Understanding the Predictability of Smartwatch Usage

Yunsheng Yao*
Indiana University
Bloomington, IN, USA
yy29@iu.edu

Xing Liu
University of Minnesota
Minneapolis, MN, USA
liu00801@umn.edu

Feng Qian
University of Minnesota
Minneapolis, MN, USA
fengqian@umn.edu

ABSTRACT

We explore the predictability of smartwatch usage using a 9-month dataset collected from 27 users through a crowd-sourced user trial. Specifically, we investigate the predictability of (1) the device energy consumption, (2) the application launch time, and (3) the screen display. Overall, we find that all three aspects exhibit reasonably good predictability. Our findings provide key knowledge and insights for developing efficient and intelligent energy management services for future smartwatch systems.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile devices**.

KEYWORDS

Smartwatch; Usage Predictability; Energy Consumption.

ACM Reference Format:
Yunsheng Yao, Xing Liu, and Feng Qian. 2019. Understanding the Predictability of Smartwatch Usage. In *The 5th ACM Workshop on Wearable Systems and Applications (WearSys'19), June 21, 2019, Seoul, Republic of Korea.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3325424.3329661

1 INTRODUCTION

Smartwatch is one of the most popular wearable gadgets. Its sales are expected to exceed 100 million by 2020 [1]. Compared to traditional mobile devices such as smartphones and tablets, smartwatches operate under much more constrained energy budget. On one hand, their small form factors and wearable nature limit their performance and battery capacity. On the other hand, the functionalities of wearables are becoming far more ubiquitous and rich. For example, researchers have developed a wide spectrum of novel applications that fully exploit watches' potentials, such as deep learning [19], sign language translation [10], and drinking activity monitoring [8]. All these features incur additional energy footprint and heat dissipation on smartwatches.

Given the potential mismatch between smartwatches' rich functionalities and tight energy budget, smartwatches need *smart energy management* at the operating system (OS) level. Despite a plethora of work on energy management and optimization for smartphones [4, 5, 11], the corresponding state-of-the-art is still quite rudimentary for smartwatches. For example, Android Wear uses static policies for controlling energy-hungry components such as display, with much room for improvement [13]. Porting existing smartphone energy management schemes to smartwatches is indeed possible. Nevertheless, we are more interested in exploring the feasibility of *leveraging wearable-specific characteristics to further facilitate energy management on smartwatches.* In particular, we hypothesize that compared to smartphones, smartwatches exhibit better predictability of users' and applications' behaviors and therefore the incurred energy consumption, due to smartwatches' small form factors that limit user interactions as well as the excessive application states. Such predictability, if exists, can be leveraged to make energy management decisions more intelligent. For example, accurate prediction of smartwatch usage in a 12-hour window helps the energy manager adaptively and proactively turn off low-priority features, allowing the battery to survive at the end of the day. Besides benefiting the energy aspect, good usage predictability also enables other important use cases such as fast app launching, smart network interface selection, and smart display brightness control.

Motivated by the above, in this study, we explore the predictability of smartwatch usage using a large 9-month dataset collected from 27 smartwatch users through an IRB-approved user study. This dataset bears to our knowledge the longest duration compared to prior datasets such as [17] and [13]. Leveraging this dataset, we answer the following questions that are under-explored by previous studies.

• *How predictable is the smartwatch energy consumption?* We feed our collected data into an accurate smartwatch energy model. We then apply off-the-shelf machine learning algorithms to predict each user's energy consumption using historically observed usage activities. We find that overall the energy consumption can be well predicted: at the granularity of five energy levels, the median prediction accuracy across users reaches around 90%. We find poor predictability is usually caused by infrequent device usage. At the hardware component level and the device state level, the energy consumption also exhibits good predictability in general.

• *How predictable are application launches?* Users can run diverse apps on smartwatches. Accurately predicting the app launch time facilitates per-app energy accounting. Meanwhile, it also helps app pre-launching, which can effectively reduce the perceived app launch time [21]. We find that many smartwatch apps' launches exhibit predictable patterns. The median precision and recall for all users' app launch predictions are 79.4% (84.6%) and 86.5% (91.2%), respectively, when the prediction granularity is 5 (15) minutes. Our further analysis indicates that performing app pre-launching

*Current affiliation: Amazon.

