PASS: Reducing Redundant Notifications between a Smartphone and a Smartwatch for Energy Saving

Jemin Lee, *Member, IEEE*, Uichin Lee, and Hyungshin Kim, *Member, IEEE*

Abstract—Smartwatches have gained significant popularity in recent years. One major use of smartwatches is notification checking as an *extended display*. Smartwatch use provides an opportunity for energy saving because it affords a reduction in the frequency of smartphone use. However, sometimes user input is limited in smartwatches due to its small screen size and users are forced to use their smartphones to fully read notifications. In such cases, the advantage of an *extended display* in the smartwatch diminishes because users need to access their smartphones, which causes extra smartphone battery consumption. We define *phone-preferable notification* that requires a user to take further actions, such as checking detailed content and replying to a message. Given that phone-preferable notifications are likely to be handled on smartphones, it is possible to *defer notification delivery* to smartwatches. In this paper, we develop a novel notification manager, called *PASS* that automatically defers phone-preferable notifications and piggybacks them on watch-preferable notifications. For model building and evaluation, we collect 15,659 notifications *in-the-wild* from 11 users for approximately 31 days. In addition, for approximately two months we gather self-reported data from five users regarding which devices were used to respond to notifications. The results show that *PASS* can save daily battery for smartwatches and daily battery for smartphones up to 43.5 and 0.9 percent without introducing any noticeable negative results on user experiences.

Index Terms—Mobile notification, battery life, energy-saving, smartwatch, machine learning

1 Introduction

Despite the rapid market penetration of mobile devices, their utilization has been dependent on their battery life. In emerging wearable devices, such as a smartwatch, the battery resource is precious because battery size is small; it is smaller than that of a smartphone [1]. Most smartwatches (hereafter called watch) are dependent upon smartphones (hereafter called phone) for data connection due to the extra service fee of LTE networks. One of the most important functionalities of the watch is its *extended display*: the secondary display of a watch allows a user to run interactive but less-featured applications on behalf of the phone [2], [3]. It is handy in terms of checking notifications because the watch is attached on the wrist and thus, more easily accessible than the phone. In general, the interaction time with a watch is shorter than that with a phone [4], [5], [6]. Also, using a watch helps save phone battery by reducing phone use frequency. Hardware components in the watch consume less power than those in the phone because they simply comprise of a small screen and low-power CPU. If notifications are properly handled on a watch, even the sum of the energy consumption on using both a phone and a watch is less than that of using only a phone.

In practice, however, users cannot deal with all notifications in a watch due to the inconvenience of interactions on a small device on the wrist [7]. In other words, even after a notification has been delivered to a watch, users need to access their phone to reply to the notifications and/or to check more detailed information. Thus, delivering a notification to the watch is less useful in such cases. In this paper, we define this type of redundant notification as a *phone-preferable notification*, where the users eventually need to access the phone for further interactions related to the notification. The other type of notification is a *watch-preferable notification*; these are notifications that can be handled on a watch, without further interaction with the phone.

Current notification delivery mechanisms are not flexible enough to deal with this kind of notification because notification handling is permission-based at the application level [8], [9]. This means that mobile systems manipulate notifications according to application-specific configurations (e.g., priority, time-based rules, etc.). Recent studies have explored a user's interruptibility to lower the burden of interrupting the user with notification delivery [10], [11], [12], [13], [14], [15], [16]. This approach basically predicts an opportune moment for notification delivery by monitoring a variation of contexts and system configurations. Our work differs from prior

- *J. Lee is with the Department of Computer Science and Engineering, Chungnam National University, Daejeon 34134, Republic of Korea, and also with the AI Research Laboratory, ETRI, Daejeon 34129, Republic of Korea. E-mail: leejaymin@cnu.ac.kr.*
- *U. Lee is with the Graduate School of Knowledge Service Engineering, KAIST, Daejeon 34141, Republic of Korea. E-mail: uclee@kaist.ac.kr.*
- *H. Kim is with the Department of Computer Science and Engineering, Chungnam National University, Daejeon 34134, Republic of Korea. E-mail: hyungshin@cnu.ac.kr.*

Manuscript received 12 Dec. 2018; revised 24 June 2019; accepted 10 July 2019. Date of publication 23 July 2019; date of current version 1 Oct. 2020. (Corresponding author: Hyungshin Kim.)
Digital Object Identifier no. 10.1109/TMC.2019.2930506

interruptibility studies in that we mainly focus on handling phone-preferable notifications in order to improve energy efficiency in multi-device environments.

In this paper, we propose a novel power-aware notification delivery mechanism, called *PASS*, that extends mobile battery life by deferring *phone-preferable notifications* from being delivered to a watch. First of all, we perform a fine-grained measurement study to identify energy efficiency issues with notification delivery. We investigate energy consumption behaviors in the cases of both phone-preferable and watch-preferable notifications.

We then build a machine learning model that can predict a notification type: phone- or watch-preferable notifications. We hypothesize that a phone-preferable notification is related to the contexts of a user/app: app name, title, location, movement, time, and so on. To demonstrate feasibility, we developed *nCollector*, an Android app for data collection. We deployed the app in daily usage scenarios such as working in an office and hanging out with friends. With *nCollector*, we gathered 15,659 notifications *in-the-wild* from 11 users for around 31 days. This data allows us to study model building and evaluation.

We evaluate *PASS* in terms of (1) the accuracy of machine learning based classifiers trained on generic and individual data, (2) the daily battery savings for a watch and a phone, and prediction overhead, and (3) the usability of deferred notification.

We built a binary classifier corresponding to each user by using three machine learning algorithms with 10 discriminating features. Our results show that across all users, *PASS* achieves an F-score of 86 percent on average. We used a model-based estimation to calculate the daily battery gains of a watch and a phone. Across all users, the daily gains ranged between 1.5–43.5 percent (mean 13.3 percent) and between −0.2–0.9 percent (mean 0.2 percent), respectively. The inference overhead shows only 8.6 percent more energy consumption as opposed to that of the default status. Finally, to investigate the usability effect, we conducted a field trial by installing an application in the phones of participants that captures ground-truth labels on which device is used to address notifications. We followed this up with a questionnaire where the participants choose from various delay options regarding further actions on phone-preferable notifications such as prompt delivery to a watch or not. The results show that it is not necessary to forward most phone-preferable notifications (71.9 percent) to the phones immediately.

The main contributions of our work are summarized as follows.

- To our knowledge, it is first work to analyze the characteristics of energy consumption on mobile notification delivery between a smartphone and a smartwatch. Our detailed measurements pinpoint where redundant energy consumption occurs during cross-device interactions.
- With collected 15,659 notifications in-the-wild, we have revealed people's preferences in receiving notifications on specific devices. To measure cross-device usage systematically and to quantify users' preferences, we systematically explored device usage in multi-device environments using our tool, called *nCollector*, which is installed on a user's devices.
- We designed a light-weight classifier to manage notification delivery on sensed contexts for on-device inference. Even though we do not require any users' feedback or labels for training models, the inference for a phone-preferable notification achieves a stable state with an over 86 percent F-score spanned across all the participants except one. Also, our classifier is based on a light-weight machine learning model for on-device inference, and the inference overhead is minimal considering the benefit of energy saving.
- We conducted a user study by deploying a self-reporting mobile app (n = 5). We collected a total of 2,371 responses and then asked users whether phone-preferable notifications need to be deferred or not. As a consequence, most phone-preferable notifications (71.9 percent) are not necessary to be forwarded on the phones immediately, and 50 percent of deferral time for phone-preferable notifications is less than 1 hour.

2 Background and Motivation

We briefly explain the notification delivery mechanism in a phone and watch paired system using the Android OS. Subsequently, we pinpoint the cause of energy inefficiency in notification delivery between a phone and a watch.

2.1 Notification Mechanism and Types

Android Wear SDK [17] released by Google is one way to span data across a watch and a phone. A developer employs *Data-layer API*, provided by Android-Wear SDK, for data synchronization between a phone and a watch over BLE or Wi-Fi. The Wear-SDK's *Data Layer* API, which is part of Google-Play services, provides a communication channel for wearable applications. Mobile application developer can send messages, images, or videos via *Data Layer*.

With *Data Layer*, Android Wear supports multiple watches connected to a phone; for example, when a user receives a notification on a phone, it automatically appears on a watch.

To synchronize data between two devices, Google's servers host a cloud node in the network of devices. Android OS synchronizes data to directly connected devices, the cloud node, and to wearable devices connected to the cloud node via Wi-Fi or BLE. If wearable devices do not connect to Wi-Fi, they can only sync data through a smartphone. According to a recent study, a large portion of watch usage time (80 percent) is limited to the pairing mode with a phone [18]. Note that Android Wear 2.0 supports a standalone watch that has direct Internet access through cellular networks. While standalone watches are gaining popularity, at this point, the dominant product type is Wi-Fi watch model due to price and service fee.

Data Layer depends on a user's configuration to decide whether a notification relays to either one of paired devices with a phone. If a user does not configure the notification delivery option, all notifications appear in both devices. Occasionally, users redundantly use both devices to check notifications owing to the nature of smartwatches (wrist worn device and small form factors). If a user needs to perform an additional task, such as replying to a message and

TABLE 1
Selected Mobile Applications and Devices

Category	Application Name
Communication	WhatsApp, Kakao Talk
Email	Gmail
Social Network	Facebook
Tool	PingMe
Device Type	**Spec.**
Smartphone	Nexus5,Android 5.1
Smartwatch	LG-Urbane, Android Wear 1.5

reading full message contents, a phone becomes involved. Based on the nature of the interactions with the two devices, we defined the notification types associated with preferable devices. Although the types vary with users, we mainly considered interaction levels to determine whether a notification is watch preferable. This is because a notification is eventually delivered to a watch by bundling with a watch-preferable notification. Also, we argue that users can be aware of all notification on their smartphones. This allows us to implicitly label notifications without gathering self-reported responses. Therefore, *PASS* only takes into account interaction levels for realistic system implementation. The detailed notification definitions are given as follows.

