Hindawi
Mobile Information Systems
Volume 2018, Article ID 7689549, 9 pages
https://doi.org/10.1155/2018/7689549

Research Article

Reducing Smartwatch Users' Distraction with Convolutional Neural Network

Jemin Lee,[1] **Jinse Kwon**,[2] **and Hyungshin Kim**[2]

[1]*Industrial Engineering and Management Research Institute, KAIST, Daejeon, Republic of Korea*
[2]*Department of Computer Science and Engineering, Chungnam National University, Daejeon, Republic of Korea*

Correspondence should be addressed to Hyungshin Kim; hyungshin@cnu.ac.kr

Received 1 June 2017; Revised 24 December 2017; Accepted 29 January 2018; Published 15 March 2018

Academic Editor: Federica Cena

Smartwatches provide a useful feature whereby users can be directly aware of incoming notifications by vibration. However, such prompt awareness causes high distractions to users. To remedy the distraction problem, we propose an intelligent notification management for smartwatch users. The goal of our management system is not only to reduce the annoying notifications but also to provide the important notifications that users will swiftly react to. To analyze how to respond to the notifications daily, we have collected 20,353 in-the-wild notifications. Subsequently, we trained the convolutional neural network models to classify important notifications according to the users' contexts. Finally, the proposed management allows important notifications to be forwarded to a smartwatch. As experiment results show, the proposed method can reduce the number of unwanted notifications on smartwatches by up to 81%.

1. Introduction

Smartwatches have become one of the first popular wearables, with the launch of high-quality flagship products by major global companies. According to recent surveys [1–3], one of the key features of smartwatches is user notification about various events, such as new messages or software updates. Several researchers have already investigated interruptions caused by notifications [4–6]. Based on the research results, mobile notifications at an inopportune moment at which users concentrate on their tasks lead to disruptive effects.

In the current notification delivery system, all notifications can be shown on all connected mobile devices simultaneously. When a user is wearing a smartwatch, they are severely distracted by notifications delivered from a smartphone because the smartwatch is a wrist-worn device [2]. To block unwanted notifications, users would manually change the settings from time to time. The burden of management causes the abandonment of smartwatches [2]. Therefore, the notification delivery system should notify only important notifications that need to be swiftly reacted considering the sender, topic, or location.

To lessen the distractive effects, many researchers have proposed several approaches that precisely predict the opportune moment to send notifications to users with machine learning techniques [7–9]. However, attention management is still lacking in emerging situations where users carry multiple mobile devices. Most previous research on notifications and interruptions only focused on a single device (e.g., a smartphone).

In this paper, we present an intelligent notification delivery system for a smartphone and a smartwatch. We extend our pilot work that filtered unwanted notifications using deep neural networks (DNNs). Unlike the pilot work, we not only collect more data from more users but also widely compare five machine learning algorithms with individual data and generic data. To train a model, we collect real users' responses and context data. Based on the users' responses and the previous work's criteria [8], we unobtrusively label notifications for a ground-truth value. The previous work [8] has found that users handle the notification within a certain time and launch related applications only if the arrival time is an opportune moment. We bring this assumption into our work to programmatically label important notifications without

any questionnaire. To infer the notification labels, we combine the user's response time (time difference between arriving and removing) and app launch (indicating whether an application is launched).

To train the convolutional neural network (CNN) models, we collected 20,352 *in-the-wild* notifications from 13 users for approximately 5 weeks. We performed an analysis on the collected data to extract features. Subsequently, we transformed the collected data to an $\mathbb{R}^{2\times5}$ matrix according to the correlation of features. To obtain better models, we compared the CNN (sparse connection) to the DNN (dense connection). As a result, the CNN outperforms other predictions models with a slightly higher *F*-score value (mean 88%) due to the correlation of features and the transfer learning effect. In addition, we compare the *F*-score of the prediction models trained on individual data with a generic prediction model trained on all users' data. Overall, the personal models are slightly better than the generic models. For predicting important notifications, the transfer learning-based CNN model achieved 91% of the precision on average. Accordingly, our model filtered unimportant notifications up to 81%. The effect of our delivery system depends on the ratio of important notifications and the model accuracy.

Our contributions are summarized as follows:

(i) With our mobile application, we collected 20,352 notifications and context data from 13 users for approximately 5 weeks. While gathering data, our tool programmatically infers users' interactions to decide the notification label.

(ii) We extracted 10 features from the collected data and analyzed their correlations to transform them into images. With the transformed data, we trained the convolutional neural network models corresponding to each user on a server equipped with high-performance GPU.

(iii) Based on the quantitative analysis, the impact of the proposed method was validated. Our results show that the trained models filtered unimportant notifications up to 81%. In addition, we reveal the impact of multidevices, which caused the inaccurate predictions by users.

2. Related Work

The human computer interaction research groups have studied various techniques to precisely infer users' interruptibility. In the desktop computer environment, they proposed the interruptibility management (IM) system for multiple applications [5, 10, 11]. For more accurate systems, context-aware interruptibility systems were proposed [12, 13], which require a user to wear sensors for extracting context. These approaches are based on sensors attached on the human body that can trigger notifications in an opportune moment by precisely recognizing the context.

In recent studies, researchers have exploited smartphones that are equipped with a variety of sensors to build the IM system [7, 8, 14–19]. For a more advanced system, nonobstructive approaches have been presented [7, 16]. To build the interruptibility models, these approaches unobtrusively monitored variation of context and system configurations without the user's involvement and questionnaires. Moreover, the notification's contents were considered as a context to build a better model [8, 15]. Turner et al. [19] proposed a decision-on-information-gain model to understand the users' microdecisions against notifications. Attelia [18] automatically mined important usage information to predict breakpoints for interruptions.