WearSys'19, June 21, 2019, Seoul, Republic of Korea
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6775-2/19/06…$15.00
https://doi.org/10.1145/3325424.3329661

based on such predictions helps hide the perceived loading time for the vast majority of the apps while incurring very small energy overhead if the prediction is inaccurate.

• *How predictable is the smartwatch screen display?* Most smartwatches use OLED display whose power consumption depends on the color of each displayed pixel. The traditional way of profiling the OLED display power is to perform pixel counting [5]. Doing so incurs non-trivial computational and energy overhead on smartwatches. We find that on smartwatches, the screen display is dominated by apps' UI as opposed to other content such as large images and videos that frequently appear on smartphones and tablets. Therefore, the pixels' color distribution can be well predicted using lightweight measurable information such as the foreground app name and its state, as confirmed by us using our data.

Overall, our preliminary findings quantitatively validate the aforementioned hypothesis, and provide key insights for developing efficient and intelligent energy management services for future smartwatches. In the remainder of this paper, §2 gives the background of our user trial. §3, §4, and §5 describe our results. §6 discusses related work, followed by the conclusion and future work in §7.

2 BACKGROUND

We conducted an IRB-approved smartwatch user study at a large U.S. university. The study was open to students, faculty, and staff members. We recruited 27 users out of more than 200 applicants. The participants consist of 18 students, 5 staff members, and 4 faculty members. 26% of the participants were female. We provided each user with an LG Urbane Android smartwatch. The watch is equipped with a quad-core Cortex A7 processor with 512MB memory, 4GB storage space, WiFi/Bluetooth radios, and various sensors. The watch runs Android Wear OS 1.3.

Despite being very familiar with conventional watches, many participants have limited or even no experiences on smartwatch. We thus provided the users with an orientation session where we went through all features of the smartwatch. Also, we started the actual data collection several months *after* giving the watch to each participant. This gives more than enough time for the participants to get familiar with the smartwatches and to develop their usage habits. The actual data collection lasted more than 9 months from late 2016 to 2017.

We developed a custom data collection system which collects a wide range of information including application activities, screenshots, screen brightness level, CPU utilization, device power state, WiFi/Bluetooth states, and user interaction events. The data collector runs in the background and is transparent to users. The collected data is automatically uploaded to our secure server over WiFi at night when the watch is being charged.

We also developed fine-grained energy models through in-lab controlled experiments. The ground truth power that is used to fit the model is measured using a Monsoon power monitor [2] hooked to the watch. Our model consists of various system components including sleep/wakeup base, CPU, display, WiFi, Bluetooth, and touching events. The error rate of the model (*w.r.t.* the overall device energy consumption) is less than 6% based on our tests using various workloads. Details of the energy model can be found in our prior work [13].

3 PREDICTING ENERGY CONSUMPTION

Motivation. Smartwatch operates under tight energy constraints. Accurate prediction of its energy consumption helps facilitate smart energy management services as described in §1. Comparing to smartphones, smartwatches have relatively homogeneous functionalities, which can possibly lead to better predictability of the energy usage.

Methodology. We study the energy usage pattern for all 27 users during the 9-months data collection period. The daily energy consumption for each user is calculated by applying our power model. We first gather usage events or statistics for CPU, display, Bluetooth, WiFi, *etc.*, and then apply our power model to obtain the total daily energy consumption. Since predicting the precise energy consumption value is not practical, we convert the actual battery drain (in mAh) into 5 energy consumption levels. For a better granularity, we further extend the number of energy consumption levels to 10 and 15.