- Watch-preferable notifications: read-only and glanceable notifications are considered watch preferable.
- Phone-preferable notifications: notifications requiring replies (e.g., RSVP) and checking detailed information (i.e., long email) are considered phone preferable.

2.2 Preliminary Experiment

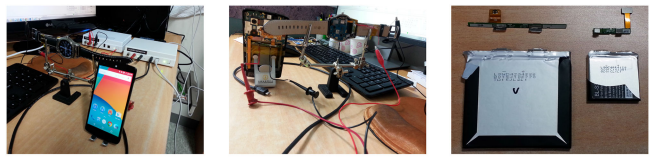
In this section, we present the quantification of how watch use can save battery in a phone or waste energy due to redundant interactions. Using the representative cases of two notification types, we measured energy inefficiency of interaction with notifications between a phone and a watch. Referring to a prior work [19], we selected five mobile applications as listed in Table 1. These applications were selected from four categories: communication, email, social network, and tool. Subsequently, we defined representative usage scenarios corresponding to notification types for the five applications.

The following scenarios are regarding watch-preferable notifications for the five applications in which a user simply checks and removes notifications without launching an application that triggers a notification.

- Email: Read a text and scroll.
- Social Network: Read a post's title.
- Messenger: Read a message.
- Tools: Check an event.

The following scenarios are with respect to phone-preferable notifications on four applications, which require further interactions on the phone such as sending a message and reading detailed contents.

- Email: Read a text and send a new e-mail.
- Social Network: Read a post's title and read full content of a post.
- Messenger: Read a message and send a new message.



(a) (b) (c)

Fig. 1. Experimental setup for directly measuring power consumption: (a) front scene, (b) back scene, and (c) the battery interface circuits of the phone and the watch.

Regarding a tool application, we were unable to define a phone-preferable notification scenario. Thus, we omitted this type of application.

To measure the power consumption during each scenario, we present an experimental setup for multiple devices with two Monsoon Power Monitors [20], as shown in Fig. 1. Unlike prior works [21], [22], we directly carved out an interface circuit from a watch battery rather than using an interface of a phone battery by the same vendor. To our knowledge, using an interface circuit of a watch battery has been unknown so far.

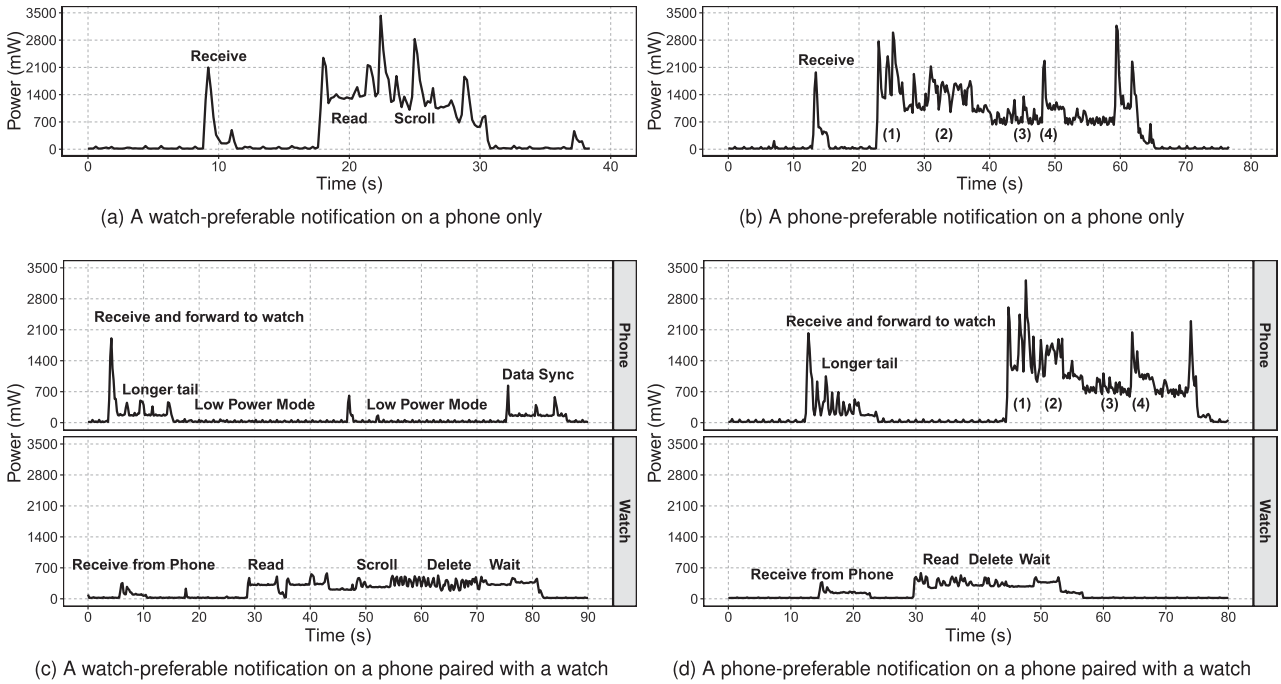
We created two test scenarios as following: a standalone phone and a phone with a watch. To compare the usage of each device, we executed each selected application on each device according to specific usage scenarios.

Fig. 2 shows the power traces of phone-preferable and watch-preferable notifications triggered by the `KakaoTalk` application on the two devices. Fig. 2a illustrates the power trace of a watch-preferable notification on a standalone phone, wherein the power consumption is divided into two regions: receiving and reading a message. Fig. 2b shows the power trace of a phone-preferable notification on a standalone phone where the power consumption is divided into four regions. We annotated each power consumption region with numerals: (1), (2), (3), and (4). These numbers correspond to reading, scrolling, waiting, and sending, respectively.

With the same functions that we used in a phone-only scenario, we measured power consumption on the phone and the watch, simultaneously. Fig. 2c illustrates the power trace of a watch-preferable notification on the phone and the watch. Interestingly, the phone stays at the low power mode for almost the entire time because the watch is active for the interaction instead of the phone. Obviously, the watch consumes much lower energy than the phone because it is equipped with a smaller screen and a low-power CPU. For that reason, increasing usage time of the watch extends the phone's battery in case of a watch-preferable notification.

Fig. 2d illustrates the power trace of a phone-preferable notification on the phone and the watch. In contrast to a watch-preferable notification, it requires to the user to use the phone to perform further actions. There are two insights from this experiment: (1) The longer network tail state by forwarding notifications to a watch over BLE and (2) redundant usage patterns on the watch because the same tasks are happening on the phone.

We measured the power consumption for each scenario for the two device configurations. Fig. 3a shows the energy consumption in terms of a watch-preferable notification. For all the five applications, the energy consumption of a *phone only* case varies from 14.40J to 30.55J, with the average



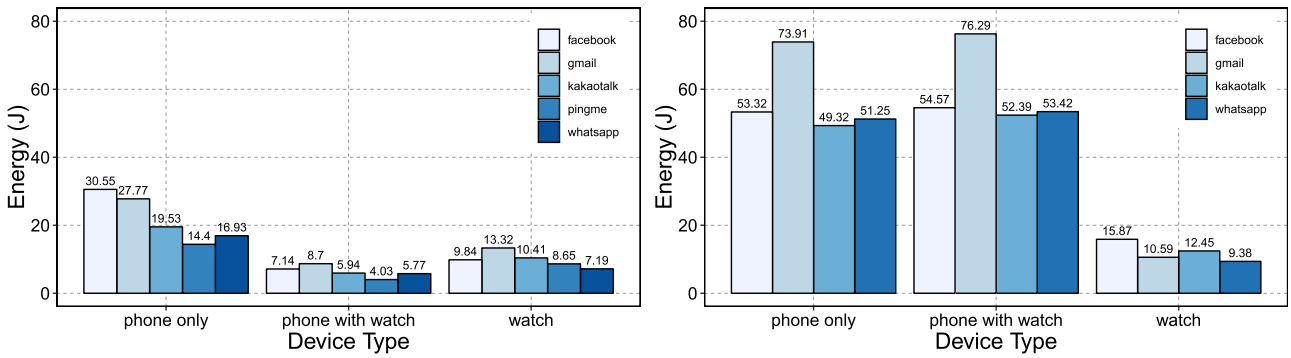
(a) A watch-preferable notification on a phone only

(b) A phone-preferable notification on a phone only

(c) A watch-preferable notification on a phone paired with a watch

(d) A phone-preferable notification on a phone paired with a watch

Fig. 2. Power consumption for Kakao-talk.



(a) watch-preferable notifications by five real applications

(b) phone-preferable notifications by four real applications

Fig. 3. Energy consumption by notification type corresponding to real applications.

being 21.84J. The energy consumption of a *phone paired with watch* case ranges between 4.04J and 8.70J, with the average being 6.32J. In this case, the *phone paired with watch* saves 3.4x more energy than a *phone only* case by offloading glanceable tasks to the watch.

Fig. 3b indicates the power consumption in terms of a phone-preferable notification. For four applications, energy consumption of the *phone only* varies from 51.25J to 73.91J with the average being 56.95J. Energy consumption of the *phone paired with watch* case ranges between 52.39J and 76.29J, with the average being 59.17J. In this case, the *phone paired with watch* consumes an average of 3.75 percent (2.22J) more energy than a phone only scenario. This extra power is caused by CPU cycle and BLE power by forwarding a notification to a watch.

