Vol.2019-UBI-63 No.12
Vol.2019-ASD-15 No.12
2019/8/27

Robust Activity Data Collection with On-Device Recognition Using Long Short-Term Memory

Nattaya Mairittha[1,a)] Tittaya Mairittha[1,b)] Sozo Inoue[1,c)]

Abstract: This paper presents using a Long Short-Term Memory (LSTM)-based method for on-device deep learning recognition to alleviate the labeling effort and ground truth data collection in activity recognition systems using smartphone sensors. The novel idea behind this is that estimated activities are used as feedback for motivating users to collect accurate activity labels. To enable us to perform evaluations, we conduct the experiments with two conditional methods and evaluate it with the dataset gathered. The results show our proposed method has improvements in both data quality (i.e., the performance of classification several machine learning models) and data quantity (i.e., the number of data collected) that reflect our method could improve data collection for activity recognition systems.

Keywords: activity recognition; data collection; on-device deep learning recognition; smartphone sensors; user feedback

1. Introduction

In the field of ubiquitous computing, researches on human activity recognition technology using mobile sensors such as smartphones have been conducted [2]. Smartphone-based activity recognition systems aimed at physical activities recognition such as walking or running, based on mobile sensor data. The sensor data may be recorded directly on the subject such as by carrying smartphones that have accelerometers and gyroscopes [1]. Understanding what users are doing in the physical world allows the smartphone app to be smarter about how to interact with them. However, a central challenge in smartphone-based activity recognition is data annotation studies in order to assess the labels describing the current activity while this activity is still ongoing or recent to ensure that the dataset is labeled correctly. Collecting accurate labels (*annotation*) comes with a hefty price tag, in terms of human effort. Either to have the data labeled by third-party observers or self-labeling both are costly, time-consuming, tedious, and they have the risk of missing some of the activity labels. For instance, while employing observers to annotate labels, it is correct segments but costly [7]. By contrast, to get labels using self-labeling and experience sampling, it is lower cost, but incorrect segments [14]. Another method such as offline labeling it also takes a long time and privacy issues [8]. This is a general problem for all supervised learning methods, which not only require the presence of a big dataset but also require human supervision to annotate the dataset. Therefore, the quality and quantity of annotations can have a significant impact on the performance of the activity recognition systems. Hence, it is unavoidable to rely on the users and to keep them motivated to provide labels. To overcome the challenge of self-labeling [14], we introduce the idea of utilizing on-device deep learning recognition for optimizing activity data collection. The rapid performance increase of low-power processors and the huge demand of Internet of Things (IoT) applications brought new ways for deploying machine/deep learning models on edge devices. On-device machine learning by fusing the inertial sensors such as smartphones have been explored [18, 6, 4]. These findings allow the activity recognition system to be feasible to identify frequent behavioral patterns on edge devices; meanwhile, deep learning revolution in the field of machine learning tends to result in higher accuracy and performs exceptionally well on machine perception tasks on smaller devices with limited resources [12, 20, 11]. TensorFlow Lite [13] designed to enable it easy to perform machine/deep learning recognition on mobile, embedded, and IoT devices with low latency and a small binary size, "at the edge" of the network, instead of sending data back and forth from a server. Thus, we will exploit the power of on-device deep learning to provide estimated activities on a smartphone in order to optimize activity data collection.

In this paper, we want to show if we give estimated activities using on-device deep learning recognition through notifications as feedback to users while they are requested for labeling, we can improve data annotation tasks for activity recognition systems. The novel idea works by the

[1] Graduate School of Engineering, Kyushu Institute of Technology, 1-1 Sensui-cho, Tobata-ku, Kitakyushu-shi, Fukuoka, 804-8550, JAPAN
[a)] nattafahh@gmail.com
[b)] callmefons@gmail.com
[c)] sozo@brain.kyutech.ac.jp

Vol.2019-UBI-63 No.12
Vol.2019-ASD-15 No.12
2019/8/27

user will get estimated activities through notifications on a smartphone as feedback that motivates for efficient activity data collection. Estimated activities are automatically inferred by periodically reading short bursts of smartphone sensor data and processing them using on-device deep learning with a Long short-term memory (LSTM) model [16] without the model retrained. To evaluate this contribution, we trained the model for on-device deep learning with the open dataset [10], conducted the experiments with proposed and traditional methods to collect the evaluated dataset, validated the dataset collected using several machine/deep learning algorithms, and showed that our proposed method outperformed than the traditional method. In summary, the contribution of this paper is that **using on-device LSTM model to recognize activities in order to provide estimated activities as feedback for efficient activity data collection.**

