." In *Systems Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on*, pp. 9–pp. IEEE, 1999.

[CTI15] Justin Cheng, Jaime Teevan, Shamsi T Iqbal, and Michael S Bernstein. "Break it down: A comparison of macro-and microtasks." In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 4061–4064. ACM, 2015.

[CZW15] David Chu, Zengbin Zhang, Alec Wolman, and Nicholas Lane. "Prime: A Framework for Co-located Multi-device Apps." In *Proc. ACM UbiComp*, 2015.

[DBA06] Brian G Danaher, Shawn M Boles, Laura Akers, Judith S Gordon, and Herbert H Severson. "Defining participant exposure measures in Web-based health behavior change programs." *Journal of Medical Internet Research*, **8**(3), 2006.

[DDG15] Elizabeth Donovan, Pronabesh Das Mahapatra, Traci C Green, Emil Chiauzzi, Kimberly McHugh, and Amanda Hemm. "Efficacy of an online intervention to reduce alcohol-related risks among community college students." *Addiction Research & Theory*, **23**(5):437–447, 2015.

[DDS09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. Ieee, 2009.

[De 97] Carlo J De Luca. "The use of surface electromyography in biomechanics." *Journal of applied biomechanics*, **13**(2):135–163, 1997.

[DGL16] Thomas A. Dingus, Feng Guo, Suzie Lee, Jonathan F. Antin, Miguel Perez, Mindy Buchanan-King, and Jonathan Hankey. "Driver crash risk factors and prevalence evaluation using naturalistic driving data." *Proceedings of the National Academy of Sciences*, **113**(10):2636–2641, 2016.

[DKC15] Keith M Diaz, David J Krupka, Melinda J Chang, James Peacock, Yao Ma, Jeff Goldsmith, Joseph E Schwartz, and Karina W Davidson. "Fitbit®: An accurate and reliable device for wireless physical activity tracking." *International journal of cardiology*, **185**:138–140, 2015.

[DR85] Edward Deci and Richard M Ryan. *Intrinsic motivation and self-determination in human behavior*. Springer Science & Business Media, 1985.

[DSY15a] Han Ding, Longfei Shangguan, Zheng Yang, Jinsong Han, Zimu Zhou, Panlong Yang, Wei Xi, and Jizhong Zhao. "FEMO: A Platform for Free-weight Exercise Monitoring with RFIDs." In *Proc. 13th ACM SenSys*, 2015.

[DSY15b] Han Ding, Longfei Shangguan, Zheng Yang, Jinsong Han, Zimu Zhou, Panlong Yang, Wei Xi, and Jizhong Zhao. "FEMO: A Platform for Free-weight Exercise Monitoring with RFIDs." In *Proc. 13th ACM SenSys*, 2015.

[DT09] Van Diggelen and Frank Stephen Tromp. "A-GPS: Assisted GPS, GNSS, and SBAS." Boston: Artech House, 2009.

[EGG14] Salma Elmalaki, Mark Gottscho, Puneet Gupta, and Mani Srivastava. "A Case for Battery Charging-Aware Power Management and Deferrable Task Scheduling in Smartphones." In *6th Workshop on Power-Aware Computing and Systems (HotPower 14)*, Broomfield, CO, October 2014. USENIX Association.

[Enr08] Patrizia Tavella Enrico Bibbona, Gianna Panfilo. "The Ornstein-Uhlenbeck process as a model of a low pass filtered white noise." In *Metrologia*, Metrologia 45, 2008.

[ETS18] Salma Elmalaki, Huey-Ru Tsai, and Mani Srivastava. "Sentio: Driver-in-the-Loop Forward Collision Warning Using Multisample Reinforcement Learning." *Proceedings of the 16th ACM Conference on Embedded Network Sensor Systems*, 2018.

[EWS15] Salma Elmalaki, Lucas Wanner, and Mani Srivastava. "CAreDroid: Adaptation Framework for Android Context-Aware Applications." In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pp. 386–399. ACM, 2015.

[FGB11] Joel E Fischer, Chris Greenhalgh, and Steve Benford. "Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications." In *Proceedings of the 13th international conference on human computer interaction with mobile devices and services*, pp. 181–190. ACM, 2011.

[FHE12] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. "Android permissions: User attention, comprehension, and behavior." In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, p. 3. ACM, 2012.

[GF17] Nitesh Goyal and Susan R Fussell. "Intelligent Interruption Management using Electro Dermal Activity based Physiological Sensor for Collaborative Sensemaking." *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **1**(3):52, 2017.

[GLR14] Petko Georgiev, Nicholas D. Lane, Kiran K. Rachuri, and Cecilia Mascolo. "DSP.Ear: Leveraging Co-processor Support for Continuous Audio Sensing on Smartphones." In *Proc. ACM SenSys*, 2014.

[GNW14] Benjamin M Good, Max Nanis, Chunlei Wu, and Andrew I Su. "Microtask crowdsourcing for disease mention annotation in PubMed abstracts." In *Pacific Symposium on Biocomputing Co-Chairs*, pp. 282–293. World Scientific, 2014.

[goo] "Google Fit." `https://developers.google.com/fit/`.

[Goo15] Google. "Google Elevation API." `https://developers.google.com/maps/documentation/elevation/`, 2015. [Online; accessed 7-March-2015].

[GPL10] Santanu Guha, Kurt Plarre, Daniel Lissner, Somnath Mitra, Bhagavathy Krishna, Prabal Dutta, and Santosh Kumar. "AutoWitness: Locating and Tracking Stolen Property While Tolerating GPS and Radio Outages." In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pp. 29–42, New York, NY, USA, 2010. ACM.

[gps] "GlobalSat EM-506 GPS Module." `http://www.globalsat.com.tw/`. Accessed: 2015-09-12.

[GTM17] Kristjan Greenewald, Ambuj Tewari, Susan Murphy, and Predag Klasnja. "Action centered contextual bandits." In *Advances in neural information processing systems*, pp. 5977–5985, 2017.