Our research goal is to augment the current notification delivery mechanisms between a phone and a watch by deferring phone-preferable notification delivery to a watch. For this, we predict the notification type using a machine learning model. By doing so, *PASS* automatically manages a notification delivery between a phone and a watch.

2.3 Bundling Notifications

To quantify a bundling effect, we conducted an experiment where a phone-preferable notification was deferred until a watch-preferable notification appeared, rather than not sending a phone-preferable notification to a watch. Batch notification delivery has different energy saving effects depending on the coalescing size. If the batch size is 128 as shown in Fig. 4, we save on energy consumption of a phone and a watch by up to 8.02x and 5.84x, respectively. Such batch delivery situations help both devices consume less energy while minimizing usability degradation.

In addition, from the experiment, the batch delivery shows that the energy saving effect of *PASS* could be gradually improved when the devices paired with a phone were increased. This is because the current notification delivery system relays all the notifications to the connected devices. From our preliminary study, we discovered the burden of the phone battery to relay a notification to another device. This burden would be exacerbated when the phone is connected to several devices.

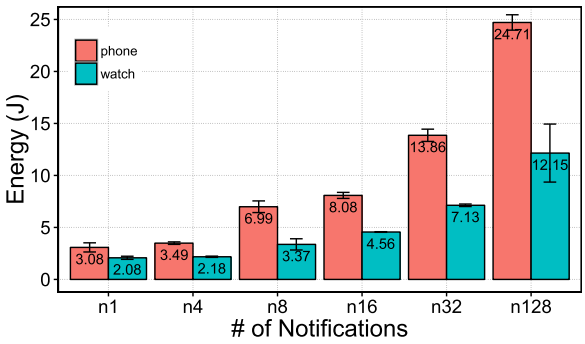


Fig. 4. Energy consumption of consecutive notification deliveries between a phone and a watch.

3 Notification Labeling Method

As mentioned before, a user can extend the battery life of a phone by using a watch instead, in the case of a watch-preferable notification. However, in the case of a phone-preferable notification, which involves the use of the phone, the batteries of the phone and the watch are wasted by data forwarding overhead and redundant interactions.

More specifically, we define that a notification is phone-preferable if an application that triggered the notification is launched within a pre-defined reaction time. In a real-world study, we need to unobtrusively label notifications without an on-body sensor or user interruption for questionnaire answering.

Fig. 5 provides a high-level description on the notification type can be determined. Our system monitors whether an application that triggers a notification and is launched before or after this notification is removed. Two time variables, denoted by $T_{launched}$ and $T_{removed}$ are related to the user's decision. Then, the elapsed time $T_{elapsed}$ is determined by $|T_{removed} - T_{launched}|$. If $T_{elapsed}$ is less than or equal to 10 seconds ($T_{reaction}$), we conclude that a user has pulled out the phone to handle further interaction with the notification. We empirically set 10 seconds as a $T_{reaction}$ because 25 percent of response time for notifications is less than 10 seconds and the curve of response time stabilizes at this point.

We designed a simple algorithm that can label phone-preferable notifications using only system information as shown in Algorithm 1. The labelling algorithm takes $T_{reaction}$, which is to decide the application trigger as an input. This algorithm produces Boolean type values, in which $TRUE$ represents a phone-preferable notification type as an output. The detailed process of the phone-preferable notification detection is described in the following section. The first step involves the initiation of $TYPE_{phone-preferable}$, which contains the result as $FALSE$. The second step is a loop that repeats until one of the arrival notifications is removed. If a notification is removed, the removal time and sender application that triggers this notification are assigned to $T_{removed}$ and $senderApp$, respectively. Then, the loop is escaped from.

The third step involves pausing the notification to capture the application usage history after removing the notification. The application usage history is then assigned to $appList$. The fourth step is a loop that repeats until $appList$ is empty. For the loop, an application name is picked from the $appList$, and it is then assigned to $usedApp$. The fifth step involves the calculation of $T_{elapsed}$ using the absolute difference between $T_{removed}$ and $T_{launched}$ if $usedApp$ is equal to $senderApp$. The loop is then escaped from. The last step is to check whether $T_{elapsed}$ is less than or equal to $T_{recation}$. If so, $TYPE_{phone-preferable}$ is assigned the value of $TRUE$.

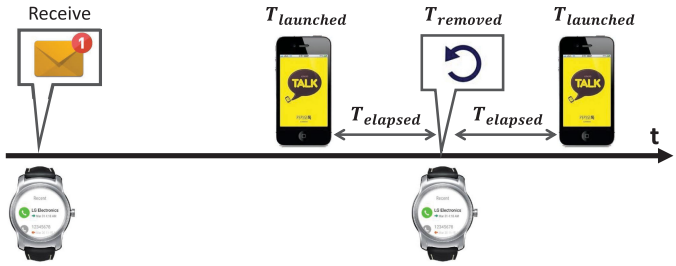


Fig. 5. Auto-labeling method for phone-preferable notification.

We hypothesize that several features are related to the notification types: name of the sender application, notification title, location, movement, time, and so on. *PASS* aims to prolong battery life by only performing watch-preferable notification delivery to the watch. To augment the synergy of energy saving, we used low-power sensors to recognize users' contexts when they react to notifications. More specifically, the movements of users were measured by using accelerometers of phones and watches. To prevent excessive energy consumption, we did not use GPS sensors that can provide fine-grained locations of users.

Algorithm 1. Phone-Preferable Notification Detection

Input: $T_{reaction}$
Output: $TYPE_{phone-preferable}$
$T_{reaction} \leftarrow certainTime$
$TYPE_{phone-preferable} \leftarrow$ FALSE
while *Is Removed Notification* **do**
  $T_{removed} \leftarrow$ current time
  $senderApp \leftarrow$ sender's app name
break
end
sleep($T_{reaction}$)
$appList \leftarrow$ recently used apps()
while *usedApp is remained in appList* **do**
  if *isEqualTo(senderApp,usedApp)* **then**
    $T_{launched} \leftarrow$ sender app's launched time
    $T_{elapsed} \leftarrow |T_{removed} - T_{launched}|$
    break
  end
end
if $T_{reaction} >= T_{elapsed}$ **then**
  $TYPE_{phone-preferable} \leftarrow$ TRUE
end

4 PASS System Implementation

To identify a phone-preferable notification, we need to collect real users' notifications and analyze them on a phone and a watch. To this end, we developed *PASS* that consists of a mobile application, a data collection server, and a data analyzer to gather users' data and build a prediction model. The details of each component are as follows:

Android Application. We developed a data collection application called *nCollector* that runs in the background to

TABLE 2
Participant Demographics

# of users	11 (8 male and 3 female)
Age	24, 25(3), 26, 27, 29, 30(2), 34, 35
Occupation	Students(7), Office Workers(3), Other(1)
Smartphone	Galaxy-s6(3), Galaxy-s4(2), Galaxy-note(2), Galaxy-a7, Galaxy-grand Max, Nexus5, Nexus5x
Smartwatch	LG-Urbane W150(11)

automatically label the notification type and context when a notification arrives. The *nCollector* also monitors connectivity between a phone and a watch to filter out pending notifications when the user does not use the phone and the watch simultaneously.

We can easily check the connectivity through an API called `CapabilityApi.getCapability()` when the phone is not connected to a watch. To prevent capturing trivial notifications, *nCollector* ignores the following notification types:

- Phone call notifications stem from *com.android.server.telecom*, *com.google.android.dialer*, and *com.android.phone*.
- Ongoing notification is defined by *FLAG_FOREGROUND_SERVICE*, *FLAG_NO_CLEAR*, and *FLAG_ONGOING_EVENT*.

nCollector[1] captures the notification type and contexts including movement, and so on. To decide the notification type, *nCollector* is built on a few APIs and Notification Listener Service [23]. To distinguish a phone-preferable notification, we implemented the labeling Algorithm 1 into *nCollector* using `UsageStatsManager` which is supported in API level 21 (Android 5.0). *nCollector* exploits the third-party library for computational social science [24], `SensorManager` and `SensorDataManager` to obtain the context information and store a large amount of data. In addition, it uses Android OS API, Activity Recognition [25] for monitoring a user's activity.

Data Server. A large amount of data, including labeled notification and contexts, is transmitted by formatting JSON from *nCollector*. To avoid a large amount of data transmission, data transfer occurs only once a day. If a user registers for participation, a server assigns a hash-ID to each user to manage data collection. The *Data server* receives JSON format data and then stores it as a file named by combining the received date and the hash-ID. We developed the *Data Server* using Java Sever Pages (JSP) and Bash shell script on Ubuntu 16.04.

Data Analyzer. We implemented *Data Analyzer* on Windows 7 using several R packages, including `jsonlite`, `doParallel`, `text-mining`, and `caret`. The `jsonlite` package supports a fast JSON parser and generator optimized for statistical data and web. The `doParallel` package provides a mechanism required to execute `foreach` loop in parallel. The `text-mining` package presents the methods for data import, corpus handling, preprocessing, metadata management and creation of term-document matrices. The `caret` package is a set of functions that attempt to streamline the process for creating predictive models. This package contains the following several functions for machine learning: data splitting, preprocessing, feature selection, model tuning using resampling, and variable importance estimation.

1. https://github.com/leejaymin/nCollector

TABLE 3
Overall Information in Collected Data

User	# of Installed Apps	# of Noti.	Act. Noti. rate
A	16	1,038	76% (788)
B	17	1,276	27% (354)
C	13	578	62% (358)
D	19	1,890	85% (1,608)
E	14	1,719	88% (1,510)
F	23	1,717	57% (978)
G	17	1,853	86% (1,602)
H	9	275	71% (196)
I	20	4,196	92% (3,888)
J	5	824	95% (782)
K	10	293	59% (173)

5 Dataset

In this section, we briefly describe how data is collected and what type of sensor data is targeted. Our research is focused on the users who can use the phone and the watch simultaneously. Given that smartwatches are not widely adopted among people [26], we purchased 11 watches (LG-Urbane W150) and distributed them for data collection. We hired participants who were willing to participate in the project without any monetary incentive.