We choose the Support Vector Machine (SVM) classification algorithm for the energy level prediction. Specifically, we use the daily energy consumption level of a week to form a 7-dimensional feature vector X. We mark the energy consumption level of the day right after the week as label Y. Then we proceed with the same procedure for every consecutive 8 days within the training window (described below) to construct the training data. The reason we choose 7 as the feature vector size is because it reflects a user's weekly schedule. We also consider larger vector sizes (history windows), such as 10 and 15 days. We use a sliding window based approach to learn and predict energy usage pattern over time. For each user, we first train the classifier with the initial 20% of the feature-label (or X-Y) pairs, and then let the classifier predict the next 5% of the dataset's labels (the data is sorted in chronological order). Next, we advance both the training and the prediction window by 5%, and so on, in order to emulate an online prediction scheme. We apply SVM with RBF kernel provided by Scikit-Learn [3].

Results. Figure 1 shows the average prediction accuracy across all users, for feature vector sizes of 7, 10, and 15 (days). Most users exhibit high prediction accuracy where the median values are around 90%. The accuracy remains similar across different vector sizes. Figure 2 shows the results with different energy consumption levels of 5, 10, and 15. As expected, the prediction accuracy improves as the number of energy consumption levels decreases.

We notice low prediction accuracy (less than 50%) for some users from Figure 1. We find most of these users barely used their smartwatches over the study period. The average usage duration (*i.e.,* when the watch is worn) for these users is only 1.53 hours per day. In Figure 3, we show the daily average usage hours versus the prediction accuracy for all users. We clearly observe two clusters of frequent and infrequent users, and that insufficient daily usage (less than 2 hours) is correlated with low prediction accuracy, whereas high usage typically leads to much better predictability.

Component-level Breakdown. We further breakdown the overall energy consumption into separate hardware components and study their individual predictability. We study three components:

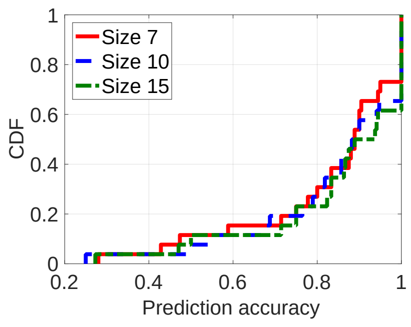

Figure 1: Prediction accuracy of daily energy consumption with varying feature vector sizes, across all users (energy levels = 5).

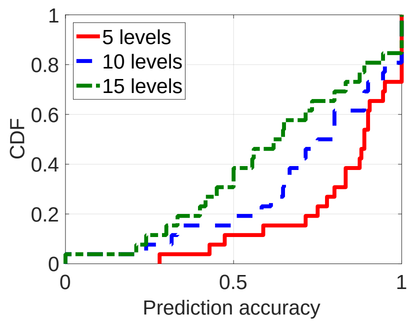

Figure 2: Prediction accuracy of daily energy consumption with varying energy levels, across all users (vector size = 7).

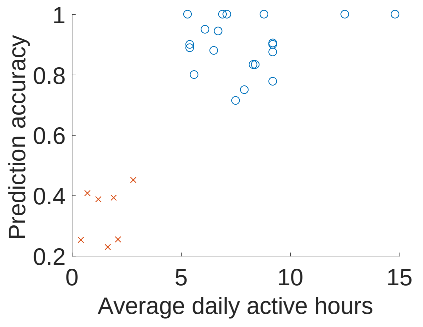

Figure 3: Users' energy consumption prediction accuracy vs. average daily active hours (energy levels = 5, vector size = 7).

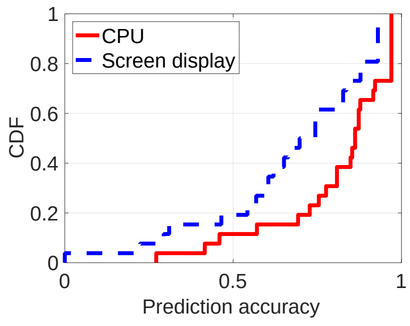

Figure 4: Prediction accuracy of CPU and display energy across all users (energy levels = 5, vector size = 7).

