NORTHWESTERN UNIVERSITY

User Experience Aware System Optimizations for Mobile Systems

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Engineering

By

Emirhan Poyraz

EVANSTON, ILLINOIS

September 2020

ABSTRACT

User Experience Aware System Optimizations for Mobile Systems

Emirhan Poyraz

Over the last few years, understanding user experience within mobile systems has become a popular phenomenon as a means to manage hardware resources. Across the many issues being studied in this area, I focus on how to utilize user satisfaction in this dissertation. Notably, I examine user experience by incorporating real end user satisfaction in mobile system designs and optimizations by conducting a variety of user studies.

While the importance of end user is not debated, the individual end user preferences are still widely ignored as the modern system designs hinge on the average user by developing a "one-size-fits-all" approach. In my work, I begin with showing that there exists a significant variation in user satisfactions across different users - even under the same workloads and/or conditions. Thus, it is certainly vital to not only focus on the raw performance of the systems, but also to concentrate on the needs and desires of the end user. I then proceed with demonstrating the tools and methods needed in order to build highly accurate prediction models for user satisfaction. For the prediction models, I use system metrics and/or built-in sensor data that is available in modern mobile devices. I also validate that the proposed models can easily be adopted in any modern mobile systems at system-level with negligible power consumption. Lastly, I end with conducting user studies in order to demonstrate how the prediction of user satisfaction can be utilized in hardware component managements for the mobile devices. Within these user experiments, I study specifically the two most power hungry components in order to maximize system-level energy savings: CPU and screen brightness.

All in all, this dissertation points toward the importance of incorporating the end user's satisfaction with the hardware component management to the mobile systems research. Various user studies that have been conducted as a part of this work illustrates that if we can succeed in placing

the end user into the design and optimization process, we can achieve significant improvements with regards to the energy efficiency of current mobile systems while maintaining - or even improving - individual levels of user satisfaction concurrently.

ACKNOWLEDGEMENTS

It would not even suffice to say that preparing this dissertation has been quite a journey. Hence, first and foremost, I would like to express my gratitude to my advisor Prof. Gokhan Memik and Prof. Seda-Ogrenci Memik for their continued support, patience and guidance along the way. Thanks to them, I have grown immensely as an individual and as an engineer.

I also would like to thank my advisor Prof. Gokhan Memik for his constant encouragement to follow my interests, even when it takes being courageous with research topics. I also wish to thank my committee members, Prof. Seda-Ogrenci Memik, Prof. Peter Dinda and Prof. Josiah Hester for their ever-present source of guidance and supervision provided during the course of my PhD. I feel really lucky to be part of Northwestern engineering community and had a chance to learn from and work with its amazing professors and staffs during my PhD studies.

I would like to thank my parents for their constant moral support and mellifluous affection which helped me to achieve every sphere of life. I also wish to extend special thanks to my brother, Efecan, for widening my perspective and showing me the presence of a whole another world in the first place. Without his mentoring and encouragement, the path forward could have looked so much more different than the opportunities I have been provided.

Last, but certainly not least, I want to thank my wife, Ipek, for all of her support during this journey. Down the road, there has been quite a few (in fact quite unorthodox and unexpected) challenges and difficulties; but her love, encouragement and believe in me has always been a constant source of inspiration for me to overcome these obstacles. I would hate to think how much more difficult my PhD would have been without her in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Recent technological advances have led mobile devices to become an indispensable part of modern society. In today's world, people spend more time with their smartphones - averaging 21 hours per week during which they are using applications 90% of the time - than using any other devices (i.e., laptops and desktop-computers). Nevertheless, the increasing dependency on smartphones also brought a diversified set of motives and workloads emerging from its users. While the smartphones were used solely for basic communication in the earlier periods, smartphones in the contemporary world has been embedded in every aspect of our everyday life as they become furnished with many features that enable "on-the-go" access to various practices (e.g., web-browsing, communication, shopping, banking, gaming, etc.).

Moreover, the expansion in the collection of the workloads and motivations for usage associated with mobile devices also brought more variation among user experiences. As the number of applications and features of smartphones have amplified, user preferences with regards to device performance or brightness settings also became more pronounced and context dependent. Indeed, it is quite intuitive to expect users to have diverging experiences based on their age, current workload of the device, activity taken underway or environmental conditions at the moment of usage.

In fact, although still quite limited in presence, some manufacturers started to introduce personalized settings in some aspects of the phones. Given that the mobile system is evaluated eventually in the hands of the end user, the importance of addressing user specific preferences as well as introducing settings tailored towards individuals are becoming the key determinant of success or failure of a system.

In this dissertation, I integrate individual users' satisfactions with the mobile system design and optimizations. Specifically, I focus on bridging the gap between hardware resource management and end user experience by conducting various user studies. Through the work conducted and

presented in this dissertation, I aim to develop mobile systems that provide better user experience with improved hardware resource management schemes.

1.1 Differences In User Preferences

Ranging from external features such as its size and shape to internal features like the CPU performance and screen brightness, humans have a diverse set of preferences regarding their mobile devices. This divergence can partially arise from demographic factors like user's age and gender as well as certain environmental or user's physical conditions. Brombacher et al. [72] refer to these differences *"in spite of meeting with the explicit product specifications, a customer explicitly complains on the (lack of) functionality of the product"*. Likewise, there still exists complaints among users about the off-the-shelf performance of phones for real workloads [146]. In return, this reveals the crucial need to better understand the individual's user preferences in order to enhance user experiences by building more competent mobile systems. Then the question becomes how can we understand the user experience to achieve this goal. One way to do so relies on the account of assessing instantaneous user satisfaction.

Indeed, as an unfortunate matter of fact, user satisfaction is discounted in modern mobile systems despite the importance of the end user. The current state of art typically generalizes the system as developed solely toward the average user while allowing too little room for implementing individualized personal settings. Inefficiency of this approach comes from the assumption that all users are expected to have maximum satisfaction with a one-size-fits-all design. In other words, the assumption goes as if all the users have the exact same or, more closely, similar set of demands and preferences regarding their mobile devices. However, following and building on some of the prior work, this dissertation shows that users' preferences and demands tend to vary significantly even under the same workloads and/or conditions. A quick solution to overcome this problem is to customize devices based on each individual's needs. Although theoretically this implementation should work, due to scalability and possibility of high-costs, this is not deemed as a practicable solution. Moreover, it is also possible that users' preferences may change over time and hence a

fixed customized solution would still fail in the long run. Maximizing the resources (i.e. maximizing performance) would still not be a good solution, due to its association with high levels of energy consumption. Thus, there is a need for systems which learn and adapt to user's preferences and needs. If we can (somehow) know end user's instantaneous satisfaction about the system, we can then utilize this information to have a number of benefits: we can create dynamic user-aware decision-making mobile systems.

However, the challenge here arises from measuring and integrating the instantaneous user satisfaction in real time. User satisfaction is inherently a subjective metric, which is affected often times by the limits of human perception and the preconceptions of an individual. The work presented in this dissertation shows some of the tools to predict instantaneous user satisfaction as well as showing how to use it as a global feedback input to better manage hardware resources in the mobile systems. In line with the previous work, I show that incorporating information stemming from an end user's satisfaction to the development of mobile systems leads to improved systems both in terms of the overall user experience and associated levels of energy consumption in comparison to standard techniques that allocate resources based on the experience of an average user.

1.2 Utilizing User Satisfaction

In this dissertation I utilize user satisfaction as a means for two distinct purposes. In the first approach, I use user satisfaction as a feedback input in order to manage hardware resources. The aim is to maximize user satisfaction while minimizing energy consumption. In the second approach, I show an alternative usage of user satisfaction where I use it to avoid harmful habits associated with smartphones. Notably, in this approach, I degrade user experience with the aim of mitigating overuse of smartphones. Below I briefly introduce and discuss these two approaches.

1.2.1 Managing Hardware Resources

The increasing dependency on smartphones brought with itself the demand for more capability and performance. To accommodate this increase in the demands, smartphones have constantly

evolved and improved by adopting new hardware features and components. It is rather common knowledge that management of these components in mobile systems is crucial as they determine level of energy satisfaction and associated user experience. In addition, given the fact that the form of a smartphone ultimately limits its battery size, and hence the amount of stored energy, the importance of an efficient management becomes even more vital. However, as was already discussed, the assumption that users do not vary in their satisfaction levels with their mobile devices causes the current systems to be depicted as inefficient. Furthermore, this assumption also fails to open up the possibility for more personalized settings, making the current management systems become sub-optimal for an individual user.

In spite of the fact that there exists a diverse set of hardware components - as well as extensive amount of studies associated with them-, many tend to focus on the subset of the components that have relatively more energy consumption (i.e. CPU, screen, WiFi). Likewise, in this dissertation, I focus on two most energy-hungry components: high performance mobile CPUs and mobile displays. As was already alluded, I integrate user satisfaction as a feedback mechanism on mobile devices. Notably, I correlate user satisfaction with other user-facing metrics and built-in sensors in smartphones as a means to manage mobile CPUs and display.

Although there exists a wealth of metrics to evaluate the phone such as its shape and weight, when it comes to managing hardware resources, not all metrics can be characterized as helpful as the others. For example, whereas the size of the phone could be a an important aspect for a customer, it is not insightful when evaluating its performance. On the other hand, frame per second (fps) of the screen can provide much more information given that smoothness of the screen is highly dependent on the level of fps. Hence, I choose specifically the metrics that give more insight on the phone's performance or the metrics that provide information from the side of the end user by evaluating the perception of the phone's performance. The details of this study can be found in Section 2.

Furthermore, sensor technology on mobile devices showed an enormous amount of improvement over the course of the decades. Whereas less than a decade ago, smartphones had only a few

sensors; today they have more than a dozen sensors with additional capabilities as well as lower power consumption levels [17]. The evolution of sensors also inevitably brought along various brand-new capabilities to mobile devices. For example, smartphones can now recognize complex human activities and gestures, detect users' stress levels or emotional states, predict pin numbers and more. In this dissertation, I utilize built-in sensors to predict user satisfaction and manage CPU and screen components on a smartphone accordingly.

1.2.2 Mitigating Smartphone Overuse

Although smartphones have helped to improve the quality of life by enabling "on-the-go" access to several activities in some sectors (e.g., web-browsing, communication, shopping, banking, gaming, etc.), the increasing dependency on smartphones also brought a growing number of concerns regarding their negative impacts associated with their excessive usage. Recent research has highlighted a number of potential problems as a consequence of mobile overuse: addiction, financial problems, and dangerous actions associated with its use (e.g., whilst driving) [83, 143]. There also exists a rising concern among parents regarding their children's excessive phone usage given the possibility of negative effects on both their social and academic life [83, 143, 145]. In addition, studies also show certain harmful health effects that might be caused by the immoderate use of phones including cancer, headache, sleep disorder, anxiety, and depression [35]. Indeed, the World Health Organization considers excessive mobile phone use as a public health concern [101], emphasizing the need for more research on preventive measurements.

Despite the importance of reducing excessive phone usage given the growing number of concerns, the number of proposed methods -as well as their efficiencies- remain quite limited and/or still unknown. Whereas a quick solution to reduce over usage would be implementing time restrictions to the applications, it has been shown that such restrictions can possibly cause significant side effects on users such as anxiety and depression [2, 95]. Thus, there exists certainly a need to develop better methods to mitigate overuse of smartphones in daily life.

In this dissertation, I degrade user experience by controlling brightness of the display with the

aim of reducing excessive phone usage. The motivation behind this study relies on the hypothesis that creating discomfort on the part of the end user by altering brightness levels (or showing pseudo pop-ups on the screen) will yield the user to spend less time with the applications and hence their phones. In order to test this hypothesis, I develop three different models where I degrade user satisfaction on the target phone by 1) gradually dimming brightness, 2) rapidly dimming brightness and 3) showing pseudo pop-ups on the screen. I show that, with a small sacrifice, excessive phone usage can be decreased by up to 37.82%. The details of this study can be found in Section 6.

1.3 Attribution

For reference, the main contributions presented in this dissertation have previously appeared in the following conference publications or currenly under review:

- Emirhan Poyraz, Prethvi Kashinkunti, Matt Schuchhardt, Michael Kishinevsky, Niranjan Soundararajan and Gokhan Memik. *Understanding the Impact of Number of CPU Cores on User Satisfaction in Smartphones*. 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. MobiQuitous 2019.

- Emirhan Poyraz and Gokhan Memik. *Using Built-In Sensors to Predict and Utilize User Satisfaction for CPU Settings on Smartphones*. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies. IMWUT/UbiComp 2019.

- Emirhan Poyraz and Gokhan Memik. *Using Built-in Sensors to Control Screen Brightness In-the-Wild*. Under review.

- Emirhan Poyraz and Gokhan Memik. *Quantifying the Importance of Energy Savings on Smartphone User Satisfaction*. Under review.

- Emirhan Poyraz and Gokhan Memik. *Phone Free: Mitigating Smartphone and Application Overuse*. Under review.

- Emirhan Poyraz and Gokhan Memik. *Analyzing power consumption and characterizing user activities on smartwatches: summary*. 2016 IEEE International Symposium on Workload Characterization. IISWC 2016.

As the citations show, the work in this presentation comes from a set of collaborative work with several co-authors who have made critical contributions to the work. Thus, it is important to note that any mentions of "we" in the subsequent chapters refer to myself as well as my co-authors.

1.4 The Organization of the Thesis

The motivation of this dissertation follows from a set of sequential research questions. Some of the main ones are presented below:

RQ1: How do users' performance experiences differentiate in the mobile devices?

RQ2: Can we correlate user satisfaction with system metrics in order to manage CPU settings in smartphones?: *CPU management using system metrics – Chapter 2*

RQ3: How about built-in sensors? It is possible that some sensors can give insights about the current satisfaction level of users. So can we utilize sensors to correlate user satisfaction to manage CPU settings?: *CPU management using built-in sensors – Chapter 3*

RQ4: How about other components? Can we utilize sensors to manage display brightness?: *Controlling brightness using built-in sensors – Chapter 4*

RQ5: Better management techniques can lead to significant energy savings, but do users really care about these optimizations? If they do how much they really care?: *Quantifying the importance of energy savings – Chapter 5*

RQ6: How else can we benefit from the instantaneous user experience in mobile devices? Can we utilize user satisfaction in order to avoid bad habits?: *Mitigating smartphone and application overuse – Chapter 6*

Once I discuss my studies in Chapters 2-6, I present Related Work in Chapter 7 and I conclude my dissertation in Chapter 8.

CHAPTER 2

IMPACT OF NUMBER OF CPU CORES ON USER SATISFACTION IN SMARTPHONES

Smartphones have already become an essential part of everyday life. Today, there are more than 2.5 billion active smartphone users in the world [122]. The increasing dependency on smartphones also brings demand for more capability and performance. To accommodate these increasing demands, smartphones have constantly improved and adopted new hardware features.

One of these features is high performance mobile CPUs. In order to increase the computational performance, mobile CPUs have been improved steadily over the years. They first evolved by increasing the CPU clock frequency to enhance the performance. When this approach faced the peak frequency limit barrier due to heat and exponential increase of power consumption, mobile CPUs have shifted to multi-core architectures [142, 34]. In recent years, smartphones have finally adopted core asymmetry as "big" performance-optimized cores, and "LITTLE" energy-optimized cores to improve the power efficiency and achieve higher performance levels [34].

Despite the wide usage of multicore smartphones in the market, how much these cores impact end user experience still remains unclear. Since current operation systems and firmware manage DVFS (dynamic voltage -frequency scaling) based on the CPU utilization, end users' satisfaction is (largely) ignored. However, prior work shows that user satisfaction on multicore smartphone varies between different CPU configurations [63]. Not all users are affected the same by the changes in CPU core/frequency for a given workload. Additionally, there are still complaints among users about the off-the-shelf performance of phones for real workloads [146]. Therefore, there is a need to understand the impact of smartphone performance on user satisfaction in order to effectively manage smartphone computational resources. If users' performance satisfaction is known, we can manage mobile CPU settings in a much more efficient manner; we can reduce the power consumption if doing so does not cause dissatisfaction or increase performance when necessary.

The challenge to achieve this is to be able to predict satisfaction accurately in real time.

In this project we study CPU cores' impact on user satisfaction and power consumption through user studies. Overall, we conduct two types of experimental studies with real users. We define them as In-the-Lab and In-the-Wild experiments.

In In-the-Lab experiments we study CPU configurations' impact on user satisfaction and power consumption in varying applications. In this experiment, we ask users to repeat a simple task for seven different applications. Between repetitions, the number of cores and core frequency are altered in the background. Along with the user satisfaction reports, we also collect data for the instantaneous power consumption and user facing metrics, such as frame rate and input lag.

We first demonstrate that while users report higher satisfaction with higher core counts for some applications, there are cases where the additional cores only lead to increased power consumption without increasing user experience. Additional cores can also be utilized to reduce the power consumption of the system for a set of applications. Given the high correlation between the collected data and user satisfaction reports, we then propose a system to save energy by altering CPU core count and frequency while keeping users satisfied. The system utilizes a set of user-facing metrics (Table 2.1) to predict user satisfaction and set CPU configuration in the phone in order to save energy. We validate the proposed system by building prediction models and show that we can predict satisfaction with over 97% accuracy on average when a binary satisfaction model is used (i.e., users indicating satisfied versus unsatisfied). The prediction accuracy is over 90% on average if a 5-level satisfaction model is used.

In In-the-Wild experiments, we evaluate the proposed system with real workloads. Specifically, using the proposed system, we build two models: a static user-independent and a dynamic user-dependent model in order to predict satisfaction and set CPU configuration in daily usage. For these experiments, we ask users to use an octa-core smartphone as their primary phone for one-week. Users test the CPU settings of the two models on the given phone. We show that, compared to default scheme, total system energy consumption can be reduced by 12.3% and 11.8% on average with the user-independent and user-dependent models, respectively, without a significant change

Table 2.1: Metrics Collected For Each Application

mean-frame-per-second,	start-and-stop-handler-lag,
stdev-frame-per-second,	click-input-lag,
mean-combined-frame-per-second,	mean-max-click-handle-time,
mean-max-frametime,	stdev-max-click-handle-time,
stdev-max-frametime,	video-load-time,
mean-combined-frametime,	showing-control-time,
mean-max-touch-handler-time,	cpu utilization and frequency,
stdev-max-touch-handler-time,	mean-number-of-threads,
mean-max-image-load-time,	current and voltage (for power),
stdev-max-image-load-time,	mean-drag-time,
mean-max-drag-frametime,	stdev-drag-time,
stdev-max-drag-frametime,	relative visual performance,

in users' satisfactions. To the best of our knowledge, this is the first scientific study analyzing the impact of core count on real users and showing methods and tools to predict user satisfaction to manage heterogeneous smartphone architecture in the wild. Specifically, we make the following contributions:

- We demonstrate that users' satisfaction with the CPU performance of their smartphones vary considerably between applications and users.
- We show that smartphone CPU configurations with more cores can lead to higher energy consumption without increasing user satisfaction for some applications, but that more cores are necessary to achieve maximum satisfaction for other applications.
- We propose a system utilizes user-facing metrics to predict user satisfaction and alter CPU configuration in real-time in order to save energy.
- We propose tools and methods to develop personalized models learn from personal performance preferences online in the wild.

The rest of the chapter is structured as follows. In Section 2.1, I introduce the In-the-Lab user study. I then analyze the collected data, discuss the impact of smartphone CPUs on user satisfaction and explain the proposed system (Section 2.2). In Section 2.3, I describe our In-The-Wild study along with the tested models and least power consuming CPU configuration selection

methodology. In Section 2.4, I show our study on the data collected during In-The-Wild study, and discuss the results. I present discussion and further analysis in Section 2.5. In section 2.6, Finally, I conclude the chapter 2.7.

2.1 Experimental Study: In-The-Lab

The purpose of In-the-Lab experiments is to understand the impact of the number of cores on a smartphone on user satisfaction. This user study was performed on 20 smartphone owners. The majority of participants were college students and all participants were under the age of 50. Participants were gathered through fliers advertising the study and word of mouth.

Users were provided with an LG Nexus 4 smartphone that features a 1.5 GHz homogenous quad-core Snapdragon S4 CPU. This smartphone was chosen due to its number of cores and the flexibility in managing them. We choose to use devices with higher number of cores in our In-the-Wild study. Open source nature of the Android OS allows us to access lower level information about the smartphone's state in this device. The smartphone was loaded with seven open source applications that were intended to represent the wide variety of applications used by smartphone users. The selected applications are as follows:

- Animation: This application we designed includes a button that causes a box to animate up and down across screen when pressed.
- Drawing: Markers [112] is an open source drawing application. It includes a pressure-sensitive, multi-touch canvas to show drawings.
- Gaming: The Green Wall [97] is an interactive game where users fling fruits from the bottom of the screen towards a wall.
- Imageswipe: Universal Image Loader [127] is an application for viewing pictures. It includes multiple ways to browse and display images. We used the viewpager mode in our study, which allows users to swipe between full screen images.

- Readability: This application allows us to test the users' Relative Visual Performance [73]. Users are presented with a grid of E's tumbled in up, down, left and right orientations, and asked to click the E's in the up and down orientation.

- Video: ExoPlayer [46] is a simple media player for Android. It provides an alternative to the MediaPlayer, the Android framework default.

- Browsing: Lightning Web Browser [111] is a lightweight Internet browser application. We chose to have users browse only Wikipedia pages since it is a familiar website for most users.

These applications were instrumented to extract user-facing metrics using Android's logging system, logcat. The extracted features are listed in Table 2.1. We chose features which give more insights about the application and also give minimum overhead to logger application. Metrics, such as frames per second and frame time, were collected for all applications to capture screen events in the phone. Application-specific metrics were gathered in order to isolate different phases; such as "click input lag" feature for animation application, "relative visual performance" for readability application, "image load time" for imageswipe application etc. We should also note that, while CPU utilization, frame metrics, number of threads and average current-voltage (to calculate power) values were collected at 1 Hz, all other collected data was event derived.

For each user study session, users were first familiarized with the applications and the system. During this period, standard governor (default scheme) was used for CPU management. Then, users performed tasks in the seven applications discussed above that were given in randomized order. Users were asked to first perform a specific task for 15 seconds, and then report their subjective rating of the smartphone's CPU performance on a whole-number scale from 1 to 5 ("1" represents lowest and "5" represents highest satisfaction).

Performance of the smartphone was inferred from users' perception of responsiveness, speed, and smoothness. Each application's task was performed twice for each of the randomly assigned CPU configurations. The possible CPU configurations included one, two, or four active cores operating at minimum, middle, maximum frequencies or governed by the standard (on-demand)

governor. Using a wifi-enabled version of the Android Debug Bridge (adb) [13] made it possible to remotely alter the CPU configuration between iterations of each task. Neither the users nor the experimenter were made aware of the CPU configuration of the smartphone at any time during the study. Users were asked to base their ratings solely on their subjective evaluation of the smartphone's performance. The instrumented user facing metrics and information about the average instantaneous power consumption were compiled for each task execution. We must note that the cost of logging is negligible, never exceeding 5% additional CPU utilization during the tests.

2.2 Results and Discussion: In-The-Lab

In this section, I present our In-the-Lab experimental results. I first start by analyzing the correlation between smartphone's CPU performance and user satisfaction. Then, I analyze the power consumption and user satisfaction for different CPU configurations. Finally, I explain our proposed method on modeling user satisfaction using user-interfacing metrics.

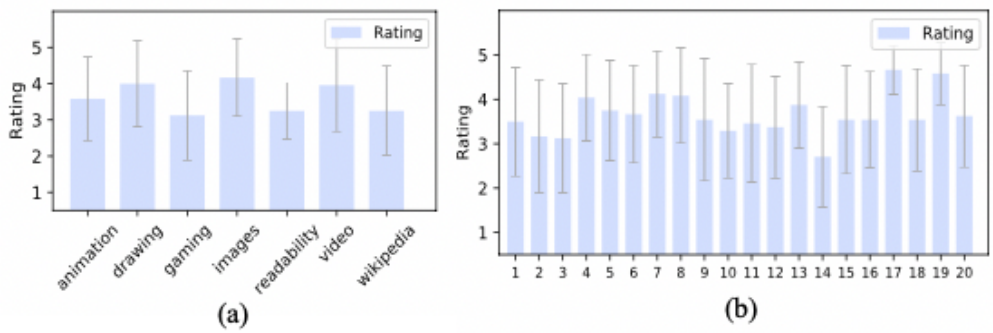


Figure 2.1: (a) Average user satisfaction over all CPU configurations for a single user. (b) Average user satisfaction over all CPU configurations for the gaming application.

2.2.1 Correlation Between Smartphone Performance and User Satisfaction

Currently, the CPU governor does not consider the differences among users' performance preferences when selecting a frequency. Instead, the CPU governor simply adjusts the frequency in response to the workload (CPU load) placed on the smartphone and the impact of these applications on system metrics such as CPU utilization [146]. This approach implicitly assumes that

maximizing the CPU performance of the smartphone is the most important task. However, not all users require the same level of performance to reach a given level of satisfaction when using a specific application. In addition, the level of performance needed to reach a given level of satisfaction varies among applications and users. If user satisfaction truly depends on more than just the level of performance, then this is a missed opportunity to optimize the CPU configuration for power consumption. The CPU governor could instead consider all factors that influence user satisfaction when assigning an operating frequency to its cores. In order to test these claims, we analyze the data collected from our user studies in two ways.

First, we plot the average ratings of each user's satisfaction for all applications. For a given user, there is a considerable variation in satisfaction across applications. Figure 2.1(a) shows this plot for a single user whose average rating over all CPU configurations demonstrates a high sensitivity to the selected application. We also observe that the standard deviation of users' average ratings, shown by the error bars, differ between applications. The same trend is observed for the majority of users. However, when average satisfaction rating for all users is plotted for all applications, the trend is less clear.

Second, we plot the average ratings of all users' satisfactions for different applications. To provide an example, Figure 2.1(b) shows the average ratings over all CPU configurations for the gaming application. We see that there is a high variation in the average ratings across all users. Additionally, the error bars show that the standard deviation of the average ratings varies between users. This shows that different users are affected in different ways to changes in the CPU configuration. In other words, while some users do not change their ratings drastically for different configurations, some are very sensitive. This trend holds for the remaining six applications.

The smartphone's average CPU performance remains constant because the same set of CPU configurations is used for every application. If performance were the sole contributor to user satisfaction for all users and applications, then both visualizations of the data would yield plots with invariable average ratings. Therefore, it is apparent that user satisfaction is influenced by more than just the performance of the smartphone. We can additionally conclude that these influencing factors

affect each user differently. We explore other factors that potentially determine user satisfaction in later sections.

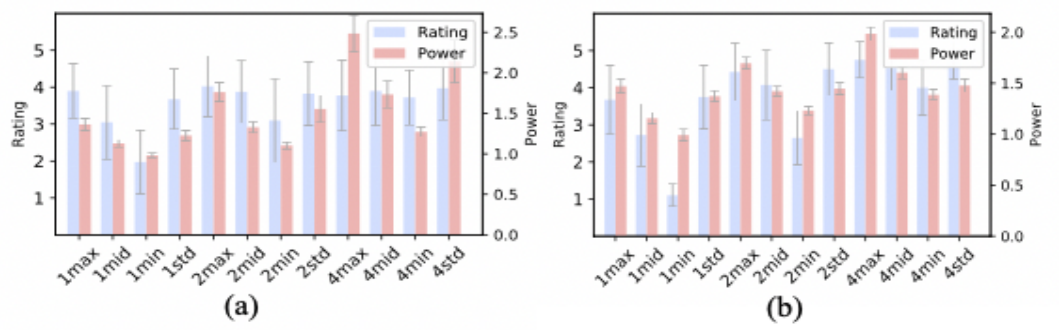


Figure 2.2: (a) Average user satisfaction and average relative power consumption of all users for the readability application. (b) Average user satisfaction and average relative power consumption of all users for the video application.

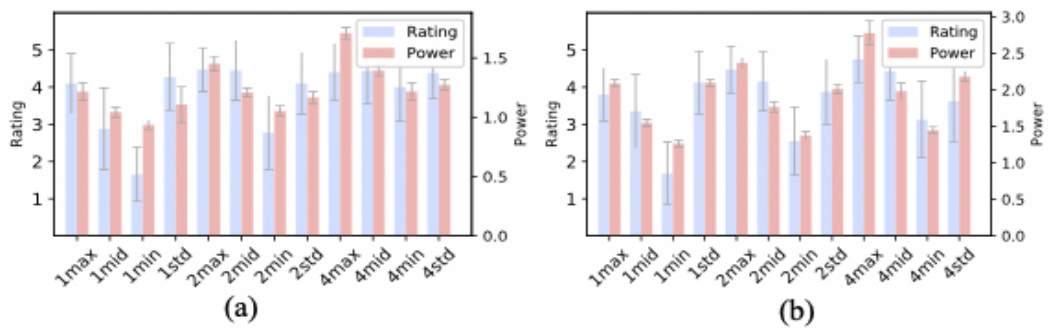


Figure 2.3: (a) Average user satisfaction and average relative power consumption of all users for the animation application. (b) Average user satisfaction and average relative power consumption of all users for the gaming application.

2.2.2 Effect of CPU Configurations on Power Consumption and User Satisfaction

It is important to consider how CPU configurations influence a smartphone's power consumption. The power consumption of an application is a key factor in understanding its impact on the battery life of the smartphone: applications with sustained periods of higher power consumption shortens the battery life more. While it is difficult for users to perceive the instantaneous power consumption of their smartphone, they will generally perceive its eventual impact on the battery life from the total consumed energy. This indirect effect on a user's satisfaction with their smartphone justifies

the importance of power consumption as a metric of satisfaction. In fact, battery lifetime is one of the important marketing tools for smartphones.

The optimal CPU arrangement will provide the highest satisfaction with the lowest power consumption. In order to determine the optimal CPU configuration, we compare the average user satisfaction rating across all users with average instantaneous power consumed during runtime for each application. Since each application runtime is fixed and same, their energy consumption comparisons are as same as power comparisons.

Figure 2.2 and 2.3 show graphical examples of the comparison of average relative power (the power consumption for the single core configuration with the lowest frequency is taken as the base), average satisfaction rating, and CPU configuration for four different applications. The labeling on the x-axis indicates the CPU configuration.

The first number indicates the number of active cores, while the term afterwards indicates the frequency of the active cores: maximum (max), medium (mid), minimum (min), and on- demand governor (std). For example, while "2max" indicates that CPU operates with 2 active cores each running at the maximum frequency; "4std" indicates 4 active cores running in (standard) on-demand governor.

Y-axis shows the user rating (left) and power consumption (right). As seen in the figures, the trends between these three factors (rating, CPU configuration, power consumption) are not always constant across the applications. For each application, the frequency at which cores operate leads to an intuitive ordering of power consumption: power consumptions of minimum frequency CPU configurations are lower than the power consumptions of middle frequencies, which are lower than the power consumptions of the maximum frequencies. The power consumption of the standard governor is usually higher than the middle frequency and is always lower than maximum frequency.

However, the power consumption for CPU configurations with a different number of active cores does not always follow a trend. Some applications, such as the animation application (depicted in Figure 2.3(a)), demonstrate that having additional cores, after having 2 cores, is not beneficial from user satisfaction and power consumption perspectives. In fact, increasing the number of

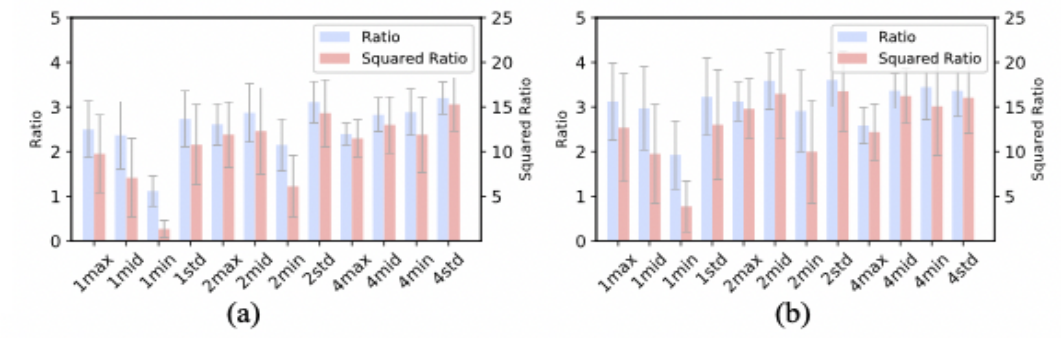


Figure 2.4: (a) Ratio of average user satisfaction and average user satisfaction squared to average power consumption for the video application. (b) Ratio of average user satisfaction and average user satisfaction squared to average power consumption for the imageswipe application.

cores leads to increased power consumption without any increase in user satisfaction. Therefore, for these applications, the optimal CPU configuration is not the standard configuration; please note that the standard configuration is 4 cores active using the standard governor, i.e., 4std. Conversely, it is necessary for some applications to have more cores in order to achieve maximum user satisfaction. Figure 2.2(b) Figure 2.3(b) shows that for the video and gaming applications, additional cores lead to higher satisfaction relative to CPU configurations with fewer cores.

As shown in the Figure 2.2 and 2.3, there is a trend for higher user satisfaction with higher core numbers and frequencies for some applications. This trend can be explained by two complementary ways. Clearly, higher number of cores and frequencies provide better response times and smoothness on the screen. Since in some applications these visual problems are easier to be noticed (skipping frames in video application or delays in touch inputs in game application) users may be more sensitive to them. But it is still not enough to explain large variation in the same application in same configurations. Because even in the game application, which requires high sensitivity to user inputs, it is obvious that, not all users are affected same from these problems (depicted in Figure 2.1(b)). Thus, user's personal preferences and expectations also play a vital role on explaining large variations of user satisfaction reports on same CPU configurations.

To help visualize the power-satisfaction relation more clearly, we introduce plots that show the ratio of user rating to power, as well as the ratio of user rating squared to power. Dividing the satisfaction rating by the power consumption allows us to see the satisfaction provided per unit power

consumption. This measure demonstrates the satisfaction efficiency of the CPU configuration with respect to power consumption.

Using these metrics, we are able to distinguish CPU configurations that have high satisfaction efficiency and high satisfaction rating through weighting this satisfaction efficiency by the rating. The squared rating/power metric (similar to the Energy-Delay2 Product which puts more weight on performance over energy) provides another way to succinctly represent the satisfaction rating over power consumption of a particular CPU configuration. To illustrate this point, we plot these two metrics for the video and imageswipe applications. Figure 2.4(a) shows the ratio of rating (left -y axis) and the ratio of rating squared (right -y axis) to power consumption for the video application. We see both metrics in the figure suggest that a CPU configuration of four cores operating at standard governor frequency is the optimal configuration for the video application. In fact, this conclusion is corroborated by the original data visualization shown earlier in Figure 2.2(b). However, although we determine that default CPU configuration is the optimal configuration for the video application, this is not the case for all applications (i.e. imageswipe application shown in Figure 2.4(b)).

Table 2.2: Optimal CPU Configurations by Applications

Application	Optimal CPU configuration	
	for all users	for bottom 25 percent
Animation	1std	1std
Drawing	2std	2std
Gaming	4mid	4max
Imageswipe	2std	4mid
Readability	2mid	4mid
Video	4std	4std
Wikipedia	4std	4max

Table 2.2 (middle column) lists the optimal configurations for all applications. We discover that the optimal CPU configuration is not the standard configuration for five of the seven applications we test. We must note that the optimal configurations found by the rating per power and rating squared per power were identical.

To further understand the relationship between user satisfaction, power consumption, and CPU

configuration, for each application we isolate the executions that produce user satisfaction in the bottom 25 percent of all executions. This allows us to study the preferences of the most unsatisfied users and compare it to the preferences of all users. Since the users in the top 75 percent of executions are already satisfied with their smartphone's performance, it may be important for designers to optimize for the least satisfied users to increase average user performance satisfaction. We then produced the same style of plots as used previously to determine which configuration the most unsatisfied users preferred.

Table 2.2 (rightmost column) summarizes the results of the optimal CPU configuration for 25 percent least satisfying executions. Five of the seven applications required all four cores to be active to reach the optimal operation for these "more demanding" users, which further motivates the higher number of cores in these systems.

In fact, similar to previous work [25], we have observed low thread level parallelism (TLP). In addition the utilization of all 4 cores were rare on the selected applications. Although more cores are rarely utilized in general, they are necessary in the "bursty" computation periods. As a result, they are required a) to achieve highest satisfaction in certain applications across most users and b) to achieve highest satisfaction across many applications for "pickier" users.

When all executions are considered, the standard CPU configuration is the optimal configuration for only the video and browsing applications.

The standard governor frequency (std) is optimal for five applications, and four cores are optimal for three applications. There are four applications for which the optimal CPU configuration of the bottom 25 percent of ratings differed from that of all users. The optimal CPU configurations in these cases are one that included more cores or a higher operating frequency. Additionally, the standard CPU configuration is the optimal configuration for only one of the seven applications for the most unsatisfied users.

We have further analyzed our results for their statistical significance. Specifically, we ran t-tests to see how user satisfaction and power consumption of the optimal configurations differ from standard 4-core configuration (4std). In these tests, we considered 5 applications (animation, draw-

ing, gaming, imageswipe, readability), which have an optimal CPU configuration that is not the standard 4-core configuration (Table 2 middle column). Across all applications, we observed p value as 0.912 for the hypothesis H-(satisfaction-optimal) $>$ H-(satisfaction-4std), indicating that the optimal configurations have a higher user satisfaction. We also observed p value as 1.000 for hypothesis H-(power-optimal) $<$ H-(power-4std), indicating that standard 4-core configuration's power consumption is higher than optimal configurations'. Thus, the configurations selected as optimal indeed reduce power consumption and also increase user satisfaction on the selected applications.

Based on the overall results, more cores are necessary for smartphone CPU configurations since they can lead to reduced power consumption and improved satisfaction for certain applications. However, all cores do not need to be active at all times since this increases power consumption without improving user satisfaction for some applications. Therefore, the default CPU configuration is overprovisioned for some applications even for the pickiest users. Understanding when it is necessary to run all cores could be very lucrative for smartphone CPU designers as it could lead to a significant power reduction without compromising user satisfaction. Within this motivation we discuss our methods on setting CPU configuration in the wild and show energy savings in real workloads later in In-The-Wild study section.

These results underscore the fact that user's performance satisfaction is tied to more than just CPU configuration. If we wish to fully understand what influences user satisfaction, we should consider alternative metrics.