2. Methods

In this section, we provide a descriptive view of the proposed on-device deep learning inference for efficient activity data collection system. **The architecture of this system is depicted in Figure 1.** The system is composed of several technical building blocks including the following: (1) to build an LSTM-based deep learning model used for on-device inference, (2) to collect accelerometer sensor data and activity labels efficiently, and (3) To provide estimated activities as feedback through smartphone notifications for efficient data collection. **The architecture of this system is depicted in Figure 1.**

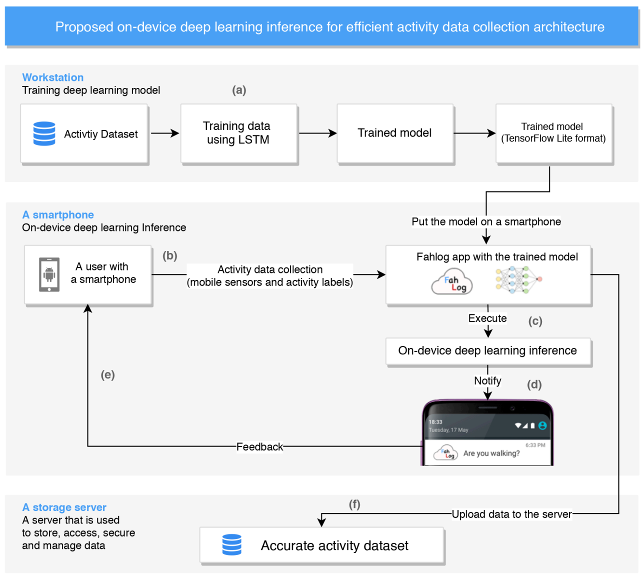

Proposed on-device deep learning inference for efficient activity data collection architecture
Workstation
Training deep learning model
(a)
Activtiy Dataset
Training data using LSTM
Trained model
Trained model (TensorFlow Lite format)
A smartphone
On-device deep learning Inference
Put the model on a smartphone
A user with a smartphone
(b)
Activity data collection (mobile sensors and activity labels)
Fahlog app with the trained model
Execute
(c)
On-device deep learning inference
(e)
Notify
(d)
Feedback
Are you walking?
A storage server
A server that is used to store, access, secure and manage data
(f)
Upload data to the server
Accurate activity dataset

Figure 1 The system architecture of the proposed on-device deep learning recognition for activity data collection

2.1 To build on-device recognition model

In this section, we propose how to build an LSTM model used for on-device recognition. We employ the open dataset provided by the Wireless Sensor Data Mining (WISDM) Lab [10] to build an activity recognition model for on-device deep learning recognition.

We use the WISDM dataset mentioned to build an activity recognition model. The reason why we first build the model by employing an existing dataset, and we then utilize it for our proposed on-device recognition method because we concern the issue that our system cannot draw any inferences for users since it has not yet gathered sufficient information. This problem usually occurs in computer-based information systems which involve a degree of automated data modeling. It is a well-known and well-researched problem, so-called the cold start problem [19]. The human activity recognition dataset built from the recordings of 29 subjects performing regular activities while carrying a waist-mounted smartphone with embedded inertial sensors. This dataset contains 1,098,207 examples and 6 attributes, including, user, activity, timestamp, x-acceleration, y-acceleration, z-acceleration without missing attribute, collected through controlled, laboratory conditions. There are 6 activity types of movement that we try to classify: Walking (38.6%), Jogging (31.2%), Upstairs (11.2%), Downstairs (9.1%), Sitting (5.5%), Standing (4.4%). The dataset's description is detailed in [10].

