Inferring Phone Location State

Steven Chen
Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, United States
scchen@berkeley.edu

Won Park
Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, United States
wonpark@berkeley.edu

Joanna Yang
Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, United States
joannayangg@berkeley.edu

David Wagner
Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, United States
daw@berkeley.edu

***Abstract*—Smartphone sensors are becoming more universal and more accurate. In this paper, we aim to distinguish between four common positions or states a phone can be in: in the hand, pocket, backpack, or on a table. Using a uniquely designed neural network and data from the accelerometer and the screen state, we achieve a 92% accuracy on the same phone. We also explore extending this to different phones and propose an acceleration calibration technique to do so.**

***Index Terms*—Sensors, Smartphone, Position**

I. Introduction

With the recent proliferation of smartphones with sensors, such as light sensors and accelerometers, combined with the wide usage of smartphones over traditional phones, there has been an explosion of applications utilizing sensors, including activity detection [1] and user authentication [2], [3]

Knowing, a priori, the position of the phone (e.g., in a bag, in a user's pocket, etc.), which we will call *phone state*, can prove to be useful. For example, a phone's vibration intensity can be adjusted based on where the phone is positioned on the person, potentially prolonging battery usage [4]. In another example, CO2 and pollution sensors could be turned off automatically if the phone were deemed to be in a pocket or backpack [5].

Not only that, but phone state can also be a powerful information tool that can enhance the quality of other classifiers. For example, Martín et al. has shown that feeding phone state as an input to a classifier doing activity prediction will increase accuracy in certain instances [1]. As another example, user authentication performance was augmented when a prior step was taken to infer phone state [3].

In this paper, we discuss our methodology of automatically detecting and differentiating between four commonly identified phone states: table, backpack, hand, and pocket. To do this, we explore useful phone features that can be easily accessed from sensor data available to Android phone models, namely data from the accelerometer and the screen. We then introduce a neural net architecture well-suited to process and learn from these features, and demonstrate that this architecture can achieve 92% accuracy in classifying a phone's state provided data from a single phone model. We then extend this to a different phone model and show that it could be possible to evaluate states on a different phone, even if the phone has an accelerometer sensor that is calibrated differently. To do this, we found that the acceleration values between the two phones can be adjusted with a simple offset. In two cross phone studies we achieve accuracies of 78% and 91%. Unfortunately, we were unable to investigate the root cause of this discrepancy further. While the work on this is preliminary, we believe it provides an important step toward the goal of a universal classifier that can detect different phone states across phone models.

II. Related work

There has been much previous work involving smartphone sensors. Note that this is a different set of work than the previous work that exists for wearable sensors, [6], [7]. We focus on smartphone sensors since we believe they are more readily accessible, prevalent, and versatile than wearable sensors.

Work by Khan et al., utilized accelerometer values to classify phone state but was limited to distinguishing between whether the phone was in the upper or lower half of the body [8]. Similarly, Miluzzo et al. proposed the use of various sensors to classify different phone locations, but only recognized two states: inside and outside of a pocket [5]. In contrast, our work distinguishes between four states: hand, pocket, backpack, and table.

Several other works tried to classify and distinguish between more states (e.g. hand, bag, pocket, etc.), but only used data from the accelerometer resulting in accuracies between 74.6% to 84% [4], [9]. Our work improves on this accuracy by utilizing additional data from the phone screen.

Park et al. achieved an accuracy of 94% in distinguishing between hand, ear, pocket, and backpack, but limited the data to record instances where the user was walking [10]. In contrast, our work does not require that the phone user perform a specific task in order to distinguish between states. Rather, we train and validate our model on phone data collected throughout the

entire day, including times when the phone may be still and not actually on the user.

Other works improved accuracy by using several additional sensors [11]. For example, Wiese et al. explored the idea of adding sensors that utilize capacitive sensing, multi-spectral properties, as well as light and proximity sensing [12]. They were able to achieve accuracies of 85% to 100%. Similarly, Martín et al. used sensors like light, proximity, and acceleration sensors to obtain phone state accuracy of 92.94% [1], but at a cost in file size. Our work achieves similar accuracy rates but with data from fewer sensors.

In a somewhat different approach, work by Wijerathne et al. [13] took advantage of smartphone accelerometers to help monitor road conditions. Similar to our work, the paper attempted to generalize between all positions the phone could be in, but instead of detecting the exact position of the user's phone, the authors attempted to detect rough roads and bumps while cycling.