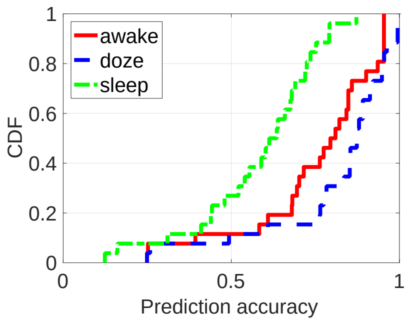

Figure 5: Prediction accuracy of each device state's energy across all users (energy levels = 5, vector size = 7).

CPU (consuming 29% of the overall energy), display (30%), and Bluetooth/WiFi (together 3.5%). The remaining energy, which is difficult for us to separate, is drained by smartwatch's system-on-chip (SoC) and other components such as sensors. We find that for CPU and display, which dominate the overall battery drain, their energy consumption level is highly correlated with the overall energy consumption level, with the median Pearson Correlation Coefficient (PCC) across all users being 0.97 and 0.98, respectively. As a result, they can also be well predicted. As illustrated in Figure 4, SVM achieves a median prediction accuracy of 88% for CPU and 75% for display. On the other hand, some other hardware components exhibit a sparse usage pattern. For example, unlike on smartphones, wireless radio (WiFi and Bluetooth) on smartwatches are lightly used. Its energy consumption accounts for only 3.5% of, and poorly correlates with the device-wide energy consumption, with a median PCC less than 0.4. As a result, it is difficult to predict its usage and henceforth the energy utilization.

State-level Breakdown. We also breakdown the overall energy consumption by smartwatch states and study their individual predictability. The smartwatch states and their energy consumption percentages are awake (27.2%), dozing (56%), and sleeping (16.8%). On our LG Urbane watch, the awake state is triggered when a user actively interacts with the watch. If the user stops the interaction for 5 seconds, the smartwatch will go to the dozing state (a "shallow" sleeping mode with the watch face darken). If there is no further interaction for 35 minutes, the watch will go to the sleeping mode. We then apply SVM to predict the daily per-state energy consumption level for all users, and get the results in Figure 5. The median prediction accuracy for the awake state is around 82%. The dozing state has slightly better accuracy, with a median of 87.3%. The sleeping state presents the lowest predictability, with a median of around 59%. A possible explanation is that due to the long timer (35 minutes), entering the sleeping mode is a relatively rare event, whose energy consumption thus exhibits more randomness.

Summary. For active smartwatch users, the daily energy consumption level can typically be accurately estimated, with the median value of around 90%. For a small number of users, their prediction accuracy is low due to a lack of active usage. We also find generally good energy consumption predictability at the component level and the state level. The above findings can be leveraged to improve the power management and battery life forecast on today's smartwatches.

4 PREDICTING APP LAUNCHES

Motivation. Accurately predicting the app launch time facilitates per-app energy accounting. It also helps app pre-launch, which

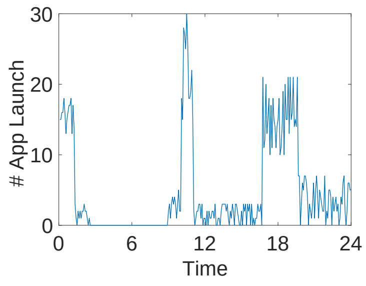

Figure 6: One user's temporal usage pattern of the Fitness app.

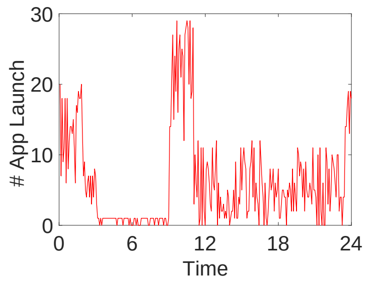

Figure 7: One user's temporal usage pattern of the Weather app.

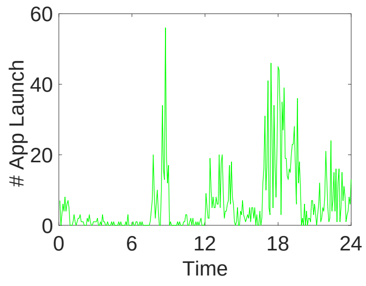

Figure 8: One user's temporal usage pattern of the Google Map app.