In addition, the OS-level IM system was designed in terms of privacy protection and deep context extraction [17]. For wearable devices, Kern and Schiele [20] proposed a delivery mechanism that relays notifications corresponding to six contexts that they defined.

However, all prior works have focused on predicting an opportune moment in a single mobile device. Those works have not yet considered emerging situation in which many people carry multiple mobile devices daily. Unlike previous works, we have focused on reducing notification delivery to a smartwatch from a smartphone to reduce user distraction.

3. Dataset

We focus on users who use a smartphone and a smartwatch simultaneously. Typically, not all users wear a smartwatch on the wrist. To collect data, we hand out the LG-Urbane W150 to 13 participants who are willing to join our experiments even without any monetary incentive. Table 1 lists the participant demographics. These participants consist of 10 males and 3 females with the age span between 24 and 35 years. As shown in Figure 1, the data were gathered during approximately 5 weeks on average. Across approximately 5 weeks, we finally collected 20,352 notifications.

4. Data Collection

In this section, we briefly describe how the notification type is unobtrusively labeled and what the type of sensor data that are collected.

4.1. Implementation. To collect notifications, we implemented an Android application that runs on a smartphone as a background service to programmatically decide notification labels as well as monitor the contexts when a notification is received. For deciding important notifications, our application was developed a few APIs, for example, Notification Listener Service (https://developer.android.com/reference/android/service/notification/NotificationListenerService.html) and UsageStatsManager (https://developer.android.com/reference/android/app/usage/UsageStatsManager.html) that are supported in API level 21 (Android 5.0). With Notification Listener Service, we can identify the arrival and removal times of the notifications. With UsageStatsManager, we can observe the states of the mobile application usage. By combining the two APIs, our mobile application can automatically decide important notifications. In addition, our mobile application exploits a third party library for computational social science [21], as well as SensorManager and SensorDataManager to obtain the contexts and store a large amount of data, respectively.

Table 1: Participant demographics.

Number of users	13 (10 males and 3 females)
Age	24, 25 (4), 26, 27 (2), 29, 30 (2), 34, and 35
Occupation	Students (9), office workers (3), and others (1)
Smartphone	Galaxy-s6 (3), Galaxy-s4 (2), Galaxy-note 3, Galaxy-note (2), Galaxy-a8, Galaxy-a7, Galaxy-Grand Max, Nexus 5, and Nexus 5x
Smartwatch	LG-Urbane W150 (13)

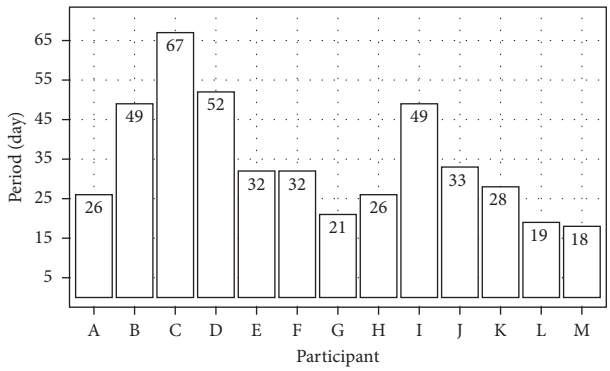

Figure 1: Data collection period across 13 participants.

In addition, it uses the Android OS API, Activity Recognition (https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi), for monitoring a user's activity. Table 2 lists the 10 contexts collected by our monitoring application.

4.2. Automated Labeling in Incoming Notifications. To infer the interactions and responses without any questionnaires, we combine a user's response time (indicating how quick a user responds) and the application launch (application relevant to the notification). We begin by cross-referencing the arrived notifications with a user's response time and the application launch that triggers the notifications. We consider a user's response time as a key factor to infer a user's interruption level because the delay to respond to notifications is highly relevant to important level of notifications [2, 8, 22, 23].

Figure 2 shows the high-level example of label notification when a user is interacting with a smartphone, while wearing a smartwatch. For a delay time threshold, we apply 10 minutes to each notification. The threshold was determined based on a previous work [8], which showed that approximately 60% of the interactions with notifications occur within 10 minutes. To prevent the missing of important notifications, our assumption for labeling is conservative. According to a previous work [24], while not all kinds of notifications are important, many were clicked within 30 seconds. Consequently, the threshold of 10 minutes implies enough margin to avoid missing them.

Figure 3 illustrates the results for labeling. Figure 3(a) shows the distribution of each label condition. Figure 3(b) shows the final labeling by cross-referencing. Finally, the result shows that the users represent various proportions of important notifications. According to the automated labeling based on combining a user's response time and application launch, the important notification rate of each user ranges from 35% to 90%, with the average being 68%. From the labeling results, we observed that user K is distracted by most of the notifications because user K has the lowest important notification rate.

Table 2: Feature group from the collected sensor data.

Group	Features
Notification's contents	Sender application name, Priority, and Title
Physical activities	Classifying activities into six classes: InVehicle, OnBicycle, OnFoot, Running, Still, Tilting, Walking, and Unknown
Time	Time of day, Day of the week, and Glance time
Sensor data	Recent phone usage, Phone's screen status, and Proximity

5. Building Prediction Models

In this section, we briefly describe how we handled the data for training the machine learning models, model structures, and how we trained them.