In this paper, we take this idea further and utilize data from only two sources: the accelerometer and the phone screen. Data from both sources is readily available, does not require specific user permissions, and is less energy draining than previously studied sensors. Furthermore, unlike previous work, we also train and validate our model on phone data collected throughout the entire day, and not just during specific tasks (e.g. walking). This includes times when the phone may be still and not actually directly on the user. An example of this may be if the phone is left on a table or in a backpack.

III. Approach

A. Problem

We want to predict the state of an user's phone based on the sensor data collected on the phone. From our observations and prior research, the most common states of the phone would be in a user's: backpack, pocket, hand, or on a table. Upon a cursory observation of accelerometer traces, these phone states also appeared to be distinguishable and motivated our approach to use deep learning to classify these states. Sample accelerometer traces for the states, measured on a Nexus 5X smartphone are shown in Figure 1.

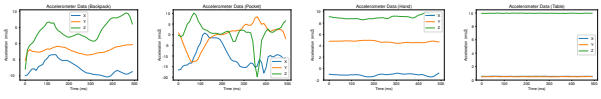

Fig. 1. Typical acceleration graphs for our four states for a Nexus 5X.

Our goal is to allow the classifier to predict phone states no matter what action the user may be performing. This includes times when the phone is not physically on the user, including cases that may be very difficult to differentiate: a phone being in a still backpack versus on a table, for example. Later, we propose a potential solution to this problem.

B. Features

For creating features, the relevant sensor data were the accelerometer readings (X, Y, Z values), number of unlocks, number of screen touches, and number of times the screen turned on/off. For each window of 0.5s of these raw sensor readings, we generate the following features:

1) *Total number of phone unlocks*
2) *Total number of phone touches*
3) *Fraction of window that phone screen was on*
4) *Mean acceleration in each of X, Y, Z*
5) *Std. deviation of acceleration in each of X, Y, Z*
6) *Mean magnitude of acceleration in each of X, Y, Z*
7) *Std. deviation of magnitude of acceleration in each of X, Y, Z*
8) *Phone is flat (handcrafted feature explained below)*

The "Phone is flat" feature is a boolean feature derived from the raw accelerometer readings. The feature is 1 if the three equations below all hold, and 0 otherwise.

$$\text{(Mean X Accel. Magnitude)} < 1.0$$
$$\text{(Mean Y Accel. Magnitude)} < 1.0$$
$$|9.8 - \text{(Mean Z Accel. Magnitude)}| < 1.0$$

In practice, features 1-3 (phone unlock count, phone screen touch count, and phone screen on time) will usually be 0 or 1, since there is unlikely to be more than 1 such event in the window.

Other sensors that we considered to be relevant for predicting phone states are the batched light sensor and step count. However, the batched light sensor was not used because of its inability to distinguish outdoor nighttime darkness and the darkness from an enclosed backpack. We could not use the step count sensor because the sensor data we collected showed that this sensor was not reliable for our phones.

We do not utilize a overlapping or 'rolling' window. Instead, we take distinct chunks of 0.5 seconds. We believe that this is acceptable since the window size is small enough that we would not miss most transitions. This decision is supported by previous work that found no notable difference in accuracy between using overlapping and non-overlapping windows [1].

C. Architecture

Our architecture has two parts. The first consists of convolutional layers, while the second part contains dense fully connected layers.

In the convolution section, we use the raw acceleration data, which includes the acceleration in the x, y, z directions. After multiple one-dimensional convolution layers, max and global pooling, and dropout layers, we concatenate the 16 features above in order to incorporate the features that do not involve the data from the accelerometer. Together, these features are fed into the second part of the net. We chose to separate the features in this way in order to take advantage of the potentially periodic behavior of a user's acceleration in certain positions (e.g. walking). The features that use the acceleration in the X, Y, Z are separated from the other binary features because we