Table 2 lists the participants that we recruited. Both male and female participants were chosen, with their age ranging between 24 and 35 years. They consist of seven students, three office works, and one other (job-seeker). In addition, we allowed them to use their own phone with a watch that we distributed as shown in Table 2.

The data was gathered between January and March 2016. The collection period per user was typically 3–6 weeks, with the average being 31 days. The reason why data collection period is different is that each user joined this study on a different date. Furthermore, we filtered out data when users do not use a phone and a watch at the same time.

We have collected 15,659 notifications as listed in Table 3. More specifically, the collected data consists of notification type and context features. Each user has received 275–4,196 notifications for 21–45 days. Then, we calculated a phone-preferable notification rate in each user. The phone-preferable notification rate of each user ranges from 27 to 95 percent, with the average being 73 percent. Given observation in the preliminary study, we reveal an opportunity to prolong battery life by deferring phone-preferable notifications. Therefore, the users who receive many notifications with higher phone-preferable rates have higher energy gains.

6 Phone-Preferable Notifications

In this section, we describe how we build a model to predict a phone-preferable notification based on our hypothesis: a phone-preferable notification depends on its information and the user's contexts. Subsequently, we analyze how long phone-preferable notifications are deferred in nature.

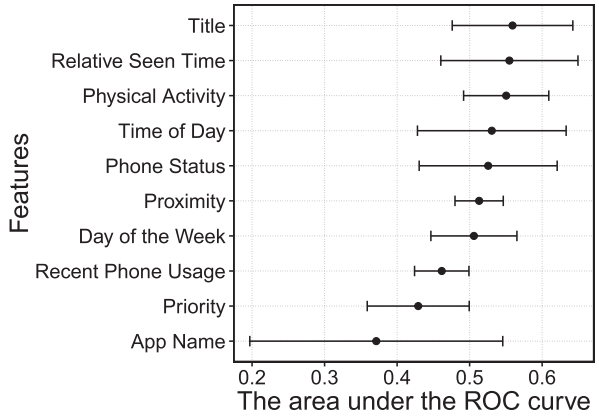


Fig. 6. Ranking of features (mean and SD) from the *nCollector* dataset.

6.1 Prediction Model

To build prediction models, we used three machine-learning algorithms: Naive Bayes, Random Forest, and Support Vector Machine. These models were trained by using the 10 features extracted from raw sensing data. The detailed description of these features is as follows.

Sender App Name. Intuitively, a phone-preferable notification is closely related to the application that triggers the notification. Package name of application can be used to retrieve the application's name from Android OS.

Title. The notification title provides a good indication of whether a user accesses the phone to reply to a message or read detailed contents. Basically, the title consists of the sender's name and a short description with regard to new information such as `Bob, How are you?` To improve matching rate among notifications' titles, we cleaned text data using the following pre-processing techniques: (1) transforming all texts to lower case, (2) removing numbers, (3) removing 174 stop-words predefined by *caret* package such as `I`, `being`, and `than`, (3) removing punctuation, and (4) removing additional white space.

Time of Day. Apparently, a phone-preferable notification is highly relevant to the time of day. Notification reception history of previous days at the same time could provide useful cues to infer the phone-preferable notification on a certain day.

Day of the Week. This feature is meaningful in phone-preferable notification inference, as users have different interaction patterns that rely on whether they are working. To extract the feature from a millisecond time-stamp, we categorized the time-stamp values into seven numbers corresponding to the day of the week.

Recent Phone Usage. Basically, a user prefers a phone over a watch due to the rich UI. Hence, recent phone usage leads to using a phone. To determine recent phone usage, *nCollector* monitors the latest screen ON time. When a notification arrives, the *nCollector* computes the time difference between the latest screen ON and notification arrival time. If the time difference is greater than one minute, we decided that recent phone usage is *TRUE* in that moment when a notification has arrived.

Proximity. The proximity sensor in the phone can detect the presence of objects including a human without any physical contact. When a notification arrives, this value is *TRUE* if a user is close to the phone. However, *FALSE* is the dominant value in this sensing data because the proximity sensor requires a human to be much close. Therefore, it is of little informative value.

Priority. Mobile application developers can assign a priority when they define notifications using API. However, many priorities are *zero*, which is the default value because assigning a priority is not mandatory.

Physical Activity. nCollector captures a user's activity from a phone. Our tool relies on Google's Activity Recognition API, which classifies it into eight classes: (1) In-Vehicle, (2) On-Bicycle, (3) On-Foot, (4) Running, (5) Still, (6) Tilting, (7) Unknown, and (8) Walking.

Phone Status. nCollector checks the screen status of a phone to know that a user interacts with the phone. To do that, *nCollector* used `isInteractive` API. This API returns *TRUE* if the device is in an interactive state in which the device basically is awake.

Relative Seen Time. This feature indicates when a notification is seen by a user. To achieve this, *nCollector* employs an unlock event of a phone. We assume that users can recognize all the notifications in the notification bar when the screen is turned ON. The *nCollector* records the arrival time and screen ON time. Then, it computes the time difference between the arrival time and the screen ON time by the user.

6.2 Feature Importance

To understand the value of the selected features, variable importance analysis was performed using the *caret* package in the R environment.

To show the importance of each feature, this package usually uses a receiver operating characteristic (ROC) curve, which is commonly used to examine the trade-off between the detection of true positives, while avoiding the false positives. The perfect classifier has a curve that passes through the point at a 100 percent true positive rate and a 0 percent false positive rate. This means that the classifier can correctly identify all the positives without any negative result. A more detailed estimation process for variable importance involves building each predictor using each feature. Then, we conducted an ROC curve analysis with regard to each predictor. Finally, the *trapezoidal rule* is used to compute the Area Under the ROC Curve (AUC). This area is used as the measure of variable importance. AUC ranges from 0.5 (for a classifier with no predictive value) to 1.0 (for a perfect classifier).

Fig. 6 shows the average and the standard deviation of AUC in each feature across all participants. As shown in Fig. 6, the points and both whiskers represent the mean of the AUC and the standard deviation in each feature.

The results show that the notification's title, relative seen time, and physical activity are the most important features. As stated previously, we know that a notification's title is intuitively relevant to the notification type. In the title feature, there is an opportunity to increase the variable importance. Many participants did not want to provide detailed notification information even though we only captured the title. To protect privacy, participants configured a setting of a messenger not to show detailed contents of a message; the title just shows `new message` under this configuration instead of `Bob, How are you?` Hence, several notifications contain similar text. To overcome this limitation, Lee et al. [15] suggested OS-level user attention management. Given that the

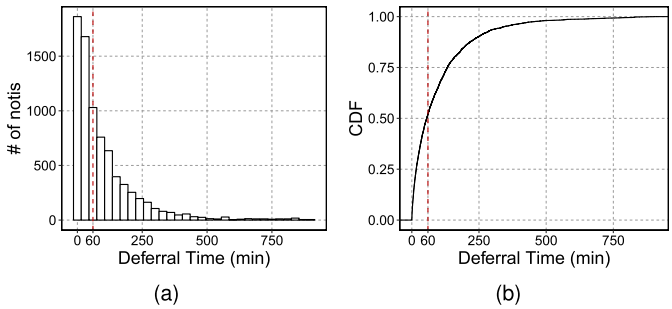


Fig. 7. Deferral time when watch-preferable and phone-preferable notifications are bundled together.

operating system can directly access all the information of every application without an API, the OS-level collection not only makes a high-quality dataset, but only reduces power consumption by data collection.

On the other hand, recent phone usage, priority, and application name are not relatively informative. In the `caret` package, we used `nearZeroVar` which diagnoses that the feature has very few unique values across entire samples. These features have a near zero variance for each user. However, we still use those features to build a classifier because those features can be used to alleviate the overfitting problem. To show which model is better, we constructed models with all features or related features. Using a recursive feature elimination (RFE) algorithm,[2] the personal *Naive Bayes* model was built using the most relevant feature corresponding to each user. Hence, the optimal feature selection ranges from one to all of the features.

Subsequently, we compared the performance of the prediction models trained on the RFE feature dataset with the prediction models trained on all the features. The experimental result indicates that the prediction model trained on all the features outperformed the RFE based model. This result is attributed to the overfitting problem (high variance). The RFE algorithm selected the optimal feature subset using only the training data. If the model is generated with only the optimal features, the overfitting problem might occur with regard to unseen data. For stable performance, we consider all the features.

6.3 Deferral Time of Phone-Preferable Notifications

To find a proper threshold of deferral time for phone-preferable notifications, we analyzed the collected notifications by calculating the difference between phone-preferable and watch-preferable notification occurrences. To remedy UX degradation, deferring phone-preferable notifications should be explored using real data. Piggybacking phone-preferable notifications on a watch-preferable notification could remedy negative effect of awareness. Hence, we analyze the collected notifications to calculate how long phone-preferable notifications are deferred when phone-preferable and watch-preferable notifications are bundled together. The calculation sequences for deferral times are as follows. First, the data was split to contain unique user and day. Second, we considered a specific time range (08:00 to 23:00) to prevent exaggerated deferral time. Fig. 7 shows the deferral time for piggybacking phone-preferable notifications across 11 users. As a result, approximately 50 percent of the deferral time for phone-preferable notifications are less than 1 hour. Therefore, *PASS* defers phone-preferable notifications up to 1 hour if they are not piggybacked in nature.