2.2.3 Proposed System: Predicting User Satisfaction

In the previous section, we showed that user satisfaction varies across users and the frequency of the CPU alone is not sufficient to determine it. To understand what influences user satisfaction, we gauged how well the user facing metrics collected during the user studies correlate with the subjective ratings. Therefore, we analyzed the accuracy of predicting user satisfaction (ratings) within the given metrics. Specifically, we propose a system that takes the user-faced metrics (listed

Table 2.3: Summary of Predictive Models

Application	Selected Features	Error
Animation	mean_fps, threads, stdev_animation_fps	9.12%
Drawing	cpu_util(0-3), mean_frame_time, max_touch_handler_time, mean_fps, mean_touch_handler_time, stdev_touch_handler_time	8.27%
Gaming	mean_frame_time, stdev_fps, mean_combined_touch_frametime	10.07%
Imageswipe	cpu0_util, mean_frame_time, stdev_fps, stdev_drag_frametime, mean_fps, stdev_image_load_time	9.35%
Readability	cpu0_util, mean_fps, rvp, stop_handler_lag	11.60%
Video	cpu1_util, max_video_frametime, mean_video_fps, video_load_time, mean_video_frametime,	8.93%
Wikipedia	cpu_util(0-2), max_frame_time, mean_fps, stdev_fps	11.92%
Global (All applications)	cpu0_util, cpu1_util, max_frame_time, mean_fps, mean_frame_time, stdev_fps, stdev_frame_time, threads	12.77%
Application-Specific Model Average		9.90%

in Table 2.1) as input and predicts the ratings given by the user.

We built prediction models using the proposed system. As discussed in Section 2.2.1, user ratings were collected on a whole-number scale from 1 to 5. In the models, the ratings were normalized to have a mean of zero and a standard deviation of one. This normalization was applied to standardize the satisfaction ratings among users. Using Weka-tool [136, 137], we first used a feature selection algorithm, "Correlation-based Feature Selection", to filter metrics that are correlated with one another. Then, using M5P tree algorithm in Weka-tool, we developed two types of predictive models: an application-specific model using metrics collected for that application specifically and a global model that uses metrics collected for all applications. We observed that M5P algorithm gives the best accuracies among other supervised algorithms on predicting normalized user ratings. Table 2.3 summarizes the features that were chosen by the feature selection algorithms, and the average absolute user satisfaction prediction error for each model. The average error is calculated by using a 10-fold cross-validation. Therefore, training and test data do not overlap. Note that runtime overhead of M5P algorithm on the phone was insignificant (¡1%), indicating that such a dynamic system is plausible to deploy. We measured the overhead by repeatedly comparing CPU utilization of the phone with and without classifying the data.

We make several observations about the user satisfaction models. First, we notice that the features selected for the application-specific models are quite intuitive. For example, the touch

handler time is included in the drawing application model, and the video frame time and frames per second were included in the video application model.

We also see that all the application-specific models have a lower average prediction error rate than the global model. It appears that metrics that are specific to a certain phase of an application (e.g., touch handlers, frame rate during a touch) are better indicators of user satisfaction than general collected metrics. However, even the global model achieves low error rates, 12.8% across all applications and users. Finally, metrics such as frames per second, frame time, and CPU utilization are included as features for a majority of the models. It is worth noting that the models produced by Weka are identical regardless of whether the user field was included as input in the analyzed data set.

Next, we carried out our study Into the Wild to predict user performance satisfaction and manage phone CPU settings in order to save energy in real workloads.

2.3 Experimental Study: In-The-Wild

In this section, I explain our In-the-Wild study. In this study, we evaluate the proposed system by working with real workloads in the hands of real users. We start by introducing our target device. Then, we define the CPU configurations selected for the study. Further, we describe our user independent and user dependent models. Finally, we explain our test setup in the wild.

2.3.1 Test Device

We used Huawei Google Nexus 6P smartphone that is powered by an octa-core Qualcomm Snapdragon 810 processor. This device has a heterogeneous big.LITTLE core (four 1.55 GHz Cortex-A53 and four 2.0 GHz Cortex-A57) architecture [1]. Nexus 6P smartphone was specifically chosen for our In-the-Wild tests because of the trend in the smartphone market for increased heterogeneous core architectures (Huawei's kirin 960, Samsung's Exynos, Apple's A11 etc.) [8]. The reason we didn't choose Snapdragons 820/821 architectures is because they have quad-core and not yet that common in the market. The software in our test device is a rooted version of Android 8.0 (Oreo).

Open-source Android platform enables us to collect most of the proposed user facing metrics and rooted version enables us to alter CPU configuration programmatically.

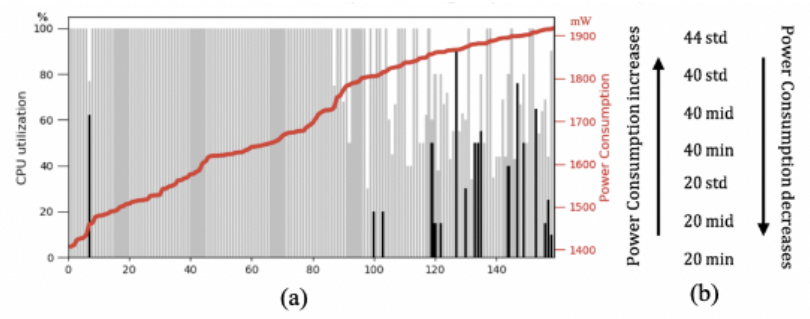


Figure 2.5: (a) CPU utilizations for least 160 power consuming configurations represented as little (gray bars) and big (black bars) core groups and (b) selected 7 configurations ordered based on their power consumptions to be used during user tests.

2.3.2 Choosing CPU Configurations

We identify CPU configurations, which cause least power consumptions on a set of applications in our test device. Our target smartphone has a heterogeneous octa-core architecture with four little cores, each supporting 11 different frequencies: 384, 460.8, 600, 672, 768, 864, 960, 1248, 1344, 1478.4, 1555.2 (all in MHz); and four big cores, each supporting 15 different frequencies: 384, 460.8, 600, 672, 768, 864, 960,1248, 1344, 1440, 1536, 1632, 1728, 1824, 1958.4 (all in MHz). Each core group (big and little) can be set to any number of active cores and frequencies available. Hence, our CPU has 2774 possible active core/frequency combinations. In order to identify the least power consuming combinations from this large set, we develop a power-logger application and compare all CPU configurations under 3 different applications: game, video, and the animation applications described in Section 2.2.

We started by activating only one little core running at minimum frequency (384MHz) and tried all available frequencies by activating/deactivating one more core each time. We also set the frequencies to on-demand governor (std), which allows standard governor to decide frequency among all available frequencies. Since little cores and big cores are homogenous in their own groups, in the test, combinations were arranged as group-wise. We also tested big-little core group

combinations to see how activating cores from both groups effect power consumption on the tested workloads.

We used Monkeyrunner in Android [98] to create same workloads and ran each configuration by 3-min in three different applications: gaming, video, and animation (described in Section 2.2, in In-the-Lab tests). Since all configurations were tested with same amount of time, their energy consumptions showed the same order with their power consumptions. In Figure 2.5(a), x-axis represents the least power consuming 160 configurations and y-axis shows the CPU utilizations of big and little core groups (gray color is for little cores and black is for big cores) (left) and power consumptions in mW (right). Our results show that, as expected, power consumption decreases when CPU utilizes little cores more, even the utilization goes to highest levels. By sorting all configurations based on their power consumptions, we make the following observations on big and little core groups:

- When cores are activated from both groups, not surprisingly, CPU tends to use only little cores. CPU uses big cores either when there is no little core, or active little cores have high utilizations (60+%). This is also one of the main principles of heterogeneous multicore architecture [126].

- Least power consumptions occur when CPU is provided with one or two little cores with minimum frequencies. But having only one active core causes skipped frames and delays in handling touch events on tested workloads.

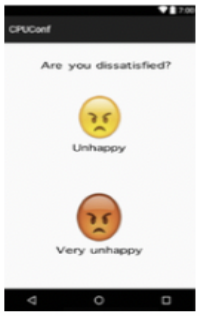
- When compared in equal conditions (only active cores are from one group in same count and same frequency), big cores cause more power consumption than little cores.

- Increasing frequency increases power consumption considerably for each core group. On-demand governor's frequency consumes less power than maximum frequency as we also observed in our In-the-Lab study.

By looking at these constraints, we define our 7 optimal CPU configurations (to be used In the Wild tests) as follows: 2 or 4 active little cores only with minimum (384MHz), medium

(960MHz), and on- demand governor frequencies (6 configurations); and all cores activated with the on-demand governor frequency (default scheme). In other words, we try to use little cores as much as possible in our configurations and activate the big cores by switching to default scheme if only they are needed to maintain user satisfaction.

Figure 2.5(b) depicts the selected configurations in ascending order (from bottom to top) based on their power consumptions. For each configuration, first digit indicates active little core number, second digit indicates active big core number, and third term is the frequency. Thus, for example, 20min means only two active little cores operating at minimum frequency and 44std means all eight cores are active managed by the standard governor.



(a)

Collected User Facing Metrics and System Statistics
CPU utilizations and frequencies of all 8 cores, Mean and standard deviation of frame per sec, Mean, standard deviation and max. of frame time, Mean number of threads, WiFi (transferred bytes), Screen on/off, Current CPU configuration (20min, 20mid,…, 44std), User satisfaction as 3 labels: default 2-happy, 1-unhappy, 0-very unhappy Current-voltage and active CPU time (to calculate power and energy)

(b)

Figure 2.6: (a) GUI on the phone to collect 2-level (unhappy and very unhappy) user reports about phone's CPU performance and (b) collected user facing metrics in in-the-wild studies.

2.3.3 Logger Application

For our in the wild tests, we have developed a logger application. The logger application predicts user satisfaction from collected data and sets least power consuming CPU configurations (described in previous section) in the wild. The application is developed as a regular ART (Android Runtime) executable using the Java standard libraries available in the Android framework. Therefore, it can be used in any rooted Android smartphone. At a high-level, the application consists of two parts: (1) a GUI part (2) and a background service part:

- *The GUI part*: When application starts in the phone, a notification is shown on the upper bar of the phone screen to get instant user satisfaction input in 3 levels: 2-happy (default),

1-unhappy and 0-very unhappy, in the wild (Figure 2.6(a)). We placed the notification on the upper bar to enable users (by swiping down) an easy access to evaluate whenever they feel dissatisfaction from the phone's performance.

- *Background service*: The background service implements four main tasks: 1) logging user-facing metrics and system statistics shown in Figure 2.6(b), 2) building prediction model from collected user data, 3) predicting current user satisfaction, and 4) setting optimal CPU configuration in real-time. To prevent perturbation, logging occurs in every 2 seconds, predicting the satisfaction and setting CPU configuration occurs in every 10 seconds. Also, the service periodically looks for a network connection and sends the logs back to our server. We observe that our logger increases the CPU utilization less than 1% and adds less than 80 mW additional power consumption on average. In case our background service was implemented in kernel space instead, its overhead to the system would be even smaller.

Figure 2.6(b) shows the collected user facing metrics and system statistics by our logger application. We collected the same metrics we used in the global (application-agnostic) model in Section 2.3.3. This enables us to evaluate global model features in real workloads. We must note that, these metrics are easily accessible in system level programmatically without any additional software or hardware support.

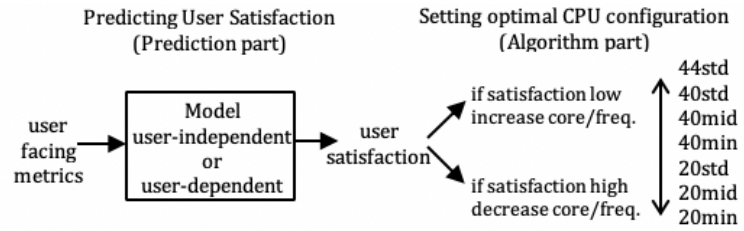


Figure 2.7: Flow of predicting user satisfaction and setting optimal CPU configuration in the test device in real-time.

2.3.4 Predicting User's Satisfaction In the Wild

Figure 2.7 shows the flow for predicting user satisfaction dynamically in the wild. As explained in Section 2.3.3, we utilized user-facing metrics to predict user satisfaction in real-time. We built

two different models to see how collected data from multiple users differ from personal data on predictions:

- **User-independent model**: We built user-independent model from 5 users' (different from In-the-Wild test users) cumulative data as offline and loaded to the test device beforehand. As workloads we chose 4 different applications (two game applications, one video application, and browser application). Users interacted with each application for one-hour duration. During these interactions, CPU configurations were altered randomly among optimal configurations described in Section 2.4.2. Users pushed the dissatisfaction buttons (unhappy or very unhappy) from the GUI of the logger application whenever they felt so. Along with the user reports, our logger application constantly logged the data shown in the table in Figure 2.6(b). Again we used M5P tree algorithm from Weka Machine Learning [137] tool to create the user independent model offline. Similar to Section 2.3.3, we observed that M5P algorithm gives the best accuracy in predicting the satisfaction (on average ¡8% absolute relative error).

- **User-dependent models**: User-dependent models were created online during the experiment for each user. Before starting the experiment, each user was first familiarized with the test phone for an hour. During this period, CPU was set to defined optimal configurations (described in Section 4.2.) randomly. In this pre-test period users were trained to use notification bar and logger application GUI to report their dissatisfaction whenever they felt so. Similarly, along with the user reports, our logger application constantly logged the data shown in the table in Figure 2.6(b). Throughout the In-the-Wild experiments user-dependent models were re-created from current accumulated user data dynamically. For user dependent models, first kMeans Clustering in Weka Java Libraries was used to cluster all collected user data. kMeans Clustering was used to group and identify configurations that cause dissatisfaction for each user. We found kMeans clustering especially helpful in situations where user does not provide enough dissatisfaction indication and the data look more sparse. Then again M5P tree algorithm in Weka Java Libraries was used to create user-dependent model

in the wild.

During the one-week *In-the-Wild* experiments, users test either one of these two user models or default scheme each day in a random order. Every time user-dependent model is chosen, a new user-dependent model is built from all the collected data. Therefore, unlike the *user independent* model, which was fixed during the experiment, *user-dependent* models were rebuilt dynamically using the current accumulated user data.

2.3.5 Setting CPU Configuration In the Wild

For both user- models (user independent and user dependent), in every 10 seconds, current user satisfaction is predicted from the given user facing metrics. Then CPU configuration was set dynamically in the wild. Figure 2.7 also visualizes the flow of CPU configuration setting process in real-time.

Users report their satisfactions in three levels: 2 (default-happy), 1 (unhappy), and 0 (very unhappy). Therefore, the user satisfaction predictions of the models also range between 0 and 2. If the predicted satisfaction is less than 1.25, that means user may be dissatisfied with the current CPU configuration and our algorithm increases either the active core count or frequency. If the satisfaction is higher than 1.75, that means user is already satisfied with the configuration and our algorithm does the opposite in order to save energy. In other words, our algorithm tries to keep the satisfaction high, always more than 1.25 and typically around 1.75.

We determined the upper (1.75) and lower (1.25) bounds by conducting experiments with several levels in our lab. It is possible that other levels may be preferable by different developers. These levels could also be changed to make our system more (or less) aggressive. However, we found these bounds to work well in general.

In Figure 2.7, we also show the CPU configurations that are set based on the predictions. As shown in Figure 2.5(b) the configurations are ordered by their power consumptions. Thus, transitions between them are also done in the same order shown in the figures. Predicting satisfaction and setting CPU configuration programmatically in real-time also presented in pseudo-code 1.

Algorithm 1 Pseudo code for predicting user satisfaction and setting CPU configuration programmatically in real time.

```
1:  procedure LOGGER-AND-CPU-SETTER
2:      Make calls to classes to user facing metrics and system statistics
3:      predict-satisfaction-and-set-CPU()
4:      post handler for 10 seconds
5:  function PREDICT-SATISFACTION-AND-SET-CPU()
6:      if current model is user-independent model then
7:          user satisfaction = get user-independent model prediction
8:      else if current model is user's user-dependent model then
9:          user satisfaction = get user's model prediction
10:     if user-satisfaction > 1.75 then
11:         decrease-core-count-frequency()
12:     else if user-satisfaction < 1.25 then
13:         increase-core-count-frequency()
14: function DECREASE-CORE-COUNT-FREQUENCY()
15:     if current CPU is 44std then
16:         set CPU to 40std
17:     else if current CPU is 40std then
18:         set CPU to 40mid
    ...
19: function INCREASE-CORE-COUNT-FREQUENCY()
20:     if current CPU is 20min then
21:         set CPU to 20mid
22:     else if current CPU is 20mid then
23:         set CPU to 20std
    ...
```

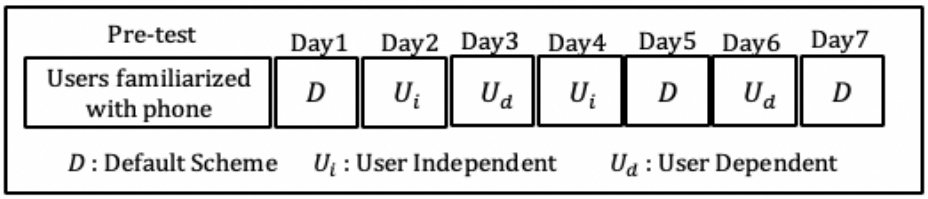


Figure 2.8: An example random model selection for a user during the one-week long In-the-Wild experiment.

2.3.6 Test Setup

We tested the two models and the default scheme on 20 smartphone owners for one-week duration each. The majority of participants were young professionals and all participants were under the age of 50. Participants were gathered through fliers advertising the study and word of mouth. Each user, before starting the one-week experiment, was introduced to the phone for an hour by using varying applications. Test phone starts collecting data from this pre-test period, until end of the experiment (one-week) for each user. Therefore, we collected total 140 days of data. During the experiment, users used the given phones as their primary phones by inserting their sim cards and installing/using applications they typically use. Users tested one model each day (the order of models were randomized). Since the experiment takes 1 week, we made sure that all users test the two models and default scheme exactly twice. Figure 2.8 visualizes a random model selection for a user. In the analysis, we ignored the first day as it may create artificially high energy consumption for the chosen model assuming that users will install their favored applications during the day. Throughout the experiment, users pushed dissatisfied buttons placed on the GUI of the phone whenever they felt so. These on-the-fly user reports were then used on building new user-dependent models in the wild.

Additionally, through a popup questioner, we asked users' daily overall satisfactions of the phone's CPU performance and its battery management three different times in each day to analyze models' performances on daily usage. In the questioner, users answered the question "How was your phone's CPU performance and battery management today?" from a 5-scale radio style selection: 1 is worst, 5 is best.

2.4 Results and Discussion: In-The-Wild

In this section, I present the results of our In-the-Wild study. I first discuss results of the power consumption and user satisfactions of our 20 users. Then I explain our Amazon Mechanical Turk study, where we compare default scheme to static model in a bigger crowd and analyze their differences in user satisfaction.

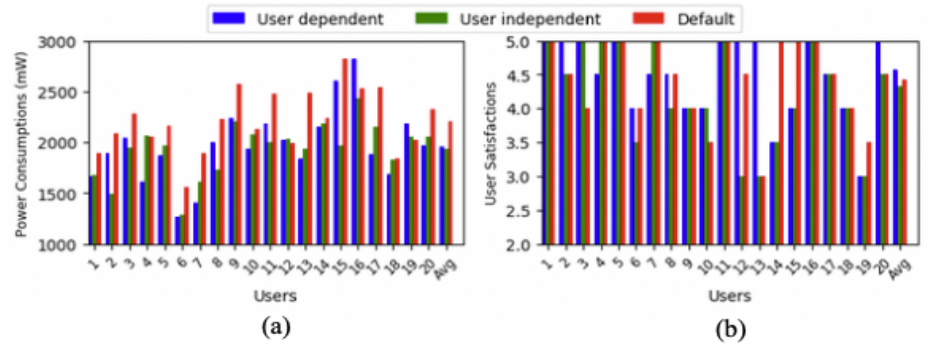


Figure 2.9: (a) Average daily power consumptions in mW and (b) average of daily reported satisfaction levels of each model for 20 users.

2.4.1 Power Consumption and User Satisfactions

Figure 2.9 shows power consumptions and daily overall user satisfaction averages of the default scheme and two tested models (user independent and user dependent) for our 20 users. In the figure, y-axes represent consumed average power in mW (a) and average user satisfactions from daily user reports ranging from 1 to 5 (b); x-axis shows our 20 users and their averages at the end.

Since each user's workload and usage pattern are different, we see a large variation across users' power consumptions and satisfactions. Nevertheless, in most cases, user independent and user dependent models consume considerably less power than default scheme while users' satisfactions look similar. When compared to default scheme, we see up to 30.2% power saving using the user independent model (user 15) and up to 26.2% saving with the user dependent model (user 13). We see only 4 users have higher power consumption with either one of the user models than default scheme. On average, with user independent and user dependent models, we see 12.3%

and 11.8% system level power savings, respectively. We must note that, since each user used the phone same amount of time, we observe similar active CPU times for each day (thus for each model). Therefore, we detect their energy consumptions comparisons are same as their power consumptions'.

Although our algorithm always tries to select the least power consuming configurations in the given order in Section 4.4, we must note that the user independent model spends 27.7% its time in the 44std configuration (default scheme – 8 cores are active running in standard governor) across our 20 users, while this ratio is 35.5% for the user dependent models. In other words, our algorithm frequently set CPU to the default scheme to meet the current user satisfaction. Thus, higher core counts are necessary to maintain user satisfaction on some users and certain applications. This result also matches the same conclusion we had from the In-the-Lab study regarding the need for a high number of cores to maintain user satisfaction for some applications.

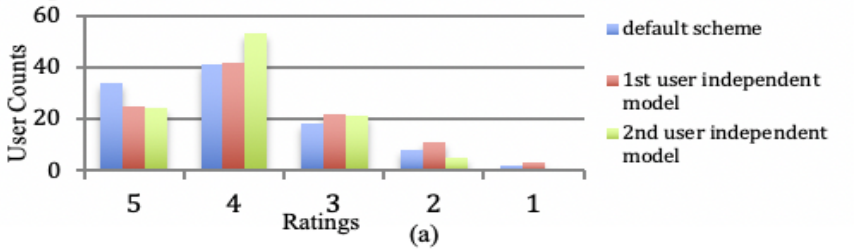
We also observe that, while user independent model provides higher power savings, on average, it also causes least satisfaction when compared to other models. Users are more "satisfied" with their own personalized models than the user independent model. However, the average user satisfactions do not vary significantly across the three models. Default scheme, user independent, and user dependent models have mean values of 4.57, 4.32, and 4.42 and standard deviations of 0.51, 0.69, and 0.59, respectively. To see the differences between these 3 different models, we perform pair-wise t-test and an equivalence test for our 20 users' satisfaction values. In t-tests, we see p-values as 0.20 and 0.39 for user independent model- default scheme and user dependent model - default scheme comparisons respectively. Since the p-values are higher than the significance level (0.05), the analysis concludes that the means do not differ. Using the equivalence test on the same data, we see p-values as 2.2e-04 and 1.17e-05 for user independent and user dependent models comparisons respectively, which indicates that there is sufficient evidence to claim that the means are similar enough.

2.4.2 Further Analysis on In-the-Wild Study

Limitations of application dependent models. As discussed in Section 2.4.4, for in-the-wild study, we build two models to predict satisfaction: a fixed user independent model and a dynamic user dependent model. We built both models as application-agnostic for two reasons. First, considering the (growing) volume of applications in the market and user selection variety on them, it is not feasible to develop a prediction model using a fixed or dynamic application list. Second, (depending on the sampling frequency) keeping track of the applications' usages along with the time they spend in CPU is quite costly in terms of overhead and memory in application level. Therefore the proposed system is application independent. However, for analysis purposes, we have collected application names and the time they spent in CPU via our logger application. We set sampling frequency to 5 min and limit application count to maximum 20 in order to keep logger overhead minimum. Specifically we were interested on the number of CPU-intense applications used by per user since they may be more influential on user's rating decision. While there is a large variation in application selections, we observe a maximum of six and a minimum of two CPU-intensive applications are used by each user. We consider applications that utilize the CPU more that 20% as CPU-intense. We also observe that, in active usage, average CPU utilization ranges between 2% and 100%, with an average utilization of 42% (user-dependent and user-independent models' utilizations are 43% and 45%, respectively, while the average utilization for the default model is 38%). These results show that even though some applications may use components like accelerators or GPU more and lower CPU utilization, the average CPU usage is still high. Moreover, we observe bursts in CPU utilizations. Since the overall utilization is not low, we see that a) the default scheme uses higher than necessary frequencies and b) lowering the frequency has a significant impact on the overall power consumption.

Users' satisfaction evaluations on tested models. Figure 2.9(b) shows average user satisfactions from daily user reports for each user. In addition to evaluations in Section 2.5.1, it is also possible to analyze user ratings individually. Specifically, instead of taking averages of all user reports, we were interested in the individuals who are "unsatisfied" with the proposed system. We

observe that while 17 users rated either user independent or user dependent model as better than or equal to default model, only 3 users rated default model as the best. Since proposed system is implemented as a software mechanism, it can be optional to users and can be turned on/off based on users' selection (i.e., it can be deactivated for the 3 users). Additionally, in order to improve battery life in case of emergencies, these models can be made an option for different power saving modes on the smartphones.



(a)

Model Comparisons	*t-test*	*equiv. test*
default vs 1st user-independent model	0.14	6.80e-08
default vs 2nd user-independent model	0.46	1.96e-11
1st user-independent model vs 2nd user-independent model	0.38	2.03e-10

(b)

Figure 2.10: (a) AWS Mechanical Turk users' satisfaction distribution from 1 to 5 in three videos. (b) Statistical difference comparisons of three tested models.

2.4.3 Mechanical Turk Study

We conducted a further study to compare the user independent model and default scheme on a bigger crowd, using the Amazon Web Services (AWS) Mechanical Turk (mTurk) [115]. In mTurk study, we observe 110 users' evaluation on the phone's CPU performance under these two models. User-dependent models were not considered since they are personal for our 20 In-The-Wild users.

In order to compare the models, we recorded a CPU-intense car race [20] gameplay videos on our test device and made the mTurk users watch these gameplays and rank their satisfaction (from 1 to 5: 1 is the worst, 5 is the best) based on the smoothness and frame skipping on the video. We recorded one video with phone CPU is managed by the default scheme, and two more videos while it is managed by user independent model. Our first video in the user independent model

was the first-time the application was started and the second one was after the first 1 minute of playing. The reason for recording in these two different times is because for the user independent model, it may take some time to set the CPU in desired configuration, and this duration may create dissatisfaction for the users. Hence, we decided to test these time states separately.

MTurk users watched the three videos (gameplays where CPU is managed by default scheme or user-independent models) in a random order. Figure 2.10(a) shows satisfaction distributions of mTurk users with the default scheme, user-independent model first and user-independent model second videos. X-axis shows the ratings (1 to 5) while the y-axis shows the ratios of reported satisfactions. Their means and standard deviations are 3.98, 3.77, 3.89 and 0.96, 1.02, 0.80, respectively. The default scheme and second user independent model seem to have similar ratings, while the first user independent model seems to have a slightly lower rating. To understand the statistical differences of these three group ratings, we performed t-test and equivalence tests as described in the previous section (Section 5.1). Table in Figure 2.10(b) shows the results that all three groups have similar means, while the difference between the default and the second user independent model is hardly distinguishable. As a result, we can claim that there may be slight changes in user satisfaction at first, but once the model settles the differences between the models becomes indistinguishable. Therefore, the proposed system does not cause a significant decrease in user satisfaction, but reduces the energy consumption considerably.

2.5 Summary

In this project, we study CPU management on mobile systems with respect to end user experience by conducting two experimental studies with real users. We define them as In-the-Lab and In-the-Wild experiments.

In-the-Lab, we showed that for some applications, a CPU configuration with more cores led to increased power consumption without increasing satisfaction. For others, it was necessary to have more cores in order to achieve maximum satisfaction. By observing the high correlation between collected data and user satisfaction reports, we propose a system to save energy by altering CPU

core count and frequency while keeping users satisfied.

We evaluate the proposed system in-the-wild by building two prediction models: a user-independent (user-oblivious) and user-dependent (personal). Our users test the two models and the default scheme for one-week duration, which composes 140 days of worth of data. When compared to default scheme, our results show that, without impacting satisfaction, user-independent and user-dependent models save 12.3% and 11.8% of total system energy on average, respectively.

CHAPTER 3

USING BUILT-IN SENSORS TO PREDICT USER SATISFACTION FOR CPU SETTINGS

Smartphones have already become an essential part of everyday life. Today, there are more than 2.5 billion active smartphone users in the world [122]. The increasing dependency on smartphones also brought demand for more capability and performance. To accommodate these increasing demands, smartphones have constantly improved and adopted new hardware features.

One of these features is high performance mobile CPUs. In order to increase the computational performance, mobile CPUs have been improved steadily over the years. They first evolved by increasing the CPU clock frequency to enhance the performance. When this approach faced the peak frequency limit barrier due to heat and exponential increase of power consumption, mobile CPUs have shifted to multi-core architectures [33, 142]. In recent years, smartphones have finally adopted core asymmetry as "big" performance-optimized cores, and "LITTLE" energy-optimized cores to improve the power efficiency and achieve higher performance levels [33].

Sensor technology on mobile devices is another feature that has also evolved over the years. Less than a decade ago, smartphones had only a few sensors; today they have more than a dozen sensors with additional capabilities and lower power consumption levels [106]. The evolution of sensors also brought various capabilities to mobile devices. For example, smartphones and smartwatches can recognize complex human activities and gestures, detect users' stress levels or emotional states, predict pin numbers and more. Despite the wide usage of multicore smartphones in the market, how much these cores impact end user experience still remains unclear. Since current operation systems manage DVFS (dynamic voltage-frequency scaling) based on the CPU utilization (CPUload), end users' satisfaction is (largely) ignored. However, prior work shows that user satisfaction on multicore smartphone varies between different CPU configurations [62]. Not all users are effected same from the CPU core/frequency changes even in the same workloads. Ad-

ditionally, there are still complaints on off-the-shelf phone performances in practice among users [150]. Therefore, there is a need to understand the users' performance satisfactions with their smartphones in order to manage computational resources on the phone. If users' performance satisfaction is known, we can manage mobile CPU settings in a much more efficient manner; we can reduce the power consumption if doing so does not cause dissatisfaction or increase performance when necessary. The challenge to achieve this is to be able to predict satisfaction accurately in real time.

In this project, we propose a system that utilizes motion sensor data, audio records, and touch events gathered from a smartphone and a smartwatch to predict user satisfaction and alter CPU configurations in the phone in order to save energy. The aim of the system is to maintain user satisfaction, while minimizing power consumption on smartphones. We evaluate our system by conduction two IRB approved user studies with a total of 30 users. In both studies, users were provided with a heterogeneous octa-core smartphone and a smartwatch. In the first study, we create a user-independent model and user-dependent models from users' workloads on seven common applications. In the second study, we make users operate the phone with the applications they choose to use while the user-independent model, their user-dependent models, or the default scheme controls the CPU. Our results show that, without impacting user satisfaction, user-dependent models and the user-independent model save on average 10.12% and 8.96% total system energy, respectively when compared to the default scheme.

Moreover, we study the relation between the sensor data, user satisfaction, and CPU configuration in more detail. We show that collected sensor data is strongly correlated with users' satisfaction. We also discuss the possible reasons behind the correlation of sensor data and user satisfaction.

Overall our contributions can be listed as below:

- We show that, user satisfaction can be predicted accurately using the sensor data gathered from both smartphone and smartwatch devices or from smartphone alone.
- We develop lightweight tools and methods to model user satisfaction using sensors from two

mobile platforms and alter CPU core/frequency based on these predictions in order to save energy in real time.

- We demonstrate that there is a strong correlation with CPU configuration, user satisfaction and collected sensor data.

The rest of the chapter is structured as follows. I introduce our test devices and logger application in Section 3.1. In Section 3.2, I describe our user tests and models. In Section 3.3, I explain our results. I present the analysis on the correlation of sensor data, user satisfaction, and CPU configurations in Section 3.4. I discuss on three important aspect of the proposed system in Section 3.5.

3.1 Methodology

In this section, I discuss our user study methodology. I first introduce the devices used during the studies. Then, I explain how we select the tested CPU configurations. Finally, I discuss our logger application along with utilized sensors.

3.1.1 Test Devices

We use a Huawei Nexus 6p smartphone and an LG Urbane 2 smartwatch in our user tests. The smartphone is powered by an octa-core Qualcomm Snapdragon 810 processor. This device has a heterogeneous big.LITTLE core (four 1.55 GHz Cortex-A53 and four 2.0 GHz Cortex-A57 cores) [1]. Nexus 6P smartphone was specifically chosen for our tests because of the trend in the smartphone market for increased heterogeneous core architectures [8]. Since Qualcomm Snapdragon 820/821 have quad core architectures, we chose Snapdragon 810 to cover a wider range of smartphones in the market (i.e. Huawei's kirin 960, Samsung's Exynos, Apple's A11, etc.). The software in our test device is a rooted version of Android 7.1 (Nougat). Open source Android platform enabled us to collect sensor data and rooted version enabled us to alter CPU configuration. We use LG Watch Urbane 2 as our smartwatch since its one of the most popular watch brands/models in

the market [121]. The software on the smartwatch is the open-source Android Wear platform (Android Wear 2.0) [52], and a general framework of C, C++, and Java code. The framework includes the Android Runtime Environment (ART), a variant of Java implemented by Google.

3.1.2 Choosing CPU Configurations

As I described in Section 2.3.2, we identified the least power consuming configurations by testing all possible CPU configurations. Thus similar to Section 2.3.2 we choose six configurations that only use little cores: 2 or 4 active little cores with their frequencies fixed at either minimum (384MHz) or medium (960MHz) or determined by the on-demand governor. These six configurations span the power spectrum in Figure 1(a). In addition, we allow our system to choose the default configuration (all cores activated with on-demand governor frequency). Overall, we utilize little cores as much as possible and activate the big cores by switching to default scheme if they are needed to maintain user satisfaction.

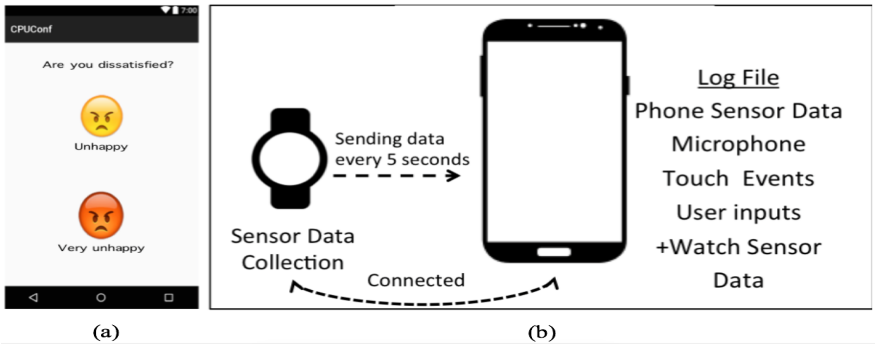


Figure 3.1: (a) GUI on the phone to collect 2-level (unhappy and very unhappy) user reports about phone's performance and (b) logger application's data collection from both smartphone and smartwatch devices.

3.1.3 Logger Application and Sensors

We develop a sensor-logger application to collect sensor data from both test devices. The application generates two apks (phone and wear) that can be installed to both devices independently. Additionally, if these two devices are connected to each other through Android Wear application [16] (Google's default application to establish and maintain connection between smartphones and

Table 3.1: Collected sensors and features for our models. "P" indicates that the corresponding sensor/feature was collected from the phone and "W" indicates that corresponding sensor/feature was collected from the watch.

Sensor/Feature	Explanation
Accelerometer (P, W)	We collect acceleration force along the x, y and z axis in 5Hz sampling rate.
Gyroscope (P , W)	We collect rate of rotation around the x, y and z axis in 5Hz sampling rate.
Touch events (P)	We collect touch events from smartphone using logcat [10]. Events include touch times, touch sizes in max, min and avg. and touch x-y coordinates in 2D phone screen. Data collection occurs by user inputs.
Microphone- Audio (P)	We collect ambient audio amplitude (in decibel) in raw, max, min and avg. values in 2Hz sampling rate using the microphone.
Heart Rate (W)	We collected heart rate (beat per minute) data in 1Hz sampling rate.

smartwatches using Bluetooth), once the application is installed to one of the devices, it automatically syncs and pushes its connected device's apk from one to another to be installed.

The logger is developed as a normal Android Runtime (ART) executable using the Java standard libraries available in the Android and Android Wear frameworks. Thus, it runs on all Android smartphones and Android Wear smartwatch devices without any special hardware or OS support. At a high-level, the logger application consists of two parts: (1) a GUI, which creates a notification on the upper bar on the screen and looks like a normal smartphone application and (2) an associated background service to provide logging functionality.

- *The GUI application:* GUI application is designed only for the smartphone. When opened by the user, the GUI begins the background services in both smartphone and connected smartwatch. When it is stopped, along with its own service, it also stops smartwatch's background service. This control over smartwatch helps us to synchronize the data collection from these two independent devices. GUI application also pops a notification on the upper bar of the phone's screen. This notification always stays on the screen to gather current user satisfaction reports about the phone's performance in 3 levels. We made this notification to enable an easy access for user inputs. Figure 3.1(a) shows the notification bar when swiped down. In the figure, faces correspond to states as unhappy (above) and very unhappy (below). We log them as user satisfactions in 3-level: 0 very unhappy, 1 unhappy, and 2 (default) happy

in numeric values.

- *Background Service:* Background service is the main part of the application. It is developed to handle functions such as data collection and setting CPU configuration in the test devices. In the smartwatch, background service is responsible for collecting sensor data and sending it to the phone every 5 seconds. In the smartphone, background service is responsible for 1) collecting sensor data and receiving smartwatch data, 2) predicting current user satisfaction using the developed models, and 3) altering CPU configurations to reduce power consumption while maintaining user satisfaction. Figure 3.1(b) explains the data collection from background services for both devices. The data is accumulated as log files in the phone. To sync watch's data with phone's data in the log files, we also put time stamps on both devices. We observe that our background services are lightweight mainly because of the small frequency of data collection. We observe that logger application increases CPU utilization by no more than 3% on both devices. Additionally, activating the logger increases energy consumption by 2.8% on the phone and 9.2% on the watch on average. We measured the overhead of the framework by using the experimental setup described in Section 2.3.2 (using Monkeyrunner-tool on three different applications).