An LSTM takes many input vectors to process them and output other vectors. In our case, the "many to one" architecture is used: we accept time series of feature vectors (one vector per time step) to convert them to a probability vector at the output for classification. The inputs are raw signals obtained from multimodal-sensors, which is a discrete sequence of equally spaced samples $(x_1, x_2, ..., x_T)$, where each data point x_t is a vector of individual samples observed by the sensors at time t. These samples are segmented into windows of a maximum time index T and fed into LSTM-based deep learning model. Each generated sequence contains 200 training with 3 input parameters (3-axis accelerometer) per time steps. The model is trained for a maximum of 50 epochs by 2 fully-connected and 2 LSTM layers (stacked on each other) with 64 units each. We use rectified linear units (ReLUs) for the hidden layers to increase the robustness of the model as well as remove any simple dependencies between the neurons preventing over fitting, and use the dropout technique to avoid overfitting in our model, where a rectified linear unit has output 0 if the input is less than 0, and raw output otherwise. Also, we use an optimization algorithm called Adam [5] to minimize the cost function by backpropagating its gradient and updating model parameter. The core hyper-parameters explored in this model are listed in Table 1.

For validating the trained model against test data, we apportion the data into training and test sets, with an 80-20 split. After each epoch of training, we evaluate the performance of the model on the validation set. We select the epoch that showed the best validation-set performance and apply the corresponding model to the test-set. As a result, we opt the final epoch that the accuracy and weighted F1-score both are reached over 97% (0.975 and 0.972, respectively) and loss is hovered at around 0.2. Note that

Table 1 Some core parameter definitions for the training

Parameter	Value
LSTM layer	2 fully-connected
epochs	50
hidden layer units	64
output classes	6
input features per timestep	3 (acc_x, acc_y, acc_z)
timesteps per series	200
learning rate	0.0025
batch size	1024

the class distribution of the WISDM dataset has the sample imbalances among activity classes which can affect machine learning [9]. We could not collect more data that could balance our classes; however, we show the weighted F1-score for additional performance metric that is preferable if there is a class imbalance problem, not just only accuracy [17]. Since the smartphone is attached on the waist and each series to classify has just a 200 sample window, those predictions are extremely accurate. If we have a look at the confusion matrix of the model ' s predictions in Figure 2, we can see that our model performs real good. Although we can see some notable exceptions that there are difficulties in making the difference between Walking, Upstairs and Downstairs, the model is almost always able to identify the movement type on a smartphone correctly. The visual insight of the results are presented in Figure 2.

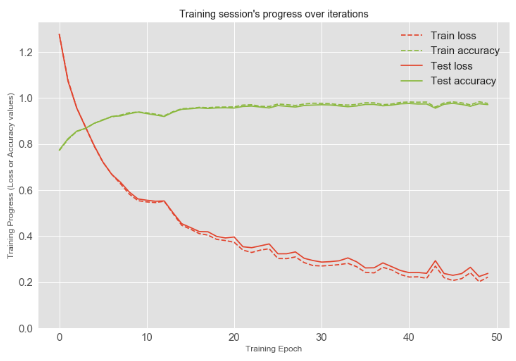


Figure 2 Training session's progress over iterations

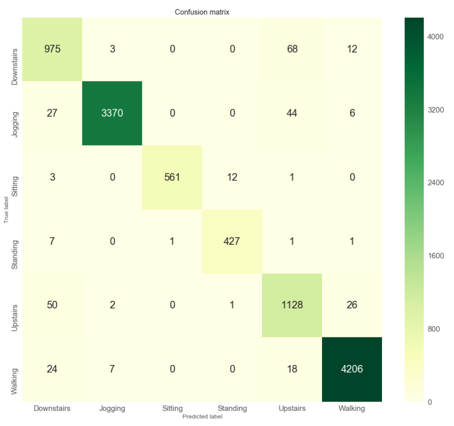


Figure 3 Training session's progress over iterations

2.2 To collect sensor data and activity labels

We request participants to carry a waist-mounted Android smartphone (Wiko Tommy3 Plus (Android 8.1)) with embedded inertial sensors, install the mobile app on smartphones to select and record their daily life activities from the list of predefined labels. Information about the demography of participants and the duration of the experiment are reported in Section 3. The mobile app is extended from our work [15] called **"FahLog"**, as shown in Figure. 4, when a user selects the activity, the labels for each activity class will be put into the right column as shown in Figure. 4-(d). Then the user has to record it by pushing the button to start and stop recording while they are carrying out the activity by following the steps as shown in Figure. 4-(d)(c)(b). Each time the user taps an activity label box, it will transition to before start (▶) → doing activity (⊙) → finish (✓) so a user can record the start and end of the activity. Since another activity may be performed while performing one activity, multiple activity labels can be started and ended in parallel. The activity labels can then be uploaded to the server when it is connected to the network. Otherwise, data will be stored on the smartphone until there is internet access. Moreover, we capture sensors and activity labels through smartphones to recognize activities using smartphone sensors continuously. Hence, we set the sampling rate of the app for the 'standard' settings of Android programming API, which is the slowest settings where they are sampled 200 milliseconds when they are not busy, then we take 1 minute time windows for calculating time windows, it is enough sampling rate for such data collection.