[GTV14] Juan M García-Gómez, Isabel de la Torre-Díez, Javier Vicente, Montserrat Robles, Miguel López-Coronado, and Joel J Rodrigues. "Analysis of mobile health applications for a broad spectrum of consumers: A user experience approach." *Health informatics journal*, **20**(1):74–84, 2014.

[gym] "Gymwolf." `http://www.gymwolf.com/`.

[HBN17] Bo-Jhang Ho, Bharathan Balaji, Nima Nikzad, and Mani Srivastava. "Emu: engagement modeling for user studies." In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, pp. 959–964. ACM, 2017.

[HCD09] Tamara L Hayes, Kofi Cobbinah, Terry Dishongh, Jeffrey A Kaye, Janna Kimel, Michael Labhard, Todd Leen, Jay Lundell, Umut Ozertem, Misha Pavel, et al. "A study of medication-taking and unobtrusive, intelligent reminding." *Telemedicine and e-Health*, **15**(8):770–776, 2009.

[HCD12] Chia-Liang Hung, Jerome Chih-Lung Chou, and Chung-Ming Ding. "Enhancing mobile satisfaction through integration of usability and flow." *Engineering Management Research*, **1**(1):44, 2012.

[hea] "Apple HealthKit." `https://developer.apple.com/healthkit/`.

[HHJ11] Peter Hornyack, Seungyeop Han, Jaeyeon Jung, Stuart Schechter, and David Wetherall. "These aren't the droids you're looking for: retrofitting android to protect data from imperious applications." In *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 639–652. ACM, 2011.

[HKF10] Eija Haapalainen, SeungJun Kim, Jodi F Forlizzi, and Anind K Dey. "Psychophysiological measures for assessing cognitive load." In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pp. 301–310. ACM, 2010.

[HKK14] Elina Helander, Kirsikka Kaipainen, Ilkka Korhonen, and Brian Wansink. "Factors related to sustained use of a free mobile app for dietary self-monitoring with photography and peer feedback: retrospective cohort study." *Journal of medical Internet research*, **16**(4), 2014.

[HKO15] Rebecca J Haines-Saah, Mary T Kelly, John L Oliffe, and Joan L Bottorff. "Picture Me Smokefree: a qualitative study using social media and digital photography to engage young adults in tobacco reduction and cessation." *Journal of medical Internet research*, **17**(1), 2015.

[HMF15] Helen Henshaw, Abby McCormack, and Melanie Ann Ferguson. "Intrinsic and extrinsic motivation is associated with computer-based auditory training uptake, engagement, and adherence for people with hearing loss." *Frontiers in psychology*, **6**:1067, 2015.

[HMS15] Bo-Jhang Ho, Paul Martin, Prashanth Swaminathan, and Mani Srivastava. "From Pressure to Path: Barometer-based Vehicle Tracking." In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pp. 65–74. ACM, 2015.

[HMT15] László Harmat, Örjan de Manzano, Töres Theorell, Lennart Högman, H[a]kan Fischer, and Fredrik Ullén. "Physiological correlates of the flow experience during computer game playing." *International Journal of Psychophysiology*, **97**(1):1–7, 2015.

[HNT13a] Samuli Hemminki, Petteri Nurmi, and Sasu Tarkoma. "Accelerometer-based Transportation Mode Detection on Smartphones." In *Proc. ACM SenSys*, 2013.

[HNT13b] Samuli Hemminki, Petteri Nurmi, and Sasu Tarkoma. "Accelerometer-based Transportation Mode Detection on Smartphones." In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pp. 13:1–13:14, New York, NY, USA, 2013. ACM.

[HON12a] Jun Han, E. Owusu, L.T. Nguyen, A. Perrig, and J. Zhang. "ACComplice: Location inference using accelerometers on smartphones." In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pp. 1–9, Jan 2012.

[HON12b] Jun Han, Emmanuel Owusu, Le T Nguyen, Adrian Perrig, and Joy Zhang. "Accomplice: Location inference using accelerometers on smartphones." In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pp. 1–9. IEEE, 2012.

[HXZ13] Tian Hao, Guoliang Xing, and Gang Zhou. "iSleep: Unobtrusive Sleep Quality Monitoring Using Smartphones." In *Proc. 11th ACM SenSys*, 2013.

[IAZ05] Shamsi T Iqbal, Piotr D Adamczyk, Xianjun Sam Zheng, and Brian P Bailey. "Towards an index of opportunity: understanding changes in mental workload

during task execution." In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 311–320. ACM, 2005.

[IB10] Shamsi T Iqbal and Brian P Bailey. "Oasis: A framework for linking notification delivery to the perceptual structure of goal-directed tasks." *ACM Transactions on Computer-Human Interaction (TOCHI)*, **17**(4):15, 2010.

[jah] "jahmm: An implementation of Hidden Markov Models in Java." `https://code.google.com/p/jahmm/`.

[JCC08] Charlene Jennett, Anna L Cox, Paul Cairns, Samira Dhoparee, Andrew Epps, Tim Tijs, and Alison Walton. "Measuring and defining the experience of immersion in games." *International journal of human-computer studies*, **66**(9):641–661, 2008.

[jef] "JEFIT Workout Tracker." `https://www.jefit.com/`.

[Jen00] Morgan Jennings. "Theory and models for creating engaging and immersive ecommerce websites." In *Proceedings of the 2000 ACM SIGCPR conference on Computer personnel research*, pp. 77–85. ACM, 2000.

[JGC13] Junghyun Jun, Yu Gu, Long Cheng, Banghui Lu, Jun Sun, Ting Zhu, and Jianwei Niu. "Social-loc: Improving indoor localization with social sensing." In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, p. 14. ACM, 2013.

[JLY12] Younghyun Ju, Youngki Lee, Jihyun Yu, Chulhong Min, Insik Shin, and Junehwa Song. "SymPhoney: a coordinated sensing flow execution engine for concurrent mobile sensing applications." In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pp. 211–224. ACM, 2012.