Android smartphones have built-in sensors that measure motion, orientation, and various environmental conditions. These sensors are capable of providing raw data with high precision and accuracy, and are useful in many situations (e.g., monitoring three-dimensional device movement or positioning). In our proposed system, to be able to predict user satisfaction in real time, we collect data from various sensors. Table 3.1 lists the selected sensors/features and sampling rates used for our models (we describe the reasons behind this selection in Section 3.4). As shown in the table, we collect motion sensor, audio records and touch events from the target smartphone.

Smartwatches are sensor-enabled technologies for mostly health and fitness purposes. Since smartwatches are worn and embedded to arm, they may provide higher quality data on capturing some user behaviors compared to smartphones. Intuitively, it is possible that a user can hold the phone still while moving the arm with the watch based on her/his emotional state (e.g., boredom).

Table 3.2: Seven most common applications used in the lab to develop prediction models.

Application	Task
Google Chrome	Users search for news, games, and weather.
Game	A CPU intensive car race game is played [28].
Video	A preloaded video is watched from SdCard through built-in Media application on the phone.
YouTube	Users search and watch a video.
Google Maps	Users search cities from different continents and zoom-in/zoom-out.
Facebook	Users check their news feed.
Instagram	Users check their home/search tab.

Moreover, prior work shows that heart rate is a good indicator for detecting high/low arousal situations [44]. Therefore, in order to see the smartwatch sensors' effect on detecting users' behaviors based on their satisfaction levels, we use the target watch's motion and heart rate sensors in our study. We found these features give more insight about users' current satisfaction. We explore each feature and the correlation of sensor data and user satisfaction in Section 3.4. Additionally, we collect current and voltage values at 2Hz sampling rate to calculate power consumption on the smartphone.

3.2 Experimental Study

In this section, I discuss the user studies we perform. We create user-independent and user-dependent models in the first user study and then test them in the second one. The purpose of these experiments is 1) to help us understand the impact of the number of cores of a smartphone on user satisfaction, 2) to analyze how user satisfaction correlates with the collected sensor data, and 3) to examine how user-independent and user-dependent models differ from each other and the default scheme in terms of user satisfaction and power consumption.

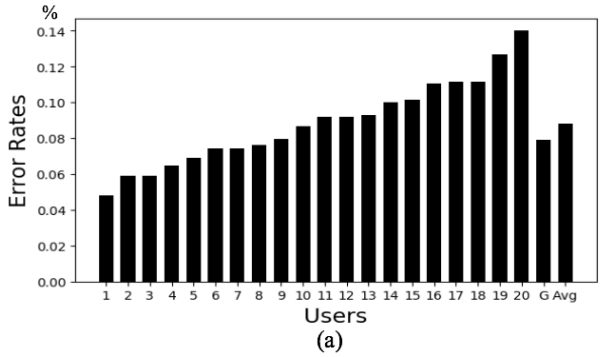
3.2.1 First User Study: Building User-independent and User-dependent Models

We first conduct user tests with 30 real users in order to develop a user-independent and user-dependent models. All the users were state-of-art smartphone owners and were under 50 years old.

Participants are gathered through fliers advertising the study and word of mouth (i.e., snowball sampling) and were provided gift cards with an amount of $10 for their participation. Users are provided with the test devices (a Nexus 6p smartphone and an LG Urbane 2 smartwatch) described in Section 3.1.2. Before starting the experiment, users are first familiarized with the devices for 10 minutes. In this introduction state, we make users login to their Google, Facebook, and Instagram accounts to be used during the test. For the users, who do not have these accounts, we create one at the moment. Then, each user performs seven predefined tasks (we choose the seven most commonly used applications [99]) in random order. Table 3.2 shows selected applications and the performed tasks. Since most of the applications require Internet connection, we conduct tests in a high speed WiFi environment with a minimum speed of 24.27Mbps.

User test begins when sensor-logger application is started by opening its GUI on the smartphone. While users are performing the tasks, sensor-logger application collects the sensor data shown in Table 3.1 and sets the CPU to one of the seven selected configurations (described in Section 3.1.3 and depicted in Figure 3.1 (b)) in random order. Hence, users do not know about the data collection or core/frequency changes on the phone. We make users experience each configuration for 1-minute duration for each application. Hence, a typical user test takes less than an hour (about 50-55 minutes with the introduction at the beginning). We specifically keep the experiment under an hour to keep users' attention on the tasks. Users are asked to use the notification bar to report their dissatisfaction whenever they felt so during the experiment. This process is also practiced during the introduction state. We use these reports to label collected data as 0, 1, or 2. Therefore all collected data along with the current CPU configuration is labeled as either 0,1 or 2 based on the inputs provided from users. We then use these labels (inputs) to form supervised learning models.

After collecting 30 users' data, we first create a *user-independent model* by accumulating first 10 users' data. For the remaining 20 users, we create *user-dependent models* for each of them from their own data. These last 20 users participate in our second user study later, where they compare their own user-dependent models, the user-independent model (note that the user-independent model was created using the first 10 users, hence it does not contain any information from this



(a)

	User-dependent Models			User-independent Model		
3-level satisfactions	TP	FP	Precision	TP	FP	Precision
0 - very unhappy	0.934	0.011	0.955	0.927	0.016	0.941
1 - unhappy	0.943	0.004	0.953	0.939	0.017	0.931
2 - happy	0.985	0.054	0.970	0.969	0.047	0.967

(b)

Figure 3.2: (a) Absolute mean error rates of 20 user-dependent models, user-independent model (G), and their averages (Avg). (b) Average of True Positive, False Positive and Precision of 3-level satisfaction reports (0-very unhappy, 1-unhappy and 2-happy) for both user-dependent and user-independent models.

second study group), and the default scheme. We found that REPTree algorithm in Weka-machine-learning-tool [139] provides highest accuracy with small overhead on the predictions of both user-independent and user-dependent models. We use 10-fold cross validation to prevent training and testing data overlap. Figure 3.2(a) shows the mean absolute error rates of the user-independent and 20 user-dependent models in ascending order. As shown in the figure, there is a large variation on accuracies of the user-dependent models. For some users, collected sensors can predict user's satisfaction of the phone with higher accuracy than others. This variation could be explained in two ways. First, not all users are affected the same from changes in the CPU; some users are more erratic than others. Also, different types of sensors on mobile devices may provide different levels of insight into users' satisfaction. As a matter of fact, we observe a similar variation when we match user reports with the configurations. Even for the same configuration in the same workload, user reports show differences. Nevertheless, with the selected sensors, we can still accurately predict user satisfaction. As shown in the Figure 3.2(a), majority of the error rates are less than 10%: 8.6% on average for the user-dependent models and 7.9% for the user-independent model.

Table 3.3: Mean absolute error rates of the user-independent model with different user groups.

Training/Testing Group	1/2+3	2/1+3	3/1+2	1+2/3	1+3/2	2+3/1
Mean Absolute Error	8.743%	8.850%	7.851%	9.881%	8.968%	9.246%

We also extract true and false positive rates of satisfaction predictions for both user-dependent and user-independent models. Since we can classify satisfactions to set CPU configurations instead of absolute predictions, false positive rates provide a better view of how accurate we can set the CPU core/frequency to keep satisfaction high. In order to classify satisfactions, we convert our 3-level satisfaction numeric values to nominal values as "zero", "one" and "two". Using Weka-tool with REPTree algorithm we measure the confusion matrix. Table in Figure 3.2(b) shows the false positive, true positive and precisions of satisfaction classification averages of all user-dependent models and the user-independent model. As shown in the table, although average of user-dependent models' precisions are slightly better than the user-independent model, both models can still accurately classify user's satisfaction in 3-level. Due to lack of space, we do not present the detailed results when the satisfaction is converted to a binary metric; however, it is worth mentioning that the prediction accuracy increases drastically when we reduce the number of levels to two (i.e., 0 and 1 mapped to "unhappy" and 2 mapped to "happy"). We measure absolute prediction accuracy as 97.1% if only binary satisfaction levels are predicted.

3.2.2 Building the User-independent Model with Different Groups and Group Sizes

To understand the impact of number of users in the training and testing groups, we conduct the following experiment. We randomly divide the data from our 30-user group into three groups of 10 users each: Group 1, Group 2, and Group 3. Then, in the first category of predictions, we use one group's data to train the model and use the model to predict the satisfaction of the remaining two. Then, in the second category of tests, we combine two groups' data for training and use the model to predict the satisfaction of the third group. All results are presented in Table 3.3. The caption indicates the training and test data in the form of 'training set/testing set'. There are two conclusions that can be drawn. First, the predictions are relatively stable across the group

selections. For the first three results, the mean absolute error rates vary between 7.9% and 8.9% and for the second set of results, the error rates vary between 9.0% and 9.9%. Second, when the training set size is increased, the error rate increases. We advocate this to overfitting. It is possible that the testing set of 10 users have a few outliers. When the training set is large, the model will overfit to the general trends and perform worse for these outliers. To further investigate the impact of group size, we have focused on the first experiment (Group 1 training and Groups 2 and 3 tested). Specifically, we have randomly divided the users in Group 1 into two groups of five (Group 1A and 1B). Then, we trained the modeled with each and tested on the remaining groups (Group 2 and 3). The mean absolute error was 15.5% for the training set of Group 1A and 14.1% for Group 1B. The increase in the absolute error rates is not surprising: with only 5 users, there is not enough training data to capture variety of behaviors. These results suggest that the set of 10 users is a good size for training for our participant population.

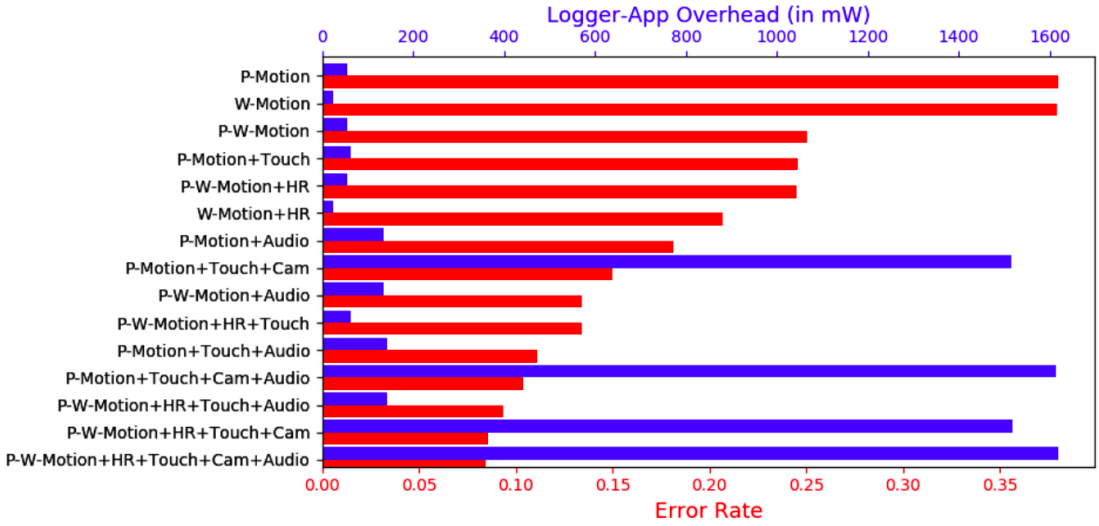


Figure 3.3: Average absolute mean error rates (sorted in red bars) for 3-level satisfaction predictions of the models and average power consumption (blue bars) of logger application for each sensor/feature combinations.

3.2.3 Accuracy of the Models With and Without Smartwatch Sensors

While studying the models, we also tested different sensor combinations at different sampling rates using the methodology described in Section 3.1.2. Along with the smartphone sensor combinations

we also include smartwatch sensor. Considering that not all smartphone users have smartwatches in their daily lives, we were interested in discovering the accuracy of our models without the smartwatch.

Figure 3.3 shows the importance of each sensor in the models and their overhead to the phone. In the figure, y-axis shows the sensors and features tested to build models, top x-axis (blue bars) shows the logger-application's power consumption on the phone and bottom x-axis (red bars) shows the error rates of satisfaction predictions. In the figure, "P" stand for phone, "W" stands for watch, "Cam" stands for camera, "HR" stands for heart rate (beat per minute), and "Audio" is the ambient noise in decibel collected from microphone. As shown in the figure, phone-only sensors (P-Motion + Touch + Audio) still provide a reasonable accuracy (89.0%) with reasonable overhead (on average 1.8% increase in CPU utilization and 144mW increase in power consumption). We observe microphone in audio feature consumes 80-100mW more power on average, while touch and motion sensor collection stays in the 10-30mW range. More importantly, false positive rates with phone-only data is also small, 5.5% on average. Since each test is performed same duration during power comparisons, sensors' energy consumption comparisons are also same.

Smartwatch sensor data drops the relative absolute error rate by 1.8% and false positive rate by 0.5% on average in the models when combined with the phone data. We also observe background service on the watch increase its power consumption by 9.2% on average (motion sensors consume 5-10mW and heart rate consumes less than 5mW power on average).

Even though watch data slightly improves accuracy in the models, a designer may decide to ignore it due to its overhead; in such a case, we can use only phone sensors to predict satisfaction. However, in order to analyze the relation between smartwatch sensors and user satisfaction as well, we use the watch data in both user studies.

We should also note that, as shown in the Figure 3.3, we tested camera feature as well while choosing sensors/features. We use Google's Face API [11] to detect smile probability of the users using the phone's front camera in 6 different sampling rates from 0.2Hz to 5Hz. Even though we observe a slight improvement of accuracy (1.0%), camera feature incurs a large overhead:

20% increase in CPU utilization and around 40% (1000-1300mW) additional power consumption. Thus, in order to keep the CPU overhead minimum, we excluded the camera feature and built the models using features listed in Table 3.1, which give the most accurate results among the users without using the camera (Figure 3.3).

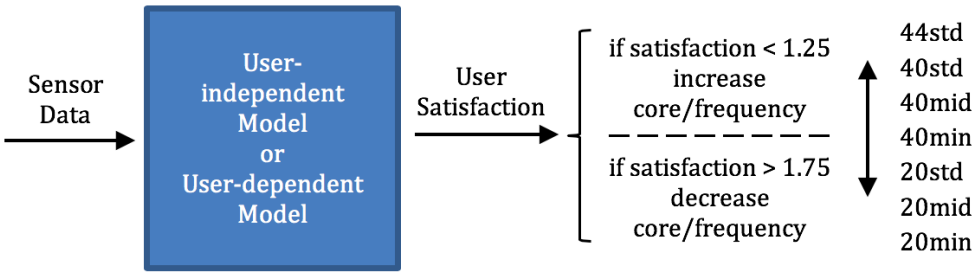


Figure 3.4: Flow of predicting user satisfaction and setting CPU configuration in real time.

3.2.4 Second User Study: Comparing the User-independent Model, User-dependent Models and the Default Scheme

Once the user-independent and the user-dependent models are developed, we conduct a second user study to compare their performance on predicting user satisfaction and altering CPU configuration in real time. In this study, we use the same 20 users who have a user-dependent model (from the previous user study). Users are again provided with the test devices: Nexus 6p smartphone and LG Urbane 2 smartwatch. Before starting the study, users are asked to install or login the applications they typically use. In this study, we do not put any limitation on the application selection for the users.

As described in the first user study (Section 3.2.1) we build models by setting CPU configurations in random order. In the second user study, CPU configurations are set based on to the two pre-loaded models' (user-independent and user-dependent) satisfaction predictions on the user. We make the logger application to decide tested models in random order and change them every 20 minutes. In order to compare both prediction models with the current default scheme, we also make users test the default scheme for 20 minutes. During the default scheme, no sensor data is collected by the logger application and CPU is set by the standard on-demand governor. During

the study, users do not know which model (or scheme) they are testing. They use the given devices and applications similar to the way they use their own phones.

Figure 3.4 provides an overview of the flow of prediction and setting CPU configuration in real time. A more detailed pseudo-code is same as described in Algorithm 1 in Section 2. As shown in the figure, first, collected data is fed to developed models. Model can be either user-independent or current test user's user-dependent model (note that the models are built in the first study and loaded to the phone offline).

Then, instantaneous user satisfaction is predicted using these models. Since models are trained with 3-level satisfaction inputs (as 0-very unhappy, 1-unhappy and 2-happy), their predictions also range from 0 to 2. We predict the satisfaction every 5 seconds. If the prediction is higher than 1.75, user is satisfied enough with the current CPU configuration and it can be dropped in the order to save energy (e.g., switch it to 20std if the current configuration is 40min as shown in the order in Figure 2.5(b)). If the prediction is less than 1.25, CPU configuration is moved up (i.e., performance is increased) to prevent any potential user dissatisfaction. Additionally, each user tests the default scheme, where core counts and frequencies are set by the default on-demand governor in Android OS. We determine the upper (1.75) and lower (1.25) bounds by conducting experiments with several levels in our lab. It is possible that other levels may be preferable by different developers. These levels could also be changed to make our system more (or less) aggressive. However, we found these bounds to work well in general.

During the second user study, in order to minimize the variation across different runs, users are asked to repeat the same applications every 20 minutes. A supervisor guides them throughout the experiment. We also collect users' overall satisfaction reports at the end of each 20 minutes through a pop up questioner on the phone screen. After the experiment starts, the questioner is shown to users every 20 minutes. In the questioner, users answer the question "How was your phone's CPU performance for last 20 minutes?" from a 5-scale radio style selection: 1 is worst, 5 is best. We collect ratings for last 20 minutes' performance in order to analyze how different CPU management models affect the overall user experience.

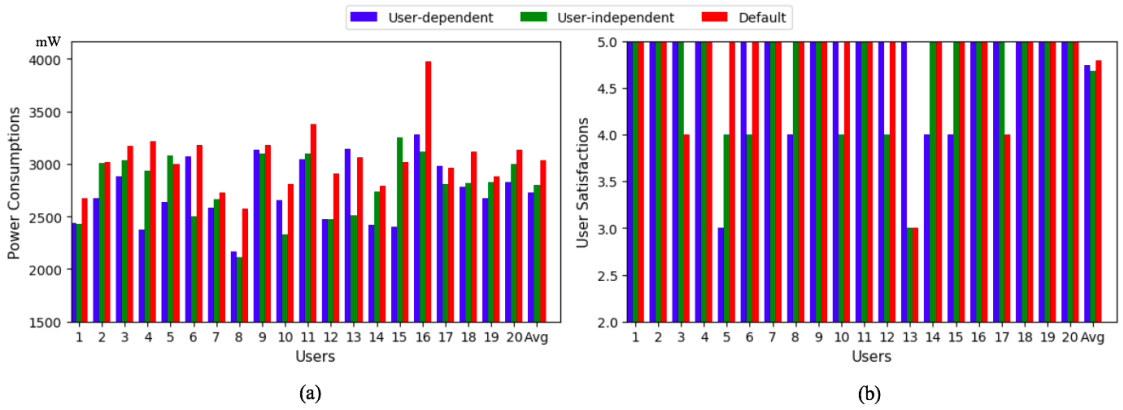


Figure 3.5: (a) Average power consumptions (in mW) and (b) user satisfaction reports for each model for 20 users.

3.3 Test Results

In this section, I present results of the second user study to compare the user-independent model, the user-dependent model and the default scheme. Figure 3.5 presents (a) the average power consumption (in mW) and (b) the reported overall satisfaction for the 20 users in our experiments.

As shown in Figure 3.5(a), the default scheme's power consumption is highest for most of the users; 16 out of the 20 users experience the highest power consumption with the default scheme. Since we do not limit application selection for our users, we also observe a large variation between power consumption levels of each user. Overall, our models are successful in saving power: user-independent (for user 16) and user-dependent models (for user 15) can save up to 21.58% and 20.4% of the total system power, respectively, compared to the default scheme. On average, user-independent and user-dependent models save 8.96% and 10.12% system power, respectively. Please note that these power savings are achieved despite the fact that our logger increases the power consumption since it collects sensor data, which is not done during the experiments involving the default scheme.

Figure 3.5(b) plots the satisfaction reports gathered from each user at the end of each model testing (through pop up questioner on the phone screen). As shown in the figure, there is a large variation in user satisfaction levels. We observe that some users are more sensitive to the CPU changes and even a short latency on responses make them drop their scores to lower rankings on the

Table 3.4: Statistical difference comparisons of satisfaction ratings. (a) Mean and standard deviation values of satisfaction ratings for each model. (b) Results of Friedman and ANOVA tests.

Models	Mean	Std
User-dependent	4.73	0.55
User-independent	4.68	0.57
Default	4.78	0.52

(a)

Friedman Test		ANOVA Test	
Chi-Squared	p-value	F – Score	p-value
0.451	0.797	0.166	0.847

(b)

tested model. On average, the default scheme seems to slightly outperform the user-dependent and user-independent models: the average satisfaction levels are 4.78, 4.73, and 4.68 (Table 3.4(a)) for the default, user-dependent, and user-independent schemes, respectively. A closer look, however, indicates that the differences between the schemes are negligible (in fact, we provide a statistical analysis in the following that shows this is indeed the case). First, the 0.05 difference is a single point from a single user. In other words, if one of the users rated their user-dependent scheme one point higher, the average would reach the same level as the default scheme. Second, out of the 20 users, only 1 rated the default scheme higher than both models. On the other hand, 3 out of 20 users selected either the user-dependent and/or the user-independent model as the best. This indicates that most users could not distinguish the differences between the schemes. Since the proposed system is implemented as a software mechanism, it can be optional to users and can be turned on/off based on user's selection (i.e., it can be deactivated for 1 user who rated default scheme higher). The power consumption levels are presented in Table 3.5(a). User-dependent and user-independent models save 10.12% and 8.96% of the system power, respectively.

In order to analyze the satisfaction and power consumption results further, we performed ANOVA [19] and Friedman tests [50] to see how these three models' results differ from each other. Table 3.4(b) shows the p-values of satisfaction comparisons and Table 3.5(b) shows the p-values of power consumption comparisons. As shown in the Table 3.4(b), p-values for satisfaction comparisons are higher than 0.05, which indicates that result is not significant and means do not differ. Since p-values are not significant, we did not conduct pair-wise t-test on the models. However as shown in the Table 3.5(b), p-values of power consumption comparisons are lower than

Table 3.5: Statistical difference comparisons of power consumption levels. (a) Mean and standard deviation values of power consumptions (in mW) for each model. (b) Results of Friedman and ANOVA tests. (c) Results of post-hoc pair-wise t-test p-values.

Models	Mean	Std
User-dependent	2728.73	291.44
User-independent	2763.10	296.82
Default	3035.939	284.53

(a)

Friedman Test	
Chi-Squared	p-value
19.3	6.443e-05
ANOVA Test	
F- Score	p-value
5.683	0.0056

(b)

Models	Default	User-independent
User-dependent	0.0072	0.5387
User-independent	0.0262	-

(c)

0.05, which indicates that result is significant. In order to understand how each model differs from each other, we conduct post hoc pair-wise t-tests [102]. Table 3.5(c) shows the p-values acquired from the pair-wise t-tests. As shown in the table, p-values for user-dependent and user-independent model vs default model are less than 0.05, meaning that there are differences in the means of these groups. By observing both Table 3.4 and Table 3.5 we can conclude that proposed models can indeed save power without impacting user satisfaction.

3.4 Correlation of Sensor Data With User Satisfaction

In this section, I analyze the relation between collected sensor data and user satisfaction in more depth. First, I show the correlation of sensor data and user satisfaction using all accumulated data (with all 30 users) to see the general patterns. Then I discuss how each user's sensor values change based on their satisfaction.

3.4.1 Correlation of Sensor Data With User Satisfaction

As described in Section 3.2.1, we use REPTree algorithm in our models. In the algorithm, predictions were made based on the branches formed by the information gained from each feature. To be able to see how much information is gained from each feature, we extracted them using Weka-tool's "InfoGainAttributeEval" evaluator from Attribute Selection [138]. Table 3.6(a) shows accumulative information gain of each feature group. To make a clear view in the table, we accu-

Table 3.6: (a) Total information gained from each sensor group. (b) Average user satisfactions for each configuration and defined configuration groups.

Feature Group	Information Gain
Audio	2.291
Touches	1.275
Watch's accelerometer	0.714
Phone's accelerometer	0.231
Heart rate	0.122
Phone's gyroscope	0.046
Watch's gyroscope	0.038

(a)

CPU Configurations	Average Satisfaction	Core Groups
44 std	1.95	Higher Cores
40 std	1.98	
40 mid	1.96	
40 min	1.61	Middle Cores
20 std	1.54	
20 mid	1.25	
20 min	0.23	Lower Cores

(b)

mulated same set of sensors and grouped them in the row. In the following sub-sections, we discuss each sensor individually. Table 3.6(b) presents the average user satisfactions reported for each CPU configuration. As shown in the table, four active cores operating in middle and on-demand governor frequencies have similar satisfaction averages. Same similarity shows itself for two active cores operation in middle and higher frequencies. Specifically, we group these 7 configurations into 3 groups based on the average satisfaction levels reported: 40mid, 40std, and 44std have all close to perfect satisfaction and are labeled "Higher", configurations 20mid, 20std, 40min have slightly lower satisfaction levels and are labeled as "Middle", and 20min is marked as "Lower".

As shown in Table 3.6(b), we observe that in general when users are provided with fewer active cores and lower frequencies, they become dissatisfied with their phone's performance. Apparently, one of the obvious reasons is the long latencies in responses and delays in computations of these fewer cores. But since not all users are affected same from these fewer cores and frequencies, user preferences and expectations are also vital on understanding impact of number of cores on user satisfaction. It is also worth to note that, while satisfaction is increasing with the higher number of cores and frequencies, it slightly drops when the CPU is set to default scheme (44std). Since standard governor dynamically scales core and frequency depending on the CPU load, fewer cores and lower frequencies may be set in time and users may be affected negatively from these changes. This is also a similar result we acquired in default scheme's reports in Section 3.2.

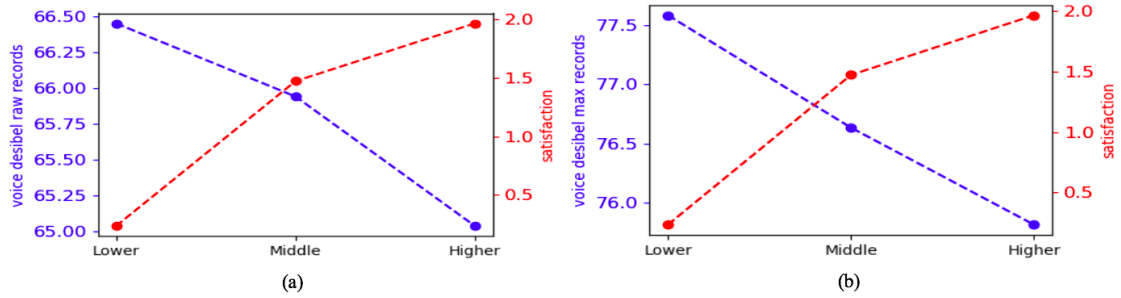


Figure 3.6: (a) Averages of raw audio records in decibel and (b) averages of maximum audio records in decibel.

Audio

Audio is the most information gained feature as shown in Table 3.6(a). We collect ambient audio amplitude (in decibel) from our users in 2Hz sampling rate using Android AuidoRecord library [21] with the microphone. Target device has built-in MEMS microphones, which enable ambient noise canceling to improve input voice quality [43, 41, 7]. Along with the raw records, we also logged their statistics such as mean, maximum, and minimum values in 5-second time windows. From these statistics, we observe raw audio records and maximum audio records give the highest information. In Figure 3.6(a) and Figure 3.6(b), we plot averages of these audio records for each CPU configuration group (lower, middle, higher). In order to compare the data with user satisfaction levels, we also plot average user satisfaction for each configuration group on the right y-axis (red).

In the user studies, we observe that when users become unsatisfied with their phones' performance (due to long latencies on responses or skipped frames on the screen etc.), they tend to lose their interest with the tasks/applications. This results in some behavioral changes among the users including starting a verbal communication with the supervisor, adjusting sitting, or changing phone holding positions.

We first monitored these changes in audio records. We observe on the majority of the users that when they face increased latencies or unresponsiveness on the phone, they chose either to start a conversation, yawning, or tapping until the phone becomes responsive again. Thus, when

they are unsatisfied, users tend to give up on the current task and devolve into other activities [75]. Therefore, in Figure 3.6(a) and Figure 3.6(b), we observe negative strong correlation with decibel records and user satisfaction reports: decibel records drop as satisfaction increase.

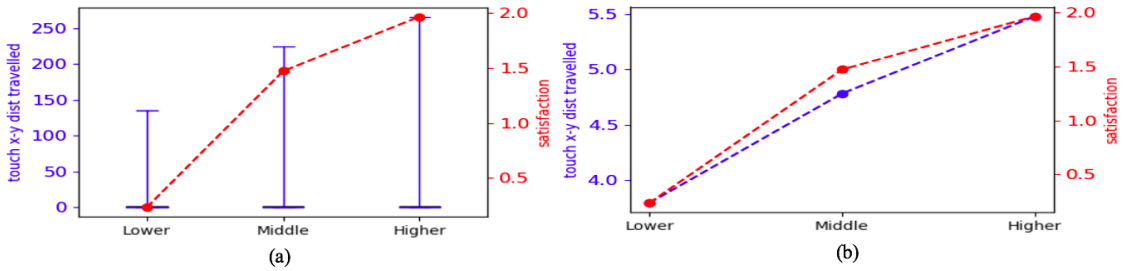


Figure 3.7: (a) Distribution of x-y coordinate distances on the screen for each configuration group and (b) averages of x-y coordinate distance on the screen touch for each configuration group.

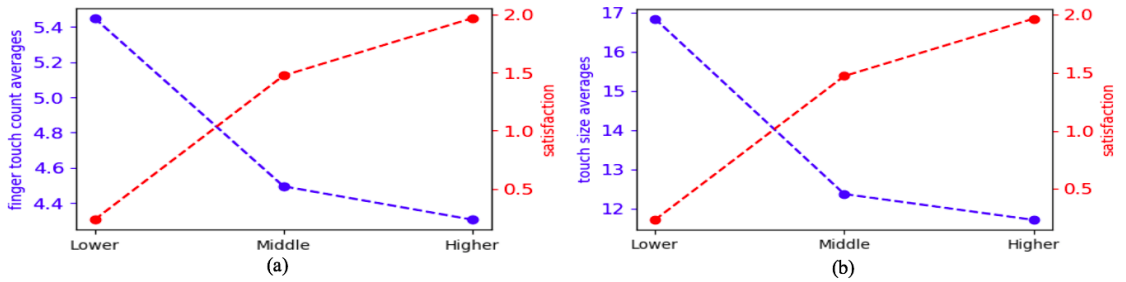


Figure 3.8: (a) Averages of number of touch counts on the screen for each configuration group and (b) averages of touch inputs sizes for each configuration group.

Touch Events

We collect touch events as shown in Table 3.1. From these events, we observe recorded x and y coordinate values on the screen give the highest information gain (they compose 1.13 of 1.27 information gain) in the models. These values show the touched screen point in two dimensions (2D). In order to plot a clear view, we calculate the distances from (0,0) point to each touched point with the Pythagorean formula:

$$distance = \sqrt{x^2 + y^2}$$

We observe a large variation between the distances in configuration groups. Figure 3.7(a) shows the distributions of the calculated distances for each configuration group. In the figure, whiskers

extended to include all data. As shown in the figure, variation in the distances increases as the satisfaction increases. Their standard deviations for lower, medium, and higher CPU configurations are 11.67, 16.14 and 17.43 respectively. Similarly, we plot the averages of these distances in Figure 3.7(b) and observe a similar trend. Finally, we plot the average of total finger touch counts and average of the touch sizes in Figure 3.8(a) and Figure 3.8(b). We calculate touch sizes from the horizontal and vertical elliptic areas on the screen given by "getevents" in Android [12]. For both figures, we observe that there is a trend that averages of touch counts and touch sizes decrease while satisfaction increases. Also, least satisfied configuration defined as "lower cores" (20min, i.e., only 2 active little cores with minimum frequency) has the highest number of touches and highest touch size on average.

As being the second most information gained feature, touch events are also significant component of our models. We observe that users give more touch inputs with bigger touch sizes when they are less satisfied with their phone. But we also observe that, their x-y coordinate touch distance variations and averages stay in minimum (Figure 3.7(a) and 3.7(b)). Assuming that unsatisfied configurations cause unresponsiveness, users may become more demanding by giving same or similar inputs (commands) on the screen. On the other hand, we observe that, in general x-y coordinate distance averages and variations increase as the user satisfaction increases. This shows that users touch (give input commands) more diverse positions on the screen with lighter touches when they are satisfied.

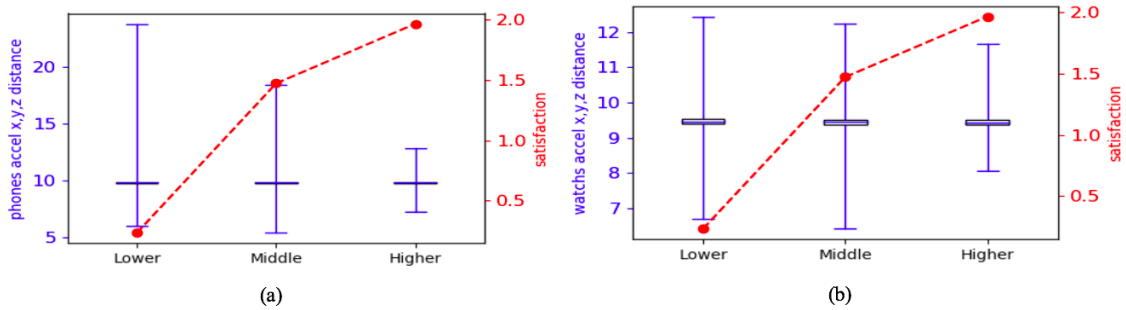


Figure 3.9: (a) Distribution of phone's accelerometer distances for each configuration and (b) distribution of watch's accelerometer distances for each configuration group.

Accelerometer

We collect acceleration force along the x, y and z-axis from both the smartphone and the smartwatch. These values show the force applied to the device from all three physical axes (including the force of gravity). Accelerometer sensor is commonly used in motion detections (shake, tilt, etc.). In order to get a clearer view of these three axes, we calculate the distance of them with the formula:

$$distance = \sqrt{x^2 + y^2 + z^2}$$

Figure 3.9(a) and Figure 3.9(b) show the boxplots of calculated distances from the collected sensors on the smartphone and the smartwatch, respectively, for each configuration group. Again we observe a large variation in the distribution of distance values across configuration groups. As depicted in the figures, the range and variation become smaller as the satisfaction increases. The standard deviations for lower, medium, and higher CPU configurations are 0.40, 0.21, and 0.12 for the phone and 0.24, 0.19, and 0.16 for the watch, respectively. We observe a clear decrease in the variation between distributions in phone's accelerometer data in Figure 3.9(a). For the most unsatisfied configuration (20min), we observe the accelerometer data is the most scattered.

Similar to the previous justifications on user behaviors, we notice that, users tend to move more when they become unsatisfied with their phone. This includes adjusting the seat or sitting position, shifting phone to other hand, shaking the phone, etc. Since these movements include 3D physical actions, we see a large variation with accelerometer data. On the contrary, we observe that users become more involved and focused with the tasks and applications when they are satisfied with their phone. As shown in the figures, this yields to more stable holding (lower variations in the distributions) of the phone and watch.

Heart Rate

In user tests, we collect heart rate at 1Hz rate from the smartwatch. Heart rate is measured from the wrist through a green LED lights sent to skin. Similar to the other sensors, we use Android's

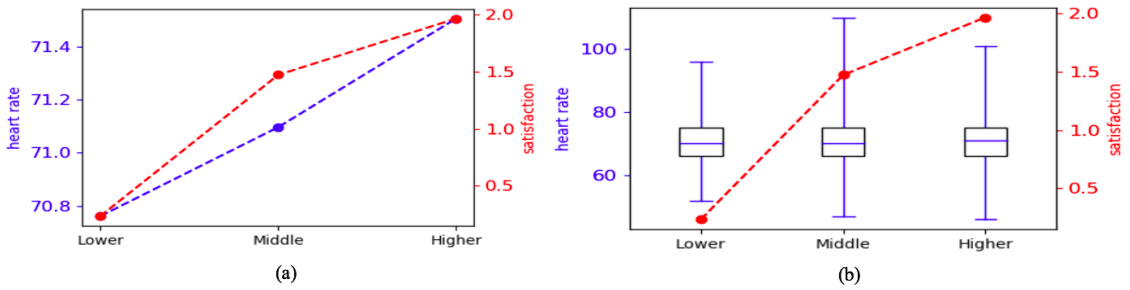


Figure 3.10: (a) Observed heart rate (beat per minute) averages from smartwatch's built in heart rate sensor for each configuration group. (b) Distribution of heart rate values for each configuration group.

built in API to monitor the heart rate. The API report heart rate as discreet current beat per minutes (BPM) values. For simplicity, we refer beat per minute values as "heart rate values" in this chapter. Figure 3.10(a) shows the averages of observed heart rate and Figure 3.10(b) shows the distribution of the these values, for each configuration. Although the average heart rate difference is small (70.6 for the lower versus 71.5 for the higher) and information gain from it is relatively small, we still observe a correlation between the heart rate values and user satisfaction. As shown in the Figure 3.10(a), heart rate averages increase when users are more satisfied with the phone's performance. Moreover, we observe standard deviations as 7.78, 8.60, and 8.06 for lower, middle and higher core groups, respectively.

There are many studies observing variation in heart rate and average heart rate under different emotional states. Studies [26, 44] show that users feel high arousal and higher heart rate values when they feel stress or anger. They make these observations by conducting stress inducing user tests (ice bucket test, social stress test etc.). Unlike these studies, our user experiments do not include any stress inducer workload. Moreover we select the seven most common Android applications when building the user-independent model (in the first user study) and do not limit any application selection in the second user study (users choose whichever application they want to use). Thus, when users are dissatisfied with their phone due to long latencies and unresponsiveness, we are potentially observing boredom rather than stress for most of our users. Boredom is defined as lack of interest [37] and a state of relatively low arousal and dissatisfaction, which is attributed to an inadequately stimulating situation [68]. There is a significant amount of studies

exploring boredom [39, 93, 68]. These studies mostly observe behavioral changes under boredom inducing experiments (i.e., time slowing experiment [93], reading boring story experiment [39], etc.). Unlike the stress inducing experiments, their results also show low arousal along with low variation in heart rate values when people face with boredom, which is similar to our observations.