5.1. Preprocessing. To train the models, the categorical data we collected should be transformed into numerical data. Simply, we changed the nominal data (category name) to a unique digit number. As a result, each feature represents quite a different numerical range. For example, the title feature ranges from 1 to 152 in a user. However, the proximity feature has a binary number of either 0 or 1. Different scales of the features make the training difficult and slow to be converged. To remedy this issue, we normalized the data ranging from 0 to 1 with the following equation:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}, \tag{1}$$

where $x = (x_1, \ldots, x_n)$ and z_i is ith normalized data. To train the DNN, an input data shape of an $\mathbb{R}^{10\times1}$ matrix is used. A correlation of the input data is not important because the DNN is densely connected. However, in a case where we trained the CNN, a correlation of input data should be considered due to sparse connection and weight sharing. To consider the correlations among features, we computed the following Pearson correlation:

$$\operatorname{corr}(x, y) = \frac{\operatorname{cov}(x, y)}{\operatorname{sd}(x) \cdot \operatorname{sd}(y)}. \tag{2}$$

Figure 4 represents the feature correlations of user J. It results in an $\mathbb{R}^{2\times5}$ matrix. Based on the correlation result, the input data of user J is the following matrix:

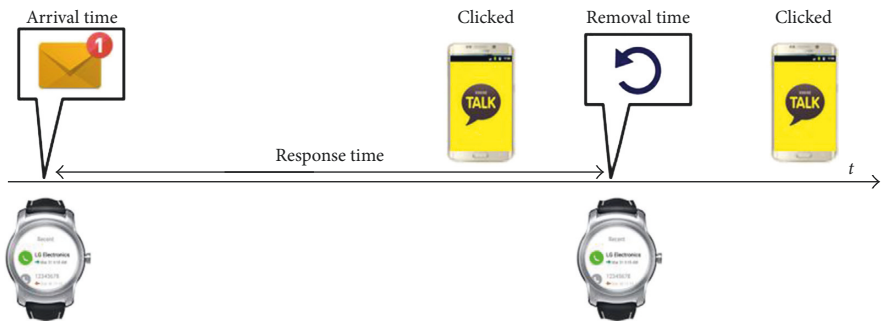

Figure 2: Labeling a notification on how quickly it responded and whether an application that triggers it is launched.

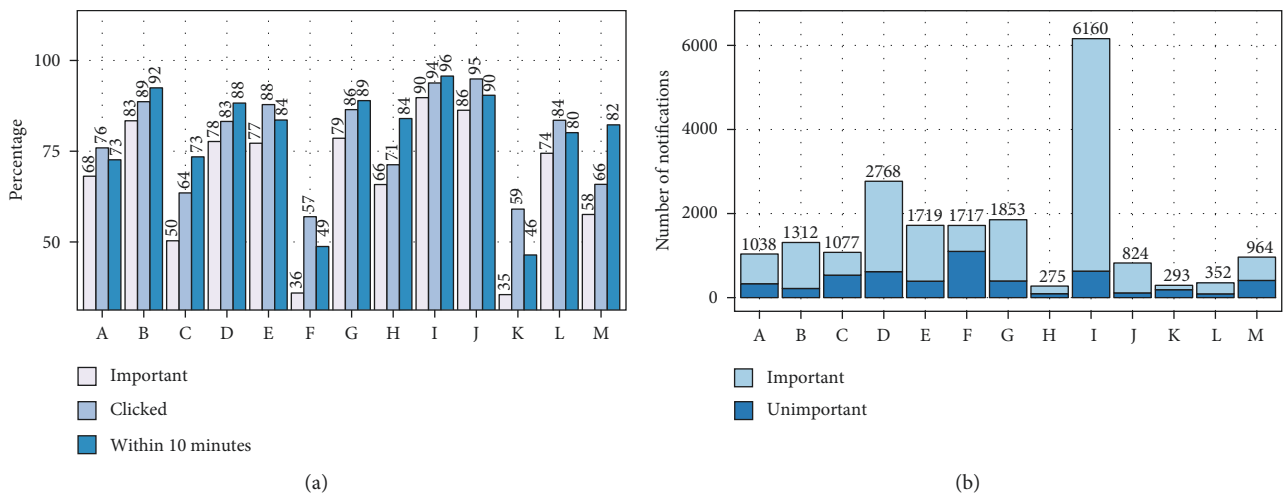

Figure 3: 20,352 notifications among 13 users for approximately 5 weeks; the bar plots of (a) the distribution of label types and (b) the distribution of important and unimportant notifications.

$$\begin{bmatrix} F_{\text{App name}} & F_{\text{Title}} & \cdots & F_{\text{Recent phone usage}} \\ F_{\text{Proximity}} & F_{\text{Priority}} & \cdots & F_{\text{Seen time}} \end{bmatrix}. \quad (3)$$

Likewise, we applied data transformation to the other users. As space is limited here, we have omitted all users' correlation data. For data preprocessing, we used the caret package (http://caret.r-forge.r-project.org/) in *R*.

5.2. Machine Learning Models. We used the following machine learning algorithms to predict important notifications: (1) naive Bayes (NB), (2) support vector machine (SVM), (3) random forest (RF), (4) deep neural networks (DNNs), and (5) convolutional neural networks (CNNs). To apply data into the naive Bayes algorithm, we used the categorical data type. In the support vector machine, we selected the nonlinear kernel function which is the radial basis function (RBF). In this case, the RBF kernel-based SVM was much better than the linear kernel-based SVM. To find the opportune radial kernel, we adjusted the C and sigma parameters with autotuning methods in the caret package. The ranges of the optimal C and sigma options for each user are 0.25–5 and 0.08–0.11, respectively. Likewise, in the random forest, we also explored a variety of hyperparameters. The best options were mtry = 45 and ntrees = 500 across all users.