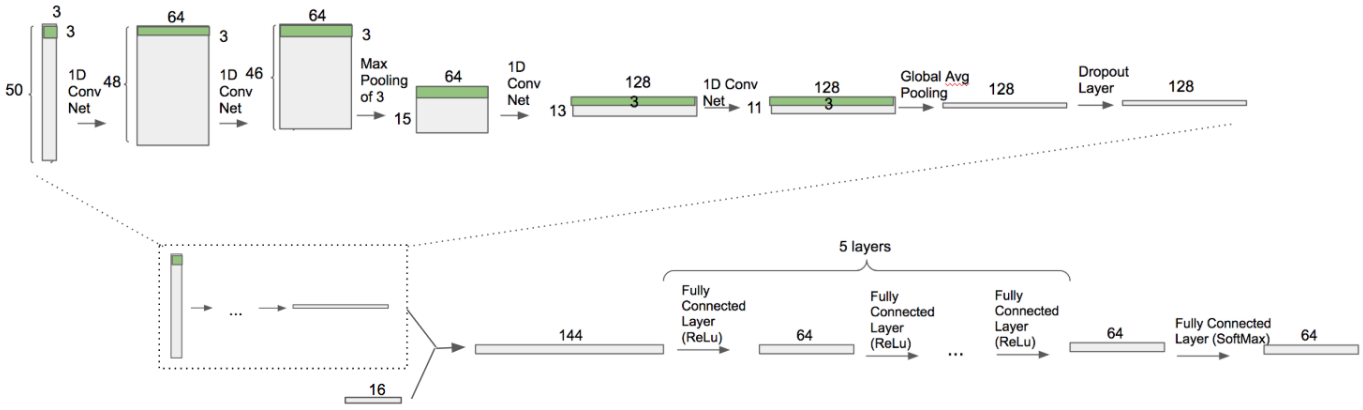

Fig. 2. The architecture of our convolutional neural net

Layer	Filters	Outputs	Activation / Note
Conv1D	64	48×3	ReLU
Conv1D	64	46×3	ReLU
MaxPooling1D	64	15×3	Stride = 3
Conv1D	128	13×3	ReLU
Conv1D	128	11×3	ReLU
GAP1D	128	128×1	
Dropout	—	128×1	Rate = 0.5
Concat	—	144×1	(128×1) + (16×1)
Dense	—	64	ReLU; 6 of these in succession
Dense	—	4	Softmax

TABLE I
MODEL ARCHITECTURE. FOR ALL CONVOLUTIONAL LAYERS, THE KERNEL WAS 3×6, AND GAP1D IS SHORT FOR GLOBALAVERAGEPOOLING1D. IN THE CONCATENATION LAYER, THE COUNT FEATURES (16×1) ARE CONCATENATED WITH CONVOLUTIONAL OUTPUTS TO FORM THE INITIAL INPUT INTO THE DENSE LAYERS.

wanted to capture the time series data of the different phone states. The other binary features are independent of the time, so these features are added in after the acceleration features go through the convolution layers.

After the concatenation of inputs, our model has 6 dense layers culminating in 4 outputs, which match the four classes listed previously. Our model is shown in Figure 2 and a description is shown in Table I. We have experimented with other architecture, such as separate binary linear classifiers for each phone state and separate neural net classifiers for each phone state. However, we found out this multiclass neural net classifier works best and has the highest accuracy rates. We also experimented with the number of convolutional and dense layers as well as the number of layer units and width of the convolutional filters.

IV. EVALUATION

A. Data Collection Tools

a) Data Collection Software: To collect our data, we used AppMon, an Android mobile logging application. AppMon logs data from various smartphone sensors, including:

1) Accelerometer
2) Screen unlock count
3) Screen touch count
4) Screen On-off count

In our project, we will be using data from the sensors to create features in order to classify and predict the location of the phone.

The app collects accelerometer readings (X, Y, Z directions) in units of m/s^2 every 10ms. It also registers the timestamp at which various trigger events occur (e.g. phone unlocks, screen on, etc.)

b) Phones: We used diary studies on two phones to collect data for the classifiers, a Nexus 5 and a Nexus 5X.

B. Diary Study

To obtain data for training and validation, three 'diary studies' were performed, all by the same participant.

In each study, the participant carried the phone around with the monitoring app running and underwent a daily routine. Whenever a change of state occurred, the participant recorded the time (according to the phone) and the new state.

The first diary study was performed for 14.5 hours using the Nexus 5X which was also used as the source of training as well as validation data. The second and third diary studies were performed using the Nexus 5 for 7 and 6 hours, respectively. This data was used to check for cross-phone validation.

The distribution of the states during the diary study are shown in Table II.