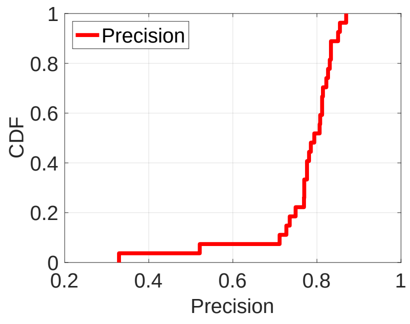

Figure 9: Precision of app launch prediction, across all users.

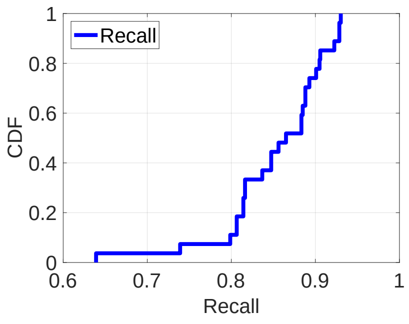

Figure 10: Recall of app launch prediction, across all users.

can effectively reduce the perceived app loading time [21]. We notice that lengthy app launch time is in particular a problem on smartwatches whose computation capability is far lagging behind compared to modern smartphones. For example, on LG Urbane, launching Google Maps takes about 8 seconds, and launching a web browser takes even longer. Meanwhile, today's smartwatches do have reasonably large memory (512MB to 2GB), making caching app launches feasible. Also, compared to smartphone apps, smartwatch apps are used in more limited contexts. We thus envision it is more feasible to predict the latter's launch time.

We exemplify the launch patterns for 3 apps (Fitness, Weather, and Google Maps) used by 3 different users. Figure 6 plots the Fitness tracker's launch time for a user. The X-axis represents the time-of-day, and the Y-axis counts the app launches over the entire study period, at a 5-minute granularity. The count is higher at certain hours, indicating strong temporal patterns (*e.g.*, a higher usage in the evening when the user goes to the gym). Similar observations are made in Figure 7 and 8, which show the launch patterns of Google Map and Weather, respectively, for two other users. Overall, due to their wearable nature, smartwatches' app usages are tightly coupled with a user's daily activities that oftentimes exhibit strong predictable patterns.

Methodology. We next detail our methodology. Smartwatch apps can be largely classified into two categories: (1) apps that require users' interaction to launch, and (2) background apps such as email and instant messengers that are usually passively launched when a push notification is received. Apps belonging to the second category typically have a service stub running in the background so the launching delay is small. We thus focus on the first category of apps whose long launching delay negatively affects the QoE.

We apply decision tree classification for launch time prediction. Decision tree is suitable here as a binary decision (either to launch or not) needs to be made at a certain time. We first describe how we generate the training and testing set. Both sets are constructed on a per-user and per-app basis. For each day, we divide its 24 hours into 288 5-minute slots. If the app was launched within a slot, we set the slot to 1, otherwise 0. Thus for each day we can get a 0-1 vector of length 288. We use the initial 20% of the days as the training set and apply the learned prediction model to test the next 5% days (from 20% to 25%). We then repeat this process by sliding both training and testing set "forward" by 5% of the days, until we cover all days. Similar to the scheme used in §3, we use 7 vectors from the consecutive days in a week as the feature and the 8th vector of the day right after the week as the label.

Results. The precision and recall results for all eligible apps across all users can be found in Figure 9 and 10, respectively. The median precision is 79.4%, indicating our trained model, despite being simple, can reasonably avoid type I errors (false positives). The median recall rate is 86.5%, indicating a good coverage *i.e.,* most app launches can be detected at the 5-minute granularity. Increasing the bin size from 5 minutes to 15 minutes further improves both the precision and recall rates by about 5%: 84.6% median precision and 91.2% median recall across all users.

When using the above prediction results for app pre-launching, type I errors (false positives) can lead to energy overhead (*i.e.,* pre-launching an app that the user does not actually launch). Here

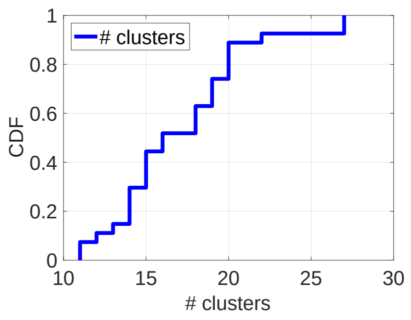

Figure 11: Number of generated clusters across all users.