A variety of deep learning models have been proposed for diverse applications and sensor types. Examples include the CNN, widely used for image classification [25, 26] and recently for text classification [27], and the DNN, used for speech recognition [28]. In our pilot work [29], we assumed that an important notification depends on the notification's contents and the user's context. Subsequently, we trained the fully connected 11-layer feedforward neural network consisting of 9 hidden layers. We bring this assumption and the mode into the current work by extending the input size to 10. Figure 5 illustrates the slightly extended DNN structure. In addition to the DNN model, we trained CNN model with same data to directly compare with the DNN model. Unlike the DNN, the CNN handles data correlation according to the kernel size. Specifically, the CNN generates useful features via its learnable filters. As shown in Figure 6, we implemented the CNN with two convolutional layers, followed by a pooling layer, a fully connected layer, and a logistic regression layer (sigmoid).

The convolution operation sums the contributions from different dimensional data in the input layer as follows:

$$y_l = \sum_{k=1}^{K} h_k * w_{kl}, \quad (4)$$

where y_m is the mth plane of the output data from each convolutional layer, h_k is the kth plane of the input data that

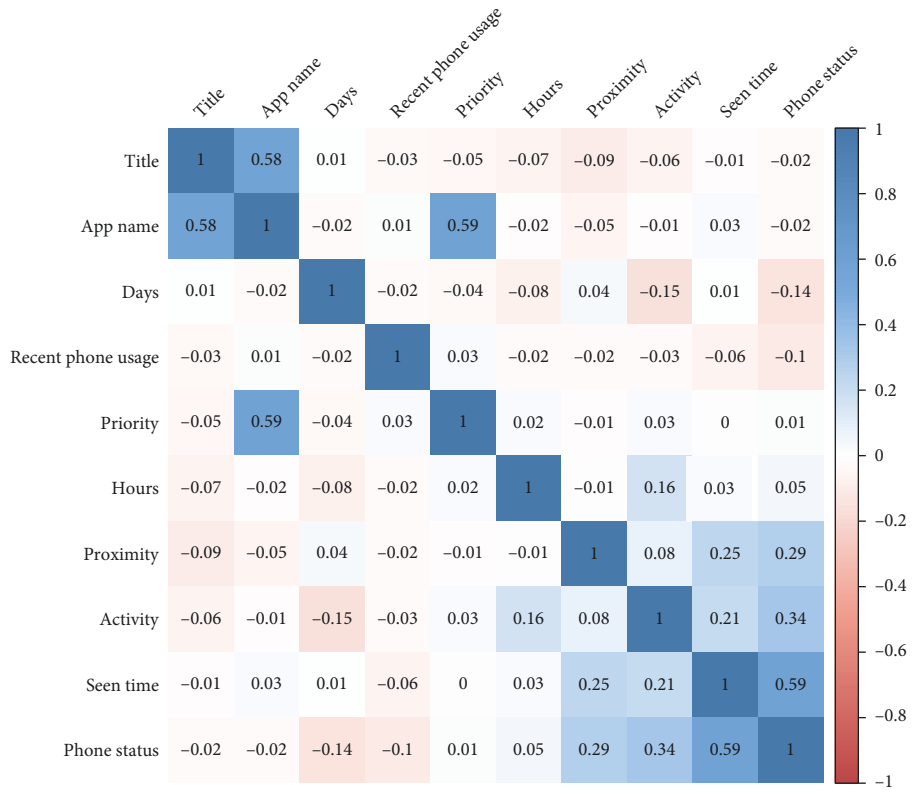

Figure 4: Pearson correlations of features in user J.

has K planes in total, and w_{km} is the kth plane of kernel m. We used a two-dimensional input layer. Therefore, K represents a single channel. After the input layer, K is the size of the activation map. The detailed convolutional operations are as follows:

$$z_{i,j,k} = b_k + \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} \sum_{k'=1}^{f_{n'}} x_{i',j',k'} \cdot w_{u,v,k',k}, \tag{5}$$

with

$$\begin{aligned} i' &= u \cdot s_h + f_h - 1, \\ j' &= v \cdot s_w + f_w - 1, \end{aligned} \tag{6}$$

where $z_{j',j',k'}$ is the output of the neuron located in row i' and column j' in feature map k' of the convolutional layer (layer l). s_h and s_w are the vertical and horizontal strides, respectively; f_h and f_w are the height and width of the receptive field (i.e., filter size), respectively; $f_{n'}$ is the number of feature maps in the previous layer (layer $l-1$). For the vertical and horizontal strides, we used one value. To maintain the same height and width for all layers, we used *zero padding*. $x_{i',j',k'}$ is the output of the neuron located in layer $l-1$, row i', column j', and feature map k' (or channel k' if the previous layer is the input layer). b_k is the bias term for feature map k (in layer l). $w_{u,v,k',k}$ is the connection weight between any neuron in feature map k of layer l and its input located at row u, column v (relative to the neuron's receptive field), and feature map k'.

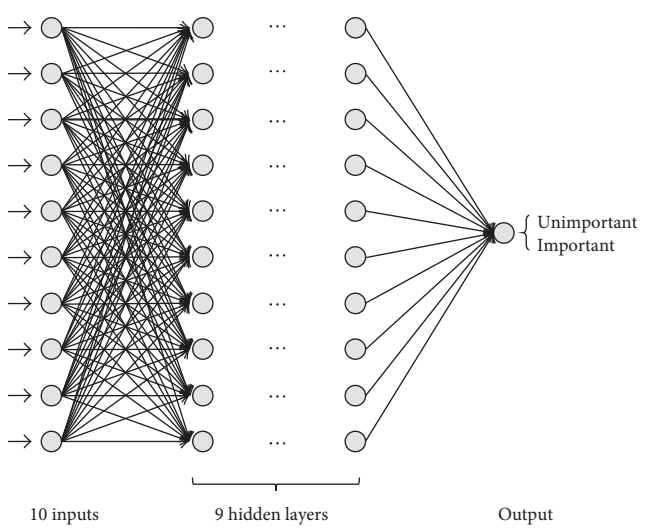

Figure 5: Fully connected feedforward neural network structure.