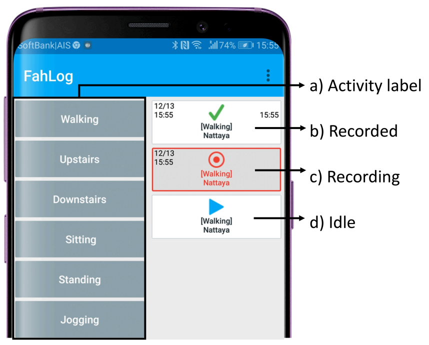
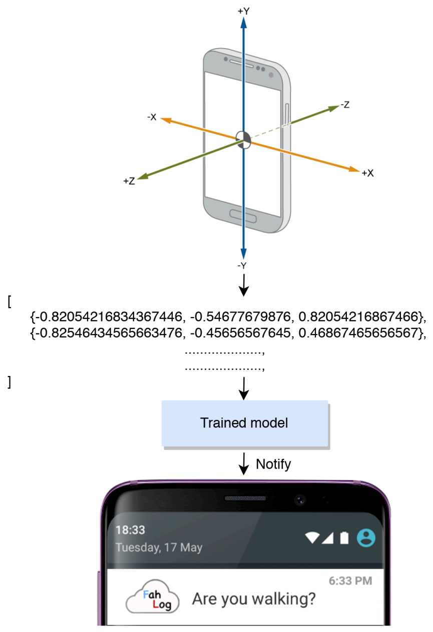


Figure 4 FahLog: A mobile app for collecting sensor data and activity labels

2.3 To provide estimated activities as feedback

We interpret the results that retrieve from model recognition. We use a list of probabilities that the model returned. We then meaningfully map them to relevant categories (activity classes) and present it on mobile notification to the user. Figure 5 presents an example of the results that are displayed on a notification. Note that to prevent excessive interruptibility and to optimize resources, we stop activity reporting if the device has been still for a while, and use low power sensors to resume reporting when it detects changes in the user's activity (e.g., changing from walking to running) with mean recognition time of 2846.0 ms. Also, when we put the deep learning model on the device and use a battery monitor application for Android smartphones to monitor the battery level, it increases energy consumption on its smartphone by 5% on average compared with the traditional manner without the on-device model. Therefore, showing that it works fast and does not waste a lot of energy.



```
[
    {-0.82054216834367446, -0.54677679876, 0.82054216867466},
    {-0.82546434565663476, -0.45656567645, 0.46867465656567},
                    ....................,
                    ....................,
]
```

Figure 5 Steps to show a estimated activity as feedback to a user

3. Experimental evaluation

In this section, we will evaluate the proposed method using a standard activity recognition chain [1] by comparing its performance with the traditional method, as shown in Table 2. We design and conduct the experiments, describe the dataset collected, preprocess the data collected, build the recognition model, and evaluate it.

3.1 Experimental setup

The participants were required to carry a waist-mounted Android smartphone, install the FahLog app on the phones, to select and record their activities from the list of predefined labels (depicted in Figure 4), get notifications, and submit data to our server. Each participant performs the experiments for 6 days. Table 2 shows the detail of the proposed method and traditional. We propose that if we give estimated activities using on-device deep learning recognition as feedback to users through smartphone notifications, they can improve activity data collection. Therefore, to compare our proposed method with the traditional method, we created notifications on smartphones that displays 2 different versions. Each version only differs in the user interface where the proposed method will show estimated activities using on-device deep learning recognition when the device detects changes in the user ' s activity. On the other hand, the traditional method will show messages "What are you doing?", without estimated activities once every 15 minutes. We also request the users click the push notifications sent to assure that the users have seen the notifications. Each participant will receive both 2 conditions, each of which will show 3 days. We randomly display the conditions for each participant to ensure that they are not affected by the day of experiments for each term. The participants were instructed with detailed instructions on how to do all process step by step using the same protocol provided. During data collection, the dataset was collected in the "wild" because the subjects provided data from their daily lives.