2. https://topepo.github.io/caret/recursive-feature-elimination.html

7 Evaluation

In this section, we evaluate *PASS* in terms of model accuracy, energy saving effectiveness, and *PASS* overhead.

7.1 Prediction Results

To verify the possibility of predicting phone-preferable notifications with sensed data, we built models using three different machine-learning algorithms: Naive Bayes (NB), Support Vector Machine (SVM), and Random Forest (RF). In addition, we built two types of models with individual and generic datasets. We evaluate each model using the *k-fold cross validation* approach with $k = 10$ in terms of the following metrics: 1) The sensitivity refers to the proportion of actual positives that are correctly identified. To calculate recall, the number of phone-preferable notifications that are correctly predicted is divided by the number of notifications that are actually phone-preferable. 2) The specificity refers to the proportion of actual negatives that are correctly identified. To measure specificity, the number of watch-preferable notifications that are correctly predicted is divided by the number of notifications that are actually watch-preferable. 3) The F-Score refers to a combination of *precision* and *recall*, which is calculated as $\frac{2 \times precision \times recall}{precision + recall}$.

As shown in Fig. 8, the personal (or individual) model outperformed the generic model. In light of this result, the individual data is more adequate for modeling. Our results demonstrate no significant difference in the performance among the three machine learning algorithms. Therefore, feature selection is more important for building an accurate model in the personal dataset.

The worst performance stems from incorrectly labeled data generated in an unobtrusive manner. If a user responded to a notification on the laptop, our profiler, *nCollector* did not capture any interaction. Therefore, this notification was categorized as a watch-preferable notification. For that reason, the phone-preferable notification can be less than actual.

7.2 Energy Saving Effectiveness

To verify the effectiveness of *PASS*, we used a theoretical model to calculate the net battery saving effects in both devices by combining the energy consumption of preliminary experiments, Naive Bayes model performance, and collected notification distribution. Among the three machine-learning algorithms, only the Naive Bayes model was used because of its low computational load and high specificity. In addition, given the difficulty of theoretically computing the energy benefit, we only considered that *PASS* does not forward watch-preferable notifications to a watch rather than deferring them until a certain time. The following constants were used to compute the phone battery benefit of PASS: $P_{savingBattery} = \frac{P_{savingEnergy}}{P_{batt} \times C_{Rate} \times 3600(s) \times V}$, where $P_{savingEnergy}$ is defined as $P_{gain} + P_{penalty}$, P_{gain} is $N_{phone-preferable} \times Sensitivity \times P_{notiGain}$, and $P_{penalty}$ contains $N_{watch-preferable} \times (1 - Specificity) \times P_{notiPenalty}$.

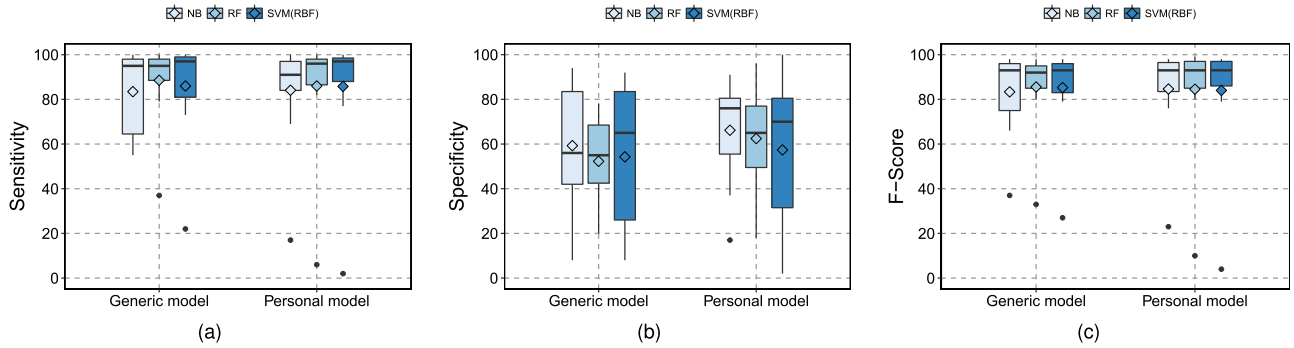


Fig. 8. Prediction results of *Naive Bayes* classifier by using 10 features extracted from *in-the-wild* 15,659 notifications: (a) sensitivity, (b) specificity, and (c) F-Score.

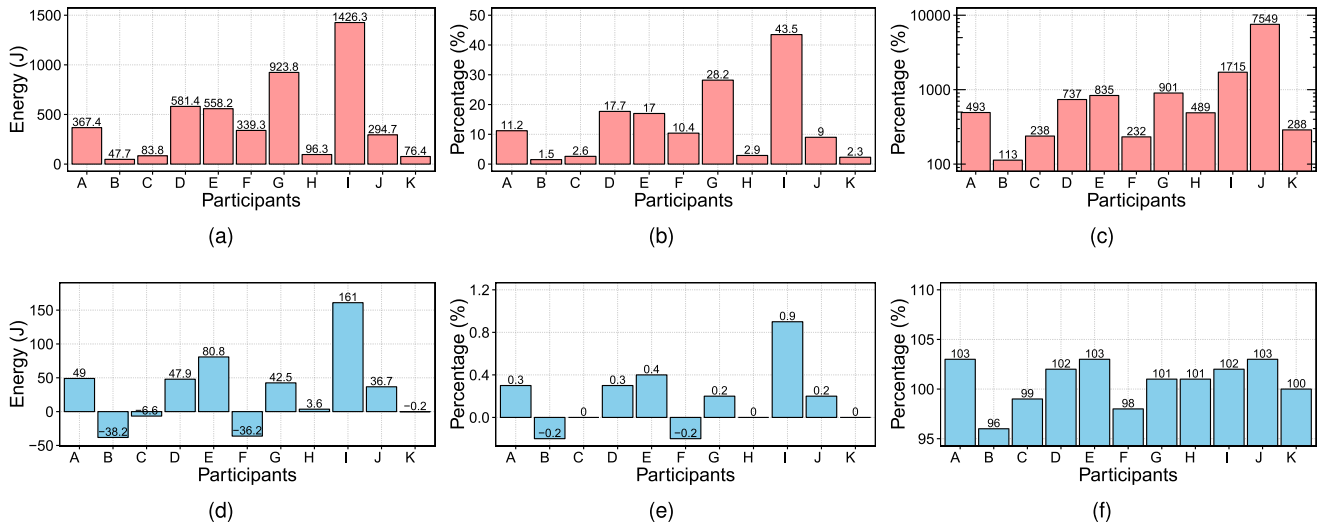


Fig. 9. Energy and battery saving effects by *PASS*: (a) daily energy saving of a watch, (b) daily battery saving of a watch, (c) efficiency ratio in a watch, (d) daily energy saving of a phone, (e) daily battery saving of a phone, and (f) efficiency ratio of a phone.

Here, P_{batt} is nominal 2300mAh, being equal to that of Nexus 5. C_{Rate} is the available capacity rate to convert from the nominal capacity to actual capacity. A lithium-ion battery can usually be discharged by up to 60 percent due to recharging and safety [27]. V is the operating voltage, and it is equal to 3.7V. $N_{phone-preferable}$ is the number of daily phone-preferable notifications for each user (listed in Table 3). $P_{notiGain}(2.22J)$ is the energy gain of a phone from preliminary experiments of phone-preferable notifications. $P_{notiGain}$ is attributed to saving notification delivery energy (CPU and BLE) to a watch. $P_{notiPenalty}(-15.52J)$ is an energy penalty of a phone computed from preliminary experiment of watch-preferable notifications. $P_{notiPenalty}$ is responsible for *False Positive* that leads to the redundant usage of phone. Owing to incorrect predictions, a watch-preferable notification will not be relayed to a watch. We can compute this penalty by subtracting *phone only* from *phone with watch*.

The following constants were used to compute the phone battery benefit of PASS: $W_{savingBattery} = \frac{W_{savingEnergy}}{W_{batt} \times C_{Rate} \times 3600(s) \times V}$, where $W_{savingEnergy}$ is defined as $W_{gain} + W_{penalty}$, W_{gain} is $N_{phone-preferable} \times Sensitivity \times W_{notiGain}$, and $W_{penalty}$ contains $N_{watch-preferable} \times (1 - Specificity) \times W_{notiPenalty}$. Here, W_{batt} is nominal 410mAh, being equal to that of LG urbane W150. C_{Rate} is 0.6 as mentioned above. V is the operating voltage, and it is equal to 3.7V. $N_{watch-preferable}$ is the number of daily watch-preferable notifications for each user (listed in Table 3). $W_{notiGain}(12.07J)$ is the energy gain of a watch from the preliminary experiments of phone-preferable notifications. For $W_{notiGain}$, the energy can be saved by not using a watch. $W_{notiPenalty}(9.88J)$ is an energy penalty of a watch computed from the preliminary experiments of watch-preferable notifications. For $W_{notiPenalty}$, the penalty is still positive because eventually predictions that are wrong do not involve watch use.

Figs. 9a, 9b, 9d, and 9e show the theoretical daily effects of *PASS* in terms of energy and battery. Each term is computed by combining each user's data and the aforementioned equations. To know the daily effect, we assume that a user can recharge the battery at least once. $N_{phone-preferable}$ and $N_{watch-preferable}$ have the daily number of notifications.