As an activation function, we used a rectified linear unit (ReLU) because this unit not only helps models to converge but also to avoid the vanishing gradient problem in models. $z_{j',j',k'}$ is applied to the ReLU as

$$h_{i',j',k'} = \max\left(0, z_{i',j',k'}\right). \tag{7}$$

In the last layer, the CNN represents a logistic regression to convert continuous data into notification labels. The last of the layer is computed as

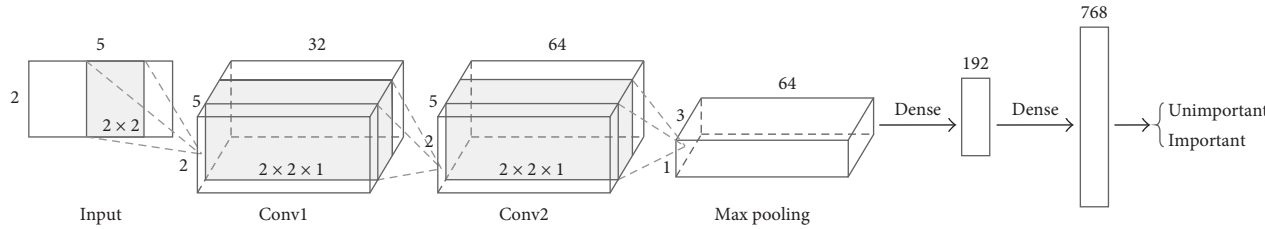

Figure 6: The convolutional neural network structure with 2 convolutional layers and 1 fully (dense) connected layer.

$$\hat{y} = \frac{1}{1 + \exp(-w^T x)}. \quad (8)$$

To compute the training error, we used the following cross entropy:

$$\text{Cost} = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log \hat{y} + \left(1 - y^{(i)}\right)\log(1 - \hat{y})\right]. \quad (9)$$

Subsequently, we trained the neurons with the *back-propagation* algorithm that propagates the cost to whole layers.

5.3. Model Training. To train diverse models, we used two frameworks: caret (*R* environment) and TensorFlow (Python environment). The caret package (http://topepo.github.io/caret/index.html) supports many traditional machine learning algorithms. Therefore, we exploited this package for the model training of NB, SVM, and RF.

We trained the DNN and CNN on Google's TensorFlow (https://www.tensorflow.org/) with the training data, by dividing the collected data into two parts, 70% for training and 30% for testing.

To successfully train the DNN with small individual data, we used the whole dataset from all the users. Subsequently, we reused the lower layers of the trained networks: this is called transfer learning. TensorFlow provides the tf.stop_gradient() function to freeze particular layers for fine tuning. Therefore, with personal data, it is easily possible to fine tune some higher-level portions of the network. We explored how many layer needs to be retrained. As a result, we found no significant difference among the layers. To reduce overfitting and computational problems, we decided to retrain only the final layer and completed building the neural network models.

To avoid overfitting concerns, we took two steps: first, we applied the *dropout* in *conv1* and *conv2* layers and the fully connected layer. Dropout is widely used in CNNs to avoid overfitting during model training [30].

In our case, we set up 10% rate as the dropout option. Second, we implemented an early stop mechanism, which forces training to be terminated when the validation error starts to increase. We initialized the weight vectors with the *Xavier* method [31] and the learning rate at 10^{-5}. In addition, we chose the Adam optimizer to train the weight vectors [32]. We performed more than 20,000 epochs. However, we stopped the training process early when the difference between the validation and training costs was larger than 10^{-2}. In addition, we performed full-batch learning, which means that the entire training data were used for a single gradient descent. We implemented our two models on a local server equipped with NVIDIA GTX 1080 (8.8 TFLOPS). After the whole training process, we finally built the deep learning models to infer whether a notification is relayed to a smartwatch with the sensed contexts.

6. Evaluation

In this section, we elaborate how accurate our classifier is in predicting important notifications. In addition, we compare the accuracy of the prediction models trained on user's personal data with a generic prediction model trained on all users' data.

6.1. Evaluating Prediction Models. We evaluated five prediction models by comparing the prediction results with the ground-truth labels that are unobtrusively determined on the testing data, by using the testing data, which accounts for 30% of the collected data. Figure 7 shows the precision, recall, and *F*-score of all models.

Our results demonstrate no significant difference in performance among the five machine learning algorithms. However, the *F*-score of fine tuning the CNN (mean 88%) is slightly better than others. We expect that performance gap among other models is increased by more users' data. As a recall, the CNN (mean 91%) outperformed the others.

The low precision results in the unimportant notification deliveries to a smartwatch. More severely, the low recall leads to negative effects that reduce information awareness. Therefore, to reduce the negative effects, we focus on maximizing the recall by sacrificing the precision.

In spite of the 91% recall on average, a few predictors show relatively poor results. However, we claim that missing an important notification on a smartwatch is not a critical issue because the smartwatch is a secondary device. Therefore, based on previous works [33], a smartphone is the key device that checks for notifications when users carry four devices including a tablet, a PC, a smartwatch, and a smartphone. Basically, users can check all the notifications on their smartphones even if they are not relayed to the smartwatches. To show the effectiveness, we calculated the number of notifications that were filtered out. To calculate them, we simply subtracted the notifications that were classified as important from all notifications. Figure 8 shows the filtered notification rate for each user. The filtered notification rate is 36% on average. Obviously, the filtering effects relied on the number of unimportant notifications and the recall of models.