C. Data Preprocessing

a) Partitioning for k-fold cross validation: In normal k-fold cross validation, the dataset is randomly partitioned into k-folds. However, we wanted to ensure that for each fold,

TABLE II
DISTRIBUTION OF STATES FOR DIFFERENT DIARY STUDIES

Diary/Phone	State	Duration	Percent
Diary 1 Nexus 5X	Table	9.29 hours	63.5%
	Pocket	0.49 hours	3.4%
	Backpack	4.26 hours	29.2%
	Hand	0.58 hours	3.9%
Diary 2 Nexus 5	Table	2.85 hours	40.0%
	Pocket	1.45 hours	20.4%
	Backpack	0.15 hours	2.1%
	Hand	2.67 hours	37.5%
Diary 3 Nexus 5	Table	0.44 hours	6.1%
	Pocket	0.60 hours	8.5%
	Backpack	0.47 hours	6.6%
	Hand	4.93 hours	69.4%

the validation set was not biased towards the training set. Specifically, for any state, two samples collected at sequential timesteps (e.g. sample A at 11:30:00 and sample B at 11:30:30 while the phone was in the user's pocket) are likely to be very similar. Then, if one sample were partitioned into the training set and the other sample into the validation set, we suspected that it could be the case that we observe a high validation accuracy, but instead of learning a generalizable decision rule, the network may have only learned to remember the class of the sample in the training set and regurgitate that class when seeing the very similar other sample in the validation set. Our suspicions were confirmed when we observed that the cross-validation accuracy was 9.22% higher when the dataset was partitioned randomly instead of sequentially.

Accordingly, we decided to partition the data sequentially (i.e. without randomization) for cross-validation. However, since the data from our diary studies did not have equal proportions of each class, it was possible for some folds to have a training set with barely any instances of a class and then validate on many instances of the same class that the network had limited instances to train/learn from. To ensure that each fold had comparable training and validation distributions for each class, we preprocessed the diary study data before sequentially partitioning it. Specifically, we divided the data of each diary study into its contiguous segments (e.g. 10:00-11:15, Table or 13:55-14:30, Hand), grouped those segments by class, and then concatenated the segments together for each class. The result was four homogenous datasets, one of each class, such that for each fold, we could construct the validation set by aggregating together $\frac{1}{k}$ of each class dataset and then construct the training set by compiling the remaining $\frac{k-1}{k}$ of each class dataset. Within these homogenous class datasets, the data were not randomized but kept sequential.

b) Accelerometer calibration across phones: We had hoped that phone accelerometers would be consistent across phones and models (i.e. both the Nexus 5 and Nexus 5X in the same state would have the same accelerometer readings, save for some noise). However, cursory comparison of the accelerometer data between the two phones (while both phones had been on the table) revealed both differing accelerometer readings and readings that did not equal the expected values of $X = 0m/s^2$, $Y = 0m/s^2$, and $Z = 9.8m/s^2$.

In order to account for the inconsistent calibrations between the accelerometers of different phones, we attempted to re-calibrate the accelerometer data for each phone before using it with the network. First, to see what type of recalibration model was necessary (e.g. $accel_{calibrated} = accel_{raw} + C$ or $accel_{calibrated} = K(accel_{raw}) + C$), we taped the Nexus 5 and Nexus 5X together, and then recorded the phones' accelerations in the four states. Plots of the two phones' accelerations against each other, Figure 3, over the same time period then suggested a linear relationship of the form $accel_{calibrated} = accel_{raw} + C$.