[KAL15] Haik Kalantarian, Nabil Alshurafa, Tuan Le, and Majid Sarrafzadeh. "Non-invasive detection of medication adherence using a digital smart necklace." In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*, pp. 348–353. IEEE, 2015.

[Kap95] Leon A Kappelman. "Measuring user involvement: a diffusion of innovation perspective." *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, **26**(2-3):65–86, 1995.

[KCS08] Aniket Kittur, E Chi, and Bongwon Suh. "Crowdsourcing for usability: Using micro-task markets for rapid, remote, and low-cost user measurements." *Proc. CHI 2008*, 2008.

[KLL10] Adil Mehmood Khan, Young-Koo Lee, Sungyoung Y Lee, and Tae-Seong Kim. "A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer." *Information Technology in Biomedicine, IEEE Transactions on*, **14**(5):1166–1172, 2010.

[KS15] H Kalantarian and M Sarrafzadeh. "A smartwatch-based system for audio-based monitoring of dietary habits." In *The Fifth International Conference on Ambient Computing, Applications, Services and Technologies*, 2015.

[KSK11] Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E Kraut. "Crowdforge: Crowdsourcing complex work." In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 43–52. ACM, 2011.

[KT10] SCOTT THURMand YUKARI IWATANI Kane and Scott Thurm. "Your Apps Are Watching You." *WALL ST. J.(Dec. 17, 2010), http://on. wsj. com/wq7Wiw*, 2010.

[Lau13] Brenda Laurel. *Computers as theatre*. Addison-Wesley, 2013.

[LFN15] Kyungmin Lee, Jason Flinn, and Brian Noble. "The case for operating system management of user attention." In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pp. 111–116. ACM, 2015.

[LFR12] Hong Lu, Denise Frauendorfer, Mashfiqui Rabbi, Marianne Schmid Mast, Gokul T Chittaranjan, Andrew T Campbell, Daniel Gatica-Perez, and Tanzeem Choudhury. "StressSense: Detecting stress in unconstrained acoustic environments using smartphones." In *Proc. ACM UbiComp*, 2012.

[LGM15] Nicholas D. Lane, Petko Georgiev, Cecilia Mascolo, and Ying Gao. "ZOE: A Cloud-less Dialog-enabled Continuous Sensing Wearable Exploiting Heterogeneous Computation." In *Proc. 13th ACM MobiSys*, 2015.

[LGQ15] Nicholas D Lane, Petko Georgiev, and Lorena Qendro. "DeepEar: robust smartphone audio sensing in unconstrained acoustic environments using deep learning." In *Proc. ACM UbiComp*, 2015.

[lgw] "LG G Watch R." `http://www.lg.com/us/smart-watches/lg-W110-lg-watch-r`.

[Lin07] Judy Chuan-Chuan Lin. "Online stickiness: its antecedents and effect on purchasing intention." *Behaviour & information technology*, **26**(6):507–516, 2007.

[LKA15] Seungwoo Lee, Yungeun Kim, Daye Ahn, Rhan Ha, Kyoungwoo Lee, and Hojung Cha. "Non-obstructive room-level locating system in home environments using activity fingerprints from smartwatch." In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 939–950. ACM, 2015.

[lla] "Llama - Location Profiles Application." `https://play.google.com/store/apps/details?id=com.kebab.Llama&hl=en`. [Online; accessed 9-Mar-2016].

[LLL13] Robert LiKamWa, Yunxin Liu, Nicholas D Lane, and Lin Zhong. "Moodscope: Building a mood sensor from smartphone usage patterns." In *Proc. 11th ACM MobiSys*, 2013.

[LLN11] Sungyoung Lee, Hung Xuan Le, Hung Quoc Ngo, Hyoung Il Kim, Manhyung Han, Young-Koo Lee, et al. "Semi-Markov conditional random fields for accelerometer-based activity recognition." *Applied Intelligence*, **35**(2):226–241, 2011.

[loc] "Locale Application." `https://play.google.com/store/apps/details?id=com.twofortyfouram.locale&hl=en`. [Online; accessed 9-Mar-2016].

[LPH12] Jie Liu, Bodhi Priyantha, Ted Hart, Heitor S. Ramos, Antonio A. F. Loureiro, and Qiang Wang. "Energy Efficient GPS Sensing with Cloud Offloading." In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, pp. 85–98, New York, NY, USA, 2012. ACM.

[LPS10] Izabella Lejbkowicz, Tamar Paperna, Nili Stein, Sara Dishon, and Ariel Miller. "Internet usage by patients with multiple sclerosis: implications to participatory medicine and personalized healthcare." *Multiple sclerosis international*, **2010**, 2010.

[LSL12] I-Min Lee, Eric J Shiroma, Felipe Lobelo, Pekka Puska, Steven N Blair, Peter T Katzmarzyk, Lancet Physical Activity Series Working Group, et al. "Effect of physical inactivity on major non-communicable diseases worldwide: an analysis of burden of disease and life expectancy." *The lancet*, **380**(9838):219–229, 2012.

[LSN14] Nuno Luz, Nuno Silva, and Paulo Novais. "Generating human-computer microtask workflows from domain ontologies." In *International Conference on Human-Computer Interaction*, pp. 98–109. Springer, 2014.

[LTA14] Thomas D LaToza, W Ben Towne, Christian M Adriano, and André Van Der Hoek. "Microtask programming: Building software with a crowd." In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pp. 43–54. ACM, 2014.

[LYL10] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. "The Jigsaw continuous sensing engine for mobile phone applications." In *Proceedings of the 8th ACM conference on embedded networked sensor systems*, pp. 71–84. ACM, 2010.

[map] "MapMyRun." `http://www.mapmyrun.com/`.

[MBM16] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. "Asynchronous methods for deep reinforcement learning." In *International conference on machine learning*, pp. 1928–1937, 2016.