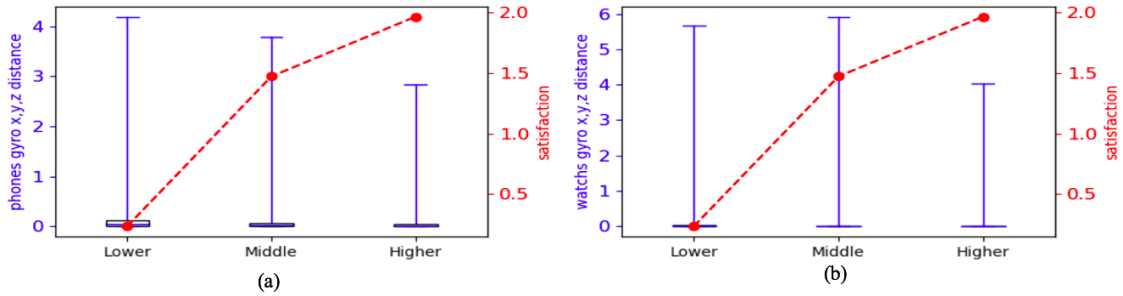


Figure 3.11: (a) Distribution of phone's gyroscope distances for each configuration group and (b) distribution of watch's gyroscope distances for each configuration group.

Gyroscope

We collect smartphone's and smartwatch's rate of rotations around each physical axis (x, y, z) using the gyroscope sensor on both devices. Gyroscope sensor is commonly used in rotation detection (spin, turn, etc.). In order to get a clearer view of these three axes, similar to Accelerometer Section, we calculate the distances of them using formula in Accelerometer Section.

Figure 3.11(a) (for phone) and Figure 3.11(b) (for watch) show the gyroscope distance distributions for each configuration group. Similar to Figure 3.9(a) and 3.9(b) (which present accelerometer data), in gyroscope data, we generally see a reduction in boxplot whisker sizes when user satisfaction increases and larger variation when users are unsatisfied. Their standard deviations for lower, medium and higher configurations are 0.29, 0.16, 0.12 for the phone and 0.32, 0.17, 0.09 for the watch, respectively.

3.4.2 Correlation of Sensor Data and User Satisfaction For Each User

In the previous section, I show the common patterns between sensor values, user satisfaction, and CPU core groups from all users' cumulative data. In this section I discuss the sensor values and

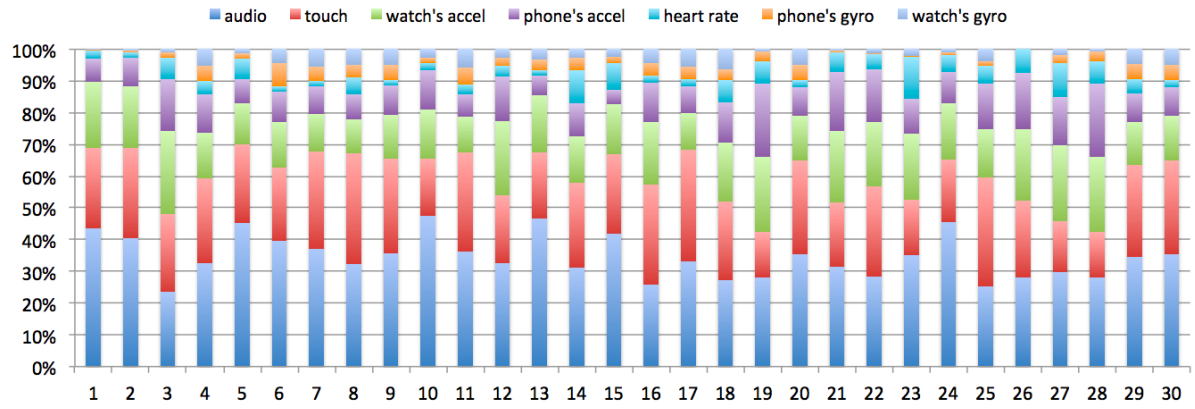


Figure 3.12: Proportions of sensor values based on their information gain for each user-dependent model.

user satisfaction relation for each user and show how each user's prediction models differ from one another.

Figure 3.12 shows the proportion of information gain from each sensor group for each user. For these tests, we have developed user-dependent models for the first 10 users as well (note that these users provided the data for the user-independent model and did not participate in the second user study where the user-dependent models we utilized). In the figure each bar sums up to 100 and each color represents a specific sensor group. Similar to Table 3.6(a), we accumulate same set of sensors and group them as one (for example, we accumulated all features acquired from the screen touches as described in Table 3.1 as "touch" in the figure).

In our experiments, we observe a large variation in user behaviors when they face dissatisfied CPU configurations. While some users do not notice and/or show any reaction to CPU changes, some other users are respond more erratically to these changes. However, we still observe audio and touch features as the most informative features across all users. This is not surprising since we still observe a low absolute mean error rate (10.82%) when we combine all users to build the user-independent model in Section 3.2.2. If the features would have been completely different across users, a) it would be hard to see a general pattern described in Section 3.4.1 and b) we should be seeing a higher error rate and higher dissatisfaction rates from the user-independent model in the second user study. Moreover, as described in Audio section, audio sensor records ambient noise

through built-in microphones in the target phone. Since the phone's or the user's position also effect the recorded ambient noise, audio sensor is sensitive to other environmental changes. For example, for user 3, while we do not observe any verbal indicator (tapping, yawning, etc.), we observe constant 3D movements when the user becomes dissatisfied. Since any physical change in the user or in the phone affects the recorded ambient noise of the environment, we still observe a significant information gain from audio sensor for this user. Overall, we observe audio and touch features roughly holds approximately between 40% to 70% of the total information gain for all users.

Motion sensors (accelerometer and gyroscope sensors for both devices) are another feature group that shows large variation among users. We observe that majority of the users make (big or small) physical movements when they become dissatisfied due to long latencies or unresponsiveness. On the other hand, there are still some users who show minimum (or no) movement and therefore their motion sensor's information gain is small (e.g., user 1). We also observe more information gain from the watch's sensor since it is embedded to arm and more exposed to any physical movement. Overall motion sensors are an important part of the models and they constitute 40% to 60% of total information.

Heart rate is the number of beats per minute and its sedative state is different from individual to individual. However, there is an expected range in sedative state heart rate based on some internal and external factors for the individuals (e.g., age, sports history, environment temperature etc.) [64]. Due to similarities of these factors in our users, we do not observe drastic variations in the heart rate values. Specifically, all selected users are between age 20 to 50 and none of them are either athlete nor have any heart disease. Moreover, since we do not include any high or low arousal inducer workload/activity in the studies, we do not observe a large variation in heart rate values. Having said that, there are still some users whose heart rate is an important indicator for their user-dependent models. For example, for user 23, heart rate constitutes almost 10% of the total information gain. When we look at the user 23 in more detail, we observe that user's heart rate value averages show more diversity compared to other users (we observe average heart rate values

as 63.69 and 68.73 in dissatisfied and satisfied conditions, respectively). Furthermore, in 9 out of 30 users, we observe higher heart rate values when they face dissatisfaction. We believe that these users feel anger rather than boredom and show high arousal signals. In fact, heart rate variability in emotional states is still a discussion in the field of Psychology. Overall, although heart rate values are important for some users in their user-dependent models, for other users information gain from heart rate is small.

3.5 Discussion

Limitations of using "best" configurations for each user. In our system, we use sensors to predict users' satisfaction level on their smartphone's CPU performance in order to manage CPU configurations. Instead of using sensor values, one could identify some best CPU configurations for each user and manage CPU with these configurations. However there are two limitations of this approach:

- Configurations may be different for each application. As explained in the first user study (Section 3.2.1), we change CPU configurations randomly while running seven most common Android applications and ask users report their dissatisfaction through a notification bar on the screen. In this study, we observe large variations in user satisfactions with different applications. For example, user 5 did not report any dissatisfaction during the "Instagram" application; but s/he marks all configurations as dissatisfied except 40mid and 44std (4 little cores running in medium frequency and all 8 cores are active in standard governor) for "Google Chrome" application. If we were to find the best configurations for this user, we need to find the lowest power consuming configurations where the user will always be happy (i.e., 40mid). The result of selecting this configuration will be increased energy consumption for the applications where user can be satisfied with lower energy consuming core/frequency configurations. Moreover, it is still not guaranteed that selected best configuration(s) is the best for all applications. In this study, we work with seven applications and for some other applications user's best configuration(s) may be different. One solution to this problem could be identifying best configurations for each application. However, as the list

of applications that can be used is extensive, it would be more expensive (in terms of memory and computation) to create such a CPU configuration/application table. Moreover every time a new application is started, a similar methodology should be run in order to find best configuration for this application, which may degrade the user experience.

- Configurations are needed to be updated/changed as the user's preferences change. Another disadvantage of identifying some best configuration(s) for users is that it needs to be updated/changed dynamically if user's preferences change. It is possible that users may require different configurations based on their current activity, usage, mood, etc. In such cases, identified best configurations may be insufficient to meet the needs of the changing user preferences.

The proposed system utilizes built-in sensors in the smartphone to predict user dissatisfaction and manage CPU based on these predictions. We make the system as application-independent as possible specifically for the reasons mentioned above. Moreover, since the proposed system depends on built-in sensors, a new prediction model can easily be built online with the new user performance preferences.

Energy savings' effect on user satisfaction. There is a tremendous effort in the smartphone industry to find solutions to extend the battery life-time. Using batteries with more capacity could be a trivial solution, but unfortunately their technological evolution does not follow the trends dictated by Moore's Law. While the computational complexity is doubled every two years according to Moore's Law, the battery capacity is doubling only every decade. Designers have reverted to architectural and system-level optimizations to keep energy consumption down. Despite the importance of end user in smartphones, how much these efforts affect user experience is still not clear. Any improvement in performance or a new hardware feature is easily observable and can have a direct effect in user experience. However energy savings' effect is non-trivial and (possibly) has influence only in the long run. Thus, it is challenging to measure and factor its effects in instantaneous user experience/satisfaction. If users knew their smartphone's current energy savings, it is possible that their experience may be different. They may even be willing to sacrifice more (if it leads to more power savings). Similarly, if manufacturers knew their efforts' effects on user

experience, they can focus more (or less) on extending battery life-time.

In order to get an insight of how much battery life is important for users, we conduct a survey in Amazon Mechanical Turk (mTurk) [115] with 500 users. In the survey, we asked users to rank the most important feature they look in their smartphones. We give them only performance, battery, and brightness options. 58% of the mTurk users select performance as the most important feature in a smartphone while 38% choose battery. If we replace the performance with other hardware/software components (i.e., camera, operating system, etc.) we observe more than half of the users (53%) choose battery over any individual component. Other online surveys also reflect similar results [49, 51, 17]. Battery life-time is certainly an important aspect of user experience in smartphones but to the best of our knowledge there is no work on measuring its quantitative impact on user experience. Although we do not know energy savings' instantaneous effect on user experience, due to its importance to users, it is reasonable to expect that our system would be rated even higher since it is saving energy compared to the default mechanism.

Using the proposed system in real-world deployment. As explained in Section 3.2.3, in the proposed system, sensor collection and CPU setting is done by a lightweight logger application in real-time. Therefore, the proposed system does not require any special hardware or software and is executable in any Android smartphone and smartwatch. However, Android OS requires root permission (superuser) in order to alter CPU configurations in system-level. Since root access is not granted as default and having a root access violates the manufacturer's guarantee/insurance on the target phone, we conduct our user studies in the lab environment (Section 3.1) on 30 real users. Having said that, the proposed system can easily be adopted in real-world deployment scenarios due to its minimal energy consumption and simple to use user interface. Moreover, we use off-the-shelf devices so that inexpensive large-scale deployments can also be possible. As discussed in Section 3.1 and Section 3.2.2, activating the logger increases energy consumption by 2.8% on the phone and 9.2% on the watch on average. Even though watch data slightly improves predictions in user satisfaction in the system, a designer may decide to ignore it due to its overhead. In such a case, we can use only phone sensors to predict satisfaction and can still have high accuracy (over

89%) on predicting satisfaction in real time.

Another advantage of the proposed system is the small overhead of building a new model. In case user preferences change by time, the system can re-build the prediction model from accumulated user data on the fly and can make predictions with this updated model.

3.6 Summary

In this project, we study CPU management in mobile systems with respect to real end user experience. Specifically, we propose a system that alters the number of active cores and their frequencies in a multicore smartphone in order to save energy. The system benefits from the strong correlation of user satisfaction with motion sensors, audio records, and touch events collected from a smartphone and a smartwatch device. In the proposed architecture, we first collect sensor data along with current user satisfaction levels reported by the users with the phone's performance while varying the active cores in the CPU and their frequencies. Then, we predict users' instant satisfaction and alter CPU active core/frequency configuration in real time. We demonstrate that, we can predict satisfaction with over 91% accuracy with 3-level satisfaction model and with over 97% with binary satisfaction model.

We evaluate the proposed system by developing and comparing two different models. We develop a user-independent model with 10 users' cumulative data and user-dependent models for 20 different users. We made users compare the user-independent model, their own user-dependent model, and default scheme. Our results show that, on average user-dependent models and the user-independent model save 10.12% and 8.96% of the total system energy, respectively, without impacting user satisfaction.

To the best of our knowledge, this is the first study concentrated on built-in sensors, gathered from mobile devices, rather than system metrics on understanding user experience in multicore smartphones.

CHAPTER 4

USING BUILT-IN SENSORS TO CONTROL SCREEN BRIGHTNESS

The display is the primary user interface on many computing devices: ranging from traditional devices such as desktops and laptops to the more ubiquitous devices such as smartphones and smartwatches, the display is arguably the most important interface. While earlier displays were only an output system for these devices, the introduction of touchscreen feature made them the main input component as well. The absence of a proper keyboard and mouse hardware for smartphones increased the importance of touchscreen displays. As a result, we have observed increasing display sizes over the years. While the larger displays are ideal for various functionalities (e.g., video viewing), they also cause a high consumption of available battery power. Prior work shows that display power consumption can reach to 50% of available battery resources [107, 119, 147, 105]. A quick solution to reduce display power consumption is to dim the display backlight. While this can lead to significant power savings, it also has the possible side effect of degrading user experience due to reduction in the visibility of the display.

In order to address this dilemma (power savings vs. user experience), systems typically control brightness by measuring ambient light level: if the smartphone is under direct sunlight, backlight is increased in order to provide a better visibility whereas if the phone is in a dark room, backlight is dimmed in order to decrease power consumption as well as increase user experience (generally a too bright display is not desirable in a dark room). Although theoretically this implementation works well, prior work [113, 27] shows that this mechanism can be improved significantly. Using only ambient light sensor ignores the other sensors' effect on brightness settings and using a static one-size-fits-all model ignores varying user brightness preferences. Clearly not all users are the same and it is possible that some users might prefer different brightness levels even under same conditions. Therefore, there is a need to understand the user-specific brightness preferences in order to control display brightness on the phone. The challenge to achieve this is to be able to

predict desired brightness settings accurately in real time.

In this project, we propose a system that utilizes built-in sensors in a smartphone to get information about phone's and environment's state to predict the desired brightness level and set screen brightness accordingly in real time. The aim of the system is to increase display user satisfaction, while also minimizing (or at least keeping the same) power consumption on smartphones. We evaluate our system by conducting two IRB (Institutional Review Board) approved user studies with a total of 47 users and 23 different smartphone models/brands with varying API versions. We conduct both studies in the wild by releasing logger applications to market. In the first study, we create a user-independent (user-oblivious) model from 10 random users' cumulative data offline. In the second study, we create user-dependent models online for each user (different from the first study users) and make them use their phones while the brightness is controlled by the user-independent model, their user-dependent models, and the default scheme each day in a random order. User-dependent models are re-built constantly as the models learn more about users' brightness preferences dynamically in the wild. Our results show that, user-dependent models and the user-independent model can predict users' brightness preferences with over 95% accuracy and increase display brightness satisfaction by 12.62% and 4.81% on average, respectively, when compared to the default scheme. Moreover, in the proposed system, we implement a gradually dimming mechanism in order to save power especially in long usage scenarios. We set the dimming limit to up to 10 brightness level in order to prevent any user dissatisfaction and show that proposed system saves 8.26% and 5.08% system level power consumption with user-dependent and user-independent models, respectively, when compared to the default scheme.

Table 4.1 shows the brands/models and Android versions of the participants' phones. Note that, in both user studies, we collect data by releasing logger applications to Google Play Store. Therefore, we did not attempt to control or select any of our participants. The users are anonymous people who downloaded our logger application from the market and use it just like any other application they install from the market.

Moreover, we study the relation between the sensor data and screen brightness preferences of

users in more detail. We show that the collected sensor data is strongly correlated with users' brightness preferences. We also discuss possible reasons behind this correlation.

Overall our contributions can be listed as below:

- We show that we can accurately predict users' brightness preferences using built-in sensors on a smartphone.
- We show that using only ambient light sensor is not sufficient for capturing the variation in user brightness preferences and present data from other sensors that are highly correlated with these preferences.
- We show tools and methods to build and learn from users' brightness preferences in the wild and control brightness in real time.

The rest of the chapter is structured as follows. In Section 4.1, I present background information and motivate our approach. I then explain our experiments in Section 4.2. In Section 4.3, I discuss the results. I analyze the correlation of sensor data and user brightness preferences for all users in Section 4.4. I present further discussion in Section 4.5.

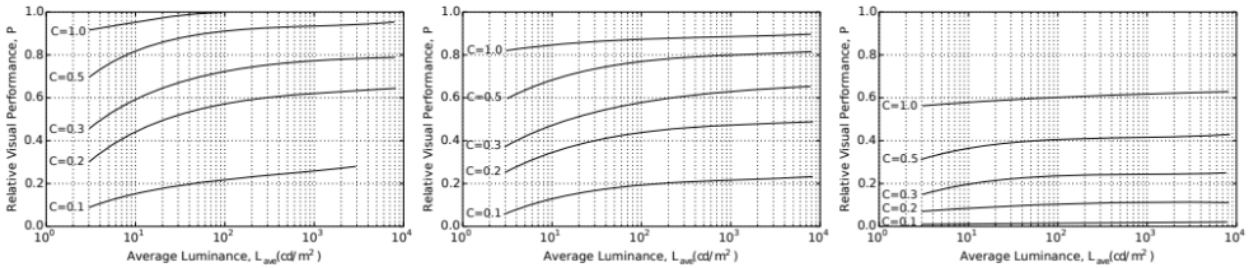
Table 4.1: Participants' Phone Brands/Models and API Levels

Advan eie Tablet 3G - 23	Samsung Galaxy S5-9 - 24, 25
Alco-RCT6973W43MD - 24	Samsung Galaxy S9 Plus - 24, 26
Huawei Mate 10 Lite - 24	Samsung Galaxy S7 Edge - 23, 24
Huawei MediaPad - 24	Samsung Galaxy on5, on7, - 26
Huawei - Nexus 6P - 25	Samsung J2, J5, J7 - 25
Huawei P9 - 24	Samsung Note 5 - 24
Lava Xolo - era3x - 24	Samsung-SM - 24, 28,29
Lenova K6 - 24, 25	Sony Xperia Z3 - 23, 24
Motorola-Moto G (4) - 24	TCL-5009A - 24
Motorola-Moto G (5) Plus - 25	Vivo - 28
Pixel-2 - 28, 29	ZTE - Blade V8 - 23

4.1 Background

4.1.1 Visual Performance

Even though there are many metrics to evaluate the quality of a display, we focus on visual performance, which is one of the most commonly used metrics. Rae et al. [109] define visual performance as the speed and accuracy of processing visual information and show that visual performance is mostly related to luminance and contrast ratio. Luminance is defined as the intensity of light per unit area as either emitted or reflected. The contrast ratio is the ratio of the luminance of the brightest color (white) to that of the darkest color (black) that the smartphone display is producing. A high contrast ratio is a desired aspect of any display. For instance, in a completely darkened room, the contrast ratio is precisely determined by the white and black pixels in the phone display. However, in a well-lit environment which involves a significant amount of ambient light, display's optical characteristics necessarily do not absorb all of the incoming ambient light and some of that light is reflected back to the viewer. This reflected light reduces the resulting contrast ratio. Therefore, modern smartphones use ambient light sensor to measure the incoming light level and control brightness accordingly in order to increase contrast ratio (i) for better user experience and (ii) for screen energy saving.



(a) 20 year old, 1.5' detail (b) 50 year old, 1.5' detail (c) 75 year old, 1.5' detail

Figure 4.1: Aggregated readability curves for users of varying ages. As people age, their ability to discern detail drops significantly at a given luminance and contrast ratio level. Source: [74]

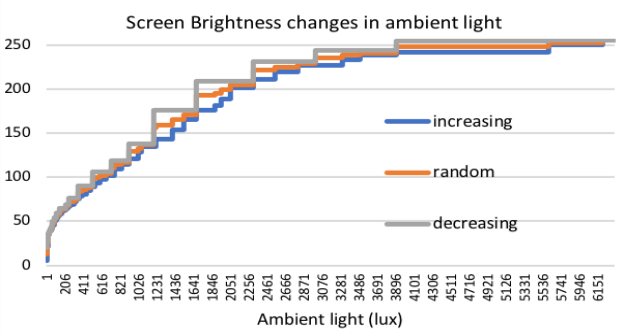


Figure 4.2: Default scheme's (ambient-only solution's) screen brightness settings in increasing, decreasing, and random order ambient light changes.

4.1.2 Mobile Displays

Liquid Crystal Display (LCD), Organic Light Emitting Diode (OLED), and Active Matrix Organic Light Emitting Diode (AMOLED) are the most common display technologies used in mobile devices. LCD is the most widely used display technology in the market. It is made up of liquid crystals that get illuminated by a fluorescent backlight. The liquid crystal matrix contains an array of red, green, or blue crystal light filters and controls the colors on the screen. On the other hand, OLED is a completely different display technology. It is a thin-film display technology that contains OLED, an organic material which emits light when current is passed through it. OLEDs display blacks better and consume less power (5.7% less on Samsung smartphones [31]) when displaying darker colors as OLEDs are always off unless electrified individually. AMOLED adds a layer of semiconducting film behind the OLED panel, which in turn allows it to activate each pixel faster. The increased speed makes it ideal for larger, higher definition displays with a large number of pixels. AMOLED screens also tend to have better contrast, as the light on the screen comes from each individual pixel rather than a backlight; when it needs to create a black color it simply dims or turns off the relevant pixels, for a true, deep black. Although all of these technologies have their pros and cons, the fraction of phones with AMOLED display technology is increasing in the market. Since we conduct our user studies by releasing logger applications to market, we do not control the display technology used. Fortunately, as shown in Table 1, our participants use various models of phones; thus we were able to test all display technologies.

4.1.3 Limitations of Ambient-Only Mechanism

As explained earlier, phones typically control brightness by measuring ambient light in the environment (we call this the "default scheme" for the rest of the chapter). There are three main limitations of this approach.

First, default scheme ignores varying user brightness preferences by implementing a static one size fits all model. This implementation basically assumes that all users have the same requirement and a static model is sufficient for good user experience. However, prior work shows that user brightness preferences vary among users. Kelley et al. [74] study RVP (Relative Visual Performance) of users in varying age groups and in varying contrasts. They find that there is a strong correlation with the required contrast ratio and age. Figure 4.1 shows that a 75-year-old person requires roughly twice the contrast ratio that a 20-year-old person does at the same luminance and contrast ratio to achieve a similar RVP. Moreover, Schuchhardt et al. [113] conduct a user study to observe users' brightness preferences under varying luminance. They find that, even under the same conditions, users' preferences show diversity both within each other and from the default scheme's settings.

Second, default scheme ignores other sensors' effect on user brightness settings/requirements. This implementation assumes that using only ambient light sensor is sufficient to control brightness. However, studies [113, 27, 22, 82] show that brightness can be controlled more efficiently by utilizing more metrics/contexts. For example, as we discuss more in the Related Work section, authors in [113] show that location is also important factor on predicting screen brightness. This is also intuitive: the brightness requirement may be different if the user is sitting on a couch versus walking (even if the ambient light may be the same). In the proposed brightness prediction system, we utilize 16 features acquired from a variety of components of a smartphone including motion and environment sensors, screen touch counts, and audio records. We also discuss the correlation between brightness preferences and these features in Section 4.4.

Third, default scheme is not sensitive to all ambient light changes. In order to understand how brightness is controlled by default scheme, we reverse-engineer the method by observing

brightness values under increasing, decreasing, and random order ambient light environments. We use two devices for this experiment: a Nexus 6p and a Samsung Galaxy S7. Both observed screen brightness level and ambient light reading data are retrieved from the operating system. The continuous model of this data is presented in Figure 4.2. Note that we observe the same outcomes for both devices. As shown in the figure, in overall, the screen brightness follows the square root of the ambient light levels, reflecting Steven's Law [123]. When we look more carefully, we make two more observations from these experiments. First, for the same ambient light level, we observe different brightness settings depending on the given ambient light order (increasing, decreasing, or random). For example, for ambient light 1200-1220 lux, we observe default scheme sets brightness to 135 when increasing ambient light order is given. The brightness is set to 176 (from 0-255 discreet levels) when a decreasing ambient light is given. There is a big difference between these brightness levels and their power consumption [114, 105, 147]. Second, for a large range of ambient light windows, brightness is not changed. Especially when ambient light is above a certain level (i.e., 1800 lux), we observe that brightness stays longer at the same level. Manufacturers may do these implementations in order to avoid any user dissatisfaction caused by a sudden or wrong brightness changes. However, a finer grain adjustment can be used in order to save more screen power.

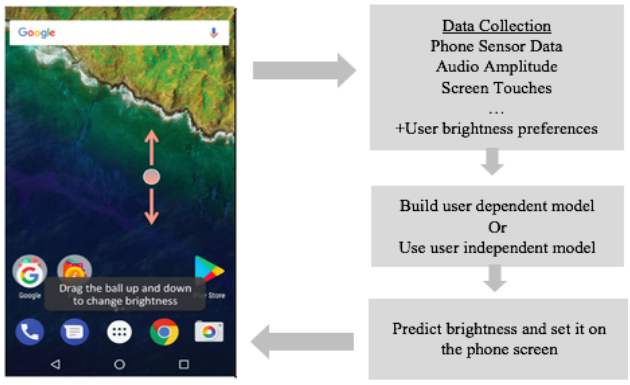


Figure 4.3: GUI part of the phone (on the left): sliding bar on the screen to collect brightness preferences and background service functions from data collection to brightness settings (on the right).

4.2 Experimental Setup

In this section, I discuss our user studies. I first introduce our logger application along with collected sensor information and the overhead of our tools. Then, I explain our two IRB approved user studies where we build user-independent and user-dependent models.

4.2.1 Logger Application and Set of Sensors

We develop a sensor-logger application to collect sensor data from target devices and release it on the Google Play Store. The logger is developed as a regular Android Runtime (ART) executable using the Java standard libraries available in Android frameworks. Thus, it runs on all Android smartphone devices without any special hardware or OS support. At a high-level, the logger application consists of two parts: (1) a GUI, which creates a sliding ball on the screen and looks like a regular smartphone application and (2) an associated background service to provide the logging functionality.

- *The GUI application* is designed to collect user brightness preferences through a sliding ball on the screen. This ball always stays at the front screen. When it is dragged up, screen brightness is increased, on the other hand, when it is dragged down brightness is decreased. We made this ball to enable an easy access for users to set brightness (Figure 4.3 left image). We scale the ball's movement in 0-255 range with the target phone's screen size in order to set minimum and maximum available brightness on the screen.

- *Background service* starts as the GUI starts. The service is responsible for 1) collecting sensor data, 2) building user-dependent model from user's sensor data, 3) predicting current user brightness preference using the developed models, and 4) setting screen brightness in order to maximize user satisfaction while minimizing power consumption. Figure 4.3 (right side) shows the background service's tasks. Also, the service periodically looks for a network connection and sends the collected data back to our server when the collected data size exceeds 500KB.

Table 4.2: Utilized sensors and collected features

Sensor/Feature	Explanation
Accelerometer	We collect acceleration force along the x, y and z axis in 5Hz sampling rate.
Gyroscope	We collect rate of rotation around the x, y and z axis in 5Hz sampling rate.
Ambient air pressure	We collect air pressure in 5Hz sampling rate.
Ambient light	We collect ambient light in 5Hz sampling rate.
Screen touches	We collect screen touches as binary (touch: 1, no touch: 0)
Ambient audio amplitude	We collect ambient audio amplitude (in decibel) in raw, max, min and avg. values in 2Hz sampling rate using the microphone.
Time	We collect time as in 24 hours scale every hour and in 4 phase of the day as morning, noon, evening, and night
Battery level	We collect battery level in a scale 1-100 by listening every battery change event.
Activity	We collect user activities every 30 sec using Google Activity Recognition API [55]
Screen brightness	We collect screen brightness in 1Hz sampling rate.

Android smartphones have built-in sensors that measure motion, orientation, and various environmental conditions. These sensors are capable of providing raw data with high precision and accuracy, and are useful in many situations (e.g., monitoring three-dimensional device movement or positioning). In our proposed system, to be able to predict users' brightness preferences in real time, we collect data from various sensors using default Android APIs. Table 4.2 lists the selected sensors/features and sampling rates used for our models. We find that these features give more insight about users' desired brightness settings. We explore the relationship of these features and brightness preferences in Section 4.4. As shown in the table, we collect motion data, audio records, and touch events from the target smartphones. All of these features can be monitored programmatically with basic user permissions (i.e., microphone, body sensors, etc.) that can be verified through our application(s). Since collected features are in different sampling rates, we also do re-sampling to make them all in 5Hz rate. We should note that, we could also collect some other features like screen frame per second and complex touch events (i.e. size, 2D coordinate, etc.) from logcat [10] in the target phones. However, since monitoring logcat requires super-user permission (i.e., root access), we do not include them in our collection set.

We use MonkeyRunner [98] tool in Android to measure overhead of the logger application.

MonkeyRunner tool provides an API for writing programs that control an Android device or emulator from outside of Android code. We specify commands in Python scripts and sent them to two different Android phones (a Nexus 6P and a Samsung Galaxy S7) through adb shell [13]. We create the same workloads (same touch events, same run-times, etc.) and test each sensor/feature combinations (comparing metrics with and without the sensor/feature) for three minutes on three different applications: a CPU-intensive game, CPU-moderate video, and a CPU-low animation application. We observe that our background service is lightweight mainly because of the small frequency of data collection. We observe logger application increases CPU utilization by no more than 3% on the smartphones. Additionally, activating the logger increases energy consumption by 2.8% on the target phones on average. With the given sampling rate, microphone in audio feature consumes 80-100mW more power on average, while touch, motion, and environment sensor collection overhead stays in the 10-30mW range. In addition, we collect current and voltage values at 2Hz sampling rate to calculate power consumption of the smartphone (note that the reported power levels are for the whole phone).

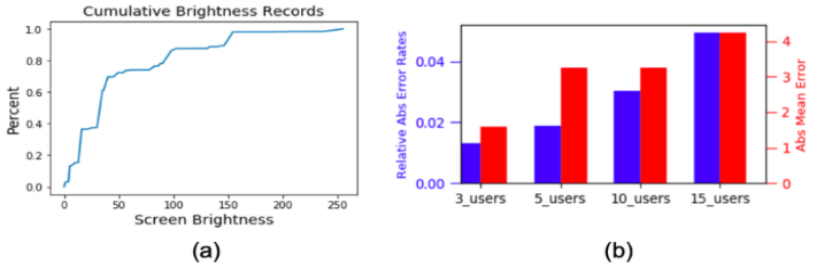


Figure 4.4: (a) Cumulative brightness records and (b) Relative absolute error rate of user-independent model in varying user sizes.

4.2.2 First User Study: Creating User-Independent Model

We release two versions of the logger application to the market (one version for each user study). In the first study, we conduct user tests with real users in order to develop a user-independent model. Participants for this study are gathered through job platforms (i.e., Amazon Mechanical Turk and MicroWorkers), fliers advertising the study, and word of mouth.

User test begins when the logger application is installed (from the Google Play Store) and started by opening its GUI on the target smartphone. In this first version of the application, we do not make any dynamic brightness changes; we only collect brightness preferences along with the sensor data. To achieve this, every 30 seconds we overwrite screen settings to turn on default (adaptive) brightness settings. Therefore, we constantly force screen brightness to be set automatically by default scheme and monitor users' manual brightness changes through the sliding ball (GUI of the application) on the screen.

In the first user study, we use 15 users who have the largest logged activity. Please note that this does not mean users with the longest amount of log time, it means users whose smartphone logged the longest (i.e., longest "screen on" times). The cumulative log data represents over 140 hours of real user activity. Figure 4.4(a) shows the cumulative distribution of the observed brightness records. As shown in the figure, we observe that brightness is under level 50 almost 60% of the time. In order to form the training data, we balance all brightness preferences by randomly downsampling the higher occurred settings in each brightness level.

From 15 users, we accumulate 10 random users' data to build user-independent model. We specifically choose 10 users' data in order to prevent any individual bias and/or overfit in the prediction model. However, in order to see the effect of group sizes on the user-independent model's accuracy, we build the user independent model with different user groups. Specifically, we formed the training data with all unique combinations of groups with 3, 5, 10, and 15 users out of total 15 users.

We find that REPTree algorithm in Weka-machine learning tool [139] provides highest accuracy with a small overhead for the predictions. We use 10-fold cross validation to prevent training and testing data overlap. Figure 4.4(b) shows the mean absolute relative error rates and mean absolute errors of the user-independent models with varying user sizes. Note that the model uses the features listed in Table 4.2 as its input.

As seen in the figure, error rate increases as the group size increases. This is intuitive: since models are built from each user's individual sensor data, unique user behavioral patterns can be

a good indicator to predict brightness preference for users in smaller size groups. However, it is also possible to see an overfit in the models with the smaller sizes. When the group size increases, these unique patterns become less distinguishable and results in increase in the error rates. Having said that, when we use all users (15 users) to build the user-independent model, error rate is still low (less than 6%). This shows that, although there are individual differences in collected sensor values, there are still common patterns where users' sensor values show similarities based on their brightness preferences. We explore such common patterns and user-specific differences in Section 4.4.

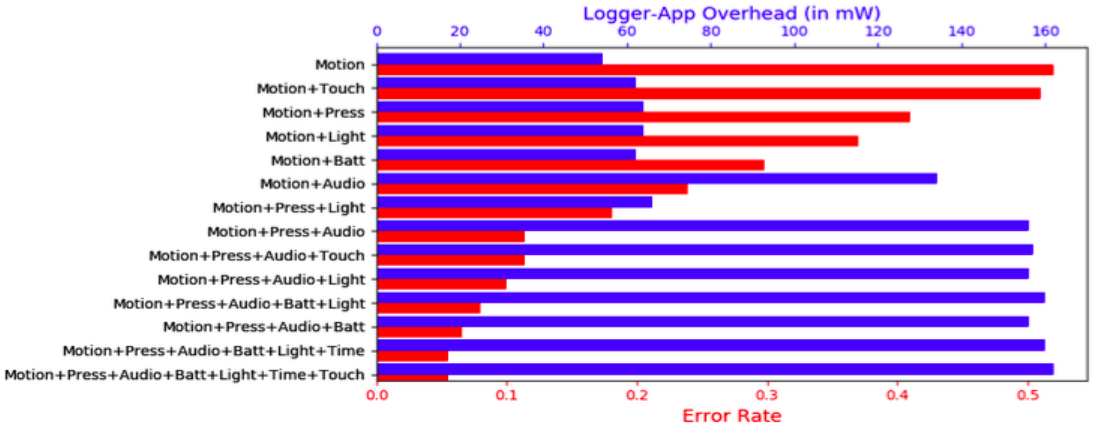


Figure 4.5: Average absolute mean error rates (sorted in red bars) for brightness predictions of the models and average power consumption (blue bars) of logger application for each sensor/feature combinations.

4.2.3 Accuracy of the User-Independent Model With/Without Each Sensor

While studying the user-independent model, we also tested different sensor combinations using the methodology described in Section 4.2. Specifically, we were interested in discovering the accuracy and overhead of our model with all different sensor combinations.

Figure 4.5 shows the importance of each sensor in the models and their overhead to the phone. In the figure, y-axis shows the sensors and features tested to build models, top x-axis (blue bar) shows the logger-application's power consumption on the phone and bottom x axis (red bar) shows the error rates of brightness predictions. In the figure, "Motion" stand for motion sensors, "Press"

stands for pressure sensor, “Batt” stands for battery level, “Light” stands for ambient light, and “Audio” is the ambient noise in decibel collected from microphone.

As shown in the figure, we can achieve the best accuracy when we include all features (Motion, Press, Audio, Batt, Light, Time, and Touch) in our prediction models with a negligible power cost: we observe the average power consumption as 162mW. Moreover, we observe that microphone in audio feature consumes relatively more power (80-100mW) on average compared to other sensors. On the other hand, adding audio feature also increases accuracy considerably (by 4.02%) when included with other sensors. We should also note that, as shown in Table 2, we tested “Activity” feature in the model as well. However, we observe that 96.3% of the activities recorded as “still”. Thus, when we include activity feature, we do not observe any improvement in our models, hence we do not include it in our user-independent model.

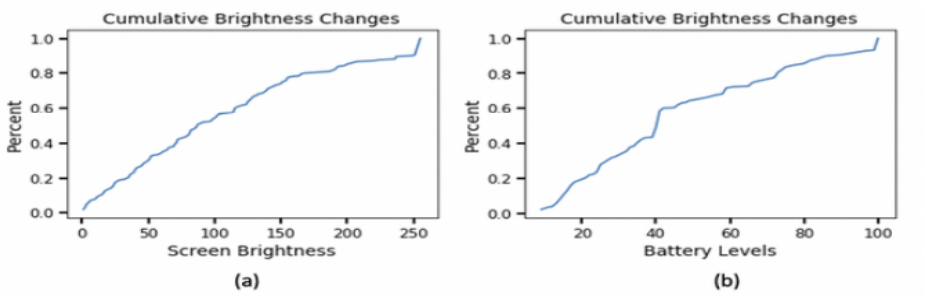


Figure 4.6: (a) Cumulative brightness change counts in brightness levels and (b) Cumulative brightness change counts at different battery levels.

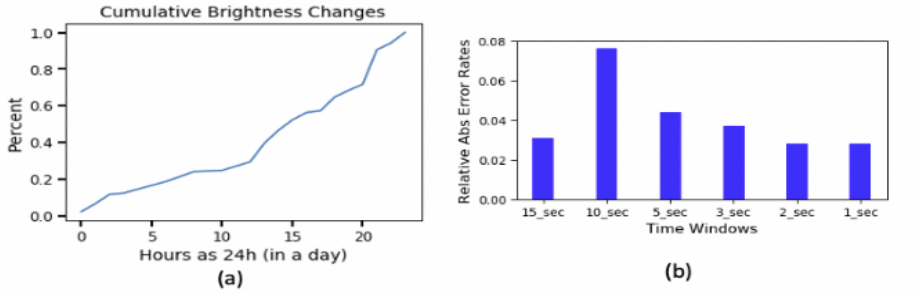


Figure 4.7: (a) Cumulative brightness change counts during the day and (b) Relative absolute error rates and absolute mean error of change prediction model in varying time windows.