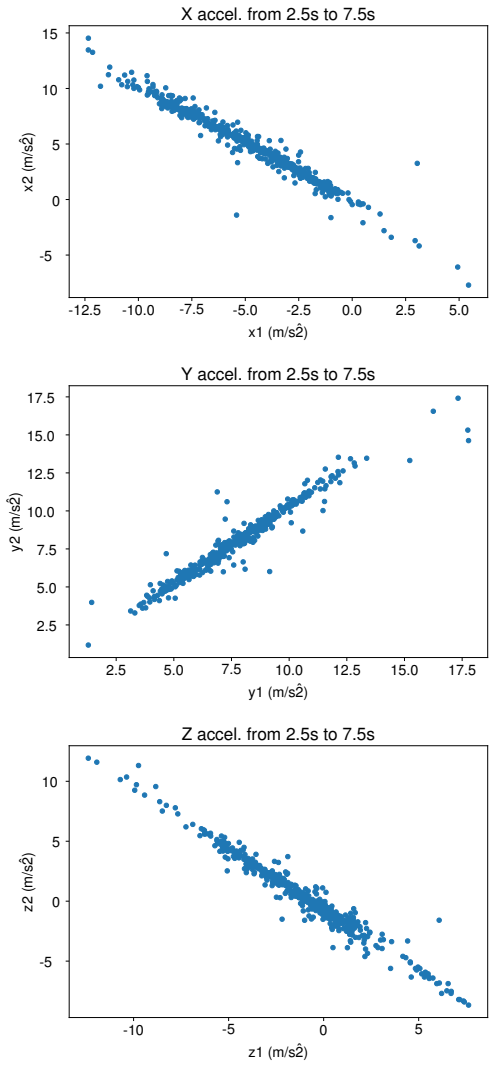

Fig. 3. Graphs of the acceleration of the Nexus 5X (x1/y1/z1) against the Nexus 5 (x2/y2/z2) while physically taped together over the same period of time.

To find the calibration constants for each phone, we collected accelerometer data from when the phone was flat on a table, measured the acceleration in the X, Y, and Z directions and then computed the offsets from the expected values of $0m/s^2$, $0m/s^2$, and $9.8m/s^2$. These offsets were then added to all of

the accelerometer data for that phone model. For practical applications of our phone state classifier, we believe that this calibration process could then be reasonably achieved by asking phone users to place their phones flat on a table for a short period of time (e.g. a minute) to allow the above calibration constants to be calculated. To note, we do not know at this point if the reason the sensor calibration is off is a property unique to the phone or the phone model or both. Regardless, we believe that this phone calibration method is a potential solution for any of the above cases.

V. Results

A. Single Phone Model

Using the architecture detailed in section 3, we trained the network for 10 epochs and measured its effectiveness using 10-fold cross-validation. Training/validation data was compiled from the diary study conducted with the Nexus 5X.

The average accuracy across all folds was 92%, and the corresponding confusion matrix is shown in Table III. The network is able to learn how to classify the Table state very accurately, which is understandable given the consistent nature of the state (i.e. always in a still, flat position). The states of Backpack and Pocket were also classified with pretty moderate accuracy, with the misclassifications likely stemming from the more diverse nature of the two states (i.e. the user could have been moving or still when the phone was in their backpack or pocket). The Hand state was classified with the least accuracy, and misclassified most often instead as the Backpack state. These misclassifications likely also stem from the varied positions of the Hand state, as in addition to the user using the phone actively in their hand, they may also walk with their phone in hand but just to hold the phone and not actually using it.

One problem our classifier had was in distinguishing between cases where the phone was flat on a table or still in a non-moving backpack. For future work, we propose a post-processing step after obtaining the classification output. In this step, classifications from timeframes before and after the one in question would be considered to help 'smooth' the outputs.

TABLE III
Confusion matrix of the network predictions on Diary Study 1 (Nexus 5X). Each entry indicates the percent of total instances that were predicted as the predicted class by the network and labeled the actual class.

	Actual			
Predicted	Backpack	Pocket	Hand	Table
Backpack	5.1%	.8%	.3%	.1%
Pocket	1.1%	6.0%	.2%	.5%
Hand	.1%	.1%	7.2%	.2%
Table	.4%	3.3%	.3%	74.1%

B. Multiple Phones

We also attempted to validate our networks across different phones, by training on data collected by one phone model and then validating on data collected on a different phone model. Specifically, we first applied the calibration process described in section 4.3 to both the Nexus 5X diary study data (Diary Study 1) and Nexus 5 diary study data (Diary Study 2 and Diary Study 3). We then trained a network of our proposed architecture on the Nexus 5X data for 20 epochs, and validated the network on the two separate Nexus 5 diary studies.

Validating on the first Nexus 5 diary study (Diary Study 2), our network had an accuracy of 78%, significantly lower than the accuracy observed in the single phone model case. The confusion matrix shown in Table IV.

TABLE IV
Confusion matrix of the network predictions on Diary Study 2 (Nexus 5).

	Actual			
Predicted	Backpack	Pocket	Hand	Table
Backpack	34.2%	1.6%	1.7%	13.1%
Pocket	.1%	18.5%	.0%	.0%
Hand	5.7%	.0%	.4%	.0%
Table	.0%	.3%	.0%	24.5%

Validating the second diary study, with the Nexus 5 (Diary Study 3), our trained network had a validation accuracy of 91%–an accuracy much closer to the cross validation accuracy demonstrated in the single phone model case. The confusion matrix is shown in Table V.

TABLE V
Confusion matrix of the network predictions on Diary Study 3 (Nexus 5).