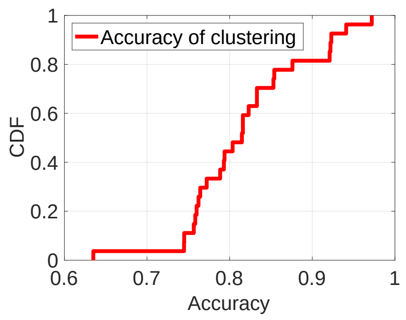

Figure 12: Clustering accuracy across all users.

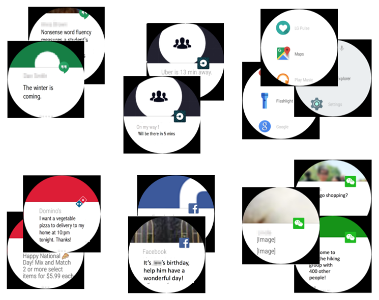

Figure 13: Example screenshots belonging to six clusters of a user.

we estimate the energy waste incurred by a single type I error. Assuming CPU is 100% utilized during a pre-launch and the pre-launch takes 8 seconds (the Google Map case), the overall energy (including the base energy consumption) wasted by a single type I error will be around 0.27 mAh according to the power model. On average, using the above decision tree model, type I errors occur 10 times a day for an average user, so the total energy waste is less than 1% of the overall battery capacity of our smartwatch (410 mAh). On the other hand, Type II errors (false negatives) do not bring any energy overhead, since the app will still be launched normally as triggered by the user.

Summary. We found that many smartwatch apps' launches exhibit predictable patterns. The median precision and recall rates across all users' app launch predictions are 84.6% and 91.2%, respectively, at the 15-minute granularity. Our approach will remove the perceived loading time for up to 91% of the app launches, with negligible energy overhead when type I errors occur. Regarding the training overhead, recall that the training only needs to be periodically performed (*e.g.,* on a daily basis). Therefore, we can offload the training process onto a cloud or edge server when the smartwatch is charging.

5 PREDICTING SCREEN DISPLAY

Motivation. On smartwatches, despite the small screen size and the aggressive sleeping policy, the display is still the most power-hungry component (§3). Therefore, it is important to accurately account for the display energy consumption. For OLED display, which is very popular among mobile devices, since its power consumption is determined by each pixel's RGB value [5], the typical approach for modeling and measuring the display power is to count the pixels of different colors. However, this approach itself incurs high energy and computational overhead especially on resource-constrained wearable devices, even if sampling is used. Instead, our key observation is that compared to a smartphone, the smartwatch display is more consistent at a per-app or per-app-state basis: the smartwatch screen is mostly dominated by UI while for smartphone, the screen may contain not only UI, but also more dynamic and custom content such as videos and images; in addition, the UI components and layout theme are usually fairly consistent for smartwatch apps, likely due to the small screen size and limited user interaction. For example, Google Hangout has a dark green frame and white chat dialog box. As a result, for the same app (state), the pixel-wise color histograms are usually very similar. Note that users may change an app's display scheme, but such events can either be directly detected or indirectly learned. Based on these observations, we propose to infer the color distribution of the displayed content using ultra-lightweight information such as the foreground app name and the app state.

Methodology. We apply unsupervised learning (clustering) to validate the above assumption. We first extract the color histograms from all screenshots of all users. There are roughly 100,000 screenshots captured during the user study (a screenshot is captured every 30 seconds). We notice that taking a single screenshot indeed incurs a high overhead – taking about 1 second on our LG Urbane watch. For each screenshot, we extract all pixels' color and construct the corresponding color histogram, which is then converted into a feature vector. We then apply clustering on the color histograms for each user. We use Bisect k-means [16], which is similar to k-means but without requiring specifying the number of clusters that is typically unknown beforehand. Bisect k-means works by iteratively splitting a cluster with the lowest internal similarity into two clusters, until the internal similarity values of all clusters are above a pre-defined threshold. The threshold is empirically chosen to be the average Euclidean distance between histograms of all application pairs (obtained using a small training dataset).