4.2.4 When Do Users Change Their Brightness?

As described in the previous section (Section 3.2), in our first study, we collect brightness changes through the sliding ball on the screen. We record each touch and drag movement of the ball on the screen in order to see (i) how many times users perform manual brightness changes and (ii) what is the scale of the brightness change (in the 0 to 255 brightness level range). We observe that all our participants used the sliding ball to change the brightness during the studies. While some users become less dissatisfied with the default scheme's settings and change brightness relatively less frequently (minimum number of changes we observe is 2.0 changes per day of usage), others change brightness far more frequently (maximum number of changes we observe is 29.1 changes per day of usage). This also shows the insufficiency of default scheme on controlling brightness and the need for user-aware models on brightness settings. Note that, in the collected data, we do not count a movement in the sliding ball as a brightness changes unless its movement corresponds more than 10 level brightness change (out of the possible 255 levels). The analysis of the collected sensor data and brightness changes reveal the following observations:

- As shown in Figure 4.6(a), almost half of the changes occur when the default scheme set the brightness between 0 and 100. Yet, as shown in Figure 4.4(a), users rarely use their phones in higher brightness levels; the fraction of time they spent over 150 is less than 10%. However, the fraction of changes that are made in this range is over 25%. This shows that, whenever users are in higher brightness levels, they are more likely to do a brightness change. Moreover, we observe that 81.8% of the changes in the 150-255 range are to decrease it (the average reduction is 150 levels), while in the 0-50 range, 66.6% of changes are to increase the brightness (the average increase is 108 levels).

- Similarly, we count screen touch events. These touch counts represent basic user interaction with the phone. Not surprisingly, most of the touches occur when the phone is in the 0-50 brightness range. For the 0-50 brightness range, we observe a roughly 1:28 ratio of brightness changes to touch counts, meaning that roughly in every 28 touches, users change

their brightness. This ratio is 1:12 in the 150-255 range.

- Over 60% of the brightness changes occur when the remaining battery is less than 50% and most of the changes occur when the remaining battery is between 40% and 45% (Figure 4.6(b)). These results show that brightness changes are slightly more frequent when the battery is running out, yet there are still significant amount of changes when the remaining battery level is high.

- As shown in Figure 4.7(a), users do not change the brightness frequently during the morning or night: about 40% of the changes occur between 12pm and 8pm and similarly about 40% of the changes occur between 8pm and 12am.

- Finally, we observe that overall 57.6% of the changes are to decrease the brightness with an average change of 76.6 levels; while the rest (42.4%) is to increase the brightness with an average increase of 74.6 levels.

Moreover, we build a prediction model to predict users' brightness changes. Specifically, we are interested to see how good we can predict the brightness change times with the collected sensor data. If we can predict the time users change their brightness prior to the user request, we can use this prediction for a finer-grained control on brightness settings to provide better user experience. In order to avoid sparsity on manual changes, we use the data of 15, 10, 5, 3, 2, and 1 seconds before the user has requested a change. Again we find that REPTree algorithm in Weka-machine learning tool [139] provides highest accuracy with a small overhead on the predictions. Similarly, we use 10-fold cross validation to prevent training and testing data overlap. Figure 4.6(b) shows the mean absolute relative error rates and mean absolute errors of models in varying time windows. Although error rates are low in all windows (<8%), taking 15 sec time window gives the lowest error rate and lowest mean absolute error.

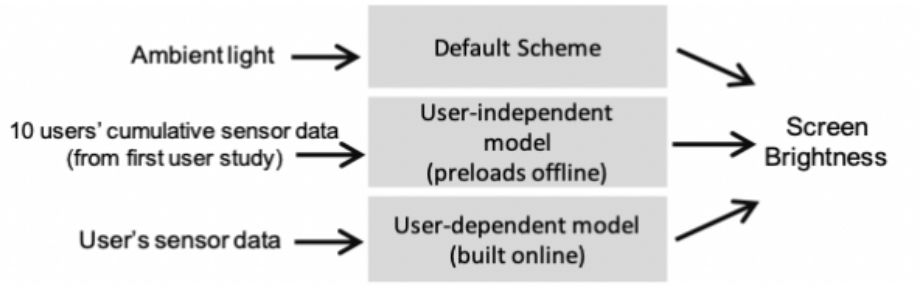


Figure 4.8: Tested 3 brightness control models (default scheme, user-independent, user-dependent) in the second user study.

4.2.5 Second User Study: Building User-Dependent Models and Controlling Brightness in the Wild

After creating the user-independent model in the first user study (Section 3.2), in the second user study (i) we build user-dependent models for each user on-the-fly, and (ii) let users test their own (user-dependent) models, the user-independent model, and the default scheme for 6 days duration.

For the second user study, we release the second version of the application on Google Play Store. In this version, we first pre-load the user-independent model (created in the first user study with 10 users' cumulative data offline). Additionally, we update our application to build user-dependent models from the user's accumulated data online in order to make brightness predictions by using only the user's data. We should note that, we release this version as a new application with a different package name and icon in the Google Play Store, so that the users in this study are different from the 10 users in the first version, whose data is used to build the user-independent model.

User test begins when the logger application is installed (from the Google Play Store) and started by opening its GUI on the target smartphone. Thus, similar to the first user study, during the experiment, users use their own phones as they typically use. In the second user study, users test their user-dependent model, the user-independent model, or the default scheme each day in a random order. We made sure that all users test the two models and default scheme exactly twice. Thus each user experiment takes 6 days in total. (Figure 4.8 shows the tested models in the study):

Algorithm 2 Pseudo code for predicting and setting screen brightness programmatically in real time.

```
1: procedure LOGGER AND BRIGHTNESS SETTER
2:     Make calls to classes to log sensor data
3:     change − tested − model()
4:     predict − brightness − and − set()
5:     sleep for 3 sec
6: function CHANGE-TESTED-MODEL()
7:     if last time model changed − current time < 24hours then
8:         return
       today's model = get new model to test
9:     if today's model is user-dependent model then
10:        build a new user-dependent model from all user data
11: function PREDICT-BRIGHTNESS-AND-SET()
12:    if today's model is user-independent model then
13:        prediction = get user-independent model prediction
14:    if today's model is user-dependent model then
15:        prediction = get user-dependent model prediction
16:    if prediction is same for 30 sec and then
17:        prediction − current brightness < 10
18:        current brightness − = 3
19:    else current brightness = prediction
```

- When the default model is selected, the application does not make any predictions and leaves the brightness control over to the default scheme. Note that the default scheme may be different based on to the Android version of the user.

- When the user-independent model is selected, application uses the pre-loaded user-independent model to predict and set brightness on the target phone.

- Every time user-dependent model is selected, a new user-dependent model is built from all the collected data of the user using the same methodology explained for the user-independent model (first user study). Therefore, unlike the user-independent model, which is loaded offline and is static during the study, user-dependent models are rebuilt dynamically using the accumulated data. Algorithm 2 provides a detailed pseudo-code used to predict and set brightness programmatically in real time for both models. While forming the training data

for user-dependent model on the target phones, we again randomly downsample the more frequent brightness settings in order to balance the data. On both tested models, predictions occur every 3 seconds

In order to benefit from human eyes' insensitivity to small gradual changes in brightness [119, 147], we also implement a gradually dimming mechanism when controlling the brightness. The aim of this implementation is to save power without impacting user satisfaction. When the required brightness is between level 50 and 255, if the previous brightness prediction is the same as the current one for 30 seconds, we dim the brightness slowly (3 level decrease every 30 seconds). We set the maximum brightness reduction to 10 in order to avoid any user dissatisfaction. For example, if predicted required brightness is 100 for a long duration, the brightness is set to 97 after 30 seconds, to 94 after a minutes, etc., until the brightness level of 90 is reached. We keep the brightness at 90 as long as the predicted required brightness level is 100.

Finally, we also ask users' overall satisfactions of the phone's brightness settings every 6 hours through a pop-up questionnaire on the screen. In the questionnaire, users answer the question "How are your phone's brightness settings today?" from a 5-scale radio style selection: 1 is the worst, 5 is the best. The aim of these questionnaires is to analyze models' impact on user satisfaction. We specifically ask the questioner 4 times in a day in order to get as much user report as possible from a model's performance on brightness setting.

4.3 Results

In this section, I present the results acquired from our second user study, where users test their user-dependent models, the user- independent model, and the default scheme on controlling their phone's brightness. In total, we receive data from 47 users. Unfortunately, not all of our participants use the application adequately (not long enough) nor give enough inputs to analyze (a common issue with conducting in-the-wild tests with real users). Once we filter out the users who don't give inputs to pop-up questionnaires and who test each model less than twice, we end up with 21 users whose results we present below.

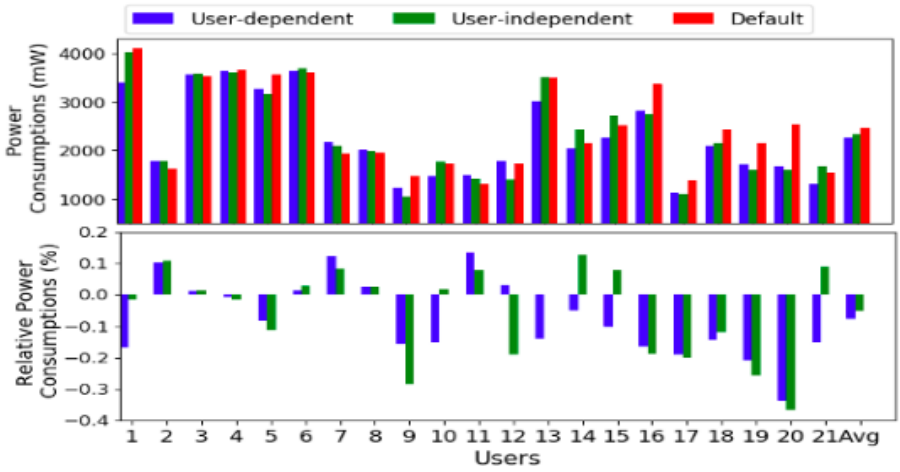


Figure 4.9: Average power consumptions (in mW) (top) and relative average power consumptions for each model for 21 users (bottom).

Figure 4.9(top) plots the power consumption values of the three tested models for our 21 users. Since usage characteristics and application selections vary, we observe a large variation between power consumption levels of each user. In order to better analyze the differences of the models, we normalize them by calculating their relative change based on the default value for each user using the formula:

$$relative_{chng} = (Power_x - Power_{default})/Power_{default}$$

,

where x represents all three models for the user. Figure 4.9(bottom) shows the relative average power consumptions of the three tested models for all users. As shown in Figure 4.9, default scheme's power consumption is highest in 9 out of the 21 users. In 15 out of 21 users, the default scheme exhibits higher power consumption when compared to the user-dependent model. Overall, our models are successful in saving power: on average, user-independent and user-dependent models save 5.16% and 7.65% system power, respectively. We also observe the maximum power savings are 36.5% and 33.7% (user20) for user-independent and user-dependent models, respectively. Please note that these power savings are system level and acquired by predicting users' brightness

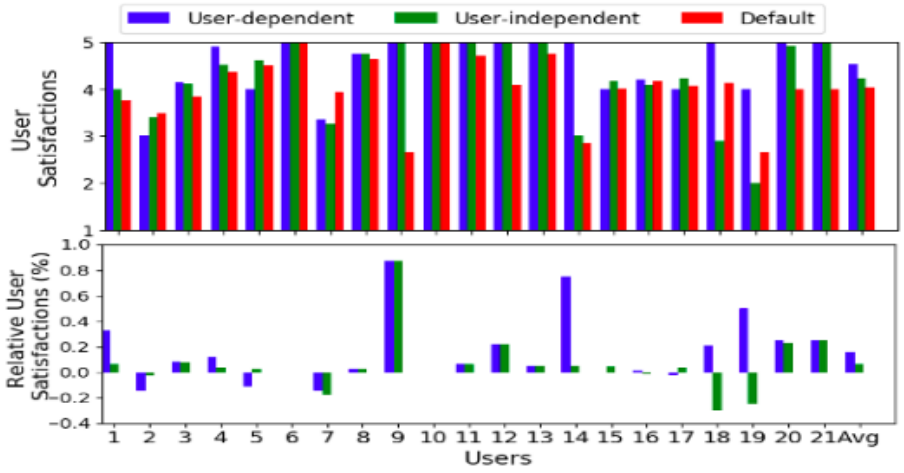


Figure 4.10: Average user satisfaction reports (top) and relative user satisfaction reports for each model for 21 users (bottom).

preferences along with a naive dimming mechanism with dimming limit up to 12 brightness level. However, it is possible to make dimming mechanism more aggressive with different thresholds to increase power saving.

Figure 4.10(top) plots the average satisfaction reports gathered from the pop-up questionnaires. Similar to Figure 8(bottom), in Figure 4.10(bottom), for each user, we calculate relative change on satisfaction reports based on their default model's report. As shown in the Figure 4.10, there is a large variation in user satisfaction levels as well. While some users are more sensitive to brightness changes and can clearly distinguish the three models (e.g., user18), others cannot distinguish any difference (e.g., user6). However there exist only 2 users (user2 and user7) who rated the default model better than other two models. On average, the user-dependent model outperforms the user-independent model and default schemes: user-independent and user-dependent models increase user brightness satisfaction by 6.03% and 15.77%, respectively, when compared to the default scheme.

In order to analyze the power consumption and satisfaction results further, we performed ANOVA [19] test on the normalized values to see how the results of these three models differ from each other. Table 4.3(a) shows the p-values of both power and satisfaction comparisons. As shown in the Table 4.3(a), p-values for power and satisfaction comparisons are 0.0772 and 0.0494,

ANOVA Test	
F-Score	p
2.6726 /3.1632	0.0772 /0.0494

(a)

Models	Default	User-independent
User-dependent	0.0096 /0.014	0.5506 /0.2163
User-independent	0.1175 /0.244	-/-

(b)

Table 4.3: Statistical difference comparisons of power consumptions and satisfaction reports. (a) Results of ANOVA tests. (b) Results of post-hoc pair-wise t-test p-values.

which indicates that results are significant and the means differ (for 0.1 and 0.05 significant levels respectively). Since p-values are significant, in order to understand how each model differ from one other, we then conduct post-hoc pair-wise t-tests [102]. Table 4.3(b) shows the measured p-values in the t-tests again for power and satisfaction comparisons. As shown in the table, for both comparisons, the p-value for the user-dependent versus default model comparison is significant, meaning that, with the user-dependent model we can indeed increase user satisfaction and also save power.

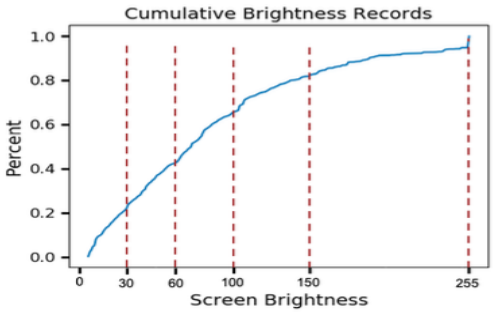
There are two reasons behind the overall power savings and increased user satisfaction. First, when we look at all the users' data, we observe 6831 manual brightness changes in total; 51.9% of them occur during the default scheme, 33.7% during the user-independent model, and the rest (14.4%) during the user-dependent model controlling the brightness. This shows that proposed models control brightness better than the default scheme in terms of user experience. Second, we observe that 63% of these manual changes are to dim the screen brightness: on average, users reduce the screen brightness by 72 levels (out of 256) when they dim it and increase it by 68 brightness levels when they brighten. This shows that default scheme sets the screen unnecessarily bright on average, which causes more power consumption.

4.4 Correlation of Sensor Data and Brightness Settings

In this section, I present the analysis of the relation between sensor data and users' brightness preferences in more detail. Specifically, when studying the relation we were interested in seeing how

Sensor/Feature Group	Info Gain
Motion Sensors	3.515
Pressure	3.382
Ambient Noise	3.13
Battery level	1.32
Ambient light	1.195
Time (in 24h)	1.152
Touches	0.156

(a)



(b)

Figure 4.11: (a) Total information gained from each sensor group. (b) Cumulative brightness records for all users along with the red dash-lines separated 5 brightness ranges.

the collected sensor data correlates with the brightness settings. We use all users' data (47 users with 23 different smartphone models) to find the relation with the given brightness preferences. I should also note that, in our analysis, we only consider "screen on" times, since the times when the screen is turned off would not reflect users' brightness preferences.

As described in Section 4.2.2, we use REPTree algorithm in our models. In the algorithm, predictions were made based on the branches formed by the information gained from each feature. To be able to see how much information is gained from each feature, we extracted them using Weka-tool's "InfoGainAttributeEval" evaluator from Attribute Selection [136]. The table in Figure 4.11(a) shows accumulative information gain of each feature group. To get a clear view in the table, we accumulated same set of sensors and grouped them in the row (for example, gain from accelerometer and gyroscope sensors are combined in the "motion" sensors group). In the following sections, I discuss each sensor group individually.

Figure 4.11(b) shows the recorded cumulative distribution of brightness settings from all users. Similar to the Figure 4.4(a), we observe that about 60% of the recorded brightness levels are less than 100. Note that the distribution of the screen brightness presented in Figure 4.11(b) and Figure 4.4(a) are similar (dimmer screen brightness is more common in both datasets). However, the exact distributions are slightly different: in the new (more extensive) dataset, we observe approximately 20% of the time spent in brightness setting of above 150 (whereas this ratio was approximately 10% for the first dataset).

To make the analysis simpler, we group the sensor data under five brightness ranges as indicated with red dash-lines in Figure 10(b) from low to high as 0-30, 30-60, 60-100, 100-150 and 150-255. Each range contains roughly the same proportion (20% $\pm$ 2%) of the data. In the following subsections, we analyze the relation of the sensor data and brightness settings in more depth according to this categorization.

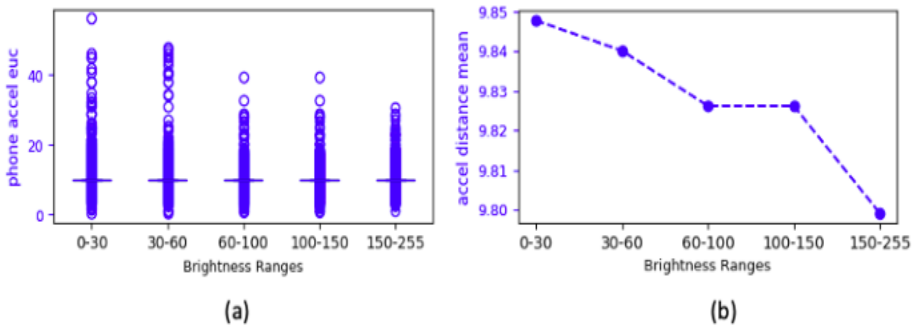


Figure 4.12: (a) Distribution of accelerometer distances for each brightness range and (b) Averages of accelerometer distances for each brightness range.

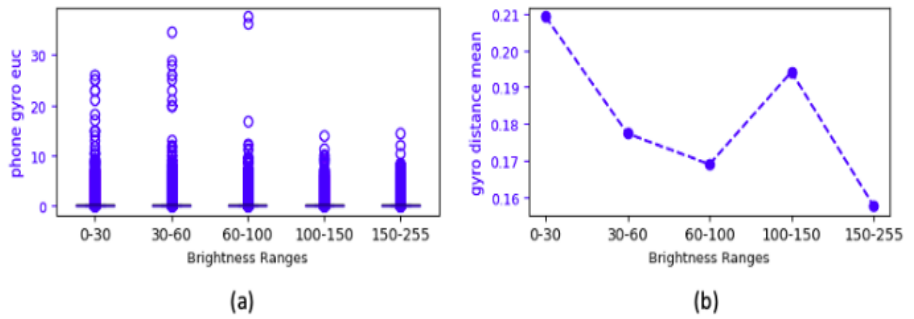


Figure 4.13: (a) Distribution of gyroscope distances for each brightness range and (b) Averages of gyroscope distances for each brightness range.

4.4.1 Motion Sensors

Motion sensors are the most information gained feature as shown in Figure 4.11(a). We collect acceleration force along the x, y, and z-axis and the rate of rotations around these axes using the accelerometer and gyroscope sensors on target phones. Accelerometer and gyroscope sensors are commonly used in motion and rotation detection. In order to get a clearer view of these raw three axes, we calculate the distance of them using the Pythagorean formula:

$$distance = \sqrt{(x^2 + y^2 + z^2)}$$

where x, y and z correspond to either accelerometer or gyroscope axes. Figures 4.12(a) and 4.13(a) show the distribution of calculated distances from the collected accelerometer and gyroscope sensors, respectively, for each brightness range. As depicted in the figures, the range and variation become smaller as the brightness level increases. The standard deviations for each brightness range (starting from 0-30) are 0.74, 0.72, 0.65, 0.64 and 0.64 for the accelerometer data and 0.53, 0.58, 0.50, 0.48 and 0.43 for the gyroscope data, respectively. In Figures 4.12(b) and 4.13(b), we show averages of these distances for each brightness range. Similar to the distribution figures, we observe drops in the average values as the brightness level increases. Assuming that, users require better brightness settings when they perform activities such as reading, playing, etc., it is possible that they may prefer higher brightness levels. Since these activities require user's attention on the screen, we see more stable movements in 3D physical actions. On the contrary, we observe higher variance and more scattered motion data when brightness is in lower levels: it is possible that users require lower brightness levels when they do activities that do not require constant focus on the screen (e.g., checking time, notifications etc.).

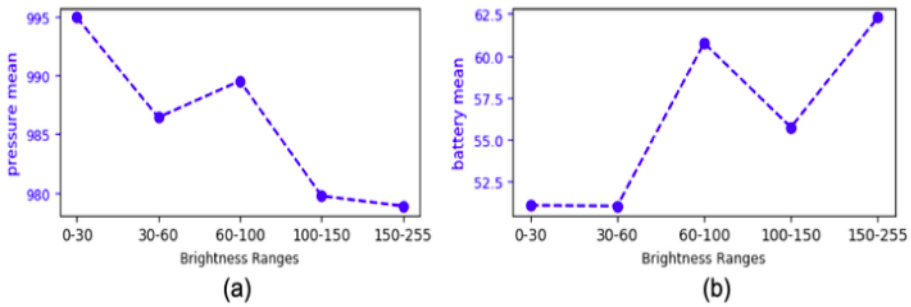


Figure 4.14: (a) Observed ambient air pressure (in mbar) means and (b) Average operating battery level for each brightness range.

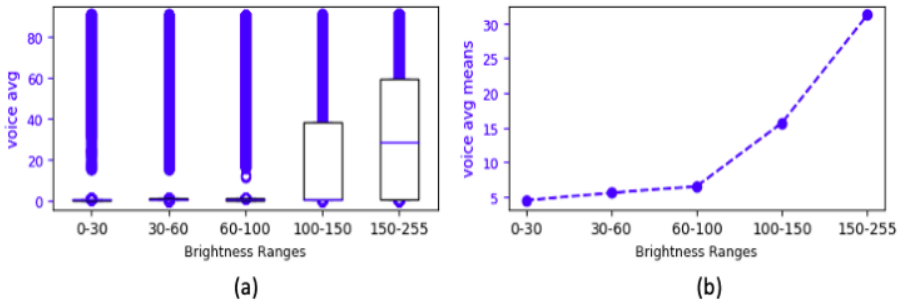


Figure 4.15: (a) Distribution of observed audio amplitude values and (b) Averages of observed audio amplitude values for each brightness range.

4.4.2 Ambient Air Pressure

We collect ambient air pressure through the pressure sensor [58] in the target phones. Pressure sensor is one of the sensors that became popular for activity recognition in recent years [55]. It is shown that especially in complex activity recognition tasks, pressure sensor increases accuracy significantly [59, 87, 96] due to high precision barometer sensors on smartphones [61] (measured relative accuracy is $\pm$ 1 meter). Figure 4.14(a) shows the observed ambient air pressure averages for each brightness range. Similar to the motion data (Figures 4.12 and 4.13), we observe a strong negative correlation between the ambient air pressure and brightness levels. This relation can be explained by the assumption that users interact with their phones and/or do activities that require attention in home/office environments (e.g., high floor environments) more, rather than outside. Given that an increase in altitude decreases air pressure, we observe lower ambient air pressure in higher brightness levels.

4.4.3 Battery Levels

We collect battery levels of phones using Battery Manager API [57] at 1 Hz rate. Figure 4.14(b) shows the observed battery level averages for each brightness range. As shown in the figure, not surprisingly, in general we observe a positive correlation between the brightness levels and battery level averages: as the battery level increases, brightness level increases as well. We observe the maximum battery average in 150-255 brightness range. It shows that when users have high battery

levels, they use their phones in higher brightness levels more. However, when they have lower battery levels (less than 52) they are more likely to use their phones in lower brightness levels due to possible battery concerns.

4.4.4 Ambient Audio Amplitudes

We collect ambient audio amplitude (in decibel) from our users at 2 Hz sampling rate using Android Auido Record library [56] with the microphone. Most of the target devices have built-in MEMS microphones, which enable ambient noise canceling to improve input voice quality [42]. In our studies, we observe the audio noise reaches up to 90 decibel. Along with the raw records, we also collect their statistics such as mean, maximum, and minimum in 3-second time windows. In Figure 4.15(a), we plot distribution of the raw audio records for each brightness range. The standard deviations of each brightness range (starting from 0-30) are 15.15, 16.31, 19.09, 25.39, and 31.88, respectively. In Figure 4.15(b), we plot the mean values of the raw audio records for each brightness ranges. As depicted in the figures, there is a positive correlation between ambient audio records and brightness levels. We hypothesize a similar intuition with motion data. When users perform tasks that require interaction with the screen (e.g., playing a game, video-chatting) they need higher brightness levels. On these activities, it is also expected for users or phone to be noisier (e.g., game music or voice), which leads to higher ambient amplitude levels.

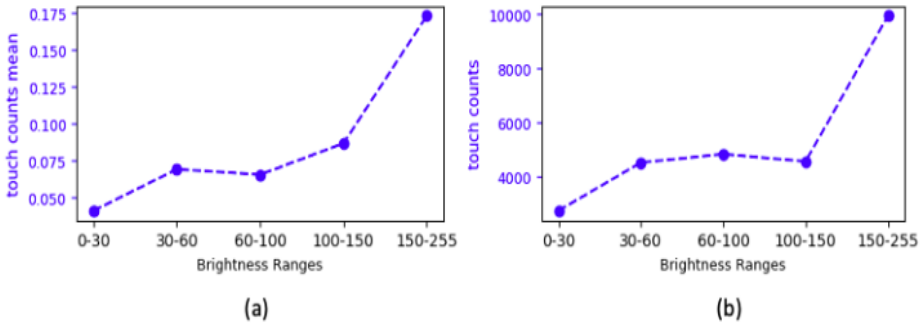


Figure 4.16: (a) Fraction of logs with touch events and (b) Total screen touch counts for each brightness range.

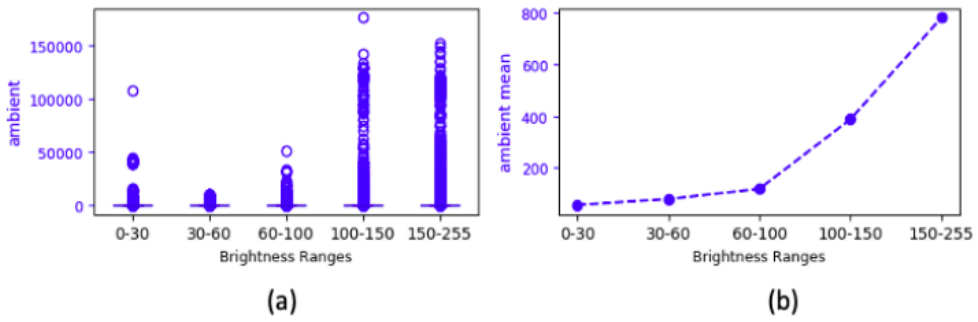


Figure 4.17: (a) Distribution of observed ambient light values for each brightness range and (b) Averages of ambient light values for each brightness range.

4.4.5 Screen Touch Counts

We collect screen touch counts from the target phones. These touch counts represent basic user interaction with the phone. Figure 4.16(a) shows the fraction of logs with touch events and Figure 4.16(b) shows the total touch counts for each brightness range. As shown in the figure, we observe a positive correlation with the brightness levels and screen touches: as the brightness level increases touch counts also increase. In fact, we observe the most touches in the 150-255 range. There are two possible reasons behind this. First, similar to previously discussed sensors: users require higher brightness levels when they perform activities such as searching, playing, etc. Since these activities involve active user interaction, we see higher touch counts on higher brightness levels. On the other hand, when users do activities which do not require active user interaction, it is possible they are comfortable with lower brightness levels. Second, as discussed in Section 4.2.4, users change their brightness more during higher levels, which causes increased number of touches on the screen.

4.4.6 Ambient Light

We collect ambient light level (illumination) in the environment using light sensors [54] on the target phones at 5Hz sampling rate. Ambient light values are collected in lx (lux) unit generally in a range between 0 to 9999. Figure 4.17(a) shows the distribution of the ambient light values for each brightness range and Figure 4.17(b) shows their averages. Not surprisingly, we observe a positive

correlation with the ambient light and brightness settings. As discussed earlier in Section 2, in order to create the desired contrast ratio on the screen, ambient light is an important indicator. However, as shown in Figure 4.11(a), we observe that ambient light sensor is the fifth most information gained feature in our models among other sensors and apparently there are other sensors that can give more insight on users' brightness preferences.

As we explained in Section 4.3, we observe 6831 manual brightness changes across all our logs. The majority of the changes occur during the default scheme operation. If a user was already satisfied with a model, no brightness changes would have been observed. As our experiments described in the previous section show, the inclusion of other sensors can improve the accuracy significantly and reduce the dissatisfaction compared to the default mechanism that only uses the ambient-light sensor.

4.5 Discussion

User selection. As discussed during the introduction of Chapter 4, all of our users are participants who downloaded our application(s) from Google Play Store just like any other application. Although it is possible to see some similarities (or differences) in user-dependent models feature selections based on users' age, gender, or being tech savvy or not categories, due to privacy concerns we do not collect personal information of our users. However, based on Google's statistics, our users are from 10 different countries where majority of them are from United States, India, Germany, and South Africa.

Android Pie. As discussed in Section 4.1.3, ambient-only solution has limitations on determining the preferred brightness level. In order to address some of these limitations, current Android OS provides a manual brightness slider even with the adaptive brightness feature turned on, claiming that manually overriding the brightness will help the adaptive brightness feature to determine the correct offset for users' preferences. However, in our user studies, we observe that there is a good mixture of both increasing and decreasing manual brightness changes from the same user even in the same ambient light. Thus, the users' brightness preference is not simply a matter of

a wrong offset but indeed a variable dependent on other factors (e.g., environment and activity conditions). Moreover, Android recently introduced a new version, "Pie" (phones with API level 28 and higher in Table 4.1), where it is claimed that a personal brightness preference is utilized. However, in our user studies, we observe that the brightness change patterns do not change across users (between our 10 Pie and 37 non-Pie users). It is possible that incorporating user preferences can help creating user-aware mechanisms on controlling the brightness, however, lack of utilizing other sensors on brightness settings still limits the capabilities. In fact, when we compare satisfactions of Pie users on tested models, we observe that user-independent and user-dependent models increase satisfaction by 0.002% and 13.5% compared to the default scheme (i.e., Pie), respectively. The average satisfaction levels are 3.84, 3.85, and 4.36 for the default, user-independent, and user dependent models, respectively. Further, we also compare the power consumptions of the tested models on Pie users and observe average power consumption reduction of proposed models are 18.9% (user-independent) and 17.9% (user-dependent) compared the Pie. Hence, we can improve the user satisfaction and reduce the power consumption drastically on the state-of-the-art brightness control mechanism.

Energy savings' effects on user satisfaction. There is a tremendous effort in the smartphone industry to find solutions to extend the battery lifetime. Designers have reverted to architectural and system-level optimizations to keep energy consumption down. Despite the importance of end user in smartphones, how much these efforts affect user experience is still not clear. Any improvement in performance or a new hardware feature is easily observable and can have a direct effect in user experience. However, energy savings' effect is non-trivial and (possibly) has influence only in the long run. Thus, it is challenging to measure and factor its effects in instantaneous user experience/satisfaction. In order to get an insight of how much battery life is important for users, we conduct a survey in Amazon Mechanical Turk [115] (mTurk) with 500 users. In the survey, we asked users to rank the most important feature they look in their smartphones. We observe more than half of the users (53%) choose battery over any individual component. Other online surveys also reflect similar results [49, 51, 17]. Battery lifetime is certainly an important aspect of user

experience in smartphones, but to the best of our knowledge there is no work on measuring its quantitative impact on user experience. Although we do not know energy saving's instantaneous effect on user experience, due to its importance to users, it is reasonable to expect that our system would be rated even higher if the users were informed about the power savings their phones attained compared to the default scheme.

4.6 Summary

In this project, I discuss our studies on controlling smartphone screen brightness. Specifically, we propose a system to control brightness on smartphones. The system utilizes motion and environment sensors, audio records, and screen touch events to predict users' preferred brightness in real time.

We evaluate the proposed system by conducting two user studies in the wild. For both user studies, we release a logger application to Google Play Store. In the first study, we develop a user-independent model by using data gathered from 10 users. In the second user study, we develop user-dependent models for each user online and make participants use their phones while brightness is controlled by their (own) user-dependent models, the user-independent model, and the default scheme. Every 6-hour we collect overall display satisfaction through a pop-up questioner. Our results show that compared to the default scheme, the user-dependent and user-independent models increase display satisfaction by 15.77% and 6.03% on average, respectively. Moreover, in the proposed system we implement a gradually dimming mechanism in order to save power especially in long usage scenarios. Our results show that proposed system saves 7.65% and 5.16% system level power with user-dependent and the user-independent models, respectively. Further, we present in depth analysis on the correlation between collected sensor data and the brightness preferences.

CHAPTER 5

DO USERS REALLY CARE ABOUT ALL THESE EFFORTS? QUANTIFYING THE IMPORTANCE OF ENERGY SAVINGS ON USER SATISFACTION

As I already alluded in earlier chapters, smartphones are a ubiquitous part of modern society. While smartphones were originally capable of only basic applications, smartphones today are capable of performing a wide variety of tasks such as messaging, streaming videos, browsing the Internet, navigation, etc. The increasing capability of smartphones typically comes with the cost of increased energy consumption. Smartphones are battery driven to allow the highest degree of freedom and the battery has to empower all the new features and applications on it, in many cases for a long duration.

Excessive energy consumption of new features is limiting the evolution of smartphones as the improvement of battery capacity is quite moderate compared to the increase of the complexity due to new hardware and services [116]. In fact, as batteries can store a fixed amount of energy, the operational time a user is able to use its phone within one charging cycle is limited as well. There is a tremendous effort in the smartphone industry to find solutions to extend the operational time. Using batteries with more capacity could be a trivial solution, but unfortunately their technological evolution does not follow the trends dictated by Moore's law: the battery capacity doubles only roughly every decade.

Despite the importance of end user in smartphones, how much of the energy saving efforts effect user satisfaction[1] is still not clear. Any improvement in performance due to changes in software or hardware system is easily observable and can have a direct effect in user experience. However energy savings' effect is not-straightforward and typically has an influence in the long run. Thus, it is challenging to measure and factor its effects in instantaneous user experience. If

[1]Throughout this chapter, user satisfaction is defined as the overall subjective satisfaction rating provided by the user for the device they are using.

users knew their smartphone's current energy savings, it is possible that their experience (or at least their perception of experience) may be different. They may even be willing to give sacrifices if they lead to more power savings. Similarly, if manufacturers knew their efforts' effects on user satisfaction, they can focus more (or less) on extending battery life time.

In order to get an insight on the importance of battery life for users, we conduct an online survey with 500 users (as I already discussed in Section 4). In the survey we want users to rank the most important feature they look in their smartphones. We observe more than half of the users (53%) choose battery over any individual component (i.e. camera, operation system, etc.). Other online surveys also reflects similar results and show battery is even more important for younger users [49, 51, 17]. Battery lifetime is certainly an important aspect of user experience in smartphones. However, to the best of our knowledge, there is no work on measuring or investigating its effect to user satisfaction *quantitatively*.

In this project, we study the relation between energy savings and user satisfaction by conducting two user studies with real users. First, we conduct an Amazon Mechanical Turk (mTurk) study with 200 smartphone users. The aim of this study is to see the energy savings' effects on user satisfaction. Thus, we compare user ratings in same conditions under different reported energy savings. In this study, we ask each participant to watch the identical video of a game play and an Instagram video (recorded on a smartphone) multiple times and rate their satisfactions about the screen brightness of the video in a 5-likert scale. During these multiple runs, we randomly choose one video and after the participant finishes the video, we display (report) a random energy saving level before asking them to rate their satisfactions. For other videos, we do not report anything before we prompt them for their satisfaction rating. Our results show that, user satisfaction increases if users are notified with the energy saving, even if the energy saving is minor: on average we observe 5.7% increase in user satisfactions when energy savings are reported in the videos. The details of this study are provided in Section 5.1.

Next, we conduct our second user study where we go *into the wild* in order to understand the energy savings' effect on users' daily usages on their own smartphones. For this study, we

develop a light-weight logger application and make it available on the Google Play Store. In the application, we design 5 different energy saving scenarios where the phone adopts and runs one for each day randomly. In the scenarios, in order to save energy, we drop screen brightness of the phone at varying levels and then either report or do not report the energy savings to users. Users rate their overall satisfaction everyday multiple times on each scenario. Similar to the first user study (mTurk study) we observe user satisfaction increases significantly when users are notified on the energy savings, even though the sacrifice on the screen brightness may be maximum. On average, we observe the highest user satisfaction (17.8% increase) by dropping the brightness 40% and reporting the saved energy. Moreover, we develop a prediction model for the user ratings and show 86.6% accuracy on predicting user satisfaction with the reported energy savings.

Overall our contributions can be listed as below:

- We show that user satisfaction increases significantly if users are notified about the energy the phone achieves with the power saving methods.
- We quantify and present detailed analysis on the relation between energy savings and user satisfaction.
- We build prediction models and demonstrate that user satisfaction can be predicted accurately with the reported energy savings.

The rest of the chapter is structured as follows. In Section 5.1, I introduce our first user study and discuss its results. In Section 5.2, I explain our second (in-the-wild) user study, our logger application and present its result. I also describe the prediction model and correlation between satisfaction reports and collected metrics in Section 5.3.