[MBN14a] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. "Gyrophone: Recognizing Speech from Gyroscope Signals." In *23rd USENIX Security Symposium (USENIX Security 14)*, pp. 1053–1067, San Diego, CA, August 2014. USENIX Association.

[MBN14b] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. "Gyrophone: Recognizing speech from gyroscope signals." In *23rd USENIX Security Symposium (USENIX Security 14)*, pp. 1053–1067, 2014.

[MCD18] Daniel Moldovan, Georgiana Copil, and Schahram Dustdar. "Elastic systems: Towards cyber-physical ecosystems of people, processes, and things." *Computer Standards & Interfaces*, **57**:76 – 82, 2018.

[MCH10] H Mahmassani, R Chen, Yun Huang, Dmitri Williams, and Noshir Contractor. "Time to play? Activity engagement in multiplayer online role-playing games." *Transportation Research Record: Journal of the Transportation Research Board*, (2157):129–137, 2010.

[ME12] A. B. M. Musa and Jakob Eriksson. "Tracking Unmodified Smartphones Using Wi-fi Monitors." In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, pp. 281–294, New York, NY, USA, 2012. ACM.

[MF94] Steven T McCaw and Jeffrey J Friday. "A Comparison of Muscle Activity Between a Free Weight and Machine Bench Press." *The Journal of Strength & Conditioning Research*, **8**(4):259–264, 1994.

[MGK08] Gloria Mark, Daniela Gudith, and Ulrich Klocke. "The cost of interrupted work: more speed and stress." In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 107–110. ACM, 2008.

[MHM16] Abhinav Mehrotra, Robert Hendley, and Mirco Musolesi. "PrefMiner: mining user's preferences for intelligent mobile notification management." In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 1223–1234. ACM, 2016.

[MKF14] Rosa Mikeal Martey, Kate Kenski, James Folkestad, Laurie Feldman, Elana Gordis, Adrienne Shaw, Jennifer Stromer-Galley, Ben Clegg, Hui Zhang, Nissim Kaufman, et al. "Measuring game engagement: multiple methods and construct complexity." *Simulation & Gaming*, **45**(4-5):528–547, 2014.

[MKM14] Kartik Muralidharan, Azeem Javed Khan, Archan Misra, Rajesh Krishna Balan, and Sharad Agarwal. "Barometric Phone Sensors: More Hype Than Hope!" In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, HotMobile '14, pp. 12:1–12:6, New York, NY, USA, 2014. ACM.

[MKS15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. "Human-level control through deep reinforcement learning." *Nature*, **518**(7540):529, 2015.

[MLK16] Akhil Mathur, Nicholas D Lane, and Fahim Kawsar. "Engagement-aware computing: Modelling user engagement from mobile contexts." In *Proceedings of the*

2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pp. 622–633. ACM, 2016.

[MLN15] Frank Mokaya, Roland Lucas, Hae Young Noh, and Pei Zhang. "Myovibe: Vibration based wearable muscle activation detection in high mobility exercises." In *Proc. ACM UbiComp*, 2015.

[MLN16] Frank Mokaya, Roland Lucas, Hae Young Noh, and Pei Zhang. "Burnout: A Wearable System for Unobtrusive Skeletal Muscle Fatigue Estimation." In *Proc. 15th ACM/IEEE IPSN*, 2016.

[MLZ11] Hendrika Meischke, Paula Lozano, Chuan Zhou, Michelle M Garrison, and Dimitri Christakis. "Engagement in ?my child's asthma?, an interactive web-based pediatric asthma management intervention." *International journal of medical informatics*, **80**(11):765–774, 2011.

[MM17a] Abhinav Mehrotra and Mirco Musolesi. "Intelligent Notification Systems: A Survey of the State of the Art and Research Challenges." *CoRR*, **abs/1711.10171**, 2017.

[MM17b] Abhinav Mehrotra and Mirco Musolesi. "Intelligent Notification Systems: A Survey of the State of the Art and Research Challenges." *arXiv preprint arXiv:1711.10171*, 2017.

[MMH15] Abhinav Mehrotra, Mirco Musolesi, Robert Hendley, and Veljko Pejovic. "Designing content-driven intelligent notification mechanisms for mobile applications." In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 813–824. ACM, 2015.

[mota] "Moto 360." `http://www.motorola.com/us/products/moto-360`.

[motb] "Moto 360 Sport." `https://www.motorola.com/us/products/moto-360-sport`.

[mov] "Moves app." `https://www.moves-app.com/`.

[MPA14] Bobak Jack Mortazavi, Mohammad Pourhomayoun, Gabriel Alsheikh, Nabil Alshurafa, Sunghoon Ivan Lee, and Majid Sarrafzadeh. "Determining the Single Best Axis for Exercise Repetition Recognition and Counting on SmartWatches." In *Proc. 11th IEEE BSN*, 2014.

[MPR08] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee. "Nericell: Rich Monitoring of Road and Traffic Conditions Using Mobile Smartphones." In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pp. 323–336, New York, NY, USA, 2008. ACM.

[MPV16] Abhinav Mehrotra, Veljko Pejovic, Jo Vermeulen, Robert Hendley, and Mirco Musolesi. "My phone and me: understanding people's receptivity to mobile notifications." In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pp. 1021–1032. ACM, 2016.

[msb] "Microsoft Band." http://www.microsoft.com/microsoft-band/en-us.

[MSG14] Dan Morris, T. Scott Saponas, Andrew Guillory, and Ilya Kelner. "RecoFit: Using a Wearable Sensor to Find, Recognize, and Count Repetitive Exercises." In *Proc. 32nd ACM CHI*, 2014.

[msh] "Microsoft Health." https://www.microsoft.com/microsoft-health/.

[MSS06] Uwe Maurer, Asim Smailagic, Daniel P Siewiorek, and Michael Deisher. "Activity recognition and monitoring using multiple sensors on different body positions." In *Proc. IEEE BSN*, 2006.