Recall that we hypothesize that the color distributions of the same app or app state tend to remain consistent. To validate this, we examine the generated clusters. Since each cluster corresponds to screenshots with similar color distributions, ideally, we expect that (1) all screenshots in each cluster belong to the same app or app state, and (2) each cluster corresponds to a unique app or app state.

Results. The number of clusters for each user is shown in Figure 11, with the median number being 16. Meanwhile, an average user has 18 apps installed on her watch. We first assess whether all screenshots within a cluster belong to the same app. We define the *label* of a cluster as the most common app name *i.e.,* the app name that is owned by the largest fraction of screenshots within the cluster. We call a screenshot to be correctly clustered if its app name matches the label of the cluster it belongs to. Then for each user, we define the *clustering accuracy* as the ratio between the number of correctly clustered screenshots and the total number of screenshots. Figure 12 plots the distribution of the clustering accuracy across all users. As shown, the median value is around 81%, which we believe to be overall decent. Note that this preliminary result can be further improved in several aspects, including (1) considering the app state such as the Activity information within an app (currently we only consider the app name) and (2) enriching the features by considering, for example, the UI layout (currently we only use the color histogram). We also examine the heterogeneity of the clusters' labels. For each user, we compute the ratio between the total number of unique labels (the most common app name within a cluster) and the total number of clusters. The median value of the ratio is around 0.85, indicating that the clusters do usually have different labels. Figure 13 exemplifies screenshots belonging to different clusters.

We look into the source of error for the results. One major type of error originates from the fact that when two apps have similar UIs, their screenshots' color histograms may also be similar. In this case, we may have two apps' screenshots mixed into a single cluster. However, for the purpose of inferring the display energy, this will not be an issue because similar color histograms lead to similar OLED screen power consumption. Upon manual inspection, we find 40% of mis-clustered screenshots belong to this category.

The above results dictate an ultra-lightweight method for profiling smartwatch's OLED display energy. Given the clustering model, which can be obtained by very sparse online training, the energy profiler simply utilizes the app (state) information to instantaneously localize the corresponding color histogram cluster and thus the display power consumption.

Summary. Despite the small screen size, smartwatch's display draws a large fraction of energy from the battery. We show that lightweight information such as app name can be utilized to estimate the color histogram and henceforth the OLED display power consumption. A simple clustering method achieves an overall clustering accuracy of 81% and a potentially even higher energy estimation accuracy.

6 RELATED WORK

Predictability Study of Mobile Devices. In the literature, efforts have been spent on improving the energy efficiency and app performance of mobile devices using predictive information, such as predicting smartphone app launch [21], predicting user mobility [6], predicting network conditions [20], and predicting the user's pose [9]. Our work instead investigates the predictability in the smartwatch context.

Smartwatch Usage. Several prior studies investigated smartwatch usage. Lyons conducted a user survey to shed light on the design of smartwatches [14]. Min *et al.* characterized smartwatch battery usage based on an online survey involving 17 users [15]. Poyraz *et al.* conducted a user study to understand the smartwatch power consumption and user activities [17]. Liu *et al.* characterized real smartwatch users' usage patterns, energy consumption, and network traffic [13]. Recently, Kolamunna *et al.* studied SIM-enabled wearables in the Wild [12]. We instead focus on the predictability of smartwatch energy consumption, app launch, and display, all of which are under-explored by prior studies. Also, our user study duration (9 months) is much longer than prior studies (*e.g.,* 70 days for [17] and 106 days for [13]).

Smartphone User Studies. In the past, researchers have launched various crowd-sourced studies of smartphones (*e.g.,* [7][18][5]). Compared to them, much fewer user trials have been conducted on wearable devices such as smartwatches.