Battery gains of the watch ranged from 1.5 to 43.5 percent, with the average being 13.3 percent. In case of the phone, the gains were between -0.2 and 0.9 percent, with the average being 0.2 percent. The battery gain of the watch was much larger than the phone. There are a few reasons for this. First, a watch is equipped with a battery smaller than that of a phone. Therefore, a slight improvement leads to greater battery saving effect. Second, the watch use frequency is reduced by not sending phone-preferable notifications to the

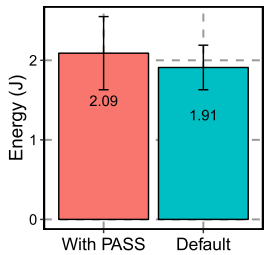


Fig. 10. Overhead of *PASS*: sensing and prediction costs.

watch. This occurs even when a watch-preferable notification is not relayed to a watch due to a False Positive.

In contrast to the watch, the battery effect of the phone is relatively small, or even negative in the case of four users. This is because the energy saving stems from notification delivery cost to a watch. In addition, False Positive penalty is seven times larger than True Positive gain. In light of the saving effect of the watch, we claim that the phone battery saving is meaningful.

The variation of the battery-saving effects is attributed to different numbers of phone-preferable notifications and model accuracy for each user. User I achieved the greatest battery effect with a watch battery gain of 43.5 percent and a phone battery gain of 0.9 percent. This was because user I showed a total 3,888 phone-preferable notifications and the sensitivity of 96.1 percent. On the other hand, the model of user B showed the worst effect. The data set of user B contains the lowest phone-preferable notification rate (27 percent). In addition, this model achieved the lowest sensitivity of 16.1 percent. These imply that there is a high penalty owing to a False Positive and a low gain owing to a True Positive. These results led to the lowest watch battery gain of 1.5 percent and the negative phone battery gain of -0.2 percent. The reasons why user B shows poor results are rooted in the auto-labeling method. In the post-interview, we discovered that user B prefers a computer to a handheld device to check notifications. This usage pattern generates misclassified notification labels because *nCollector* can not capture an action when a user attends to another device. The misclassified labels result in not only poor accuracy of the model, but also a low phone-preferable notification rate. From the optimal prediction model of user B, the watch battery gain of 4 percent and the phone battery gain of 0.1 percent can be achieved.

Figs. 9c and 9f shows the theoretical energy efficiency ratio owing to *PASS*. The efficiency ratio of the watch ranges between 113 and 7,549 percent. For the efficiency of the phone, eight users show positive ratios but the others represent negative ratios. First, the energy efficiency ratio of a watch is computed by using the following constants and equations: $W_{efficiencyRatio} = \frac{W_{defaultEnergy}}{W_{defaultEnergy} - W_{savingEnergy}}$, where $W_{defaultEnergy}$ is defined as $(N_{watch-preferable} \times W_{eWatch-preferable}) + (N_{phone-preferable} \times W_{ePhone-preferable})$. Here, $W_{eWatch-preferable}$ $(9.88J)$ is the average watch energy consumption across five watch-preferable scenarios, being equal to $W_{notiPenalty}$. $W_{ePhone-preferable}(12.07J)$ is an average watch energy consumption across four phone-preferable scenarios; it is equal to $W_{notiGain}$. Second, the energy efficiency ratio of a phone is computed by using the following constants and equations: $P_{efficiencyRatio} = \frac{P_{defaultEnergy}}{P_{defaultEnergy} - P_{savingEnergy}}$, where $P_{defaultEnergy} = (N_{watch-preferable} \times P_{eWatch-preferable}) + (N_{phone-preferable} \times P_{ePhone-preferable})$. Here, $P_{eWatch-preferable}(6.32J)$ is the average phone energy consumption across five watch-preferable scenarios. $P_{ePhone-preferable}$ $(59.17J)$ is the average phone energy consumption across four phone-preferable scenarios.

7.3 PASS Overhead

We investigated prediction overhead with both devices before and after applying *PASS*. The energy overhead consists of two parts: sensing and prediction. First, we employed low-power sensors to predict whether a notification is phone-preferable. Second, prediction overhead was responsible for the machine-learning model type. In the actual deployment, we used a Naive Bayes model due to the trade-off between accuracy and computational cost. We developed the NB classifier on WEKA for Android.[3] All overheads are mostly dependent upon CPU usage and several low power sensors such as accelerometers and proximity. To measure the overhead, we compared the difference in power consumption while triggering a notification with and without *PASS*. Regarding a notification, we repeatedly measured power consumption ten times. As shown in Fig. 10, *PASS* consumed 8.6 percent more energy than the default on average. In light of the sensing and prediction costs, we claim that *PASS* would show a small overhead, as it is not activated at all if a notification has not arrived.

7.4 Field Study

Directly manipulating notification delivery between a phone and a watch with prediction models is difficult because Android-Wear is not open-source. Instead, in order to investigate the usability, we conducted a field trial by installing a mobile application that captures ground-truth labels on which device is used for reaction to notifications by the participants. For this field trial, we recruited five participants from the same people who participated in the data collection study. Subsequently, we asked them to perform self-reports about which device they use to respond to notifications. This kind of experience sampling has been widely used in the human-computer interaction (HCI) research community to gather insights about human decision making [28].

To implement a mobile application that supports self-reporting and sensing data for inference, we combined the nCollector, Snotify [29], and Google Firebase.[4] The modified Snotify can not only store all the received notifications but can also capture all sensing data. As shown in Fig. 11, participants can choose three answers corresponding to their actions. In this survey, *Watch* and *Phone* indicate that a notification is watch- and phone-preferable, respectively. We also added *Computer* since our participants mentioned that it can be used for notification checking as well as their phones. This category is considered as "not-watch-preferable" in that further actions are required in other devices as in phone-preferable notifications. A total of 2,371 responses were collected from the participants over a period of approximately 2 months (from Oct. to Dec. 2017). As shown

3. https://github.com/andrecamara/weka-android
4. https://firebase.google.com/

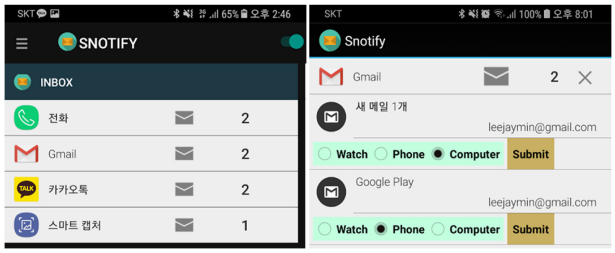


Fig. 11. Mobile app to collect users' feedback about the device they use to respond to notifications.

in Fig. 12, participants provided information about the device they used to respond to notifications.

To investigate users' thoughts about delay tolerance, we conducted a follow-up study as follows. We extracted not-watch-preferable notifications (phone and computer) for this study, because watch-preferable notifications could be forwarded to a watch without any delay. We then asked the participants to respond to seven options corresponding to further actions for not-watch-preferable notifications arrived in their phones. These further actions are categorized into the following two parts. Not-watch-preferable notifications are (i) simply ignored without delivery to the watch, and (ii) promptly relayed to the watch or deferred by a certain time. By referring to prior work [30], certain time options for deferring notifications to the smartwatches were designed as a post-survey form: prompt, 5 minutes, 30 minutes, 1 hour, 3 hours, 8 hours, 1 Day, or 3 Days. Finally, users answered questions related to the type of not-watch-preferable notifications they were allowed to either defer, or not forward to the watch. Note that even with this deferral, users can check all the notifications using their phones anytime they want.

As shown in Fig. 13, all responses were grouped into 11 categories pre-defined as in the prior study [30]. A total of 1,410 out of the 2,371 responses are relevant to a computer and a phone. As a consequence, 71.9 percent of not-watch-preferable notifications do not need to be delivered from the phones immediately. It is worth noting that further actions of phone-preferable notifications are different based on application categories. In the system and reminder categories, users responded that most phone-preferable notifications are not necessary to be received on the watches promptly. On the other hand, in social and phone categories, phone-preferable notifications tend to be received on watches for instant awareness. Therefore, using the application name to distinguish application category is a crucial feature to train a personalized model. Furthermore, to prevent usability degradation, we can maintain a (personalized) *whitelist* that allows a phone-preferable notification to be forwarded to a watch.

As shown earlier in Section 2.3, 50 percent of the phone-preferable notifications can be delivered to a watch by piggybacking on watch-preferable notifications. Furthermore, we set the maximum duration of deferral (say one hour). In practice, users are likely to frequently check their phones as part of their regular phone usage, and the actual delay that users experience would be much shorter [31], [32], [33]. For these reasons, we claim that usability does not notably diminish.

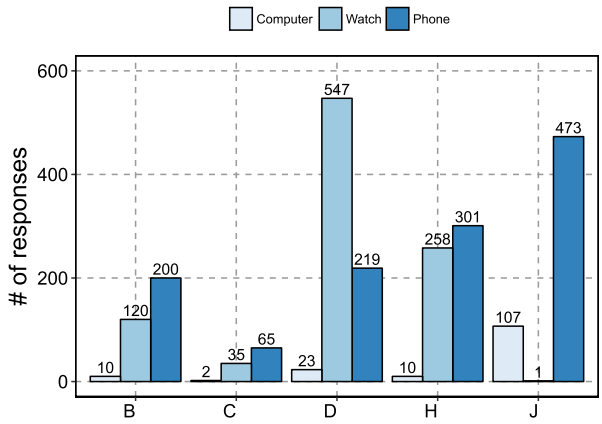


Fig. 12. Number of responses collected from real users.