[MSV15] Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. "PowerSpy: Location Tracking Using Mobile Device Power Analysis." In *24th USENIX Security Symposium (USENIX Security 15)*, pp. 785–800, Washington, D.C., August 2015. USENIX Association.

[MVB12a] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. "Tapprints: Your Finger Taps Have Fingerprints." In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pp. 323–336, New York, NY, USA, 2012. ACM.

[MVB12b] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. "Tapprints: your finger taps have fingerprints." In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 323–336. ACM, 2012.

[MVC11] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. "(sp) iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers." In *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 551–562. ACM, 2011.

[MVP15] Abhinav Mehrotra, Jo Vermeulen, Veljko Pejovic, and Mirco Musolesi. "Ask, but don't interrupt: the case for interruptibility-aware mobile experience sampling." In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pp. 723–732. ACM, 2015.

[MVS15] Verena Majuntke, Sebastian VanSyckel, Dominik Schafer, Christian Krupitzer, Gregor Schiele, and Christian Becker. "COMITY: coordinated application adaptation in multi-platform pervasive systems." In *Proc. of IEEE PerCom*, 2015.

[MWV13] Elizabeth Murray, Ian R White, Mira Varagunam, Christine Godfrey, Zarnie Khadjesari, and Jim McCambridge. "Attrition revisited: adherence and retention in a web-based alcohol trial." *Journal of medical Internet research*, **15**(8), 2013.

[myo] "Myo Band." https://www.myo.com/.

[Nat12] Suman Nath. "ACE: exploiting correlation for energy-efficient and continuous context sensing." In *Proc. 10th ACM MobiSys*, 2012.

[NDA13] Shahriar Nirjon, Robert F. Dickerson, Philip Asare, Qiang Li, Dezhi Hong, John A. Stankovic, Pan Hu, Guobin Shen, and Xiaofan Jiang. "Auditeur: A Mobile-cloud Service Platform for Acoustic Event Detection on Smartphones." In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pp. 403–416, 2013.

[NDH10] Ehsan Nazerfard, Barnan Das, Lawrence B Holder, and Diane J Cook. "Conditional random fields for activity recognition in smart environments." In *Proc. ACM Health Informatics*, 2010.

[NGG15] Shahriar Nirjon, Jeremy Gummeson, Dan Gelb, and Kyu-Han Kim. "TypingRing: A Wearable Ring Platform for Text Input." In *Proc. 13th ACM MobiSys*, 2015.

[NHS15] Inbal Nahum-Shani, Eric B Hekler, and Donna Spruijt-Metz. "Building health behavior models to guide the development of just-in-time adaptive interventions: A pragmatic framework." *Health Psychology*, **34**(S):1209, 2015.

[nik] "Nike+." `http://www.nike.com/us/en_us/c/nike-plus`.

[noa] "National Oceanic and Atmospheric Administration (NOAA) National Climatic Data Center." `http://www.ncdc.noaa.gov`. Accessed: 2015-03-05.

[nur] "Anonymous due to double-blind policy.".

[NYY15] Yuhong Nan, Min Yang, Zhemin Yang, Shunfan Zhou, Guofei Gu, and XiaoFeng Wang. "UIPicker: User-Input Privacy Identification in Mobile Applications." In *24th USENIX Security Symposium (USENIX Security 15)*, pp. 993–1008, Washington, D.C., August 2015. USENIX Association.

[OHD12a] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. "ACCessory: Password Inference Using Accelerometers on Smartphones." In *Proceedings of the Twelfth Workshop on Mobile Computing Systems and Applications*, HotMobile '12, pp. 9:1–9:6, New York, NY, USA, 2012. ACM.

[OHD12b] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. "ACCessory: password inference using accelerometers on smartphones." In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, p. 9. ACM, 2012.

[Ope15] OpenStreetMap. "Main Page — OpenStreetMap Wiki,." `http://wiki.openstreetmap.org`, 2015. [Online; accessed 7-March-2015].

[opt] "OptiTrack Prime 13." `https://www.optitrack.com/products/prime-13/`.

[ORN15] Tadashi Okoshi, Julian Ramos, Hiroki Nozaki, Jin Nakazawa, Anind K Dey, and Hideyuki Tokuda. "Reducing users' perceived mental effort due to interruptive notifications in multi-device mobile environments." In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 475–486. ACM, 2015.

[OST14] Robin Wentao Ouyang, Mani Srivastava, Alice Toniolo, and Timothy J. Norman. "Truth Discovery in Crowdsourced Detection of Spatial Events." In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pp. 461–470, New York, NY, USA, 2014. ACM.

[OT08] Heather L O'Brien and Elaine G Toms. "What is user engagement? A conceptual framework for defining user engagement with technology." *Journal of the American society for Information Science and Technology*, **59**(6):938–955, 2008.

[OTD14] Peter Organisciak, Jaime Teevan, Susan Dumais, Robert C Miller, and Adam Tauman Kalai. "A crowd of your own: Crowdsourcing for on-demand personalization." In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[OTT17] Tadashi Okoshi, Kota Tsubouchi, Masaya Taji, Takanori Ichikawa, and Hideyuki Tokuda. "Attention and engagement-awareness in the wild: A large-scale study with adaptive notifications." In *Pervasive Computing and Communications (PerCom), 2017 IEEE International Conference on*, pp. 100–110. IEEE, 2017.

[PBW16] Olga Perski, Ann Blandford, Robert West, and Susan Michie. "Conceptualising engagement with digital behaviour change interventions: a systematic review using principles from critical interpretive synthesis." *Translational behavioral medicine*, **7**(2):254–267, 2016.

[PC12] Josée Poirier and Nathan K Cobb. "Social influence as a driver of engagement in a web-based health intervention." *Journal of medical Internet research*, **14**(1), 2012.