5.1 First User Study: Mechanical Turk Study

In this section, I describe our first user study, which is conducted on the Amazon Mechanical Turk (mTurk) platform with 200 participants. The aim of this study is to see the effects of reported

Table 5.1: Order of video play and their notifications displayed to users afterwards.

Video Order	Notification Type	Displayed notification
First	Default video	”Note that, brightness of this video is set to default. Based on this video, you will rank next two videos’ brightness settings.“
Second or Third (Randomly chosen)	Report Energy Saving	”if you were watching this video on a typical smartphone, it would require {random selection between 2-12}% less energy overall and hence increase the battery life.“

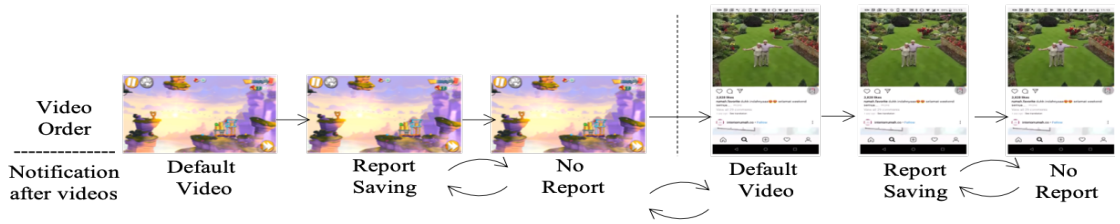


Figure 5.1: Typical order of Angry Birg game play and Instagram videos. Note that The order of applications and energy savings are reported vs not reported are selected randomly.

energy savings on user satisfaction. In the following subsections, I first explain our methodology and then discuss the results.

5.1.1 Methodology

We record 20 seconds of the screen during the use of two applications on a Huawei Nexus 6p smartphone. We selected the applications from the list of most common applications [141]: a game play of Angry Bird [18] and screen scrolling of Instagram [47]. We choose these two different type of applications specifically a) to design a user study as short as possible in order to keep the users attention high and b) to investigate how application (content) affects the relation between energy savings and user satisfaction. We conduct our mTurk study from our server by uploading the videos to YouTube [53] in order to limit any video loading time inefficiency.

Users start watching each application’s recorded video in random order. Once the application is selected, we make users watch the same video 3 times before going to the other application. Thus, users watch total 6 videos (3 times for game play video and 3 times for Instagram video). For each application, after watching its video first time, we display a notification on the screen which indicates that the video’s brightness is set to default settings. Although users watch the

Table 5.2: (a) Average user satisfactions for each video, and (b) observed p-values from t-tests of each video's satisfaction ratings

	Game application	Instagram Application
Default video	4.03	3.77
Reported video	3.98	3.91
Not-reported video	3.91	3.69

(a)

Game Application	Default video	Not-reported video
Default video	-	0.3248
Reported video	0.6942	0.5621

Instagram Application	Default video	Not-reported video
Default video	-	0.5438
Reported video	0.2417	0.06663

(b)

Table 5.3: Proportion of users and average reported energy savings of users categorized by the changes of satisfaction ratings between reported and not-reported videos.

	Game Application		Instagram Application	
change between reported and not-reported videos	average reported energy saving (%)	proportion of users	average reported energy saving (%)	proportion of users
increased satisfaction rating	7.60	32%	7.54	47%
same satisfaction rating	7.08	44%	8.16	25%
decreased satisfaction rating	5.93	24%	7.12	28%

same videos, we do that to make users set a calibration for their expectations for the next videos. In one of the next two videos, after users watch the video, we display (report) a random energy saving level, while for other video we do not report anything. The order of when energy savings are reported are selected randomly, in other words, after the first default video, we may report an energy saving or the energy saving is reported on the third play. The reported energy saving is a random number between 2% to 12%. These thresholds (i.e., reported energy saving levels) are determined by dropping screen brightness up to 30% in varying brightness levels and measuring the average energy consumption off-line on the selected applications (Angry bird game play and instagram). Figure 5.1 visualizes a typical video ordering. Table 5.1 also shows the video order and their notifications displayed to users after the videos. Once a video ends, we display a 5-likert radio-style chart (from 1 to 5, where 1 is very bad and 5 is very good) to collect subjective user satisfaction on the brightness of the videos. Thus, we compare the reported energy savings' effect on subjective user ratings under same conditions. We should also note that, in this study, we rename the videos and present them in different web pages in order to prevent users from realizing that they are watching the same videos 3 times.

5.1.2 Results

In this section, I present the results of our mTurk study. In our analysis, we categorize the videos based on their displayed notifications afterwards. Therefore as ”default videos“ we refer the first displayed videos; as ”reported videos“ we refer to the videos where energy saving is reported to the user and as ”not-reported videos“ we refer the videos where no energy notifications are displayed. Table 5.2(a) shows the averages of user satisfactions for both applications. As shown in the table, users' satisfactions increase if they are notified with energy savings. We observe the average satisfactions as 3.98 and 3.91 for the game application and 3.91 and 3.69 for Instagram application for the reported and not-reported videos, respectively. We also observe that for the game application, default video's satisfaction is highest. Although all videos are same, it is possible that users expect more screen brightness for game application and lower their ratings. Further, we perform t-tests between the satisfaction ratings of reported and not-reported videos in order to see how their means differ from each other. Table 5.2(b) shows the p-values of the t-tests. As shown in the table, especially in Instagram application, we observe a low p-value (as low as 0.066), which indicates that (for significance level of 0.1) the averages of the reported versus not-reported videos are different from each other. Similarly, it is possible that users think there is a brightness reduction in game application and they might not like it, since we do not observe a significant difference between the reported and not-reported videos on the game application.

Next, we look at the relation of reported energy saving levels and user satisfactions. Specifically, we are interested in the changes on a user's satisfaction ratings between the reported and not-reported videos based on the reported energy saving levels. Table 5.3 shows the average reported energy savings and the proportion of users based on the change in the satisfaction ratings of reported and not-reported videos. Therefore in the first column, ”increased satisfaction rating“ means, users increase their ratings in the reported video and ”decreased satisfaction rating“ means users decrease their ratings in the reported video (compared to the not-reported video). As shown in the table, for both applications, we observe more users give higher ratings on reported videos compared to not-reported videos. For game and Instagram applications, we observe 32% and 47%

of the users give higher ratings to reported video and their average reported energy savings are 7.60% and 7.54%. On the other hand for same applications, we observe 24% and 28% of the users give lower ratings to reported videos and their average reported energy savings are 5.93% and 7.12%. It is possible that, users may be less likely to sacrifice their brightness in game application comparing to Instagram application. However, since the videos are identical in both applications, these results show that there is a direct positive effect between the reported energy savings and the user satisfaction. This correlation becomes more profound as the reported energy saving levels are increased. In Section 5.2, we will further analyze this relation.

Before starting the study, we also asked users to fill a form in order to collect some personal and environmental factors. Although we make comparisons user-wise to eliminate environmental factors among users, we were interested in investigating whether it is possible to see trends in user satisfaction of brightness based on their age, phone usage, etc. Hence, we compare the satisfaction levels of reported and not-reported videos based on these factors. We first categorize the users based on their ages. We observe that users younger than 35 years old give significantly higher ratings to reported videos compared to not-reported videos, while users older than 50 years old give either same or lower ratings. We observe the highest difference among users for ages 18 to 24: we observe average ratings as 3.66 and 3.75 for game application and 3.67 and 3.91 for Instagram application for reported and not-reported videos, respectively, for this age group. This shows that, for younger people, battery lifetime is more important. Moreover, we collect users' daily smartphone usages in a 5-likert scale (1 is minimum usage, 5 is maximum usage). Not surprisingly, we observe that as the usage increase, the average satisfaction ratings of reported videos also increase. Users become more sensitive to battery lifetime as they use their smartphones more frequently. We observe similar outcome when we compare users' application usage ratings to satisfaction ratings. For both applications, users who give higher ratings to application usage (again in 5-likert scale) also give higher satisfaction ratings to reported videos. On the reported videos, we observe 12.03% and 9.90% increase in satisfactions between users with high application usage (ratings 4 or 5) and low application usage for game and Instagram applications, respectively. When

we compare the genders of the users, we also observe that while male users (compose 62% of all users) give higher usage ratings to game applications and are more sensitive to energy savings on game application, female users give higher usage ratings to Instagram application and are more sensitive to energy savings on Instagram application.

Finally, we collect ambient light level of the environment again in a 5-likert scale (1 is very dim, 5 is very bright). We observe that 25% of the users select their environment's ambient light level as maximum (5 out of 5). We also observe that, their average satisfaction ratings for reported videos also decrease comparing to not-reported videos: their averages are 4.2 and 4.66 for game application and 3.6 and 4 for Instagram application. Although users watch the same video, it is possible that, with the reported saving, they might assume they are watching the video in a dimmed brightness. While this does not cause any concerns in low ambient light environments, it may cause dissatisfaction for higher ambient light environments. Thus users' sacrifice level on brightness for more energy does also vary based on the environmental conditions.

5.1.3 Predicting User Satisfaction of mTurk Users

In the mTurk study, we observe that user satisfaction typically increases as the users are notified about the energy savings. We also observe a positive correlation between the amount of the reported energy saving and the increase in satisfaction. Having said that, there are still personal and environmental factors that might effect user satisfaction on energy savings. In order to see how accurately we can predict user satisfaction of the played videos with the collected data, we build prediction models.

We use the Weka-tool to test different prediction algorithms. For training, we use 10-fold cross-validation, therefore, training and test data never overlap. We observe that tree algorithms are relatively better on predicting the satisfaction. Our results show that, we can achieve the best accuracy of 85.23% and 86.40% using RepTree algorithm in Weka-tool for game and Instagram applications, respectively. Once we add other collected metrics (gender, age, ambient light level in the environment, phone usage level, and application usage level) we observe a slight increase on

the accuracy (85.66% and 87.02%). This shows that, although we have a slightly better accuracy by adding personal and environmental factors, we can still achieve a high accuracy on predicting user satisfaction with the given energy saving level alone.

Next, we go into the wild in order to understand how energy savings effect users' satisfaction in the daily usages with users own smartphones.

5.2 Second User Study: In-The-Wild Study

In this section, I explain our In-the-Wild study. In this study, we investigate energy savings' effects on user satisfaction in their daily usage with their own smartphones. I first explain brightness settings on smartphones. Then, I introduce our light-weight logger application. Afterwards, I explain our methodology and scenarios we use during the In-the-Wild study. Finally, I discuss our results and present our prediction model.

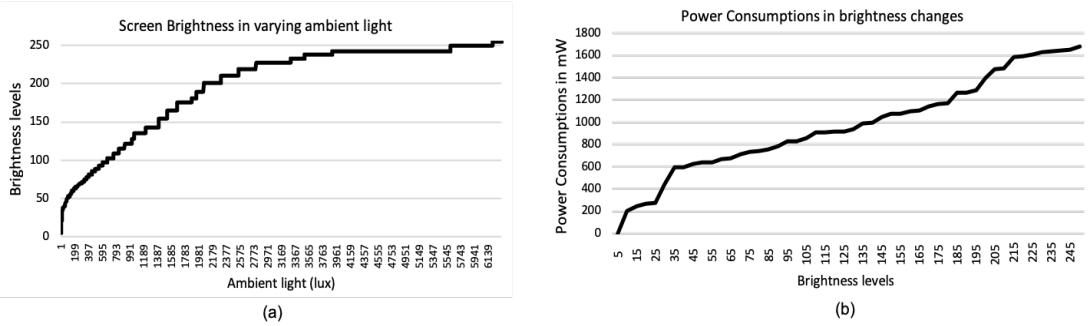


Figure 5.2: (a) Observed screen brightness levels in varying ambient light in the environment and (b) observed average power consumption for each brightness levels.

5.2.1 Setting Brightness on Smartphones

We control brightness on the target phones in order to save energy. There are three reasons we choose brightness for energy savings. First, screen is one of the most power hungry components in smartphones and any savings in screen energy will have a significant effect on overall system energy [119]. Second, controlling brightness do not require any special hardware or software support (unlike CPU scaling, which requires root access on the target phones) in application level,

which makes it possible to conduct our study in the wild with wider audience. Third, as explained in Chapter 5 introduction, in our online survey, when comparing brightness and battery, we observe 92% of users choose battery as more important than brightness, meaning that users will be more willing to sacrifice their brightness if it leads to power savings.

In smartphone displays, a high contrast ratio is a desired aspect. For instance, in a completely dark room, the contrast ratio is precisely determined by the white and black pixels in the phone display. However, in an environment with external/ambient light, display's optical characteristics do not necessarily absorb all of the incoming ambient light and some of that light is reflected back to the viewer. This reflected light reduces the resulting contrast ratio. Therefore, modern smartphones use ambient light sensor to measure the incoming light level and control brightness accordingly in order to increase contrast ratio for better user experience.

In order to understand how brightness is controlled on the smartphones, we reverse-engineer the method by observing brightness values under all possible ambient light environments a typical smartphone can measure. We use two devices for this experiment: a Nexus 6p and a Samsung Galaxy S7. Both observed screen brightness level and ambient light reading data are retrieved from the operating system. The continuous model of this data is presented in Figure 5.2(a). Note that we observe the same outcomes for both devices. As shown in the figure, in the default settings, overall, the screen brightness follows the square root of the ambient light levels, reflecting Steven's Law. In Figure 5.2(b), we also show the increase in power consumption (compared to the zero brightness setting) for varying brightness levels. As shown in the figure, increase in power consumption is almost linear with the increase in brightness levels. In our experiments, we use both observations (i) to set the brightness and (ii) to report energy savings in the wild.

5.2.2 Logger Application

For our In-the-Wild tests, we develop a logger application and make it available on Google Play Store. The application is developed as a regular ART (Android runtime) executable using the Java standard libraries available in the Android framework. Therefore, it can be used in any Android

Table 5.4: (a) Observed smartphone brands and models of the in the wild users and (b) The five scenarios tested during In-the-Wild studies.

(a)

Brand/Models
Huawei - Nexus 6P
Motorola-Moto G (5) Plus
Samsung Galaxy on5
Samsung Galaxy S5
Samsung Galaxy S7 Edge
Samsung-SM-N910H
Samsung-SM-G930T
Sony Xperia Z3
ZTE - Blade V8

(b)

Scenarios	Brightness Drop	Report Energy Saving
Scenario 1	20%	Yes
Scenario 2	40%	Yes
Scenario 3	20%	No
Scenario 4	40%	No
Scenario 5	-	No

smartphone without any hardware or software support. Table 5.4(a) shows the brands and models of the phones of the users who participated in our In-the-Wild study. At a high-level, the application consists of two parts: (1) a GUI part (2) and a background service part:

- The GUI part: When the application is first started in the phone, it asks basic user permissions such as overwriting settings to control brightness, accessing sensors to monitor ambient light sensor, etc. Also a notification is shown on the upper bar of the phone screen to inform users that the application is running in the background. Moreover, a pop-up questioner (5-likert scale chart) is displayed every 6 hours to collect user satisfaction ratings.

- Background service: The background service implements three main tasks: 1) logging user satisfaction ratings and system metrics shown in Table 5.5, 2) controlling brightness and 3) selecting and implementing daily scenarios in random order (we further discuss this in the next section). To prevent perturbation, logging and setting screen brightness occur every 5 seconds. Also, the service periodically looks for a network connection and sends the logs back to our server. We observe that our logger increases the CPU utilization less than 1% on average and adds less than 50 mW additional power consumption in the worst case. In case our background service was implemented in the kernel space instead, its overhead to the system would be even smaller.

Table 5.5 shows the collected system metrics by our logger application. As discussed in Section

Table 5.5: Collected system metrics during the In-the-Wild study.

System Metrics	Explanation
Ambient light	We collect ambient light in the environment in order to determine default brightness level
Screen Brightness	We collect screen brightness level to calculate energy savings.
Current and Voltage	We collect current and voltage to calculate power consumption.
Phone usage	We collect the "screen on" times in order to calculate energy consumption.
Battery level	We collect the battery levels on the target phones.
User satisfaction ratings	We collect user satisfaction ratings for each scenario through the pop-up questioners.

5.2.2, we collect ambient light in the environment to control brightness and we collect phone usage (in time) along with screen brightness levels to report energy savings. More specifically, we map corresponding brightness levels for all ambient light and power consumption levels for all brightness settings in our logger application in order to calculate energy savings with the phone usage. We must note that, these metrics are easily accessible (with basic user permissions) in system level without any additional software or hardware support.

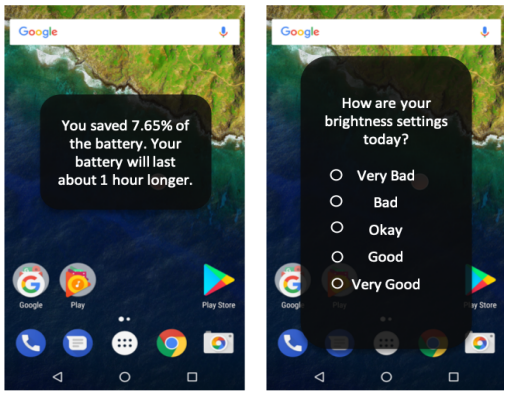


Figure 5.3: (a) Pop-up screen reporting the energy savings and (b) pop-up questionnaire to collect the user satisfaction ratings.

5.2.3 Methodology and Scenarios

We conduct our In-the-Wild study with 20 smartphone users. The majority of participants are college students and all participants are under the age of 50. Participants are gathered through fliers advertising the study and word of mouth. The experiment is started by installing the logger

application from Google Play Store and giving all requested permissions. Each user study takes 10 days. During the 10 days, users use their smartphones as they would typically use, while the logger application runs in the background. In the meantime, we monitor the logging data sent from the users' phones to our server.

During the In-the-Wild study, in order to understand energy savings' effects on user satisfaction, we define 5 different scenarios. Since a user study takes 10 days, each user tests each scenario exactly 2 times. Table 5.4 summarizes the scenarios tested in the study. As shown in the table, in Scenarios 1 and 2 we drop brightness 20% and 40% from its default value and report the saved energy value to the user, in Scenarios 3 and 4 we drop brightness again to 20% and 40%, but we do not report any savings. In scenario 5, we neither drop nor report any savings and let the brightness to be controlled by the default mechanism on the users' phones. Thus, we make users to test same brightness drops by (i) notifying and (ii) not notifying them on how much energy they save. **Please note that the reported energy savings are based on the voltage/current readings and represent the actual savings achieved by the applied scheme**. During the study, a pop-up questioner is shown to users every 6 hours to collect user satisfaction ratings. During the Scenarios 3, 4, and 5 we only display pop-up questioners for user rating. For Scenarios 1 and 2, we also display the energy savings right before the questioner. Therefore, we can see the effect of reporting energy savings on the satisfaction ratings. Figure 5.3 shows the formats of reported energy saving and user satisfaction questioner pop-ups.

We determined brightness drop levels and scenarios by conducting experiments with different combinations offline in our lab. It is possible that some alternative dropping levels or different scenarios may be preferable by other developers. The selected drops and scenarios could also be expanded to make different observations. However, we find the selected brightness drops and scenarios are good enough to analyze a variety of energy saving levels and their impact on user satisfaction within a reasonable experimental duration.

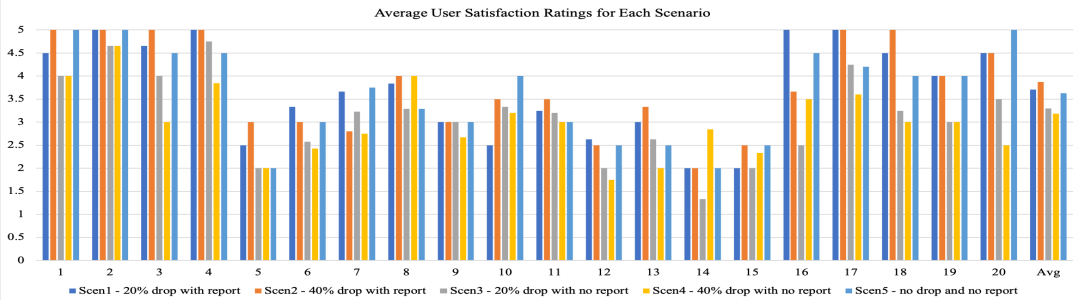


Figure 5.4: Average satisfaction ratings across users for each tested scenario.

Table 5.6: T-test result (p-values) of scenarios.

	Scen 3	Scen 5		Scen 4	Scen 5
Scen 1	0.07599	0.8035	Scen 2	0.009464	0.6335

5.2.4 Results

In this section, we present the results of our In-the-Wild study. Figure 5.4 shows the individual user satisfaction ratings for each scenario. As shown in the figure, there is a high variation across reported user satisfactions. Although some users do not notice the differences or do not get effected from the changes on the screen and constantly give low (e.g., user 15), medium (e.g., user 9), or high ratings (e.g., user 2) to all scenarios, majority of users show high sensitivity between the scenarios. More, while some users enjoy the more energy savings by sacrificing the brightness of their phones more (e.g., user 18), for some other users, reducing brightness causes dissatisfaction (e.g. user 10). Having said that, in general the first 2 scenarios, where we drop the brightness and report energy savings, have the highest satisfaction ratings (the right-most set of bars in the figure present the average ratings across all users). If we compare the reported versus not-reported scenarios with the same brightness drops (i.e., Scenario 1 versus 3 and Scenario 2 versus 4), we observe only three users give higher ratings to not-reported scenarios. We observe average ratings as 3.70, 3.87, 3.29, 3.18, and 3.63 for Scenarios 1 through 5, respectively. It is also worth to note that we observe the reported energy savings range between 2% and 23%: the averages are 5.62% and 7.91% for Scenario 1 and Scenario 2 (where we drop brightness 20% and 40%), respectively. Another interesting observation is that both reported scenarios (Scenario 1 and Sce-

nario 2) show higher satisfaction ratings than the default scenario (Scenario 5), where we do not drop the brightness. This shows that, informing users with their energy savings, even though the overall savings are somewhat small, improves the satisfaction significantly. In fact, we observe that dropping brightness 40% and not reporting energy savings (Scenario 4) is the least satisfied scenario for our users. However, if users are notified with the energy savings achieved from this brightness reduction, it becomes the most satisfied scenario (Scenario 2). Overall, we observe the highest satisfaction differences with 17.8% improvement between Scenario 2 and 4. This suggests that as the energy saving increases above a threshold, users are willing to sacrifice more (as long as they are aware of the savings).

Further, we conduct t-tests on the reported versus not-reported scenarios with the same brightness drops (i.e., Scenario 1 versus 3 and Scenario 2 versus 4). Table 5.6 shows the p-values of the t-tests. The p-value of 0.009465 for the comparison of Scenario 2 and 4 indicates that these two scenarios do not have the same mean. Similarly, the p-value of 0.07599 between Scenario 1 and 3 indicates that there is a small chance that these two scenarios have the same mean (albeit a much higher probability compared to the comparison of Scenarios 2 and 4). These results also show that when informed about the energy savings, users are more satisfied and willing to sacrifice their screen brightness more.

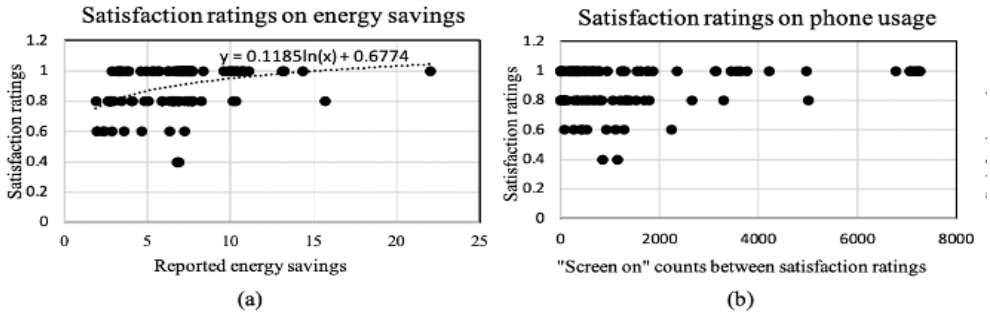


Figure 5.5: (a) Normalized User Satisfaction Ratings on Reported Energy Savings, (b) Normalized User Satisfaction Ratings on Phone usage

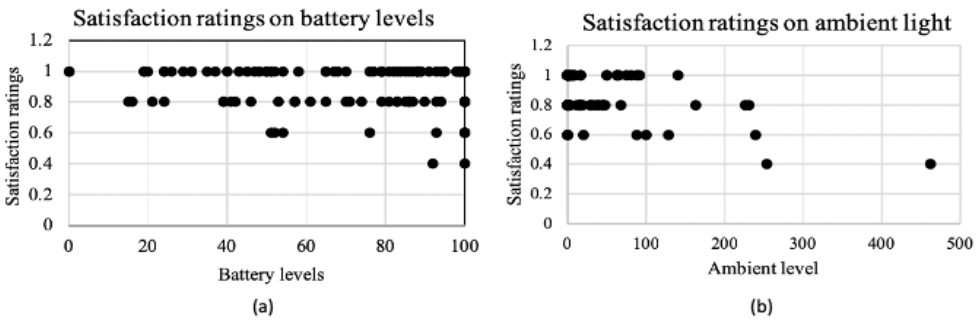


Figure 5.6: (a) Normalized User Satisfaction Ratings on Battery Levels and (b) Normalized User Satisfaction Ratings on Ambient light level in the environment.

5.2.5 Predicting User Satisfaction for In-The-Wild Users

In the previous section, we show that user satisfaction varies across users. To understand how energy saving levels impact user satisfaction, we gauge how well the reported energy savings correlate with the subjective ratings. Our goal is also to investigate the nature of the correlation between the reported energy saving levels and user satisfaction. Therefore, we analyze the accuracy of predicting user satisfaction (ratings) for a reported energy saving level. Specifically, we develop a prediction model that takes the energy savings as input and predicts the user's satisfaction. For the training data, we use all reported energy savings and their corresponding satisfaction ratings. In the model, the ratings are normalized for each user such that maximum rating of the user is set to 1 and a fixed value (1/5 = 0.2) is decreased for each sub-points. For example if user A has a maximum rating 5 and another rating 4 for reported energy savings, we normalize these ratings as 1 and 0.8 in order to reflect this 1 point drop in the satisfaction. Similarly if user B has a maximum rating 3 and another rating 2; we also normalize these ratings as 1 and 0.8. Because user B has the same 1 point drop in his/her satisfaction rating scale as the user A had. For the data, we use the reported energy savings obtained during the tests of Scenario 1 and Scenario 2 and the corresponding reported satisfactions to build the supervised learning models.

Similar to analysis in Section 5.1.3, we use Weka-tool to test different prediction algorithms. For training, again we use 10-fold cross-validation, therefore, training and test data never overlap. We observe that, regression and tree algorithms are relatively better on predicting the satisfaction.

We test a variety of linear and non-linear regression algorithms/equations to predict user satisfaction with the given energy savings. Our results show that, we can achieve the best accuracy with 86.6% rate using Non-linear Logarithmic regression with the following formula:

$$user\_satis = 0.1185 * ln(energy\_saving\_level) + 0.6774$$

Figure 5.5(a) shows the data points of normalized user satisfaction ratings on the reported energy savings along with the equation's curve. As shown in the figure, user satisfaction increases as the reported energy saving increases.

Using tree algorithms (e.g. RepTree), we can predict the satisfaction with 85.6% accuracy. Since tree algorithms are formed with branches based on their separation the data into two, we observe the energy saving value separates the user satisfaction most is 7.6979: above this value is assigned as 0.95 (closer to 1, maximum satisfaction) and under this value separates again for lower satisfaction ratings. This result shows that for the energy saving above 7.69%, most of the users give their maximum ratings.

We also analyze the relation between other collected metrics (Table 5.5) and satisfaction ratings. Figure 5.5(b) shows the satisfaction ratings for smartphone usage. We calculate the smartphone usage as the "screen on" log counts between reported satisfaction ratings. Similar to Section 5.1.2, we observe a positive correlation between the phone usage and user satisfaction: as the phone usage increases, user satisfaction on energy savings also increases. Users who use their smartphones more are more sensitive to energy savings. On the other hand, when we look at the relation between battery levels and user satisfaction ratings in Figure 5.6(a), we observe a negative correlation: as the battery level increases, user satisfaction drops. Considering that users with lower battery levels typically need more energy savings, it is reasonable for them to give higher priority to energy savings. Since energy savings would not be as crucial for users with higher battery levels, they may not want to sacrifice their brightness. Finally, Figure 5.6(b) shows the relation between ambient light in the environment and satisfaction ratings. Similar to Section 5.1,

there is a negative correlation between the ambient light in the environment and satisfaction ratings. On higher ambient light levels, users need more brightness in order to increase contrast on the smartphone screen. Therefore, dropping brightness may be causing more dissatisfaction. On the other hand, user satisfactions increase in the environments where ambient light is lower.

I should also note that, once we add all collected metrics in Table 5.5 to the prediction model we observe a similar accuracy (86.02%) on predicting the user satisfaction. Thus, given energy saving level alone is good enough to achieve high accuracy on predicting user satisfaction.

5.3 Summary

In this project, we study energy savings' effects on user satisfaction. Specifically, we quantify the importance of energy savings on user satisfaction by conducting two user studies.

We conduct our first study over the Amazon Mechanical Turk platform. In this study, we make users watch the identical game play (and instagram) video 3 times and report their satisfactions about the brightness. In order to understand how energy savings' affect user satisfaction, in one of these videos, after watching the video we display (report) a random energy saving level. Our results show that, although users watch the same videos, their average satisfaction increase when the system reports energy savings.

In our second user study, we go into the wild in order to understand the energy savings' effect on user satisfaction using users' own smartphones in their daily lives. In the wild, our users test and rate 5 scenarios, one scenario each day (each scenario twice for a 10-day study). In these scenarios, we drop brightness by either 20% or by 40% and either report or do not report energy savings (the fifth scenario is the default brightness management scheme of the phone). We then compare reported and not-reported scenarios' satisfactions to analyze energy saving reports' effects. We observe that user satisfaction is the highest on average when the brightness is dropped by 40% and the system reports saved energy.

Finally, we build prediction models to correlate user satisfaction with the reported energy savings, which to the best of our knowledge first of its kind.

CHAPTER 6

ALTERNATIVE USE OF USER SATISFACTION: MITIGATING SMARTPHONE AND APPLICATION OVERUSE

Recent technological advances have led smartphones to become an indispensable part of modern society. Today, there are more than 3.2 billion active smartphone users in the world [122]. Moreover, smartphones are nearly ubiquitous among younger adults: 96% of Americans and 93% of British aged 18 to 29 own at least one smartphone [29, 100].

Although smartphones have helped to improve the quality of life in some sectors by enabling "on-the-go" access to several aspects (e.g., web-browsing, communication, shopping, banking, gaming, etc.), the increasing dependency on smartphones also brought growing concerns regarding their negative aspects such as excessive usage. Users spend more time with their phones (on average 21 hours per week and about 90% of the time is in applications) than using any other devices (i.e., laptops and desktop-computers) [110]. Recent research has highlighted a number of potential problems as a consequence of mobile overuse: addiction, financial problems, and dangerous use (e.g., whilst driving) [83, 143]. There is also an increasing concern among parents for their children's excessive phone usage due to its possible negative effects on both social and academic life [83, 145, 143]. In addition, studies also show certain harmful health effects, which might be caused by the immoderate use of phones including cancer, headache, sleep disorder, anxiety, and depression [35]. Moreover, the World Health Organization considers excessive mobile phone use as a public health concern [101], emphasizing the need for more research on preventive measurements.

Despite the importance of reducing excessive phone use, number of methods and their efficiencies are quite limited and/or still unknown. A quick solution to reduce usage would be implementing time restrictions to the phone or applications. While this can lead to significant reductions is usage, it has been show that such restrictions can cause serious possible side effects on users

such as anxiety and depression [2, 95]. Thus, there is a need to develop better methods to mitigate overuse in daily life. In this work, we degrade user experience by controlling display brightness in order to reduce excessive phone usage.

The display is the primary user interface on smartphones. While earlier displays were only an output system for these devices, the introduction of touchscreen feature made them the main input component as well. Visibility of the display is therefore one of the most important aspects for high quality user experience. In order to create the desired visibility for the display, systems typically control brightness by measuring ambient light level in the environment: if the smartphone is under direct sunlight, backlight is increased in order to provide a better visibility whereas if the phone is in a dark room, backlight is dimmed in order to increase user experience (generally a too bright display is not desirable in a dark room). In fact brightness settings has a direct and instantaneous effect on the current user experience about the phone [74, 114, 113].

In this project, we propose Phone Free, a system to reduce excessive daily phone usage of the users. The system relies on the hypothesis that creating discomfort by altering brightness levels (or showing pseudo pop-ups on the screen) will yield users spend less time with their applications and hence phones. In order to test this hypothesis, we develop three different models to degrade user satisfaction on the target phone: gradually or rapidly dimming brightness and showing pseudo pop-ups on the screen. For the rest of the chapter we refer them as “gradual model”, “rapid model” and “pop-up model”, respectively. We evaluate our system by conducting an IRB (Institutional Review Board) approved user studies with a total of 30 real users and 22 different smartphone models/brands with varying operation system versions.

We conduct our user tests in the wild by releasing a logger application into the market. During the experiments, users test three models and the default scheme each day in a random order. Their combined data represents a total of over 240 days of worth of recorded use. We regularly collect overall display satisfaction of the users through 5-point likert chart pop-up questioners. Our results show that when compared to default scheme, our models reduce maximum application session length by up to 37.82% on average with a negligible sacrifice on satisfaction. Moreover, due to

brightness dimming mechanisms and overall reduced phone usage with the proposed models, we can save energy especially in prolonged usage scenarios. Our results also show that the proposed system saves up to 5.68% system level power, when compared to the default scheme. We discuss the session lengths and power savings for all proposed models in more depth In Section 5.

Table 1 shows the brands/models and Android versions of the participants' phones. Note that, in our user studies, we collect data by releasing a logger application to Google Play Store. Therefore, we did not attempt to control or select any of our participants. The users are anonymous people who downloaded our logger application from the market and use it just like any other application they install from the market.

Overall our primarily contributions can be listed as below:

- We show that we can reduce maximum and average application session lengths by degrading user experience over time.
- We show light-weight tools and methods to reduce excessive phone usage in real time.
- We show that by controlling brightness and decreasing application session lengths, we can achieve on average 5.68% system level power savings.
- We show analysis on how gradually and rapidly dimming brightness effect user experience and hence phone usage.

The rest of the project is structured as follows. In Section 6.1, I explain our methodology. In Section 6.2, I describe our experimental study. I discuss our results in Section 6.3. I present further discussion in Section 6.4.

6.1 Methodology

In this section, I discuss our methodology for our user studies. I first introduce our logger application along with collected sensor and system information and the overhead of our tools. Then, I explain controlling brightness in modern smartphones.

Table 6.1: Participants' Phone Brands/Models and API Levels

ASUS-X00PD - 26	ONEPLUS A5000 - 25
Google-Pixel 3 - 28	ONEPLUS A3003 - 28
Google-Pixel 3 XL - 29	Realme-RMX1971-29
Huawei-LND-AL30 - 26	Samsung-SM-G - 25, 27, 28, 29
HMD Global-Nokia 1 - 28	Samsung-SM-J - 25, 26, 27, 28
Infinix X650C - 28	Samsung-SM-N - 29
I-life-ITELL-K3500N-27	Samsung-SM-S - 28
LGE-LM-X120 - 28	Samsung-SM-T - 28
Motorola one macro-28	TCL-5032W - 28
Motorola-moto e5 cruise - 26	Vivo 1803 - 27
Motorola-XT1562 - 25	Xiaomi-Redmi 5 Plus - 27

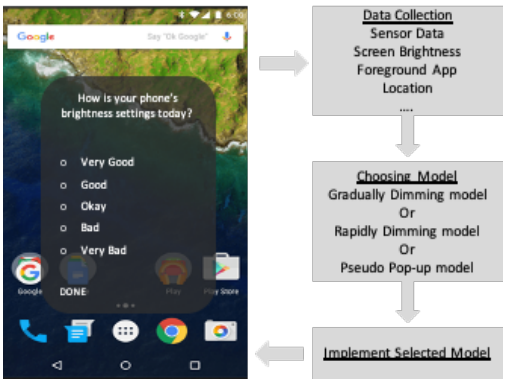


Figure 6.1: GUI part of the logger application (on the left) and pop-up questioner and background service functions from data collection to implement the selected model (on the right)

6.1.1 Logger Application and Collected Metrics

We develop a sensor-logger application to collect system metrics from target devices and release it on the Google Play Store. The logger is developed as a regular Android Runtime (ART) executable using the Java standard libraries available in Android frameworks. Thus, it runs on all Android smartphone devices without any special hardware or OS support. At a high-level, the logger application consists of two parts: (1) a GUI to collect 5-point likert scale user satisfaction ratings and (2) an associated background service to provide the logging functionality.

The GUI application is designed to collect user satisfaction ratings about the phone's brightness settings through pop-up questioners shown on the screen in every 6 hours (Figure 6.1(left)). We collect the user ratings in 5-point likert scale chart: 1 is worst, 5 is best. We analyze relation between our models and user satisfaction in Section 5.

Background service starts as the GUI starts. The service is responsible for 1) collecting system

Table 6.2: Collected System Metrics

System Metrics	Explanation
Ambient light	We collect ambient light in 1Hz sampling rate.
Screen brightness	We collect screen brightness in 1Hz sampling rate.
Application	We collect current foreground application on the screen.
Time	We collect time as in 24 hours scale every hour and in 4 phase of the day as morning, noon, evening, and night
Battery level	We collect battery level on a scale 1-100 by listening every battery change event.
Location	We collect latitude and longitude values of the phone every 20 seconds using Android Location API [14]
Activity	We collect user activities every 20 seconds using Android Activity Recognition API [55]
Power Consumption	We collect current and voltage values in 2Hz rate using Android's Battery Manager API [57]

metrics, 2) choosing the model to be tested in random order each day, 3) keeping the application session times and 4) implementing the selected model in order to reduce overuse. Figure 6.1(right) shows the background service's tasks. Also, the service periodically looks for a network connection and sends the collected data back to our server when the collected data size exceeds 500KB.