Table 2 Experimental design

Method	**Conditional detail**
Proposed	Receive notifications of estimated activities using on-device deep learning recognition
Traditional	Receive notifications with messages "What are you doing?" without estimated activities.

3.2 Data description

The dataset was collected between June 2019, from 6 subjects within an age bracket of 25-30 years, performing 1 of 6 regular activities (as shown in the left column of Table 3) while carrying a waist-mounted Android smartphone that recorded the movement data (accelerometers in smartphones). Note that we requested them to carry a smartphone in the same position as the WISDM dataset used to train the on-device recognition model. As a result, we gathered 713 activity labels from all participants.

Table 3 The number of activity labels collected

Activity class	**# labels**
Walking	247
Jogging	1
Sitting	249
Standing	153
Downstairs	36
Upstairs	27
Total	**713**

3.3 Activity recognition using smartphone sensors

Since we propose a standard activity recognition chain and a supervised learning approach for evaluations, we first preprocess the dataset collected and then evaluate it.

3.3.1 Data preprocessing

We put together the dataset by including 3-axis accelerometer sensor data and the activity labels on the smartphones without clock and time synchronization because the sensor and the labeling system are both in the same device. We used sliding windows of 1 minute with no overlapping. For each axis, average, standard deviation, maximum value and minimum value were extracted as features. Before data proceeding, we excluded missing values. As a result, we obtained multivariate data of 9,129 samples with 12 variables for feature vectors. Figure 10 shows the activity labels distribution of the data samples in our dataset. It is worth noting that the distribution is highly skewed, where some classes appear more frequently than others. Since imbalanced dataset can negatively influence the generalization and reliability of supervised learning algorithms, we employed the SMOTE algorithm: Synthetic Minority Over-sampling Technique as presented in [3] (an oversampling technique that creates new synthetic data samples in the minority classes, varying the features values of the existing data points based on their k nearest neighbors in the feature space) in order to balance our dataset. By upsampling the size of training and testing datasets separately.

3.3.2 Evaluation method

In this section, we present the effectiveness of the proposed method when we give estimated activities using on-device deep learning through smartphone notifications. The experiment was designed to test the performance of our classifier for a user-dependent scenario. In this case, the classifiers were trained and tested for each individual with her/his own data, and average accuracy and was computed. We show that several machine algorithms and LSTM-based deep learning algorithm have improvements in the classification performance. We also present the proposed method has improvements in the number of data collected compared to the traditional method. To evaluate the proposed method using a technique of supervised learning algorithm for multiclass classification. We trained each participant separately using one deep learning classifier and several standard machine learning classifiers, including LSTM in the same way of the on-device model trained, Logistic Regression (LR), Linear discriminant analysis (LDA), k-nearest neighbors (KNN), Decision tree (CART), Naive Bayes (NB), Support-vector machine (SVM), and Random Forest (RF).

To test the model's ability we used stratified k-fold cross-validation. The folds are made by preserving the percentage of samples for each class to ensure each fold is a good representative of the whole. To account for label imbalance, the model performance was presented using the weighted average of precision, recall, F1-score of each class for the multiclass task. So the average is weighted by the support, which is the number of samples with a given label.

4. Results

Following the evaluation approach discussed above, we report our results of the validation together with a discussion of such results. We show the proposed method has improvements in data quality (the classification performance) compare to the traditional method. The average classification performance of all models results are shown in Figure 6. We also present the proposed method has improvements in data quantity (the number of data collected) compare to the traditional method. Figure 10 shows the number of collected activity labels for both methods.