[PCD14] Martin Pielot, Karen Church, and Rodrigo De Oliveira. "An in-situ study of mobile phone notifications." In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*, pp. 233–242. ACM, 2014.

[PCK17] Martin Pielot, Bruno Cardoso, Kleomenis Katevas, Joan Serrà, Aleksandar Matic, and Nuria Oliver. "Beyond interruptibility: Predicting opportune moments to engage mobile phone users." *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **1**(3):91, 2017.

[PDP15] Martin Pielot, Tilman Dingler, Jose San Pedro, and Nuria Oliver. "When attention is not scarce-detecting boredom from mobile phone usage." In *Proceedings*

of the 2015 ACM international joint conference on pervasive and ubiquitous computing, pp. 825–836. ACM, 2015.

[PFB00] Michael L Pollock, Barry A Franklin, Gary J Balady, Bernard L Chaitman, Jerome L Fleg, Barbara Fletcher, Marian Limacher, Ileana L Piña, Richard A Stein, Mark Williams, et al. "Resistance exercise in individuals with and without cardiovascular disease benefits, rationale, safety, and prescription an advisory from the committee on exercise, rehabilitation, and prevention, council on clinical cardiology, American Heart Association." *Circulation*, **101**(7):828–833, 2000.

[PGG13] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. "Whole-home gesture recognition using wireless signals." In *Proc. 19th ACM MobiCom*, 2013.

[PGR08] Kevin Patrick, William G Griswold, Fred Raab, and Stephen S Intille. "Health and the mobile phone." *American journal of preventive medicine*, **35**(2):177–181, 2008.

[PHK13] Igor Pernek, Karin Anna Hummel, and Peter Kokol. "Exercise repetition detection for resistance training based on smartphones." *Personal and ubiquitous computing*, **17**(4):771–782, 2013.

[PHM17] Aditya Ponnada, Caitlin Haynes, Dharam Maniar, Justin Manjourides, and Stephen Intille. "Microinteraction Ecological Momentary Assessment Response Rates: Effect of Microinteractions or the Smartwatch?" *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **1**(3):92, 2017.

[PK11] Duck Gun Park and Hee Chan Kim. "Muscleman: Wireless input device for a fighting action game based on the EMG signal and acceleration of the human forearm." In *International Symposium on Neural Networks. ISNN*, 2011.

[PLJ10] Namkee Park, Kwan Min Lee, Seung-A Annie Jin, and Sukhee Kang. "Effects of pre-game stories on feelings of presence and evaluation of computer games." *International Journal of Human-Computer Studies*, **68**(11):822–833, 2010.

[PLL11] Bodhi Priyantha, Dimitrios Lymberopoulos, and Jie Liu. "Littlerock: Enabling energy-efficient continuous sensing on mobile phones." *Pervasive Computing, IEEE*, **10**(2):12–15, 2011.

[PM14] Veljko Pejovic and Mirco Musolesi. "InterruptMe: designing intelligent prompting mechanisms for pervasive applications." In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 897–908. ACM, 2014.

[PM15] Veljko Pejovic and Mirco Musolesi. "Anticipatory mobile computing: A survey of the state of the art and research challenges." *ACM Computing Surveys (CSUR)*, **47**(3):47, 2015.

[poc] "Step tracking made easy with the M7." http://blog.runkeeper.com/326/step-tracking-made-easier-with-the-m7/.

[PPC12] Jun-geun Park, Ami Patel, Dorothy Curtis, Seth Teller, and Jonathan Ledlie. "Online pose classification and walking speed estimation using handheld devices." In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pp. 113–122, 2012.

[PVG11a] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research*, **12**:2825–2830, 2011.

[PVG11b] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research*, **12**:2825–2830, 2011.

[PVP18] Martin Pielot, Amalia Vradi, and Souneil Park. "Dismissed!: a detailed exploration of how mobile phone users handle push notifications." In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*, p. 3. ACM, 2018.

[RBG16] Anna N Rafferty, Emma Brunskill, Thomas L Griffiths, and Patrick Shafto. "Faster teaching via pomdp planning." *Cognitive science*, **40**(6):1290–1332, 2016.

[RD00] Richard M Ryan and Edward L Deci. "Intrinsic and extrinsic motivations: Classic definitions and new directions." *Contemporary educational psychology*, **25**(1):54–67, 2000.

[RGC13] Amanda Richardson, Amanda L Graham, Nathan Cobb, Haijun Xiao, Aaron Mushro, David Abrams, and Donna Vallone. "Engagement promotes abstinence in a web-based cessation intervention: cohort study." *Journal of medical Internet research*, **15**(1), 2013.

[RGK11a] Andrew Raij, Animikh Ghosh, Santosh Kumar, and Mani Srivastava. "Privacy Risks Emerging from the Adoption of Innocuous Wearable Sensors in the Mobile Environment." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pp. 11–20, 2011.

[RGK11b] Andrew Raij, Animikh Ghosh, Santosh Kumar, and Mani Srivastava. "Privacy risks emerging from the adoption of innocuous wearable sensors in the mobile environment." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 11–20. ACM, 2011.

[Rie96] Lloyd P Rieber. "Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games." *Educational technology research and development*, **44**(2):43–58, 1996.

[RKC18] Mashfiqui Rabbi, Meredith Philyaw Kotov, Rebecca Cunningham, Erin E Bonar, Inbal Nahum-Shani, Predrag Klasnja, Maureen Walton, and Susan Murphy. "Toward increasing engagement in substance use data collection: development of the Substance Abuse Research Assistant app and protocol for a microrandomized trial using adolescents and emerging adults." *JMIR research protocols*, **7**(7), 2018.

[RLS13] Joel JPC Rodrigues, Ivo MC Lopes, Bruno MC Silva, and Isabel de La Torre. "A new mobile ubiquitous computing application to control obesity: SapoFit." *Informatics for Health and Social Care*, **38**(1):37–53, 2013.