In our proposed system, in order to reduce excessive phone usage in real time, we collect data from various system metrics using default Android APIs. Table 6.2 lists the used metrics and sampling rates used for our models. We select these metrics in order to control the brightness and implement our models in a timely manner. We also explore the relation of the collected metrics, user experience, and application usage sessions in Section 5 in more depth. As shown in the table, we collect sensor data (ambient light), foreground application information, activity, and location data from the target smartphones. All of these metrics can be monitored with basic user permissions (i.e., body sensors, usage statistics, location access, etc.) that can be verified through our application. We collect most of our metrics either as event driven or in lower sampling rates in order to decrease application overhead. We should note that, we could also collect some other metrics like CPU usage and complex touch events (i.e. size, 2D coordinate, etc.) from logcat in the

target phones for our analysis. However, since monitoring logcat requires super-user permission (i.e., root access), we do not include them in our collection set.

We use MonkeyRunner [98] tool in Android to measure overhead of the logger application. MonkeyRunner tool provides an API for writing programs that control an Android device or emulator from outside of Android code. We specify commands in Python scripts and sent them to two different Android phones (a Google Pixel 3L and a Nexus 6p) through adb shell [13]. We create the same workloads (same touch events, same run-times, etc.) and test each metric combinations (comparing overhead with and without the metric) for three minutes on three different applications: a CPU-intensive game, CPU-moderate video, and a CPU-low animation application. We observe that our background service is lightweight mainly because of the small frequency of data collection. We observe logger application increases CPU utilization by no more than 2% on the smartphones. Additionally, activating the logger increases energy consumption by less than 2% on the target phones on average. With the given sampling rate, monitoring location consumes 50-90mW more power on average, while ambient light sensor and other metrics' overhead stays in the 10-30mW range. In addition, we collect current and voltage values at 2Hz sampling rate to calculate power consumption of the smartphone (note that the reported power levels are for the whole phone).

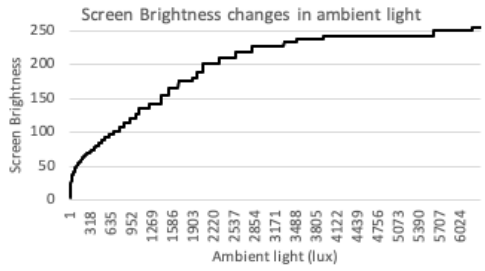


Figure 6.2: Default scheme's screen brightness settings in all ambient light changes.

6.1.2 Controlling Brightness in Smartphones

As explained in the introduction in Chapter 6, in gradual and rapid models, we control brightness in order to degrade user experience. There are two main reasons we choose brightness in our studies.

First, screen is the primary user interface, thus brightness has a direct and instantaneous effect on user experience [114, 119]. Second, controlling brightness do not require any special hardware or software support in application level, which makes it possible to conduct our study in the wild with a wider audience. Specifically, in our preliminary in-the-lab experiments, we observed that there are some other methods that can be utilized in order to degrade user satisfaction, such as slowing phone performance with CPU scheduling. However, most of these methods require root access on the target phones, which makes it impractical to reach a bigger and variety of crowds in the wild.

In smartphone displays, a high contrast ratio is a desired aspect. Modern smartphones use ambient light sensor to measure the incoming light level and control brightness accordingly in order to increase contrast ratio for better user experience. In order to understand how brightness is controlled on the smartphones, we reverse-engineer the method by observing brightness values under all possible ambient light environments a typical smartphone can experience. We use two devices for this experiment: a Google Pixel 3L and a Nexus 6p. Both observed screen brightness level and ambient light reading data are retrieved from the operating system. The continuous model of this data is presented in Figure 6.2. We observe that in the default settings, overall, the screen brightness follows the square root of the ambient light levels, reflecting Steven's Law [123]. In our logger application we map each ambient light to its corresponding brightness level. Thus, in gradual and rapid models, we use this observation to reset the brightness back to its default settings when needed.

6.2 Experimental Study

In this section, I discuss our experimental study. I first introduce our proposed three models (gradual, rapid, and pop-up) and discuss the process of degrading user satisfaction during excessive usage. Then, I explain our in-the-wild user studies.

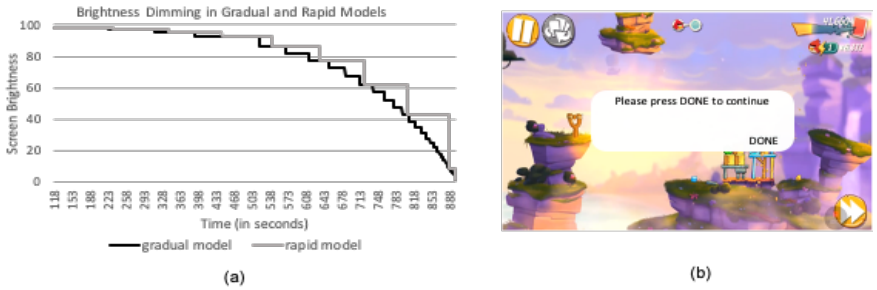


Figure 6.3: (a) Gradual and Rapid model brightness settings when offset brightness is level 100 and (b) An example of a pseudo pop-up in the pop-up model during a gameplay.

6.2.1 Tested Models

In the proposed system, we degrade user satisfaction about the phone to reduce overuse. In order to do that, we develop three different models where we gradually or rapidly dim brightness and do not change brightness but show pseudo pop-ups on the screen. In all tested models, we start degrading user satisfaction when the foreground application's usage exceeds 2 minutes without any interruption. We keep degrading satisfaction frequency in an accelerating fashion as long as user continues using the same application. If the user changes or exits foreground application or manually alter the screen brightness, we reset the models' degrading frequency back to its initial value (2 minutes).

In the studies, users test three models and the default scheme in a random order each day:

- *When the default model is selected*, the application does not create any user dissatisfaction and leaves the brightness control over to the default scheme. Note that the default scheme may be different based on to the Android version of the user.

- *When the gradual model is selected*, logger-application starts decreasing brightness gradually if the foreground application's usage exceeds 2 minutes without any interruption. In gradual model, the time between consecutive dimming events is decreased as long as user continues using the application. Whenever user changes or exits the application or manually alter the screen brightness we reset the model back to its initial settings.

- *When the rapid model is selected*, similar to the gradual model, we create user dissatisfaction

Table 6.3: Tested Models Along With Their Degrading Procedures and Reset Criteria

Model	Degrading Procedure	When to Reset
Default Scheme	-	-
Gradual Model	Dims brightness gradually in an increasing frequency over time	User exits or changes the foreground application, or user alters brightness manually
Rapid Model	Dims brightness rapidly in an increasing frequency over time	User exits or changes the foreground application, or user alters brightness manually
Pop-up Model	Shows pop-ups on the screen in an increasing frequency over time	User exits or changes the foreground application

by dropping screen brightness on the target phones. Although brightness is set to same levels in both models, the difference here is that the times between brightness dimming events are longer compared to gradual model. In other words, we make smaller number of brightness drops, but each drop is for a larger amount.

- *When the pop-up model is selected*, we do not control brightness and leave the brightness control over to the default scheme, however, we show pseudo pop-ups on the screen in order to degrade user satisfaction. Shown pop-ups always stay on the top of the screen unless there is a user input on the screen. In pop-up model, again the frequency of pop-ups increases over time as long as user continues using the application.

In our studies, we specifically want to test the rapid model in order to see how rapid changes differ from gradual changes on user experience. Additionally, we include pop-up model in order to compare effects of other methods (other than altering brightness) on user experience and eventually on application and hence phone usage. Table 6.3 summarizes each model procedures and their reset criteria. In all models, we increase the frequency of user dissatisfaction events (either dimming brightness or showing more pop-ups on the screen) with a fixed acceleration rate:

$$t_{new} = \alpha * t_{prev} \tag{6.1}$$

where t_{new} is the time to next dissatisfaction event, α is a fixed rate for each model, t_{prev} is

time to previous dissatisfaction event.

Initial value of t_{prev} is set to 120 (2 minutes in seconds) and; α is 0.86 for gradual and pop-up models and 0.95 for rapid model, where we create user dissatisfaction events with a lower frequency. As shown in equation 6.1, duration between two dissatisfaction events shortens by the time, meaning models degrade user satisfaction more frequently as long as user continues using the same application.

For gradual and rapid models, we set the brightness with an inverse exponential formula:

$$brightness_{new} = -\exp(\beta * time_{app}) + brightness_{offset} \tag{6.2}$$

where β is a fixed rate between 0.4 and 0.2 (depends on the offset brightness), $time_{app}$ is the total application session duration, $brightness_{offset}$ is the initial brightness when the application session is started.

Figure 6.3(a) shows the brightness settings over time for gradual and rapid models when offset (starting) brightness level is 100. As shown in the figure, the gradual model procedure is as follows: after 120 seconds (2 minutes) of use, brightness is dropped to level 98 (based on the equation 6.2), after 104 seconds (120 * 0.86) of continuous use, brightness is dropped to level 96; then 90 seconds (104 * 0.86) later to 93 and so on. Same procedure for rapid model is: after 120 seconds of use, brightness is dropped to level 98, after 114 seconds (120 * 0.95) of continuous use, brightness is dropped to level 96; then 108 seconds (114 * 0.95) later to 93 and so on. Although, at first, the distinction between these two models is not clear, as the use time increases, brightness drops in rapid model will be much more rapid compared to gradual model. In fact, as shown in Figure 6.3(a), when the use time exceeds 10 minutes (600 seconds), the brightness drop in rapid model is more than 18 level, whereas in gradual model this number is about 6 level.

During the session, whenever user changes the foreground application or brightness manually, models reset the next degrading event time back to 120 seconds. We make sure that, for gradual and rapid models, no matter what the offset brightness is, models drop the brightness to minimum value (level 1) after 900 seconds (15 minutes) of use: when $time_{app}$ equals to 15 minutes. We

determine α, β and $time_{app}$ levels by conducting preliminary experiments both in our lab and in the wild. Specifically, in the experiments, we observe the average and maximum phone sessions as 15.2 and 80.9 minutes, respectively. Thus, we define the maximum application session duration as 15 minutes for our models. Although we set the maximum session duration to 15 minutes, due to our recurring user dissatisfaction events, it is highly possible for users to become uncomfortable sooner. In fact, as shown in Figure 6.3(a), brightness reduced almost by half in the first 10 minutes. It is also possible some other levels may be preferable by different developers. These levels could also be changed to make our system more (or less) aggressive. However, we found these bounds to work well in general.

Figure 6.3(b) demonstrates how pseudo pop-ups are shown on the screen during an application session. As shown in the equation 6.1, similar to the other models, frequency of pop-ups increases over time as long as user continues using the same application. We also set the minimum time between pop-ups as 5 seconds in order to avoid possible stalling on screen inputs on the target phones.

I should note that, during our user tests, we exclude some applications which can be essential for phone's functionality. Specifically, we do not induce dissatisfaction events when built-in system applications or maps associated applications are used in the foreground. Moreover, we also exclude some user activities such as driving, bicycle, etc. We detect such movement-related activities by monitoring the activity and location metrics as shown in Table 6.2.

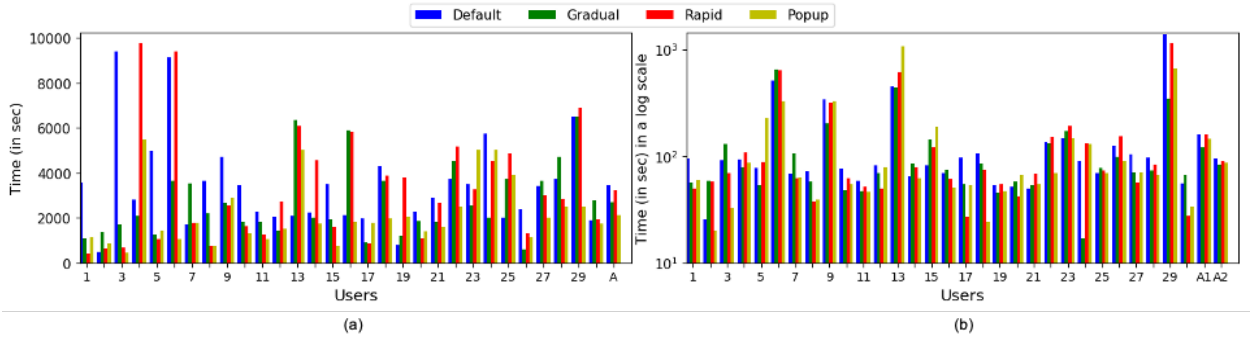


Figure 6.4: (a) Maximum application session for all users and (b) Average application sessions for all users.

Table 6.4: Post-hoc t-test results of each model

Models	Default Scheme	Gradual Model	Rapid Model	Pop-up Model
Default Scheme	-	0.0724	0.7179	0.00619
Gradual Model		-	0.3412	0.1472
Rapid Model			-	0.0422
Pop-up Model				-

6.3 Results

In this section, I present the results acquired from our user studies. We test the gradual, rapid, and pop-up models along with the default scheme in a random order each day. In total, we receive data from 58 users. Unfortunately, not all of our participants use the logger-application long enough. Also, some users did not provide enough inputs to perform an analysis (a common issue with conducting in-the-wild tests with real users). Once we filter out the users who don't give inputs to the questioner and who test each model less than twice, we end up with 30 users whose results we present below.

6.3.1 Maximum and Average Application Sessions in Each Model

We consider mobile application sessions to be consecutive periods of time during which a user interacts directly with the same foreground application on the phone. Since in the proposed schemes, models reset their timing when the foreground application is changed, we were specifically interested in seeing how proposed models perform on application session lengths.

Figure 6.4(a) shows the recorded maximum application session lengths of each users. Since usage characteristics and application selections vary, we observe a large variation between session lengths of each user. We observe the default scheme has longest sessions in 13 (out of 30) users. Moreover, there are only 2 users (user2 and user19), where default scheme's session is minimum. We observe the average maximum application sessions as (in seconds) 3441.5, 2708.3, 3227.0 and 2139.6 for default, gradual, rapid, and pop-up models, respectively. Overall, our models are successful in reducing the session lengths: compared to default scheme, on average we observe 21.30%, 6.23%, and 37.82% reductions for gradual, rapid, and pop-up models, respectively. I also

explain the possible reasons behind this in the next section (Section 5.2).

Figure 6.4(b) shows the average application sessions for our 30 users. Although, in the proposed system, we aim to reduce overuse in smartphones, we were still interested to see the performance of our models on average use as well. Similarly, due to differences in user characteristics, we observe a high variation between the sessions across users. In order to create a clear view, we use a logarithmic scale on the y-axis. We observe the average application sessions as (in seconds) 160.7, 121.4, 158.8, and 144.2 for the default scheme, gradual, rapid, and pop-up models, respectively. We also observe that, user6, user13, and user29 have significantly longer sessions. In our analysis, we also show the average values by omitting these users in "A2" in the figure, where average values drop to 94.2, 83.4, 89.5, and 85.8 for each model respectively. Overall, our models reduce average sessions by 24.46%, 1.19%, and 10.22% for gradual, rapid, and pop-up models respectively.

I should note that, in our analysis, when calculating the average sessions, we consider all application sessions. However, in a typical daily usage, there are a lot of short sessions as well. In fact, in total we observe 132859 application sessions from our users, where 37.5% of them were under 2 minutes in length. This means that, our models do not implement any dissatisfaction event 37.5% of the sessions and hence we observe relatively less reductions in Figure 6.4(b).

In order to analyze the session results further, we performed ANOVA [19] test to see how the four models' results differ from each other. In the test, we observe the p-value as 0.04843 which indicates that result is significant and the means differ. Since p-value is significant, in order to understand how each model differ from each other, we then conduct post hoc pair-wise t-tests [102]. Table 6.4 shows the measured p-values in the t-tests. As shown in the table, the p-value for the gradual and pop-up model versus default model comparison is significant (with significant level 0.1 and 0.05), meaning that, with the proposed models we can indeed reduce maximum application session lengths.

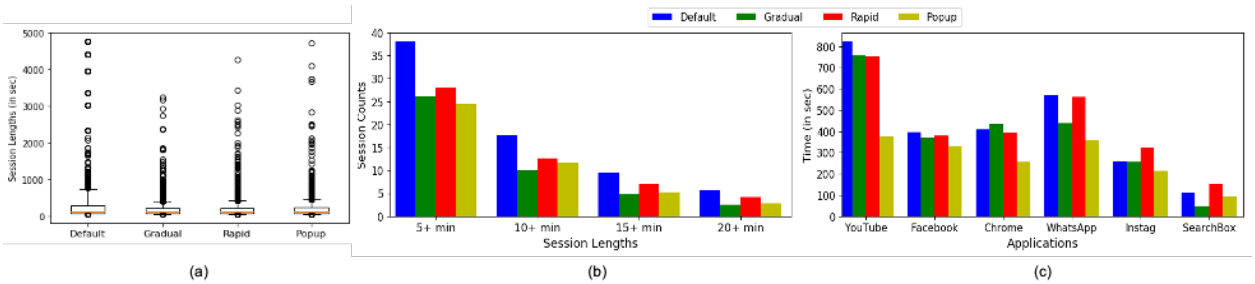


Figure 6.5: (a) Distribution of 2+ min application sessions for each model (b) Average application session counts with varying lengths of all users and (b) Averages of maximum session duration for most common applications for all users.

6.3.2 Application Session Length Analysis

Figure 6.5(a) shows the distributions of application sessions where the duration exceeds 2 minutes for each model. Since, in our models, we start to degrade user experience in longer sessions (starting at 2 minutes), we were particularly interested in the performance of our models for longer sessions. In the boxplots, whiskers are extended to include 95% of the data. As shown in the figure, average count of longer sessions in the default model is higher than all other models. In fact, we observe the total number of counts as 27618, 19312, 19710, and 16345 for the default, gradual, rapid, and pop-up models, respectively. Having said that, we still observe some long sessions in our models as well.

Figure 6.5(b) shows the average (per user) session counts of all users for each model, where the sessions exceeds 5+ minutes, 10+ minutes, 15+ minutes, and 20+ minutes respectively. As described in Section 4.1, in our proposed models, we increase the frequency of interrupts as long as the user continues using the same application. Thus, we also observe the average counts of sessions drop over time in all models. As shown in the figure, we observe default scheme has the most counts in all session lengths. Moreover, we observe the minimum session counts in the gradual model as the session lengths increase (after 10+ minutes): we observe 45.26% reduction when the session lengths reach 20+ minutes compared to default scheme.

Although we observe a high variation on both application selections and application usage characteristics of users, we extract the most common applications from our users. Figure 6.5(b)

shows the averages of maximum session lengths of the most common applications across all users. We observe the 6 most common applications of our users are YouTube, Facebook, Google Chrome, WhatsApp, Instagram, and Google Search Box. As shown in the figure, in majority of the applications, we see the maximum usage when the phone is in default scheme; and minimum usage when the phone is in the pop-up model.

When we look at Figure 6.4 and Figure 6.5, we observe that pop-up and gradual models reduce the overuse the most. For the pop-up model, the reasoning behind it could be the fact that unlike the gradual and rapid models, in order to dismiss the pop-up messages shown on the screen, users need to touch to screen. In gradual and rapid models, on the other hand, users are only exposed to dimming brightness and hence are not necessarily required to perform an action. Moreover, in pop-up model, users particularly need to change or exit the foreground application in order to reset the pop-up events, while in gradual and rapid models, users can reset the events with manual brightness changes as well. Further, we observe that the gradual model reduces overuse more compared to rapid model. Due to rapid changes in brightness, in rapid model, users notice these changes earlier than gradual model, hence alter the brightness manually sooner. Since change in brightness results models to reset their dimming events back to initial settings (2 minutes), we observe longer application sessions with the rapid model. In fact, we observe that in the rapid model, for each application session, users perform 1.91 brightness changes on average, while this rate is 1.44 for the gradual model.

6.3.3 Overall User Satisfaction

We collect ratings through a pop-up questioner four times a day in order to understand how proposed models effect the overall user experience (as explained in Section 4.2). Figure 6.6(a) plots the average satisfaction reports gathered from these pop-up questioners. In order to analyze the differences between the models, we normalize them by calculating their relative change based on the default scheme's value for each user using the formula:

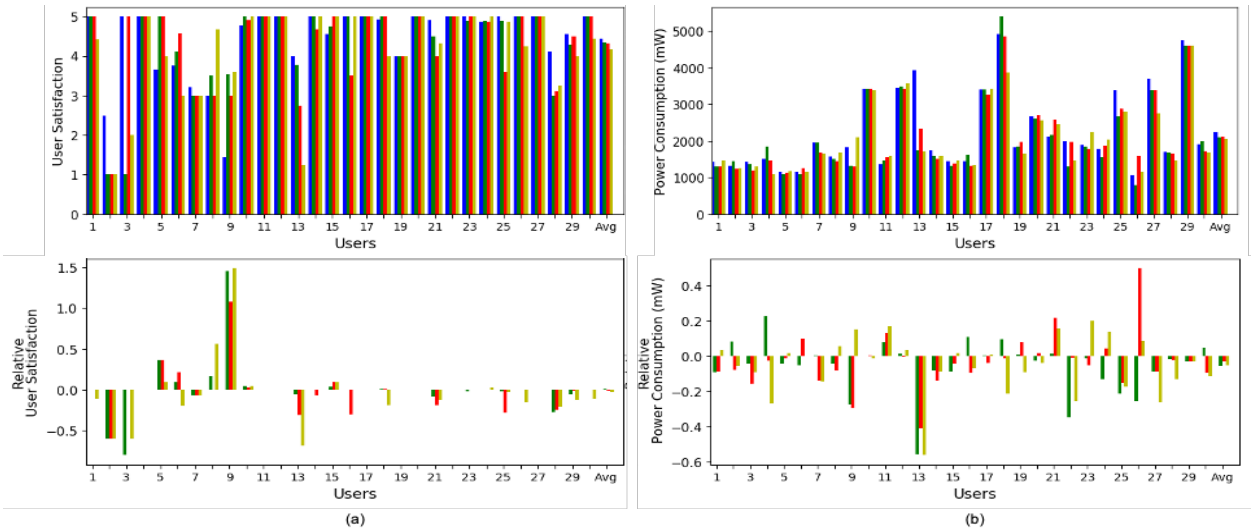


Figure 6.6: (a) Average user satisfaction ratings (top) and relative user satisfaction ratings for each model for 30 users (bottom); and (b) Average power consumption (in mW) (top) and relative average power consumptions for each model for 30 users (bottom).

$$relativechange = (Satis_x - Satis_{default})/Satis_{default}) \tag{6.3}$$

where x represents all four models for the user. Figure 6.6(a) (bottom) shows the relative average satisfaction ratings of the four tested models for all users. As shown in the Figure 6.6(a), there is again a large variation in user satisfaction levels. While some users are more sensitive to models' degrading events and can clearly distinguish the four models (e.g., user13; in fact, as shown in Figure 6.4(a) user13's maximum session lengths are also high), others do not notice any difference (e.g., user27). We observe that the default scheme and pop-up model as the most and least satisfied models, respectively. This is expected, since we also see the the most session reductions with pop-up models in Figure 6.4. Another interesting point is that although session lengths are reduced more with gradual model, we observe their ratings higher than the rapid model. The reasoning again could be that the rapid brightness changes become disturbing sooner than gradual models. Thus, for rapid model, users both give lower ratings and also alter brightness, which results in the model resetting its timing. Overall, the average ratings between models is not statistically significant: 4.44, 4.33, 4.31, and 4.17 for the default, gradual, rapid, and pop-up

models, respectively. Further, we conduct ANOVA test on the normalized values to see how the results of the four models differ from each other. We observe the p-value as 0.9649, which means that with the proposed models, we can indeed reduce maximum session lengths with no significant impact on user satisfaction.

6.3.4 Power Consumption Analysis

Figure 6.6(b)(top) plots the power consumption values of the four tested models for our 30 users. Similar to Figure 6.6(a)(bottom), in Figure 6.6(b)(bottom), for each user, we calculate the relative change on power records based on their default model's record using the equation 6.3. As shown in Figure 6.6(b), default scheme's power consumption is highest in 8 out of the 30 users. Overall, our models are successful in saving power: on average, gradual, rapid, and pop-up models save 5.68%, 3.23% and 5.08% system power, respectively. Please note that these power savings are system level and acquired by reducing phone usage and dimming brightness overuse. However, it is possible to make dimming mechanisms more aggressive with different thresholds to increase power saving. We should also note that, since we monitor the power consumption on the target phones from the operating system, the recorded power values are when the phones' CPUs are active, thus the average energy consumption comparisons are also same as power consumption comparisons.

6.4 Discussion

User selection. As discussed in the introduction in Chapter 6, all of our users are participants who downloaded our application from Google Play Store just like any other application. Although it is possible to see some similarities (or differences) in the user data based on age, gender, or being tech savvy or not, due to privacy concerns, we do not collect personal information of our users. However, based on Google's statistics, our users are from 10 different countries where majority of them are from United States, India, Pakistan, UAE, and S.Africa.

Limitations. While we are aware that the proposed system (PhoneFree) may raise some con-

cerns over usability issues for particular people (i.e. elderly people or people with visual disorder etc.), in this study we focus on mitigating smartphone overuse owing to the negative effects associated with it. Notably, such system could be pivotal for parents who aim to restrict screen time of their children and as well as individuals who want to limit their excessive phone or application usage (i.e. limiting game or social media applications etc.). Moreover, since the system is developed in application level, it is also possible to personalize the system by defining specific applications or setting different time-thresholds based on the individual preferences.

6.5 Summary

In this project, we propose a system to reduce excessive smartphone use. The system relies on the hypothesis that degrading user experience on the phone by altering brightness levels or showing pseudo pop-ups on the screen will yield users spend less time with their applications and their phones. In order to test this hypothesis, we develop three different models to degrade user satisfaction on the phone: gradually or rapidly dimming brightness and showing pseudo pop-ups on the screen.

We evaluate the proposed models by conducting user studies in the wild by releasing a logger application into the Google Play Store. Users test each model and default scheme in a random order each day, representing a combined total of over 240 days of worth of recorded use. We regularly collect overall user experience ratings about the phone display through 5-point likert chart questionnaires. Our results show that with the proposed models, on average we can reduce maximum application session duration by up to 37.82% with a negligible sacrifice on user experience. Moreover, due to brightness dimming mechanisms and reduced usage with the proposed models, we can save energy especially in prolonged use scenarios. Our results also show that the proposed system saves up to 5.68% system level power when compared to the default scheme.

CHAPTER 7

RELATED WORK

In this chapter I present related works. Even though the number of works in mobile devices and/or user experience is excessive, in this chapter, I focus on the works which are more closely related with our studies.

7.1 CPU Settings and User Experience

Although, there are many works concentrated on energy optimizations and/or CPU scaling on mobile multicore architectures, majority of these works do not consider end user. More importantly, our studies' novelty in this dissertation lies in considering the individual user and automatically customizing CPU management for each user by using system metrics or built-in sensors. Additionally, majority of the existing work perform energy optimizations on specific conditions/applications (e.g., while screen is off, while there is no user interaction). In this section, I focus on the papers that are more closely related to ours.

Shye et al. [119] perform an in-the-wild user study in order to understand smartphone usage patterns. Then they use these patterns to guide power optimizations. Specifically they propose a smooth CPU frequency-decreasing scheme in order to save energy without impacting user satisfaction. Their work does not deal with user satisfaction or individual user customization. Li et al. [89] introduce SmartCap, a statistical scheme for power adaptation for smartphone processor based on the user experience. They use system metrics in the smartphone (i.e., CPU utilization) and user reports to model user experience based on tested applications. They study 27 applications with 20 users and keep each application's satisfied configuration for each user. They show similar results on energy savings and unnecessary high frequency usage of the default scheme. Our studies differ from theirs as we do not ask the users for ratings, but rather infer the satisfaction/dissatisfaction information from the built-in sensors. Pasricha et al. [104] propose an application and user aware

power optimization scheme. Similar to the work by Li et al. [89], they monitor system metrics on the phone to understand user interactions with each application. Then with a learning model that is based on the user interaction in the current application, they implement their CPU and brightness optimizer scheme in order to save energy. Such application-dependent approaches have limitations: for each new application, considerable amount of work is necessary. Also, they do not adapt to changes in user preferences. We also discuss these limitations in more detail in Section 7. In contrast to these studies, the work presented in this dissertation does not require any application information. Matthew et al. [62] conduct a crowd sourcing study to analyze the impact of CPU changes in user satisfaction. They record gameplays while setting the smartphone CPU to different core and frequency configurations and make users rank the video based on their performance preferences. Their results show that, users react differently to changes in CPU frequency for different workloads. The variation in user performance preferences is also one of the main motivations of the projects presented in this dissertation. Shye et al. [118] develop hardware devices and monitor user physiological traits in order to understand hardware requirements in a traditional desktop. Although they conduct user studies on a traditional desktop, their test setup and findings show similarities (i.e., user expectation varies drastically). Specifically, they measure galvanic skin response and implement a force sensor on the keyboard in order to control CPU frequency. They also show significant power savings at the system level. Note that the sensors they use are specifically built for the purpose of user satisfaction prediction, whereas we use built-in sensors. In [38], authors propose a frequency controller based on trained performance and power models. They test their controller on web page loading/execution scenarios. Lo [91] predicts execution times of upcoming tasks in order to scale DVFS on smartphones. Lee et. al. [134] predict execution time of applications in the cloud in order to save memory and computation overhead on the smartphone. Lee et. al. [135] propose a battery and user aware QoS scaling system. They use users' performance expectations in different battery levels and scale CPU frequency accordingly. In our studies, we do not utilize current battery state on the phone. Although in general, these studies prove that there is significant room for improvement, they do not include tests with real users, which our study does.

Other studies [120, 131, 24, 3] do not involve real users but use QoS (Quality of Service) and QoE (Quality of Experience) metrics to define user experience and propose power optimization schemes. In these studies, user experience is measured by defining delay-thresholds to tasks and making sure that these thresholds are not surpassed. For example, Tseng et al. [131] propose a user-centric CPU scheduler by grouping applications based on their user interactions and try to minimize delays on the threads of these applications: if the user interaction lasts longer for an application, scheduler increases the priority of its threads. Although these studies prove that there is significant room for improvement, they do not include tests with real users, which our study does.

7.2 Display Brightness and User Experience

Although there are many works concentrating on display brightness in smartphones, majority of these works do not consider individual users and/or use proxy metrics. In this section, I focus on the effects of brightness setting on user experience.

Kelly et al. [74] define visual performance as the speed and accuracy of processing visual information. They show that there is a strong correlation between the screen brightness and visual performance. Studies [114, 119, 118] conduct real user tests to save energy by dimming the screen brightness while monitoring user experience. One common observation these studies made is that brightness of the screen has a direct and instantaneous effect on user satisfaction. Schuchhardt et al. [114] measure readability of the screen while gradually dimming the brightness from its default settings. They show that users try to adapt the changes until they struggle to see the screen. Their mechanism builds on the default scheme and learns user preferences. However, (a) they build on top of ambient light sensor rather than a new model, thus their models still largely depend on ambient light value, (b) they do not include the rich set of sensors we use in our mechanism, (c) they do not develop a user-independent model to compare, (d) they do not include any optimization on energy consumption, and finally (e) our study reaches a much wider audience as our applications are available on Play Store Shye et al. [119] collect user display satisfaction ratings as they grad-

ually drop the brightness. Their results show that users hardly notice a gradual changes on the brightness even after 40% reduction. However, after a threshold, they observe significant drops on user satisfactions. Both studies also show significant energy savings with small sacrifice on user satisfaction.

Sutika et al. [125] analyze the relation between brightness and smartphone usage for the elderly people. They demonstrate that users frequently change their brightness settings (manually) during their use, showing the importance of brightness settings during the phone usage. Yu et al. [149] focus on the importance of brightness levels on user experience and propose automatic color scheme adjustment to improve user experience. Guterman et al. [60] analyze user satisfaction of display panel brightness. Their primary result is that brighter display panels are not always preferable, and that overly bright displays can actually be less preferred. Studies [27, 78] examine the relation between brightness level and user experience and propose auto-brightness control mechanisms to increase user satisfaction.

The work presented in this dissertation differs from the above listed studies in various ways. All the presented works above focus on maximizing or at least keeping the same user experience on smartphone display. Whereas, we specifically aim to utilize brightness to induce discomfort on users in order to reduce excessive phone and application usage. Moreover, we conduct our experiments in-the-wild with real users, which separates our works from the majority of the above mentioned works.

7.3 User Characterization Using Mobile Sensors

There are many studies using mobile sensors for different purposes. One of the most popular research areas is activity recognition [4]. Today's mainstream smartphones with various motion sensors, including accelerometer, GPS, gyroscope, compass, etc. provide a rich data source available to be mined in order to get insights of people's daily activities (walking, jogging, standing etc.) [124]. Activity recognition is popular, because it classifies people's actions, which can be exploited in many different areas. There are many works studying activity recognition using the

embedded motion sensors on mobile devices [40, 84, 81, 6]. One of the main differences of all these prior work is either recognizing activity type (as simple or complex activities) or using different feature selection techniques and models in the predictions. Among them Dernbach et al. [40] collect accelerometer sensor data while users execute seven complex activities. While their models can predict simple activities with around 80% accuracy, with complex activities their accuracy drops to 50%. Le at al. [84] investigate combinational feature selection and reduce dimensionality on datasets in order to increase computation speed. They achieve 3 to 5 times faster performance. Kose et al. [81] compare Naive Bayes classifier and K-Nearest Neigbor (kNN) classifier in activity recognition and observe better accuracy with kNN algorithms. Although these studies focus specifically on detecting activities rather than detecting user-satisfaction related behaviors, they show similar methodologies on collecting sensor data. Emotion and stress detection using mobile devices are other widely studied areas in recent years. Egilmez et al. [44] conduct stress inducing user experiments by monitoring user behaviors with using five different mobile devices (including a galvanic skin conductor). They find distinct behavior patterns among users when they are under stress and show 88% accuracy on predicting stress. EmotionSense [108] classifies users' emotions based on speech records and location (using motion and GPS sensors). Although they can detect high arousal emotions (stress, anger) easily, they generally fail on classifying more complex low arousal emotions. StressSense [94] is another stress classifier based on the human speech patterns. The authors conduct stress inducing user studies and accurately classify stress situations [94]. However they do not study other emotions. MoodScope [90] predicts users' moods by collecting usage patterns along with users' mood reports through an application. However they do not collect any built-in sensors. Lee et al. [85] classify emotions by collecting data on various metrics, including touch events, typing speed, time, location, and weather. Their results show that they can predict seven emotional states with 67.5% accuracy. However, the relationships of each feature and emotions identified in the study have not been clearly validated as only a single user is studied.

The work presented in this dissertation differs from the above listed studies in various ways. Unlike the user characterization studies mentioned above, we use mobile sensors in order to see

hardware requirements of smartphone users. Therefore, we do not include any stress or specific emotion inducing experiments in the user studies. From this point, our work in chapter 3 is similar to the work by Shye et al. [118], where psychological traits to understand hardware requirements are monitored. However, we study hardware requirements in smartphones (rather than desktops) and use built-in sensors from two popular devices: a smartphone and a smartwatch (rather than developing dedicated hardware sensors). Moreover, the proposed system is application-independent and solely depends on user sensor data, which makes it more adaptable to changing conditions. We also discuss application-dependent implementations' limitations in Section 7. Finally, we use 30 real users to validate the proposed system, which separates the work presented in this dissertation from studies using QoS and QoE metrics for user experience. To the best of our knowledge, this is the first study that uses built-in sensors on understanding CPU requirements of users on a smartphone.

7.4 Quantifying User Experience

There are many works on quantifying user experience on smartphones. However majority of these works do not involve real users and/or use proxy metrics. I focus on the works that are more closely related to ours as below.

Huang et al. [66] conduct user studies to quantify user satisfaction in mobile cloud games. They want users play games from a PC and mobile platform and rate their experience on graphics, smoothness, and control. Then they correlate user ratings with these system parameters. Our studies differ from theirs as (i) we study energy savings' impact on user satisfaction rather than system parameters' and (ii) we conduct our studies in the wild. Matthew et al. [62] conduct a crowd sourcing study to analyze the impact of CPU changes in user satisfaction. They record game plays while setting the smartphone CPU to different core and frequency configurations and want users rank the video based on their performance preferences. Then they quantify user satisfaction with the tested CPU configurations. Their results show that users react differently to changes in CPU frequency for different workloads. The crowd source methodology and the variation in individual user prefer-

ences is also one of the motivations of the projects presented in this dissertation. However, similar to other works, the distinction of our works is in understanding the impact of energy savings. Chen et al. [30] quantify VoIP user satisfaction by analyzing the call duration from actual Skype traces. Instead of using real users, they form a user satisfaction index composed by objective source- and network-level metrics, such as the bit rate, bit rate jitters, and round-trip time. Similarly, Chen et. al. [32] quantify quality of service (QoS) and quality of experience (QoE) metrics on smartphone video streaming applications. Most of these works quantify user experience/satisfaction with system metrics rather than energy savings. Amdone et al. [9] quantify user experience on wearable fitness technologies in order to increase user satisfaction. They quantify physical comfortable of users, rather than energy savings' effect.

Studies [144, 36, 130] design battery interfaces and investigate theirs effects on users' behaviours. Among them Jung et.al. [144] propose a battery-user interaction tool to inform users about their battery level and battery consumption. Then they compare users' phone usage behaviors before and after learning battery information. Their results show that, by learning the battery information, users are more likely to change their phone usage behaviors towards energy savings. In contract to [144], in our studies, we neither provide battery-user interaction tool nor monitor users' behaviours, but instead we quantify user satisfaction based on the energy savings. Truong et al. [130] propose the task-centered battery interface (TCBI), which is a battery interface that accurately calculates the remaining battery time on a device. The TCBI offers the estimated remaining battery life on a device for a few predefined cases while taking into account currently activated applications. The interactive battery interface (IBI) proposed by Ferreira et al. [36] also shows the remaining battery life and provides functionality to terminate activated applications.

7.5 Excessive Smartphone Usage's Effects on Individuals

There are many studies focusing on the relation between excessive smartphone usage and its harmful effects on individuals. Billieux et al.[23] show strong similarities between excessive phone usage and behavioral addiction. Similary, Tossell et al. [129] investigate smartphone user behaviors

and their relation to self-reported smartphone addiction. Specifically, they give 34 non-smartphone users a smartphone and monitor their usage behaviours over a year. Their result show that at the end of one year period, 62% of the users show addictive behaviors. Xie et al. [145] show that parents' social exclusion with mobile phone interruption is a risk factor for adolescent mobile phone addiction.