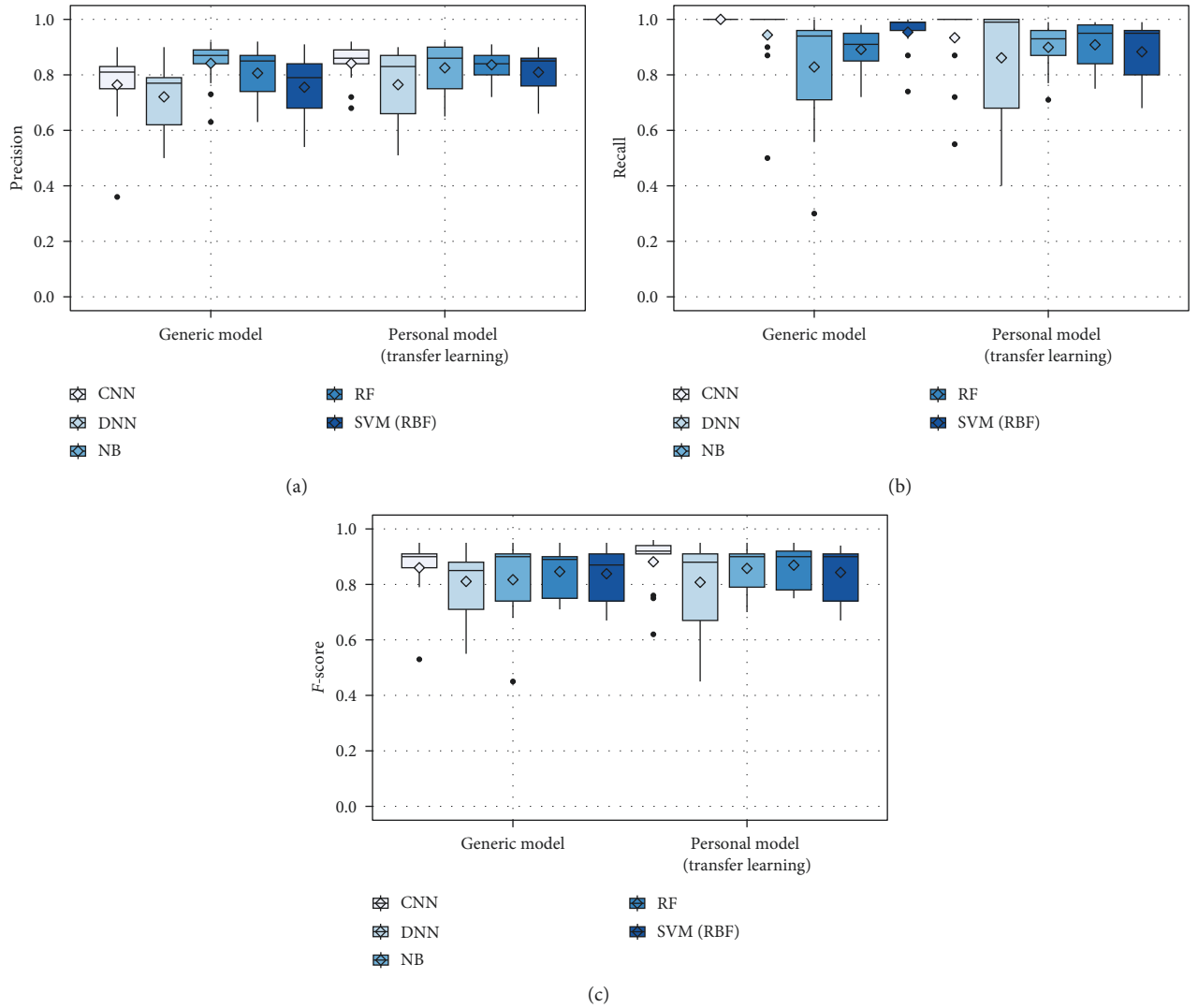

Figure 7: Prediction results of the CNN, DNN, NB, RF, and SVM models: (a) precision, (b) recall, and (c) *F*-score.

6.2. Generic versus Personal Models. We compare the performance of the predictions models trained on individual data with a generic model trained on all users' data. In most cases, the personal model is better than the generic model. However, in some cases, when the amount of data is small, the generic model shows a better *F*-score than the personal model. This is primarily due to the lack of training data. Unlike other machine learning algorithms, the personal model of the CNN stably outperformed its generic model across all users. As mentioned before, we performed fine tuning with the generic model to build the personal models: this is called transfer learning. Its benefits are even greater when we consider users with little or no training data.

7. Discussion and Limitations

We are aware that notification filtering may affect the user experience of smartwatches. However, we claim that the user experience degradation of the smartwatch is not critical for the following reasons: first, users tend to addictively check their smartphones [34]. In addition, based on ANOVA, recent works [9] have revealed that the alert type of the smartphone (silence and vibration) does not show statistically significant effects on the notification awareness. Specifically, users reported that they just missed the notifications for 14.63% of the time, even when their smartphones were in silent mode. This means that the notification awareness of users is not dependent on any other factors such as the alert type of the smartphone or the presence of the smartwatch due to frequent smartphone usage. Next, users are not dependent on the smartwatches because smartwatches are secondary devices. A recent work [3] investigated the level of discomfort if devices were running out of battery. According to the questionnaire survey, 33% of the smartwatch users reported *neutral*. However, 46% of the users responded with *very uncomfortable*. In addition, Weber et al. found that users prefer a smartphone for receiving notifications among the four devices (a tablet, a smartwatch, a smartphone, and a PC) regardless of the notification type [33]. This is because the smartphone is a key device for online connectivity and communication with other people.