[RMB10] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. "Using mobile phones to determine transportation modes." *ACM Transactions on Sensor Networks (TOSN)*, 2010.

[RMM10] Kiran K. Rachuri, Mirco Musolesi, Cecilia Mascolo, Peter J. Rentfrow, Chris Longworth, and Andrius Aucinas. "EmotionSense: A Mobile Phones Based Adaptive Platform for Experimental Social Psychology Research." In *Proc. 12th ACM UbiComp*, 2010.

[run] "RunKeeper." `http://www.runkeeper.com/`.

[San10] Alexius EG Sandoval. "Electrodiagnostics for low back pain." *Physical medicine and rehabilitation clinics of North America*, **21**(4):767–776, 2010.

[SB12] Henrik Schønau-Fog and Thomas Bjørner. "?Sure, I Would Like to Continue? A Method for Mapping the Experience of Engagement in Video Games." *Bulletin of Science, Technology & Society*, **32**(5):405–412, 2012.

[SBS15] Muhammad Shoaib, Stephan Bosch, Hans Scholten, Paul JM Havinga, and Ozlem Durmaz Incel. "Towards detection of bad habits by fusing smartphone and smartwatch sensors." In *Proc. IEEE PerCom Workshops*, 2015.

[SBV11] Christian Seeger, Alejandro Buchmann, and Kristof Van Laerhoven. "myHealthAssistant: a phone-based body sensor network that captures the wearer's exercises throughout the day." In *Proceedings of the 6th International Conference on Body Area Networks*, pp. 1–7. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011.

[SDN16] Karandeep Singh, Kaitlin Drouin, Lisa P Newmark, Ronen Rozenblum, Jaeho Lee, Adam Landman, Erika Pabo, Elissa V Klinger, and David W Bates. "Developing a framework for evaluating the patient engagement, quality, and safety of mobile health applications." *Issue Brief (Commonw Fund)*, **5**(1):11, 2016.

[SJP14] Cong Shi, Kaustubh Joshi, Rajesh K Panta, Mostafa H Ammar, and Ellen W Zegura. "CoAST: collaborative application-aware scheduling of last-mile cellular traffic." In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pp. 245–258. ACM, 2014.

[SM06] Charles Sutton and Andrew McCallum. "An introduction to conditional random fields for relational learning." *Introduction to statistical relational learning*, pp. 93–128, 2006.

[SNB13] David Silver, Leonard Newnham, David Barker, Suzanne Weller, and Jason McFall. "Concurrent reinforcement learning from customer interactions." In *International Conference on Machine Learning*, pp. 924–932, 2013.

[SOM10] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. "Earthquake shakes Twitter users: real-time event detection by social sensors." In *Proceedings of the 19th international conference on World wide web*, pp. 851–860. ACM, 2010.

[SS94] Arthur A Stone and Saul Shiffman. "Ecological momentary assessment (EMA) in behavorial medicine." *Annals of Behavioral Medicine*, 1994.

[SSA14] Hillol Sarker, Moushumi Sharmin, Amin Ahsan Ali, Md Mahbubur Rahman, Rummana Bari, Syed Monowar Hossain, and Santosh Kumar. "Assessing the availability of users to engage in just-in-time intervention in the natural environment." In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 909–920. ACM, 2014.

[SSH15] Yuanchao Shu, Kang G Shin, Tian He, and Jiming Chen. "Last-Mile Navigation Using Smartphones." In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pp. 512–524. ACM, 2015.

[SSP15] Rayman Preet Singh, Chenguang Shen, Amar Phanishayee, Aman Kansal, and Ratul Mahajan. "A Case for Ending Monolithic Apps for Connected Devices." In *Proc. USENIX HotOS XV*, 2015.

[SSS17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. "Mastering the game of Go without human knowledge." *Nature*, **550**(7676):354, 2017.

[Ste64] William Stephenson. *The play theory of mass communication.* Transaction Publishers, 1964.

[str] "Strava." `https://www.strava.com/`.

[STR16] Hillol Sarker, Matthew Tyburski, Md Mahbubur Rahman, Karen Hovsepian, Moushumi Sharmin, David H Epstein, Kenzie L Preston, C Debra Furr-Holden, Adam Milam, Inbal Nahum-Shani, et al. "Finding Significant Stress Episodes in

a Discontinuous Time Series of Rapidly Varying Mobile Sensor Data." In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 4489–4501. ACM, 2016.

[SWD17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347*, 2017.

[SZG14a] Kartik Sankaran, Minhui Zhu, Xiang Fa Guo, Akkihebbal L. Ananda, Mun Choon Chan, and Li-Shiuan Peh. "Using Mobile Phone Barometer for Low-power Transportation Context Detection." In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, pp. 191–205, New York, NY, USA, 2014. ACM.

[SZG14b] Kartik Sankaran, Minhui Zhu, Xiang Fa Guo, Akkihebbal L Ananda, Mun Choon Chan, and Li-Shiuan Peh. "Using mobile phone barometer for low-power transportation context detection." In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, pp. 191–205. ACM, 2014.

[tas] "Tasker app." `https://play.google.com/store/apps/details?id=net.dinglisch.android.taskerm&hl=en`. [Online; accessed 9-Mar-2016].

[TEA15] Edison Thomaz, Irfan Essa, and Gregory D Abowd. "A practical approach for recognizing eating moments with wrist-mounted inertial sensing." In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 1029–1040. ACM, 2015.

[UMP14] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L Littman. "Practical trigger-action programming in the smart home." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 803–812. ACM, 2014.

[usa76] *U.S. Standard Atmopshere*. U.S. Government Printing Office, Washington, D.C., 1976.

[VEL17] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. "Recognizing Detailed Human Context in the Wild from Smartphones and Smartwatches." *IEEE Pervasive Computing*, **16**(4):62–74, 2017.

[vim] "VimoFit." `http://www.vimofit.com/`.

[Vin68] T.K. Vintsyuk. "Speech discrimination by dynamic programming." *Cybernetics*, **4**(1):52–57, 1968.