8 Discussion

Energy Saving Impacts in Multi-Device Environments. Due to the increasing popularity of wearable devices, smartphones will be connected with them more and more in the future (i.e., more devices mean more messages to deliver and manage). However, the current notification delivery system relays all the notifications to connected devices. From our preliminary study, we discovered the burden of the phone battery for relaying notifications to connected devices. We strongly believe that this burden would be exacerbated when the phone is connected with more and more devices (e.g., IoT and wearable devices). To know the energy burden of increasing the number of paired devices, we conducted an additional experiment in which a smartphone is paired with two smartwatches. As a result, the power consumption of forwarding a notification from the smartphone has increased from 1199.83 mW to 1229.70 mW (9.7 percent more energy) if we pair with two watches. This result clearly shows that increasing the number of IoT and wearable devices connected to a smartphone will increasingly require more energy consumption. With the advent of IoTs and wearable devices, there will be an increasing number of wirelessly connected personal devices, and thus, our proposed optimization techniques will be more important in this computing environment.

Model Accuracy. For some users, our model accuracy shows poor prediction results. All the limitations of model accuracy stem from two reasons: auto-labeling and privacy intrusion. First, auto-labeling leads to generating misclassified labels when a user attends another device that we do not consider in the study. Therefore, we need to capture all responses on any devices a user focuses on. Second, application-level techniques can not capture detailed context such as the titles of notifications because many users hide notification content by configuration. Since the operating system has access to all information of every application, our approach should be a system service. Such a service can monitor a user's response for all the notification categories in different contexts. Therefore, with an OS-level service, we can collect a richer dataset with which to build a model.

Usability Consideration. We are aware that *PASS* may affect the nature of user experience on the watch. We claim that usability degradation on the watch is less critical for the following reasons. First of all, our results showed that most phone-preferable notifications (71.9 percent) are not

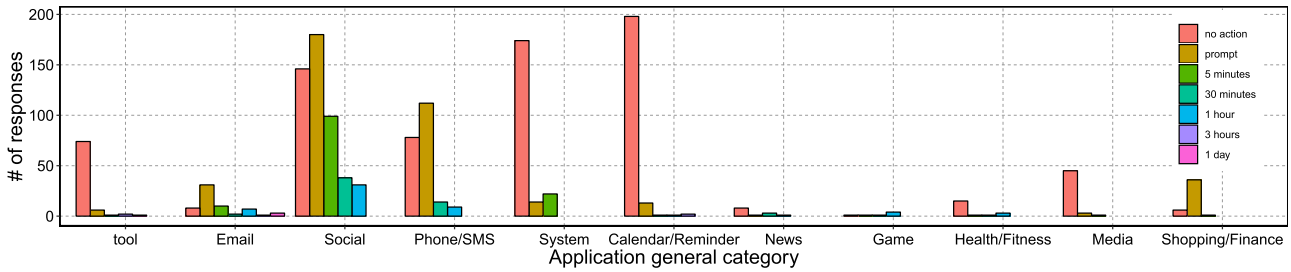


Fig. 13. Distribution of responses on phone-preferable notifications for five users and all categories.

necessary to be forwarded on the phones immediately. According to a recent survey [31], [32], [33], users check the phone approximately 150 times every day, which means that even with deferrals, perceived delay would be less significant. Second, users are less dependent on a watch because it is often regarded a secondary device. A recent work [4] investigated the level of discomfort if devices ran out of battery. According to the questionnaire survey, users answers for the watch were *neutral* 33 percent of the time. On the other hand, users responded with *very uncomfortable* 46 percent of the time for the phone. This answer was attributed to the phone being a key device for online connectivity and communication with other people. Third, more than half of notifications were forwarded to a watch when a phone was unlocked. As shown in Fig. 14, the phone screen ON rate ranges 41.3–81.8 percent, for an average of 57.4 percent, across all the participants. With the `isInteractive` API, the number of notifications is recognized when the screen of the phone is ON. Intuitively, the received notifications when a user interacts with a phone need not be forwarded to the watch. However, the current mobile systems relay all notifications to a watch regardless of its status on the phone. From this result, we anticipate that almost half of the notifications could be recognized without watch assistance in a timely manner. Last, we allow users to configure a whitelist. Note that some phone-preferable notifications may require urgent user interactions such as incoming calls. This kind of exception can be easily incorporated into our labeling algorithm. We can provide a user interface that allows users to input some rules to override our mechanism (e.g., app names, keywords, time-period).

9 Related Work

Recently, research on desktop interruptibility has been extended to mobile systems due to the popularity of mobile devices. Mobile interruptibility research has exploited a variety of sensors in smartphones to infer opportune moments.

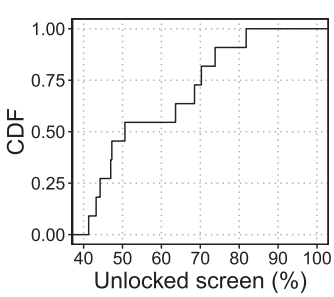


Fig. 14. Ratio of UNLOCKED screen when a notification is arrived.

Fischer et al. [12] investigated whether the breakpoints between the user's interaction with the mobile phone correlate with a right moment to deliver notifications. However, they asked users to provide feedback about their context. Such an experience sampling study leads to the extra burden of identifying the user's thoughts. As more advanced systems, unobtrusive approaches that do not require manual labels from users were presented [13], [14], [16].

To build an interruptibility model, these approaches unobtrusively monitored a variation of contexts and system configurations without user involvement and questionnaires. Moreover, some researchers built a smart notification manager based on trained models [29], [34]. A few works analyzed and managed smartwatch notifications [6], [7], [35]. Visuri et al. [6] conducted a quantitative analysis by collecting a large number of notifications (2.8 million); they pointed out that smartwatches have a limitation regarding their input capabilities, making these devices more appropriate for consuming content rather than creating new content. Cecchinato et al. [7] performed a qualitative study by interviewing existing users who use their smartwatches on a daily basis. Further, they reported that users want to optimize notification settings based on sender, topic, or even location. In those points, our work is in line with the findings of Visuri et al. [6] and Cecchinato et al. [7]. In their study of multi-device environments, Weber et al. [35] analyzed the users' device preference to respond to notifications. They show that users prefer to receive notifications on specific devices depending on their situation.

On the other hand, there are energy-optimization works of smartphones in diverse aspects such as sensing, battery charging, and battery value [36], [37], [38]. In addition, researchers have proposed mobile systems to extract human behavior patterns from smartphones [39], [40], [41]. Those works covered a variety of interaction patterns with smartphones; unlike those works, we shed light on inefficiency of notification forwarding by narrowing down notification interactions. In addition, we focus on notification optimization due to the limited space.

Several researchers have studied notification optimization on a smartphone. A notification contains messages and alarms that make devices wake-up in a certain timeframe. Through batch processing, AlarmScope [42] reduced the number of non-critical alarms that are deferrable. However, AlarmScope only targeted a smartphone and alarm induced wake-ups rather than multiple devices and a notification. To identify non-critical alarms, three static criteria are employed. Considering external notification, TailEnder [43] and Xu et al. [44] optimized tail energy consumption in wireless communication. Moreover, to reduce the amount of the wasted tail time

energy, Personalized Diapause (PD) [45] accurately predicted whether a subsequent transmission will occur in the tail time by personalized high-level network usage patterns. However, such works have studied energy efficiency when the notification arrived from outside. They have not considered energy efficiency between a smartphone and a smartwatch. To the best of our knowledge, the proposed *PASS* is the first work to consider energy consumption between a smartphone and a smartwatch in terms of notification delivery.

10 Conclusion

We identified a novel opportunity of reducing redundant user interactions with *phone-preferable notifications* to save the battery of a watch and a phone. We proposed a power-aware notification delivery mechanism, called *PASS* that infers a phone-preferable notification using machine learning models. With *nCollector*, 15,659 notifications are collected from 11 watch users for around 31 days. This dataset was used to investigate phone-preferable notifications. We extracted 10 features relevant to a phone-preferable notification and built machine learning models. We evaluated *PASS* in terms of prediction accuracy, battery gains in both devices, and usability. The personalized classifier outperformed the generic classifier. Across all users, the personalized models achieved an average F-score of 83 – 86 percent. Our model-based evaluation showed that *PASS* improved the daily battery life of the watch and the phone by up to 43.5 and 0.9 percent, respectively. In addition, we deployed our notification manager, which works on a personal classifier in a real user. Subsequently, we interviewed users with regard to their complaints users felt; they did not report any noticeable discomfort by missing out on important notifications.

In future work, we plan to extend *PASS* to support several devices such as tablets and laptops and to implement OS or cloud-level notification control to span across multiple devices.

Acknowledgments

This research was supported by the National Research Foundation (NRF) of Korea funded by the Ministry of Education (NRF-2017R1D1A1B03034705) and Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2017M3C4A7065960).

References

[1] R. Rawassizadeh, B. A. Price, and M. Petre, "Wearables: Has the age of smartwatches finally arrived?" *Commun. ACM*, vol. 58, no. 1, pp. 45–47, Dec. 2014. [Online]. Available: http://doi.acm.org/10.1145/2629633

[2] X. A. Chen, T. Grossman, D. J. Wigdor, and G. Fitzmaurice, "Duet: Exploring joint interactions on a smart phone and a smart watch," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2014, pp. 159–168. [Online]. Available: http://doi.acm.org/10.1145/2556288.2556955

[3] J. Huang, A. Badam, R. Chandra, and E. B. Nightingale, "WearDrive: Fast and energy-efficient storage for wearables," in *Proc. USENIX Annu. Tech. Conf.*, 2015, pp. 613–625.

[4] 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 *Proc. IEEE Int. Symp. Wearable Comput.*, 2015, pp. 11–18.