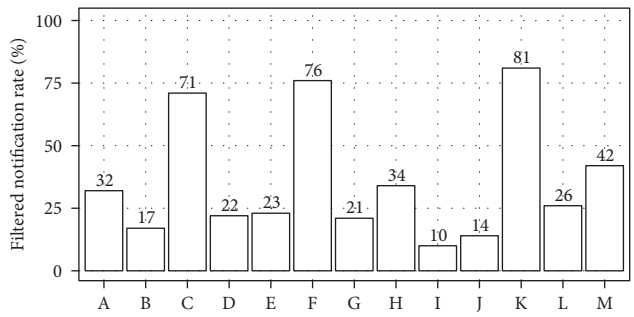

Figure 8: Filtered notification rate.

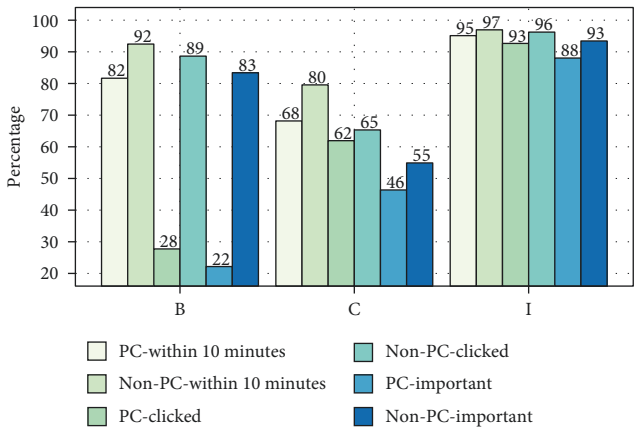

Figure 9: The labeling effects using PCs.

In a few users, our classifier shows poor prediction results. The reasons are as follows: first, some users just glanced at the wrist and their smartphones to decide whether to interrupt their current activities to deal with the notification. In this case, the proposed labeling method misclassifies this notification as dismissed because there is no interaction. Next, some users read the notifications and dismissed them on their PCs. In this case, our labeling method does not capture the interactions. To investigate the effects by using PCs, we conducted one additional experiment. The experiment was divided into two conditions: allowing three users to use PCs and not to use PCs. The experiments were conducted for a week, respectively.

Figure 9 shows the three class types under two conditions. If a user uses a PC, many notifications are misclassified as unimportant because the application launch occurs in the PC. Therefore, we forced them not to use any device except a smartphone and a smartwatch for checking notification. Despite our instructions, a few users still used the PCs for their official tasks such as sending documents. Therefore, a poorly trained model stems from misclassified notification labels.

8. Conclusion

Even though smartwatches improve the awareness of incoming notifications, they aggravate the disruptive nature of notification delivery because they are worn on the human body. To reduce a smartwatch user's distraction, we proposed a notification delivery system that relays only important notifications predicted by CNN models. To build our models, we collected 20,352 notifications and sensor data from three users using a mobile application, which unobtrusively monitors all the data. Subsequently, we implemented a binary classifier that identifies important notifications. For important notification prediction, our classifier attained 76% and 91%, respectively, for the precision and recall on average, spanned across all users. This classifier can reduce the distraction in smartwatch users without noticeable degradation in the users' awareness.

In the future, we plan to deploy our notification delivery system to real users. In addition, we implemented the OS-level software without cloud assistance not only to capture the detailed context (i.e., full text) but also to avoid privacy intrusion.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Research Foundation (NRF) of Korea funded by the Ministry of Education (NRF-2017R1D1A1B03034705)

References

[1] A. Visuri, Z. Sarsenbayeva, N. van Berkel et al., "Quantifying sources and types of smartwatch usage sessions," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*, pp. 3569–3581, ACM, New York, NY, USA, 2017.

[2] M. E. Cecchinato, A. L. Cox, and J. Bird, "Always on(line)?: user experience of smartwatches and their role within multi-device ecologies," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*, pp. 3557–3568, ACM, Denver, CO, USA, May 2017.

[3] C. Min, S. Kang, C. Yoo et al., "Exploring current practices for battery use and management of smartwatches," in *Proceedings of the IEEE International Symposium on Wearable Computers (ISWC)*, pp. 11–18, Osaka, Japan, 2015.

[4] S. T. Iqbal and E. Horvitz, "Notifications and awareness: a field study of alert usage and preferences," in *Proceedings of International Conference on Computer Supported Cooperative Work*, pp. 27–30, ACM, Shanghai, China, April 2010.

[5] E. Horvitz, P. Koch, and J. Apacible, "Busybody: creating and fielding personalized models of the cost of interruption," in *Proceedings of the ACM International Conference on Computer Supported Cooperative Work (CSCW)*, pp. 507–510, Chicago, Illinois, USA, November 2004.

[6] L. Leiva, M. Böhmer, S. Gehring, and A. Krüger, "Back to the app: the costs of mobile application interruptions," in *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services, MobileHCI*, pp. 291–294, San Francsico, CA, USA, September 2012.

[7] V. Pejovic and M. Musolesi, "InterruptMe: designing intelligent prompting mechanisms for pervasive applications," in *Proceedings of the ACM International Joint Conference on Pervasive*

and Ubiquitous Computing (UbiComp), pp. 897–908, Downtown Seattle, WA, USA, September 2014.