[VVL07] Douglas L Vail, Manuela M Veloso, and John D Lafferty. "Conditional random fields for activity recognition." In *Proc. ACM Autonomous Agents and Multiagent Systems*, 2007.

[VWC14] Rajan Vaish, Keith Wyngarden, Jingshu Chen, Brandon Cheung, and Michael S Bernstein. "Twitch crowdsourcing: crowd contributions in short bursts of time." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 3645–3654. ACM, 2014.

[WD92] Christopher J. C. H. Watkins and Peter Dayan. "Q-learning." *Machine Learning*, **8**(3):279–292, May 1992.

[Wei91] Mark Weiser. "The Computer for the 21 st Century." *Scientific american*, **265**(3):94–105, 1991.

[WH97] Jane Webster and Hayes Ho. "Audience engagement in multimedia presentations." *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, **28**(2):63–77, 1997.

[WHY15] Bo Wei, Wen Hu, Mingrui Yang, and Chun Tung Chou. "Radio-based device-free activity recognition with radio frequency interference." In *Proc. 14th ACM/IEEE IPSN*, 2015.

[Wic02] Christopher D Wickens. "Multiple resources and performance prediction." *Theoretical issues in ergonomics science*, **3**(2):159–177, 2002.

[WLR14] Ursula Whiteside, Anita Lungu, Julie Richards, Gregory E Simon, Sarah Clingan, Jaeden Siler, Lorilei Snyder, and Evette Ludman. "Designing messaging to engage patients in an online suicide prevention intervention: survey results from patients with current suicidal ideation." *Journal of medical Internet research*, **16**(2), 2014.

[WLR15] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. "Mole: Motion leaks through smartwatch sensors." In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pp. 155–166. ACM, 2015.

[wor] "Workout Labs." `http://workoutlabs.com/`.

[WRS02] Amy B Woszczynski, Philip L Roth, and Albert H Segars. "Exploring the theoretical foundations of playfulness in computer interactions?" *Computers in Human Behavior*, **18**(4):369–388, 2002.

[WSC12] Jing Wang, Susan M Sereika, Eileen R Chasens, Linda J Ewing, Judith T Matthews, and Lora E Burke. "Effect of adherence to self-monitoring of diet and physical activity on weight loss in a technology-supported behavioral intervention." *Patient preference and adherence*, **6**:221, 2012.

[WSL14] Jacob Whitehill, Zewelanji Serpell, Yi-Ching Lin, Aysha Foster, and Javier R Movellan. "The faces of engagement: Automatic recognition of student engagementfrom facial expressions." *IEEE Transactions on Affective Computing*, **5**(1):86–98, 2014.

[WVK16] Dominik Weber, Alexandra Voit, Philipp Kratzer, and Niels Henze. "In-situ investigation of notifications in multi-device environments." In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 1259–1264. ACM, 2016.

[XLL13] Chenren Xu, Sugang Li, Gang Liu, Yanyong Zhang, Emiliano Miluzzo, Yih-Farn Chen, Jun Li, and Bernhard Firner. "Crowd++: Unsupervised Speaker Count with Smartphones." In *Proc. ACM UbiComp*, 2013.

[XSW11] James Y Xu, Yuwen Sun, Zhao Wang, William J Kaiser, and Greg J Pottie. "Context guided and personalized activity classification system." In *Proc. 2nd ACM Wireless Health*, 2011.

[YGL17] Fengpeng Yuan, Xianyi Gao, and Janne Lindqvist. "How busy are you?: Predicting the interruptibility intensity of mobile users." In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 5346–5360. ACM, 2017.

[YTS14] Longqi Yang, Kevin Ting, and Mani B Srivastava. "Inferring occupancy from opportunistically available sensor data." In *Proc. IEEE PerCom*, 2014.

[YWZ16] Lin Yang, Wei Wang, and Qian Zhang. "Secret from Muscle: Enabling Secure Pairing with Electromyography." In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pp. 28–41. ACM, 2016.

[ZDH13a] Xiaoyong Zhou, Soteris Demetriou, Dongjing He, Muhammad Naveed, Xiaorui Pan, XiaoFeng Wang, Carl A. Gunter, and Klara Nahrstedt. "Identity, Location, Disease and More: Inferring Your Secrets from Android Public Resources." In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pp. 1017–1028, New York, NY, USA, 2013. ACM.

[ZDH13b] Xiaoyong Zhou, Soteris Demetriou, Dongjing He, Muhammad Naveed, Xiaorui Pan, XiaoFeng Wang, Carl A Gunter, and Klara Nahrstedt. "Identity, location, disease and more: Inferring your secrets from android public resources." In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 1017–1028. ACM, 2013.

[ZF15] Manuela Züger and Thomas Fritz. "Interruptibility of software developers and its prediction using psycho-physiological sensors." In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 2981–2990. ACM, 2015.

[ZMM17] Bonnie T Zima, F Alethea Marti, Michael McCreary, Nancy Alfaro, Kira Williams, Alpa Patel, Bo-Jhang Ho, Lily Zhang, and Lingqi Tang. "6.40 Feasibility of a Web-Based App to Optimize Early Stimulant Medication Treatment." *Journal of the American Academy of Child & Adolescent Psychiatry*, **56**(10):S290, 2017.

[ZMN10] Shumei Zhang, Paul McCullagh, Chris Nugent, and Huiru Zheng. "Activity Monitoring Using a Smart Phone's Accelerometer with Hierarchical Classification." In *Proc. 6th IEEE Intelligent Environments (IE)*, 2010.

[ZSC16] Bo Zhou, Mathias Sundholm, Jingyuan Cheng, Heber Cruz, and Paul Lukowicz. "Never skip leg day: A novel wearable approach to monitoring gym leg exercises." In *Pervasive Computing and Communications (PerCom), 2016 IEEE International Conference on*, pp. 1–9. IEEE, 2016.