7 CONCLUSION AND FUTURE WORK

Through analyzing a 9-month dataset from 27 real smartwatch users, we quantitatively confirm the good predictability of three key aspects of smartwatch usage: energy consumption, app launch, and screen display. Our findings can be leveraged to make resource management more intelligent and efficient for future wearable systems. They also facilitate other important services such as app pre-launching on smartwatches. We plan to leverage the lessons learned in this work to build a real usage prediction system on COTS smartwatches. We will then incorporate the predictor into a wearable energy manager.

REFERENCES

[1] 101 million smartwatches globally by 2020. http://bit.ly/2OSEufE.
[2] Monsoon Power Monitor. https://www.msoon.com/LabEquipment/PowerMonitor/.
[3] Support Vector Machine. https://scikit-learn.org/stable/modules/svm.html.
[4] D. H. Bui, Y. Liu, H. Kim, I. Shin, and F. Zhao. Rethinking energy-performance trade-off in mobile web page loading. In *MobiCom*, 2015.
[5] X. Chen, N. Ding, A. Jindal, Y. C. Hu, M. Gupta, and R. Vannithamby. Smartphone energy drain in the wild: Analysis and implications. *SIGMETRICS*, 2015.
[6] Y. Chon, E. Talipov, H. Shin, and H. Cha. Mobility prediction-based smartphone energy optimization for everyday location monitoring. In *SenSys*, 2011.
[7] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. Diversity in smartphone usage. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 179–194. ACM, 2010.
[8] T. Hamatani, M. Elhamshary, A. Uchiyama, and T. Higashino. Fluidmeter: Gauging the human daily fluid intake using smartwatches. *IMWUT/UbiComp*, 2018.
[9] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han. Rubiks: Practical 360-degree streaming for smartphones. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 482–494. ACM, 2018.
[10] J. Hou, X.-Y. Li, P. Zhu, Z. Wang, Y. Wang, J. Qian, and P. Yang. Signspeaker: A real-time, high-precision smartwatch-based sign language translator. In *MobiCom*, 2019.
[11] A. Jindal and Y. C. Hu. Differential energy profiling: energy optimization via diffing similar apps. In *OSDI*, 2018.
[12] H. Kolamunna, I. Leontiadis, D. Perino, S. Seneviratne, K. Thilakarathna, and A. Seneviratne. A first look at sim-enabled wearables in the wild. In *IMC*, 2018.
[13] X. Liu, T. Chen, F. Qian, Z. Guo, F. X. Lin, X. Wang, and K. Chen. Characterizing smartwatch usage in the wild. In *MobiSys*, 2017.
[14] K. Lyons. What can a dumb watch teach a smartwatch?: Informing the design of smartwatches. In *ACM International Symposium on Wearable Computers (ISWC)*, 2015.
[15] C. Min, S. Kang, C. Yoo, J. Cha, S. Choi, Y. Oh, and J. Song. Exploring current practices for battery use and management of smartwatches. In *ACM International Symposium on Wearable Computers (ISWC)*, 2015.
[16] K. Murugesan and J. Zhang. Hybrid bisect k-means clustering algorithm. In *Intl. Conf. on Business Computing and Global Informatization (BCGIN)*, 2011.
[17] E. Poyraz and G. Memik. Analyzing power consumption and characterizing user activities on smartwatches: summary. In *Workload Characterization (IISWC), 2016 IEEE International Symposium on*, pages 1–2. IEEE, 2016.
[18] F. Qian, K. S. Quah, J. Huang, J. Erman, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. Web caching on smartphones: ideal vs. reality. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 127–140. ACM, 2012.
[19] M. Xu, F. Qian, M. Zhu, F. Huang, S. Pushp, and X. Liu. Deepwear: Adaptive local offloading for on-wearable deep learning. *IEEE Transactions on Mobile Computing*, 2019.
[20] Q. Xu, S. Mehrotra, Z. Mao, and J. Li. Proteus: network performance forecast for real-time, interactive mobile applications. In *MobiSys*, 2013.
[21] T. Yan, D. Chu, D. Ganesan, A. Kansal, and J. Liu. Fast app launching for mobile devices using predictive user context. In *MobiSys*, 2012.