[8] A. Mehrotra, M. Musolesi, R. Hendley, and V. Pejovic, "Designing content-driven intelligent notification mechanisms for mobile applications," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, pp. 813–824, Osaka, Japan, September 2015.

[9] A. Mehrotra, V. Pejovic, J. Vermeulen, R. Hendley, and M. Musolesi, "My phone and me: understanding people's receptivity to mobile notifications," in *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pp. 1021–1032, San Jose, CA, USA, May 2016.

[10] S. S. Intille, J. Rondoni, C. Kukla, I. Ancona, and L. Bao, "A context-aware experience sampling tool," in *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pp. 972-973, Ft. Lauderdale, Florida, USA, April 2003.

[11] S. T. Iqbal and B. P. Bailey, "Oasis: a framework for linking notification delivery to the perceptual structure of goal-directed tasks," *ACM Transactions on Computer-Human Interaction*, vol. 17, no. 4, pp. 1–28, 2010.

[12] J. Ho and S. S. Intille, "Using context-aware computing to reduce the perceived burden of interruptions from mobile devices," in *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pp. 909–918, Portland, OR, USA, April 2005.

[13] D. Siewiorek, A. Smailagic, J. Furukawa et al., "SenSay: a context-aware mobile phone," in *Proceedings of the IEEE International Symposium on Wearable Computers (ISWC)*, pp. 248-249, Washington, DC, USA, October 2003.

[14] S. Rosenthal, A. Dey, and M. Veloso, "Using decision-theoretic experience sampling to build personalized mobile phone interruption models," *IEEE Pervasive Computing*, vol. 6696, pp. 170–187, 2011.

[15] 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 *Proceedings of the ACM International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI)*, pp. 103–112, Lisbon, Portugal, 2010.

[16] M. Pielot, R. D. Oliveira, H. Kwak, and N. Oliver, "Didn't you see my message?: predicting attentiveness to mobile instant messages," in *Proceedings of the ACM International Conference on Human Factors in Computing Systems (CHI)*, pp. 3319–3328, Toronto, ON, Canada, 2014.

[17] K. Lee, J. Flinn, and B. Noble, "The case for operating system management of user attention," in *Proceedings of the ACM International Workshop on Mobile Computing Systems and Applications (HotMobile)*, pp. 111–116, Santa Fe, NM, USA, February 2015.

[18] T. Okoshi, H. Nozaki, J. Nakazawa, H. Tokuda, J. Ramos, and A. K. Dey, "Towards attention-aware adaptive notification on smart phones," *Pervasive and Mobile Computing*, vol. 26, pp. 17–34, 2016.

[19] L. D. Turner, S. M. Allen, and R. M. Whitaker, "Reachable but not receptive: enhancing smartphone interruptibility prediction by modelling the extent of user engagement with notifications," *Pervasive and Mobile Computing*, vol. 40, pp. 480–494, 2017.

[20] N. Kern and B. Schiele, "Context-aware notification for wearable computing," in *Proceedings of the IEEE International Symposium on Wearable Computers (ISWC)*, pp. 223–231, Osaka, Japan, September 2015.

[21] N. Lathia, K. Rachuri, C. Mascolo, and G. Roussos, "Open source smartphone libraries for computational social science," in *Proceedings of the ACM International Conference on Pervasive and Ubiquitous Computing adjunct publication (UbiComp)*, pp. 911–920, Zurich, Switzerland, September 2013.

[22] M. E. Cecchinato, A. L. Cox, and J. Bird, "Working 9-5?: professional differences in email and boundary management practices," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15)*, pp. 3989–3998, Seoul, Republic of Korea, April 2015.

[23] A. S. Shirazi and N. Henze, "Assessment of notifications on smartwatches," in *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI '15)*, pp. 1111–1116, ACM, Copenhagen, Denmark, August 2015.

[24] A. S. Shirazi, N. Henze, T. Dingler, M. Pielot, D. Weber, and A. Schmidt, "Large-scale assessment of mobile notifications," in *Proceedings of International Conference on Human Factors in Computing Systems (CHI)*, pp. 3055–3064, ACM, Toronto, ON, Canada, 2014.

[25] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 1943–1955, 2016.

[26] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[27] N. Majumder, S. Poria, A. Gelbukh, and E. Cambria, "Deep learning-based document modeling for personality detection from text," *IEEE Intelligent Systems*, vol. 32, no. 2, pp. 74–79, 2017.

[28] K. Chen and A. Salman, "Learning speaker-specific characteristics with a deep neural architecture," *IEEE Transactions on Neural Networks*, vol. 22, no. 11, pp. 1744–1756, 2011.

[29] J. Lee, J. Kwon, and H. Kim, "Reducing distraction of smartwatch users with deep learning," in *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct, MobileHCI '16*, pp. 948–953, ACM, New York, NY, USA, 2016.

[30] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[31] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10)*, Society for Artificial Intelligence and Statistics, pp. 249–256, Sardinia, Italy, May 2010.

[32] D. Kingma and J. Ba, "Adam: a method for stochastic optimization, arXiv preprint arXiv:1412.6980," in *Proceedings of 3rd International Conference for Learning Representations*, San Diego, CA, USA, 2015.

[33] D. Weber, A. Voit, P. Kratzer, and N. Henze, "In-situ investigation of notifications in multi-device environments," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'16)*, ACM, pp. 1259–1264, Heidelberg, Germany, 2016.

[34] O. Turel and A. Serenko, "Is mobile email addiction overlooked?," *Communications of the ACM*, vol. 53, no. 5, pp. 41–43, 2010.