There are many studies focusing on the relation between the academic performance of college students and smartphone use due to its possible distracting or disruptive side effects to academic focus. Felisoni et al. [48] conduct an experiment to test the relationship between average time students spend using their smartphones per day and their academic performance. Their analysis yield a significant negative relationship between total time spent using smartphones and academic performance: each 100 min spent using the device on average per day corresponded to a reduction in a student's position at the school's ranking of 6.3 points, on a range from 0 to 100. Winskel et al. [143] examine smartphone usages of 389 students and they also show that the more time a student spends using their smartphone, the more at risk they are for problematic smartphone use and possible academic performance costs. Similarly, Uzun et al. [132] shows that after controlling particular demographics (i.e., age, gender, year of study, etc.), overuse of media and technology tools significantly degrades the students' academic performance.

There is a plethora of studies focusing on phone usage during the classroom. Kim et al. [77] conduct a 14-week measurement study in the wild with 84 first-year college students. Their results show that, students use their phones for more than 20 min per class. Also, they show that excessive daily and in-class use habits have a negative relationship with student grades. Waite et al. [140] examine impacts of concurrent off-task text messaging during an academic presentation on the students' performance on tests. They show that such media multitasking negatively impacts the quality of note-taking during the presentation and reduce learning.

Studies [70, 71, 79, 69, 86] focus on the impact of both smartphone and media application usage on students. Jankovic et al. [70] conduct a study with 485 students and show that in cases of a lack of time, students are more likely to sacrifice academic work, rather than time for Facebook

or smartphones in general. Similarly, studies [71, 79] show that Facebook users declared having lower levels of GPA and that they devoted fewer hours to studying per week than non-Facebook users. Huang [65] shows a negative mean correlation between social networking site usage and academic performance. Studies [69, 86] also show negative correlation between calling/texting and GPA scores.

There are also many works that show negative impacts of excessive phone usage on mental health. Ali et.al [5] show that daily excessive phone usage was significantly associated with disturbed sleep pattern and sleep quality. Valasareddy et al. [133] conduct a survey to analyze the impact of bedtime smartphone usage on sleep health. Their results show that, 50% of the bedtime smartphone users perceive sleep deprivation and 83.3% of the users perceive adverse effects such as restlessness in the morning after high smartphone usage the previous night. Studies [128, 148] show associations between long and frequent mobile phone use and mental health outcomes, such as depression and anxiety symptoms.

7.6 Reducing Smartphone Usage on Individuals

There exists substantial amount of works focused on reducing the excessive smartphone usage. Studies [92, 80, 103] introduce filtering and blocking applications in order to reduce interactions. They show significant improvements on reducing the number of user interactions with smartphones in daily use. Similarly, some smartphone manufacturers introduce methods to limit phone or application usage [67, 15]. One common characteristics of these methods is to force a hard time threshold to restrict access for a specific application or phone as a whole. Although their efficacy is unknown on reducing the phone usage in daily life, there are other studies focusing on the effects of restricting access to smartphones. Cheever et al. [2] show that on heavy and moderate smartphone users, restriction make participants significantly more anxious over time. Studies [45, 95] show that smartphone restriction could cause withdrawal symptoms on the users. In our proposed architecture, we do not implement any restriction to the phone or application, instead gradually create user discomfort on the target smartphone in order to reduce overuse. Moreover, we aim to

mitigate prolonged usages rather than reducing the number of interactions.

There exists a lot of works introducing interventions and social/peer supports to mitigate phone overuse [80, 76, 88, 117]. Kim et al. [76] propose a software-based intervention in order to encourage participation and raise awareness regarding appropriate mobile phone usage to establish social norms in college classrooms. Li et al. [88] show that there is an increasing number of adolescents suffering from depression due to mobile phone addiction and social support and positive emotions can lower levels of depression among adolescents. Shen et al. [117] introduce intervention strategies for preventing smartphone overuse. Similarly, the distinction of our studies is that, we do not create any social or peer support nor apply interventions in order to reduce overuse.

CHAPTER 8

CONCLUSION

8.1 Concluding Remarks

In this dissertation, I integrate individual users' satisfactions with the design of mobile systems and their optimizations. The motivation behind the work presented in this dissertation is grounded in addressing the suboptimality of the current mobile systems. Particularly, the current mobile systems typically generalize their respective decisions as if they are developed solely toward the average user while allowing too little room for the implementation of individualized personal settings. Inefficiency of this approach comes from the assumption that all users are expected to have maximum satisfaction with a one-size-fits-all design. In other words, the assumption goes as if all the users have the exact same or, more closely, similar set of demands and preferences regarding their mobile devices. However, following and building on some of the prior work, in this dissertation I showed that users' preferences and demands tend to vary significantly even under the same workloads and/or conditions. Thus, there is a need for systems which utilize the differences between user's preferences and needs for developing better sysstem designs. Hence, if we can (somehow) know end user's instantaneous satisfaction about the system, we can then utilize this information to have a number of benefits: we can create dynamic user-aware decision-making mobile systems or try to avoid bad habits associated with smartphones.

However, the challenge here arises from measuring and integrating the instantaneous user satisfaction in real time. User satisfaction is inherently a subjective metric, which is affected often times by the limits of human perception and the preconceptions of an individual. As I discussed in chapters 2 and 3, I correlate user satisfaction with system metrics and built-in sensors on a smartphone. To put in brief, I show that user satisfaction can be correlated with system metrics and built-in sensors in a smartphone with an accuracy of over 90%. This is important, because know-

ing user's instantaneous satisfaction about their phones can be utilized in many different areas of research. In this dissertation I utilize instantaneous user satisfaction, as a means for two distinct purposes.

In the first approach, I first utilize user satisfaction as a feedback input in order to better manage hardware resources. The aim is to maximize user satisfaction while minimizing energy consumption. Thus, I focus on bridging the gap between hardware resource management and end user experience by conducting various user studies. As discussed in Chapters 2-4, I conducted user studies both in-the-lab and in-the-wild with the aim of optimizing the two most power hungry components: mobile CPU and screen. As I show in the results, we can achieve up to 15% energy savings on average with no impact on user satisfaction.

In the second approach, I show an alternative usage of user satisfaction where I use it to avoid harmful habits associated with smartphones. Notably, in this approach, I degrade user experience with the aim of mitigating overuse of smartphones. As our user studies have showed, we can decrease the maximum phone usage up to 37% with a negligible impact on user experience.

8.2 Summary of Contributions

Summary of main contributions in this dissertation are as listed below. I should note that, since indicated in Chapter 1 (Section 1.3 more specifically) the work in this presentation comes from a set of collaborative work with several co-authors; thus, I have listed the contributions by using "we" in order to refer to myself as well as my co-authors.

- User satisfactions with the CPU performance and display brightness of their smartphones vary considerably between applications being used as well as users.
- Smartphone CPU configurations with more cores can lead to higher energy consumption without increasing user satisfaction for some applications, but more cores are necessary to achieve maximum satisfaction for other applications.

- Using only ambient light sensor is not sufficient for capturing the variation in user brightness preferences and data from other sensors are highly correlated with these preferences.
- We propose systems that utilize either user-facing metrics or built-in sensors on smartphones to predict user satisfaction to manage hardware resources (CPU and screen) in real-time in order to save energy and maximize user experience.
- We show light-weight tools and methods to develop personalized models that learn from personal performance (CPU) and brightness preferences in the wild.
- We quantify and present analysis on the relation between energy savings and user satisfaction.
- We show alternative ways to use user satisfaction. Particularly, we develop a system that utilizes brightness settings on a smartphone for the purpose of degrading user experience as a means of limiting excessive smartphone or application usage.

8.3 Possible Avenues for Further Research

While the work in this dissertation has shown considerable benefits of integrating user satisfaction in mobile devices, there still remains many avenues of research yet to be explored. I outline some potential directions below.

Possible challenges of building fully-adaptable mobile devices. During the course of my studies, I have worked on building systems that personalize specific hardware components to increase user satisfaction while minimizing energy consumption at the same time. As I discussed in chapters 2-4 at length, in the proposed systems, we have correlated user satisfaction inputs with system metrics or built-in sensors on smartphones. Although we study a specific hardware component at a given time, the proposed systems can be extended to optimize more components simultaneously. For example, a system can decrease CPU cores and increase brightness if this leads to more energy savings and increased user satisfaction. However, one challenge of such system

could be the increase in the number of scenarios. Notably, as the number of hardware components increases, it is expected that the adaptation of the system would take more time, since the system would need to encounter more scenarios in order to make the optimal decisions. We have visioned such systems for our screen study that is discussed at length in chapter 4 where we have considered altering both screen brightness and CPU configurations for (more) energy savings.

Another challenge of personalizing (and optimizing) multiple components at the same time arises from the need of more user inputs. In our studies we generally collected user feedback through the logger applications we have built. However, we observed that, users tend to lose their attention quite fast as the test duration increases. Thus, we mostly kept our studies in the duration of just a week. However, if we are to build a system that optimizes multiple components, we may need to consider more easy input mechanisms for users. We have considered such mechanisms by testing *nfc* buttons in the early years of my graduate studies.

More alternative ways of utilizing user satisfaction in mobile devices. In the work presented in this dissertation, I utilize user satisfaction mainly for two different purposes: managing hardware resources (chapters 2-4) and avoiding bad habits with smartphones (chapter 6). However, there are many more possible fields of research that can utilize user satisfaction in mobile devices. One example can be drawn from the hypothesis that based on the workload and web sites to be visited, some users may be willing to sacrifice their WiFi signal power if it leads to more power savings. For these users, a dynamically adaptive model that controls WiFi signal power could be beneficial in terms of energy savings (and possibly for better network traffic). Similar to the idea discussed in chapter 6, user satisfaction can also be utilized to avoid/limit smartphone usage under certain conditions (i.e. driving, gambling etc.).

Defining different thresholds. As I discussed in chapters 2 & 3, during our user studies, we have collected user satisfaction reports in 3-level (as numeric 0-very unhappy, 1-unhappy and 2-happy). We have correlated these reports with phone-based metrics in order to build satisfaction-prediction models. Then based on the outputs of prediction models, we determined some thresholds for the subsequent decision making. For example, when the prediction is below 1.25, we

assume the user is unhappy and alter the core count accordingly. Similarly when it is above 1.75, we assume the user is happy and decrease the core count to save energy. It is important to note that although we have determined these thresholds, it is still possible that with some other threshold, system can achieve more power savings and/or can achieve better user experience. Moreover, these thresholds can be user-specific in order to reflect user's satisfaction in a finer grain fashion.

Building the systems at the kernel level. All the work I have presented in this dissertation is built at the system-level. We do this in order to conduct our studies with a wider audience in the wild (with users' own smartphones). In the case that the models and optimizations were to be implemented at the kernel-level, it is possible to observe more energy savings and less overhead to the overall phone.

REFERENCES

[1] Huawei Nexus 6P. *Nexus 6P features*. 2020. URL: https://en.wikipedia.org/wiki/Nexus_6P.

[2] Cheever N. A., Rosen L. D., Carrier L. M., and Chavez A. "Out of sight is not out of mind: the impact of restricting wireless mobile device use on anxiety levels among low, moderate and high users." In: *Computer and Human Behaviour* 105 (2014).

[3] Das A., Walker M. J., Hansson A., Al-Hashimi B. M., and Merrett G. V. "Hardware-Software Interaction for Run-time Power Optimization: A Case Study of Embedded Linux on Multicore Smartphones." In: *ISLPED* (July 2015).

[4] *Activity recognition definition*. 2018. URL: https://en.wikipedia.org/wiki/Activity_recognition.

[5] Akhtar Ali, Sehrish Mehmood, Lubna Farooq, Humaira Arif, Nadeem Akhtar Korai, and Muhammad Aitmaud Uddolah Khan. "Influence of Excessive Mobile Phone Use on Anxiety and Academic Performance among Medical College Students". In: *Journal of Pharmaceutical Research International* (Dec. 2019).

[6] Anjum Alvina and Muhammad U. Ilyas. "Activity Recognition Using Smartphone Sensors". In: *Consumer Communications and Networking Conference* (Jan. 2013).

[7] Analog. *MEMS microphone sensitivity*. 2012. URL: http://www.analog.com/en/analog-dialogue/articles/understanding-microphone-sensitivity.html.

[8] AnandTech. *Increasing core count in smartphone market*. 2020. URL: http://www.anandtech.com/show/9518/the-mobile-cpu-corecount-debate.

[9] Ionut Andone, Konrad Blaszkiewicz, Mark Eibes, Boris Trendafilov, Christian Montag, and Alexander Markowetz markowetz. "Menthal: quantifying smartphone usage". In: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Sept. 2016.

[10] Google Android. *Android logcat*. URL: https://developer.android.com/studio/command-line/logcat.html.

[11] Google Android. *Face API*. URL: https://developers.google.com/vision/face-detection-concepts.

[12] Google Android. *Getevents*. URL: https://source.android.com/devices/input/getevent.

[13] Google Android. *Android Debug Bridge*. 2020. URL: https://developer.android.com/studio/command-line/adb.html.

[14] Google Android. *Android Location API*. 2020. URL: https://developer.android.com/training/location/background.

[15] Google Android. *Manage your child's screen time*. 2020. URL: https://support.google.com/families/answer/7103340?hl=en.

[16] *Android Wear application*. 2019. URL: https://play.google.com/store/apps/details?id=com.google.android.wearable.app.

[17] Androidauthority. *Survey on battery importance*. 2019. URL: https://www.androidauthority.com/smartphone-survey-mediatek-916566/.

[18] AngryBird. *Angry Bird Game Application*. 2019. URL: https://www.angrybirds.com/games/.

[19] *ANOVA*. 2019. URL: https://en.wikipedia.org/wiki/Analysis_of_variance.

[20] Appquest. *Car race game*. 2015. URL: http://appquest.io/blog/popular-android-racing-games-cpu-stickman/.

[21] *Audio Record in Android*. URL: https://developer.android.com/reference/android/media/AudioRecord.html.

[22] Thomas Bashford-Rogers, Miguel Melo, Demetris Marnerides, Maximino Bessa, Kurt Debattista, and Alan Chalmers. "Learning Preferential Perceptual Exposure for HDR Displays". In: *IEEE Access* (Apr. 2019).

[23] Joël Billieux, Pierre Maurage, Olatz Lopez-Fernandez, Daria J. Kuss, and Mark D. Griffiths. "Can Disordered Mobile Phone Use Be Considered a Behavioral Addiction? An Update on Current Evidence and a Comprehensive Model for Future Research". In: *Springer - Technology and Addiction* (Apr. 2015).

[24] A. S. Bischof. "User-Experience-Aware System Optimization for Mobile Systems." In: *HotMobile* (Jan. 2016).

[25] Gao C., Gutierrez A., Dreslinski R, Mudge T., Flautner K., and Blake G. "A study of thread level parallelism on mobile devices." In: *ISPASS* (Mar. 2014).

[26] Perifer C., Schulz A., Schachinger H., Baumann N., and Antony C. "The relation of flow-experience and Phycological Arousal Under Stress- Can you Shape it?" In: *Journal of Experimental Social Psychology* (2014).

[27] B. Cambodge, C. Remi, M. Gerard, and D. Xavier. "QoE-Based Brightness Control For HDR Displays." In: *QoMEX* (2017).

[28] *Car race game*. 2015. URL: http://www.m2mobileinsights.com/blog/9-popular-android-racing-games-with-the-highest-cpu-usage-rates/.

[29] Pew Research Center. "Mobile Fact Sheet". In: *Pew Research Center Internet & Technology* (June 2019). URL: https://www.pewresearch.org/internet/fact-sheet/mobile/.

[30] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei. "Quantifying Skype User Satisfaction". In: ACM SIGCOMM, Sept. 2006, pp. 399–410.

[31] X. Chen, Y. Chen, Z. Ma, and F. C. Fernandes. "How is energy consumed in smartphone display applications?" In: *HotMobile* (2013).

[32] Yu-Chieh Chen, Jen-Wei Chang, and Hung-Yu Wei. “A multi-level QoE framework for smartphone video streaming applications”. In: IEEE Globecom Workshops (GC Wkshps), Dec. 2014.

[33] Hongsuk Chung, Munsik Kang, and Hyun-Duk Cho. *Heterogenous multicore processing solutions*. URL: https://www.arm.com/ja/files/pdf/Heterogeneous_Multi_Processing_Solution_of_Exynos_5_Octa_with_ARM_bigLITTLE_Technology.pdf.

[34] Hongsuk Chung, Munsik Kang, and Hyun-Duk Cho. “Heterogenous multicore processing solutions.” In: *ARM Technologies* (2020). URL: https://www.arm.com/why-arm/technologies/big-little.

[35] Federal Communications Commission. *Effects Of Using Mobile Phones Too Much*. 2019. URL: https://ecfsapi.fcc.gov/file/7520941199.pdf.

[36] Ferreira D., Ferreria E., Goncalves J, Kostakos V., and Dey A. K. “Revisiting human-battery interaction with an interactive battery interface”. In: *UbiComp*. ACM, Sept. 2013.

[37] Fisher C. D. “Boredom at Work: A Neglected Concept”. In: *Human Relations* (3 1993), pp. 395–417.

[38] Shingari D., Arunkumar A., Gaudette B, Vrudhula S, and Wu Carole-Jean. “DORA: Optimizing Smartphone Energy Efficiency and Web Browser Performance under Interference.” In: *ISPASS* (Jan. 2018).

[39] Marty-Dugas J Danckert J. Hammerschimdt T. and Smilek D. “Boredom: Under-aroused and restless”. In: *Consciousness and Cognition* (61 2018), pp. 24–37.

[40] Stefan Dernbach, Barnan Das, Narayanan C. Krishnan, Brian L. Thomas, and Diane J. Cook. “Simple and Complex Activity Recognition Through Smart Phones”. In: *Intelligent Environments (IE)* (June 2012).

[41] Edn. *Basic Principles of MEMS microphones*. 2014. URL: https://www.edn.com/design/analog/4430264/Basic-principles-of-MEMS-microphones-.

[42] EDN. “Basic Principles of MEMS microphones”. In: May 2014. URL: https://www.edn.com/basic-principles-of-mems-microphones/.

[43] Eenewsanalog. *Microphone Design*. 2016. URL: http://www.eenewsanalog.com/news/mems-microphone-design-better-audio.

[44] Begum Egilmez, Emirhan Poyraz, Wenting Zhou, Gokhan Memik, Peter Dinda, and Nabil Alshurafa. “UStress: Understanding College Student Subjective Stress Using Wrist-Based Passive Sensing”. In: *Pervasive Computing and Communications Workshops (PerCom Workshops)* (Mar. 2017).

[45] Tine A. Eide, Sarah H. Aarestad, Cecilie S. Andreassen, Robert M. Bilder, and Ståle Pallesen. “Smartphone Restriction and Its Effect on Subjective Withdrawal Related Scores”. In: *Frontiers in Pyschology* (Aug. 2018).

[46] ExoPlayer. *ExoPlayer Source Code*. 2019. URL: https://github.com/google/ExoPlayer.

[47] Facebook-Instagram. *Instagram Application*. 2019. URL: https://www.instagram.com/.

[48] Daniel Darghan Felisoni and Alexandra Strommer Godoi. "Cell phone usage and academic performance: An experiment". In: Computers Education. Elsevier, 2018, pp. 175–187.

[49] FirstPost. *Survey on battery importance*. 2019. URL: https://www.firstpost.com/tech/news-analysis/longer-battery-life-is-mostimportant-feature-for-smartphone-users-finds-survey-3651843.html.

[50] Friedman. *Friedman test*. 2019. URL: https://en.wikipedia.org/wiki/Friedman_test.

[51] Gemalto. *Survey on battery importance*. 2019. URL: https://www.gemalto.com/brochures-site/download-site/Documents/tel-infosurvey-smartphone-usage-issues.pdf.

[52] Google. *Android Wear 2.0*. 2019. URL: https://www.android.com/wear/.

[53] Google. *YouTube Application*. 2019. URL: https://www.youtube.com/.

[54] Google-Android. *Ambient light sensor*. 2020. URL: https://developer.android.com/reference/android/hardware/Sensor.htmlTYPE-LIGHT.

[55] Google-Android-Activity. *Activity Recognition API*. 2020. URL: AndroidActivityRecognition. https://developers.google.com/location-context/activity-recognition/.

[56] Google-Android-Audio. *Audio record API*. 2020. URL: https://developer.android.com/reference/android/media/AudioRecord.html.

[57] Google-Android-Batt. *Android Battery Manager*. 2020. URL: https://developer.android.com/reference/android/os/BatteryManager.

[58] Google-Android-Pressure. *Ambient air Pressure*. 2020. URL: https://developer.android.com/reference/android/hardware/Sensor-TYPE-PRESS.

[59] Raffaele Gravina and Qimeng Li. "Emotion-relevant activity recognition based on smart cushion using multi-sensor fusion". In: *Information Fusion* (Aug. 2019).

[60] P. S. Guterman, K. Fukuda, L. M. Wilcox, and R. S. Allison. "Is brighter always better? the effects of display and ambient luminance on preferences for digital signage". In: *SID* (2010).

[61] Xianping Tao Haibo Ye Tao Gu and Jian Lu. "Scalable floor localization using barometer on smartphone". In: *Wireless Communications & Mobile Computing* (July 2016).

[62] Matthew Halnern, Yuhao Zhu, and Vijay Reddi. "Mobile CPU's rise to power: Quantifying the impact of generational mobile CPU design trends on performance, energy, and user satisfaction". In: *HPCA* (Mar. 2016).

[63] Reddi V. J. Halpern M. Zhu Y. "Mobile CPU's Rise to Power: Quantifying the Impact of Generational Mobile CPU Design Trends on Performance, Energy, and User Satisfaction." In: *HPCA* (Mar. 2016).

[64] *Heart rate chart*. 2015. URL: https://www.heart.org/en/healthy-living/fitness/fitness-basics/target-heart-rates.

[65] Chiungjung Huang. “Social network site use and academic achievement: A meta-analysis”. In: *Computers & Education* (Nov. 2017).

[66] Chun-Ying Huang, Cheng-Hsin Hsu, De-Yu Chen, and Kuan-Ta Chen. “Quantifying User Satisfaction in Mobile Cloud Games”. In: ACM MoVid’14, Sept. 2014.

[67] Apple Iphone. *Using Screen Time on Your Iphone*. 2020. URL: https://support.apple.com/en-us/HT208982.

[68] Mikulas W. L. & Vodanovich S. J. “The Essence of Boredom”. In: *The Psychological Record* (3 1993), p. 43.

[69] W.C. Jacobsen and R. Forste. “The wired generation: Academic and social outcomes of electronic media use among university students”. In: *Cyberpsychology, Behavior, and Social Networking* (2011).

[70] Branka Jankovic, Milan Nikolic, Jelena Vukonjanski, and Edit Terek. “The impact of Facebook and smart phone usage on the leisure activities and college adjustment of students in Serbia”. In: Computers in Human Behavior. Elsevier, 2016, pp. 354–363.

[71] R. Junco. “The relationship between frequency of Facebook use, participation in Facebook activities, and student engagement”. In: *Computers and Education* (2012).

[72] Evangelos Karapanos. *Modeling Users’ Experiences with Interactive Systems*. Springer, 2013.

[73] Penczek J. Kelley E. Lindfors M. “Display daylight ambient contrast measurement methods and daylight readability.” In: *Journal of the Society for Information Display,* (Mar. 2006).

[74] E. F. Kelly, M. Lindfors, and J. Penczek. “Display daylight ambient contrast measurement methods and daylight readability”. In: 2006.

[75] Amera Khanam. *Body signs on boredom*. URL: https://body-language.knoji.com/how-to-detect-the-boredom-signs-among-audiance-using-body-langauge/.

[76] Inyeop Kim, Gyuwon Jung, Hayoung Jung, Minsam Ko, and Uichin Lee. “Let’s FOCUS: Mitigating Mobile Phone Use in College Classrooms”. In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* (Sept. 2017).

[77] Inyeop Kim, Rihun Kim, Heepyung Kim, Duyeon Kim, Kyungsik Han, Paul H.Lee, Gloria Mark, and Uichin Lee. “Understanding smartphone usage in college classrooms: A long-term measurement study”. In: vol. 141. Computers & Education. Elsevier, 2019, pp. 354–363.

[78] Seung-Ryeol Kim and Seung-Woo Lee. “Auto-brightness control technology depending on user’s pupil area”. In: *IEICE Electronics Express* (Mar. 2018).

[79] P.A. Kirschner and A.C. Karpinski. “Facebook and academic performance”. In: *Computers in Human Behavior* (2010).

[80] Minsam Ko, Seungwoo Choi, Koji Yatani, and Uichin Lee. “Group-based Limiting Assistance App to Mitigate Smartphone Distractions in Group Activities”. In: *In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (Sept. 2016).

[81] Mustafa Kose, Ozlem Durmaz Incel, and Cem Ersoy. “Online Human Activity Recognition on Smart Phones”. In: *2nd International Workshop on Mobile Sensing* (Apr. 2012).

[82] A. Krause, A. Smailagic, and D. P. Siewiorek. “Context-aware mobile computing: Learning context-dependent personal preferences from a wearable sensor array”. In: *Mobile Computing, IEEE Transactions* (2006).

[83] Daria J. Kuss, Eiman Kanjo, Mark Crook-Rumsey, Fraenze Kibowski, Grace Y. Wang, and Alex Sumich. “Problematic Mobile Phone Use and Addiction Across Generations: the Roles of Psychopathological Symptoms and Smartphone Use”. In: Journal of Technology in Behavioral Science. Springer, 2018, pp. 141–149.

[84] Tuan Dinh Le and Chung Van Nguyen. “Human activity recognition by smartphone”. In: *Information and Computer Science (NICS), 2015 2nd National Foundation for Science and Technology Development Conference* (Sept. 2015).

[85] Hosub Lee, Sang Choi, Sunjae Lee, and I.P.Park. “Towards unobtrusive emotion recognition for affective social communication”. In: *CCNC* (Jan. 2012).

[86] A. Lepp, J.E. Barkley, and A.C. Karpinski. “The relationship between cell phone use, academic performance, anxiety, and satisfaction with life in college students”. In: *Computers in Human Behavior* (2014).

[87] Haobo Li, Xiangpeng Liang, Aman Shrestha, Yuchi Liu, Hadi Heidari, Julien Le Kernec, and Francesco Fioranelli. “Hierarchical Sensor Fusion for Micro-Gestures Recognition with Pressure Sensor Array and Radar”. In: *IEEE Journal of Electromagnetics, RF and Microwaves in Medicine and Biology* (Oct. 2019).

[88] Menglong Li, Xia Jiang, and Yujia Ren. “Mediator Effects of Positive Emonitions on Social Support and Depression Among Adoloscents Suffering From Mobile Phone Addiction”. In: *Psychiatria Danubina* 29 (2017).

[89] Xueliang Li, Guihani Yan, Yinhe Han, and Ziaowei Li. “SmartCap: User Experience-Oriented Power Adaptation for Smartphone’s Application Processor.” In: *DATE* (Sept. 2013).

[90] Robert LiKamWa, Yunxin Liu, Nicholas D. Lane, and Lin Zhong. “MoodScope: Building a Mood Sensor from Smartphone Usage Patterns”. In: *MobiSys* (June 2013).

[91] Suh Edward. Lo D. Song Taejoon. “Prediction-Guided Performance-Energy Trade-off for Interactive Applications.” In: *MICRO* (June 2015).

[92] Markus Lochtefeld, Matthias Bohmer, and Lyubomir Ganev. “AppDetox: Helping Users with Mobile App Addiction”. In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* (Sept. 2013).

[93] Nisbett R.E. London H. Monello L. Cognitive manipulation of boredom. In: London H. "Thought and Feeling: Cognitive Alteration of Feeling States". In: *Aldine; Oxford* (1974), 239–239.

[94] Hong Lu, Denise Frauendorfer, Mashfiqui Rabbi, Marianne Schmid Mast, Gokul T., Andrew Campbell, Daniel Perez, and Tanzeem Choudhury. "StressSense: detecting stress in unconstrained acoustic environments using smartphones". In: *UbiComp* (Sept. 2012).

[95] Cutino C. M. and Nees M. A. "Restricting mobile phone access during homework increases attainment of study goals". In: *Mobile Media Communication* (2017).

[96] Christine F. Martindale, Sebastijan Sprager, and Bjoern M. Eskofier. "Hidden Markov Model-Based Smart Annotation for Benchmark Cyclic Activity Recognition Database Using Wearables". In: *Sensors* (Mar. 2019).

[97] Tim Messick. *The Green Wall Source Code.* 2015. URL: https://github.com/awlzac/greenwall.

[98] Google Android MonkeyRunner. *Monkeyrunner in Android*. 2020. URL: https://developer.android.com/studio/test/monkeyrunner/index.html.

[99] *Most used applications*. URL: https://en.wikipedia.org/wiki/List_of_most_popular_smartphone_apps.

[100] Ofcom. *The UK is now a smartphone society*. 2016. URL: https://www.ofcom.org.uk/about-ofcom/latest/media/media-releases/2015/cmr-uk-2015.

[101] World Health Organization. "Public Health Implications of Excessive Use of the Internet, Computers, Smartphones and Similar Electronic Devices Meeting report". In: *World Health Organization* (Aug. 2014).

[102] *Paired t-test*. URL: https://www.statsdirect.com/help/parametric_methods/paired_t.htm.

[103] Chunjong Park, Junsung Lim, Juho Kim, Sung-Ju Lee, and Dongman Lee. "Don't Bother me. I'm Socializing!: Breakpoint-Based Smartphone Notification System". In: *In Proceedings of the 20th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '17)* (2017).

[104] Sudeep Pasricha, Brad Donohoo, and Chris Ohlsen. "A middleware framework for application-aware and user-specific energy optimization in smart mobile devices." In: *Pervasive and Mobile Computing* (July 2015).

[105] Pijush Pramanik, Nilanjan Sinhababu, Bulbul Mukherjee, P. Sanjeevikumar, Aranyak Maity, Bijoy Upadhyaya, Jens Bo Holm-Nielsen, and Prasenjit Choudhury. "Power Consumption Analysis, Measurement, Management, and Issues: A State-of-the-Art Review of Smartphone Battery and Energy Usage". In: *IEEE Access* (Apr. 2019).

[106] Qualcomm. "Development of sensor technology on the mobile phones over years." In: (2014). https://www.qualcomm.com/news/onq/2014/04/24/behind-sixth-sense-smartphones-snapdragon-processor-sensor-engine.

[107] QuickLogic. “Auto Brightness: How to Improve the Android Mobile Operating System in Handheld Consumer Devices.” In: (2011). URL: https://www.quicklogic.com.

[108] Kiran 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: *UbiComp* (Sept. 2010).

[109] M. Rae and M. Ouelette. “Relative visual performance. A basis for application”. In: *Lighting Research and Technology* (1991).

[110] eMarketer Report. *US Time Spent with mobile 2019*. 2019. URL: https://www.emarketer.com/content/us-time-spent-with-mobile-2019.

[111] Anthony Restaino. *Lightning Web Browser*. 2015. URL: github.com/anthonycr/Lightning-Browser.

[112] Daniel Sandler. *Markers Source Code*. 2015. URL: https://github.com/dsandler/markers/wiki.

[113] M. Schuchhardt, S. Jha, R. Ayoub, M. Kishinevsky, and G. Memik. “CAPED: Context-aware Personalized Display Brightness for Mobile Devices.” In: *CASES* (2014).

[114] M. Schuchhardt, S. Jha, R. Ayoub, M. Kishinevsky, and G. Memik. “Optimizing mobile display brightness by leveraging human visual perception.” In: *CASES* (2015).

[115] Amazon Web Services. *Amazon Mechanical Turk*. 2020. URL: https://aws.amazon.com/documentation/mturk/.

[116] Findlay Shearer. “Power management in mobile devices, chapter Hierarchical View of Energy Conservation”. In: *Newnes* (2008).

[117] Libi Shen and Anchi Su. “Multifaceted Approach to Digital Addiction and Its Treatment - Intervention of Smartphone Addiction”. In: *Book Chapter* (2019).

[118] A. Shye, Y. Pan, B. Scholbrock, J. S. Miller, G. Memik, P. A. Dinda, and R. P. Dick. “Power to the people: Leveraging human physiological traits to control microprocessor frequency.” In: *MICRO* (Dec. 2008).

[119] A. Shye, B. Scholbrock, and G. Memik. “Into the wild. Studying real user activity patterns to guide power optimizations for mobile architectures.” In: *MICRO* (Dec. 2009).

[120] Wook Song, Nosub Sung, Byung-Gon, and Jihong Kim. “Reducing Energy Consumption of smartphones using user-perceived response time analysis.” In: *HotMobile* (Feb. 2014).

[121] Statista. *Smartwatch Sales Worldwide*. 2014. URL: https://www.statista.com/statistics/422097/smartwatch-sales-worldwide-companies-market-share/.

[122] Statista. *Number of Smartphone Users Worldwide*. 2020. URL: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

[123] Stanley Smith Stevens. “Steven’s Power Law”. In: (). URL: https://en.wikipedia.org/wiki/Stevens27s-power-law.

[124] Xing Su, Hanghan Tong, and Ping Ji. “Activity Recognition with Smartphone Sensors”. In: *Tsinghua Science and Technology* 19 (3 June 2014).

[125] Thawanrut Sutika, Suree Funilkul, Tuul Triyason, and Montri Supattatham. “Quality of Smartphone User Experience Analysis: Focusing on Smartphone Screen Brightness Level for the Elderly”. In: *ACM IAIT* (Dec. 2018). URL: https://dl.acm.org/doi/pdf/10.1145/3291280.3291784.

[126] Mitra T. “Heterogenous Multi-core Architecture.” In: *IPSJ Transactions on System LSI Design Methodology Vol.8, 51-62* (Aug. 2015).

[127] Sergey Tarasevich. *Universal Image Loader Source Code.* 2015. URL: https://github.com/nostra13/Android-Universal-Image-Loader.

[128] Sara Thomée. “Mobile Phone Use and Mental Health. A Review of the Research That Takes a Psychological Perspective on Exposure”. In: *Int. J. Environ. Res. Public Health* (2018).

[129] Chad Tossell, Philip Kortum, Clayton Shepard, Ahmad Rahmati, and Lin Zhong. “Exploring Smartphone Addiction: Insights from Long-Term Telemetric Behavioral Measures”. In: *International Journal of Interactive Mobile Technologies (iJIM)* (2015).

[130] K. N. Truong, J. A. Kientz, T. Sohn, A. Rosenzweig, A. Fonville, and Smith T. “The design and evaluation of a task-centered battery interface”. In: *UbiComp*. ACM, Sept. 2010.

[131] Po-Hesien Tseng, Pi-Cheng Hsiu, Chin-Chiang Pan, and Tei-Wei Kuo. “SmartCap: User Experience-Oriented Power Adaptation for Smartphone’s Application Processor.” In: *DAC* (June 2014).

[132] Ahmet Murat Uzun and Selcan Kilis. “Does persistent involvement in media and technology lead to lower academic performance? Evaluating media and technology use in relation to multitasking, self-regulation and academic performance”. In: *Computers in Human Behavior* (Jan. 2019).

[133] Manvitha Valasareddy, Wenli Wang, Chaza F. Abdul-Al, and Steven P. Niles. “The Impact of Bedtime Smartphone Usage on Sleep Health: A Pilot Quantitative Study”. In: *Issues in Information Systems* 20 (Aug. 2019). URL: https://iacis.org/iis/2019/4_iis_2019_75-85.pdf.

[134] Lee W., Sunwoo D., Gerstlauer A., and John L. “Cloud-guided QoS and Energy Management for Mobile Interactive Web Applications.” In: *MOBILESoft* (June 2017).

[135] Lee W., Panda R., Sunwoo D., Joao J., Gerstlauer A., and John L. “BUQS: battery- and user-aware QoS scaling for interactive mobile devices.” In: *ASPDAC* (June 2018).

[136] University of Waikato. *Weka Machine Learning*. 2020. URL: http://www.cs.waikato.ac.nz/ml/weka/.

[137] University of Waikato. *Weka Machine Learning Libraries in Java.* 2020. URL: https://weka.wikispaces.com/Use+WEKA+in+Java+code.

[138] University of Waikota. *Attribute Selection Algorithm*. URL: http://weka.sourceforge.net/doc.dev/weka/attributeSelection/InfoGainAttributeEval.html.

[139] University of Waikota. *Weka REPTree algorithm*. URL: http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/REPTree.html.

[140] Bradley M. Waite, Rachel Lindberg, Brittany Ernst, Laura L.Bowman, and Laura E.Levine. "Off-task multitasking, note-taking and lower- and higher-order classroom learning". In: *Computers and Education* (May 2018).

[141] Wikipedia. *Most common applications of 2019*. 2019. URL: https://en.wikipedia.org/wiki/List_of_most_popular_smartphone_apps.

[142] Wikipedia. *Heterogenous computing*. 2020. URL: https://en.wikipedia.org/wiki/Heterogeneous_comput.

[143] Heather Winskel, Tae-Hoon Kim, Lauren Kardash, and Ivanka Belica. "Smartphone use and study behavior: A Korean and Australian comparison". In: *Elsevir* (July 2019).

[144] Dongwon Kim Wonwoo Jung Yohan Chon and Hojung Cha. "Powerlet: An Active Battery Interface for Smartphones". In: *UbiComp*. ACM, Sept. 2014.

[145] Xiaochun Xie, Wu Chen, Xiaowei Zhu, and Dan He. "Parents' phubbing increases Adolescents' Mobile phone addiction: Roles of parent-child attachment, deviant peers, and gender". In: *Children and Youth Services Review* 105 (Oct. 2019).

[146] Zhang Y., Wang X., Liu X., Liu Y., Zhuang L., and Zhao F. "Towards better CPU management on multicore smartphones." In: *HotPower* (Mar. 2013).

[147] Zhisheng Yan, Qian Liu, Tong Zhang, and Chang Wen Chen. "CrowdDBS: A Crowdsourced Brightness Scaling Optimization for Display Energy Reduction in Mobile Video". In: *Mobile Computing, IEEE Transactions* (Nov. 2018).

[148] Xiujuan Yang, Zongkui Zhou, Qingqi Liu, and Cuiying Fan. "Mobile Phone Addiction and Adolescents' Anxiety and Depression: The Moderating Role of Mindfulness". In: *Journal of Child and Family Studies* (2019). URL: https://link.springer.com/article/10.1007/s10826-018-01323-2.

[149] Jiadi Yu and Yingying Chen. "Color Scheme Adaptation to Enhance User Experience on Smartphone Displays Leveraging Ambient Light". In: *IEEE Transactions on Mobile Computing* 16 (Mar. 2017). URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7484329.

[150] Yifan Zhang, Xudong Wang, Xuanzhe Liu, Yunxin Liu, Li Zhuang, and Feng Zhao. "Towards Better CPU Power Management on Multicore Smartphones". In: *HotPower* (Nov. 2013).