[5] A. S. Shirazi and N. Henze, "Assessment of notifications on smartwatches," in *Proc. 17th Int. Conf. Human-Comput. Interaction Mobile Devices Serv. Adjunct*, 2015, pp. 1111–1116. [Online]. Available: http://doi.acm.org/10.1145/2786567.2794338

[6] A. Visuri, Z. Sarsenbayeva, N. van Berkel, J. Goncalves, R. Rawassizadeh, V. Kostakos, and D. Ferreira, "Quantifying sources and types of smartwatch usage sessions," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2017, pp. 3569–3581. [Online]. Available: http://doi.acm.org/10.1145/3025453.3025817

[7] M. E. Cecchinato, A. L. Cox, and J. Bird, "Always on(line)?: User experience of smartwatches and their role within multi-device ecologies," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2017, pp. 3557–3568. [Online]. Available: http://doi.acm.org/10.1145/3025453.3025538

[8] Google, "Notification settings," 2019. [Online]. Available: https://support.google.com/nexus/answer/6111294?hl=en

[9] L. Terveen, J. McMackin, B. Amento, and W. Hill, "Specifying preferences based on user history," in *Proc. ACM Conf. Human Factors Comput. Syst.*, 2002, pp. 315–322.

[10] S. Rosenthal, A. Dey, and M. Veloso, "Using decision-theoretic experience sampling to build personalized mobile phone interruption models," in *Proc. 9th Int. Conf. Pervasive Comput.*, 2011, vol. 6696, pp. 170–187.

[11] J. E. Fischer, N. Yee, V. Bellotti, N. Good, S. Benford, and C. Greenhalgh, "Effects of content and time of delivery on receptivity to mobile interruptions," in *Proc. ACM Int. Conf. Human Comput. Interaction Mobile Devices Serv.*, 2010, pp. 103–112.

[12] J. E. Fischer, C. Greenhalgh, and S. Benford, "Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications," in *Proc. ACM Int. Conf. Human Comput. Interaction Mobile Devices Serv.*, 2011, pp. 181–190.

[13] M. Pielot, R. D. Oliveira, H. Kwak, and N. Oliver, "Didn't you see my message?: Predicting attentiveness to mobile instant messages," in *Proc. ACM Int. Conf. Human Factors Comput. Syst.*, 2014, pp. 3319–3328.

[14] V. Pejovic and M. Musolesi, "InterruptMe: Designing intelligent prompting mechanisms for pervasive applications," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2014, pp. 897–908.

[15] K. Lee, J. Flinn, and B. Noble, "The case for operating system management of user attention," in *Proc. ACM Int. Workshop Mobile Comput. Syst. Appl.*, 2015, pp. 111–116.

[16] A. Mehrotra, M. Musolesi, R. Hendley, and V. Pejovic, "Designing content-driven intelligent notification mechanisms for mobile applications," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2015, pp. 813–824.

[17] Google, "Android wear SDK," 2019. [Online]. Available: http://developer.android.com/intl/ko/wear/index.html

[18] X. Liu, T. Chen, F. Qian, Z. Guo, F. X. Lin, X. Wang, and K. Chen, "Characterizing smartwatch usage in the wild," in *Proc. 15th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2017, pp. 385–398.

[19] M. Pielot, K. Church, and R. D. Oliveira, "An in-situ study of mobile phone notifications," in *Proc. ACM Int. Conf. Human-Comput. Interaction Mobile Devices Serv.*, 2014, pp. 233–242.

[20] M. S. Inc., "Power monitor overview," 2019. [Online]. Available: https://www.msoon.com/LabEquipment/PowerMonitor

[21] R. Liu and F. X. Lin, "Understanding the characteristics of android wear OS," in *Proc. 14th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2016, pp. 151–164.

[22] X. Liu and F. Qian, "Measuring and optimizing android smartwatch energy consumption: Poster," in *Proc. 22nd Annu. Int. Conf. Mobile Comput. Netw.*, 2016, pp. 421–423.

[23] Google, "Android notification listener service API," 2019. [Online]. Available: http://developer.android.com/intl/ko/reference/android/service/notifica tion/NotificationListenerService.html

[24] N. Lathia, K. Rachuri, C. Mascolo, and G. Roussos, "Open source smartphone libraries for computational social science," in *Proc. ACM Int. Conf. Pervasive Ubiquitous Comput. Adjunct Publication*, 2013, pp. 911–920.

[25] Google, "Activity recognition API," 2019. [Online]. Available: https://developers.google.com/location-context/activity-recognition/

[26] A. Lazar, C. Koehler, J. Tanenbaum, and D. H. Nguyen, "Why we use and abandon smart devices," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2015, pp. 635–646.

[27] Y. Barsukov and J. Qian, *Battery Power Management for Portable Devices*. Norwood, MA, USA: Artech House, 2013.

[28] N. V. Berkel, D. Ferreira, and V. Kostakos, "The experience sampling method on mobile devices," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 1–40, 2017.

[29] S. Pradhan, L. Qiu, A. Parate, and K.-H. Kim, "Understanding and managing notifications," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.
[30] D. Weber, A. Voit, J. Auda, S. Schneegass, and N. Henze, "Snooze!: Investigating the user-defined deferral of mobile notifications," in *Proc. 20th Int. Conf. Human-Comput. Interaction Mobile Devices Serv.*, 2018, pp. 2:1–2:13. [Online]. Available: http://doi.acm.org/10.1145/3229434.3229436
[31] O. Turel and A. Serenko, "Is mobile email addiction overlooked?" *Commun. ACM*, vol. 53, no. 5, pp. 41–43, May 2010. [Online]. Available: http://doi.acm.org/10.1145/1735223.1735237
[32] M. Meeker and L. Wu, "Annual Internet Trends report," 2013. [Online]. Available: http://www.kpcb.com/blog/2013-internet-trends
[33] tecmark, "Tecmark survey finds average user picks up their smartphone 221 times a day," 2014. [Online]. Available: http://www.tecmark.co.uk/smartphone-usage-data-uk-2014/
[34] A. Mehrotra, R. Hendley, and M. Musolesi, "PrefMiner: Mining user's preferences for intelligent mobile notification management," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2016, pp. 1223–1234. [Online]. Available: http://doi.acm.org/10.1145/2971648.2971747
[35] D. Weber, A. Voit, P. Kratzer, and N. Henze, "In-situ investigation of notifications in multi-device environments," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2016, pp. 1259–1264. [Online]. Available: http://doi.acm.org/10.1145/2971648.2971732
[36] S. Hosio, D. Ferreira, J. Goncalves, N. van Berkel, C. Luo, M. Ahmed, H. Flores, and V. Kostakos, "Monetary assessment of battery life on smartphones," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2016, pp. 1869–1880.
[37] D. Ferreira, A. K. Dey, and V. Kostakos, "Understanding human-smartphone concerns: A study of battery life," in *Proc. Int. Conf. Pervasive Comput.*, 2011, pp. 19–33.
[38] R. Rawassizadeh, M. Tomitsch, M. Nourizadeh, E. Momeni, A. Peery, L. Ulanova, and M. Pazzani, "Energy-efficient integration of continuous context sensing and prediction into smartwatches," *Sensors*, vol. 15, no. 9, pp. 22 616–22 645, 2015.
[39] R. Rawassizadeh, E. Momeni, C. Dobbins, J. Gharibshah, and M. Pazzani, "Scalable daily human behavioral pattern mining from multivariate temporal data," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 11, pp. 3098–3112, Nov. 2016.
[40] V. Srinivasan, S. Moghaddam, A. Mukherji, K. K. Rachuri, C. Xu, and E. M. Tapia, "MobileMiner: Mining your frequent patterns on your phone," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2014, pp. 389–400.
[41] S. Nath, "ACE: Exploiting correlation for energy-efficient and continuous context sensing," in *Proc. ACM Int. Conf. Mobile Syst. Appl. Serv.*, 2012, pp. 29–42.
[42] S. Park, D. Kim, and H. Cha, "Reducing energy consumption of alarm-induced wake-ups on android smartphones," in *Proc. ACM Int. Workshop Mobile Comput. Syst. Appl.*, 2015, pp. 33–38.
[43] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proc. ACM Int. Conf. Internet Meas. Conf.*, 2009, pp. 280–293.
[44] F. Xu, Y. Liu, T. Moscibroda, R. Chandra, L. Jin, Y. Zhang, and Q. Li, "Optimizing background email sync on smartphones," in *Proc. ACM Int. Conf. Mobile Syst. Appl. Serv.*, 2013, pp. 55–68.
[45] Y. Kim, B. Jeon, and J. Kim, "A personalized network activity-aware approach to reducing radio energy consumption of smartphones," *IEEE Trans. Mobile Comput.*, vol. 15, no. 3, pp. 544–557, Mar. 2016.

Jemin Lee received the BS and PhD degrees in computer science and engineering from Chungnam National University, in 2011 and 2017, respectively. He is a researcher at the Electronics and Communications Research Institute (ETRI). He was a post-doctoral researcher at the Korea Advanced Institute of Science and Technology (KAIST), in 2017–2018. His research interests include energy-aware mobile computing, large-scale GUI testing for Android applications, and applied machine learning. He is a member of the IEEE and ACM.

Uichin Lee received the BS degree in computer engineering from Chonbuk National University, in 2001, the MS degree in computer science from KAIST, in 2003, and the PhD degree in computer science from UCLA, in 2008. He is an associate professor with the Graduate School of Knowledge Service Engineering, Korea Advanced Institute of Science and Technology (KAIST). He continued his studies at UCLA as a post-doctoral research scientist (2008-2009) and then worked for Alcatel-Lucent Bell Labs as a member of technical staff till 2010. His research interests include social computing systems and mobile/pervasive computing.

Hyungshin Kim received the BS degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), in 1990, the MSc degree in satellite communication engineering from the University of Surrey, United Kingdom, in 1990, and the PhD degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), in 2003. He joined Chungnam National University, in 2004 and he is a professor with the Department of Computer Science and Engineering. His research interests include avionics system, energy-aware computing, and embedded system software. He is a member of IEEE and ACM.