	Actual			
Predicted	Backpack	Pocket	Hand	Table
Backpack	5.3%	4.0%	1.3%	.1%
Pocket	.6%	3.7%	.5%	.1%
Hand	.0%	1.4%	5.5%	.1%
Table	.9%	.3%	.0%	76.3%

The disparity in accuracy between the two different diary studies with the Nexus 5 was unexpected, since both were conducted within the same Nexus 5 device. We did not have time to explore the root cause for this disparity, so we cannot conclude definitely if a single network is applicable across phone models. The results from Diary Study 3 (91% accuracy) suggest that a linear calibration strategy may be effective, but further investigation is needed for definitive understanding.

VI. Conclusion

The work described in this paper describes the methodology in applying deep learning to the task of determining the state of a smartphone on the user's person. To do this, we do not require the user to be performing any specific action. Instead, we utilize data from the accelerometer and screen, which are both lightweight and readily available on Android phone models, to identify four common phone positions that span most of a user's phone behavior and appear distinguishable from the

sensor data. We show that great accuracy can be achieved when evaluating on a single phone model. Furthermore, we propose an accelerometer calibration strategy for standardizing phone accelerometer data across phone models, and show the potential generalization of a network trained on data from a single phone model to other phone models. However, our cross model results are inconclusive. For future work, we plan to strengthen these findings with the collection of more data, specifically the hand and pocket cases as well as investigate the classifier across phone models. We would also like to experiment with the 'smoothing' of outputs and determine if it can enhance accuracy.

ACKNOWLEDGEMENTS

This work could not have been completed without the help and guidance of Serge Egelman, Irwin Reyes, Michael McCoyd, and Jason Xinyu Liu.

This work was supported by NSF through grant CNS-1514457, by the Hewlett Foundation through the Center for Long-Term Cybersecurity, by gifts from Qualcomm and Google, and by Intel through the ISTC for Secure Computing.

REFERENCES

[1] H. Martín, A. M. Bernardos, J. Iglesias, and J. R. Casar, "Activity logging using lightweight classification techniques in mobile devices," in *Personal Ubiquitous Comput.* Springer-Verlag, 2013.

[2] R. Kumar, P. P. Kundu, D. Shukla, and V. V. Phoha, "Continuous user authentication via unlabeled phone movement patterns," in *2017 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE Computer Society, 2017.

[3] A. Primo, V. Phoha, R. Kumar, and A. Serwadda, "Context-aware active authentication using smartphone accelerometer measurements," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE Computer Society, 2014.

[4] K. Fujinami and S. Kouchi, "Recognizing a mobile phone's storing position as a context of a device and a user," in *Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer Berlin Heidelberg, 2013.

[5] E. Miluzzo, M. Papandrea, N. D. Lane, H. Lu, and A. T. Campbell, in *PhoneSense 2010*, 2010.

[6] K. Kunze, P. Lukowicz, H. Junker, and G. Tröster, "Where am i: Recognizing on-body positions of wearable sensors," in *LoCA*. Springer, Berlin, Heidelberg, 2005.

[7] L. Atallah, B. Lo, R. King, and G. Z. Yang, "Sensor positioning for activity recognition using wearable accelerometers," in *IEEE Transactions on Biomedical Circuits and Systems*. IEEE Computer Society, 2011.

[8] A. M. Khan, Y. Lee, S. Lee, and T. Kim, "Accelerometer's position independent physical activity recognition system for long-term activity monitoring in the elderly," in *Medical & Biological Engineering & Computing*. Springer-Verlag, 2010.

[9] D. Coskun, O. Incel, and A. Ozgovde, "Phone position/placement detection using accelerometer: Impact on activity recognition," in *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. IEEE Computer Society, 2015.

[10] J. Park, A. Patel, D. Curtis, S. Teller, and J. Ledlie, "Online pose classification and walking speed estimation using handheld devices," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012.

[11] J. Yang, E. Munguia-Tapia, and S. Gibbs, "Efficient in-pocket detection with mobile phones," in *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*. ACM, 2013.

[12] J. Wiese, S. Saponas, and A. Brush, "Phone position/placement detection using accelerometer: Impact on activity recognition," in *CHI 2013*. ACM, 2013.

[13] N. Wijerathne, S. Kv, S. Marakkalage, V. Beltran, C. Yuen, and H. Beng Lim, "Towards comfortable cycling: A practical approach to monitor the conditions in cycling paths," in *IEEE 4th World Forum on Internet of Things*, 2018.