4.1 Quality of collected activity data

Figure 6 shows F1-score, precision, and recall performance results of all machine learning models were improved with our proposed method compared to the traditional method. The F1-score was improved from 0.6240 to 0.7620 **(+0.138)** The precision was improved from 0.6440 to 0.7802 **(+0.136)** The recall of improved from 0.6366 to 0.7677 **(+0.131)**

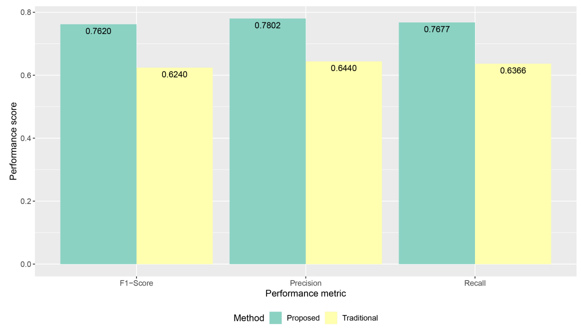


Figure 6 The average classification performance of all models for each method

Figure 7 shows F1-score performance results of all machine learning models were improved with our proposed method compared to the traditional method. The F1-score of CART was improved from 0.657 to 0.770 **(+0.113)** The F1-score of KNN was improved from 0.667 to 0.801 **(+0.134)**. The F1-score of LDA was improved from 0.604 to 0.766 **(+0.162)**. The F1-score of LR was improved from 0.623 to 0.778 **(+0.155)**. The F1-score of LSTM was improved from 0.657 to 0.783 **(+0.126)**. The F1-score of NB was improved from 0.472 to 0.606 **(+0.134)**. The F1-score of RF was improved from 0.694 to 0.815 **(+0.121)**. The F1-score of SVM was improved from 0.623 to 0.775 **(+0.152)**.

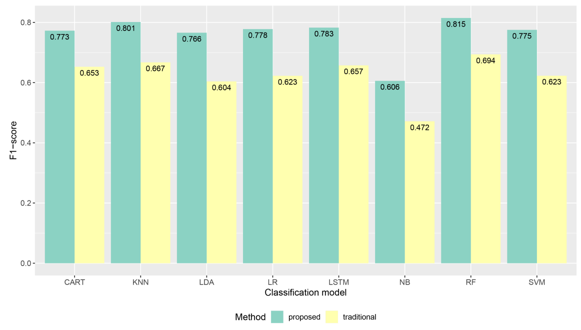


Figure 7 The F1-score performance results of several machine learning models

Figure 8 shows precision performance results of all machine learning models were improved with our proposed method compared to the traditional method. The precision of CART was improved from 0.679 to 0.805 **(+0.126)**. The precision of KNN was improved from 0.665 to 0.793 **(+0.128)**. The precision of LDA was improved from 0.611 to 0.762 **(+0.151)**. The precision of LR was improved from 0.616 to 0.759 **(+0.143)**. The precision of LSTM was improved from 0.675 to 0.803 **(+0.128)**. The precision of NB was improved from 0.593 to 0.757 **(+0.164)**. The precision of RF was improved from 0.698 to 0.813 **(+0.114)**.

The precision of SVM was improved from 0.619 to 0.738 **(+0.119)**.

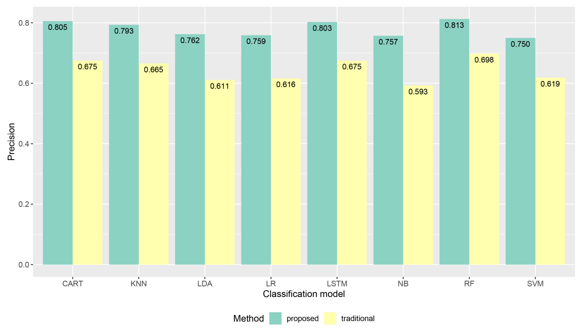


Figure 8 The precision performance results of several machine learning models

Figure 9 shows recall performance results of all machine learning models were improved with our proposed method compared to the traditional method. The recall of CART was improved from 0.648 to 0.746 **(+0.098)**. The recall of KNN was improved from 0.681 to 0.814 **(+0.133)**. The recall of LDA was improved from 0.626 to 0.780 **(+0.154)**. The recall of LR was improved from 0.657 to 0.806 **(+0.149)**. The recall of LSTM was improved from 0.657 to 0.779 **(+0.121)**. The recall of NB was improved from 0.459 to 0.556 **(+0.097)**. The recall of RF was improved from 0.696 to 0.821 **(+0.137)**. The recall of SVM was improved from 0.677 to 0.833 **(+0.156)**.

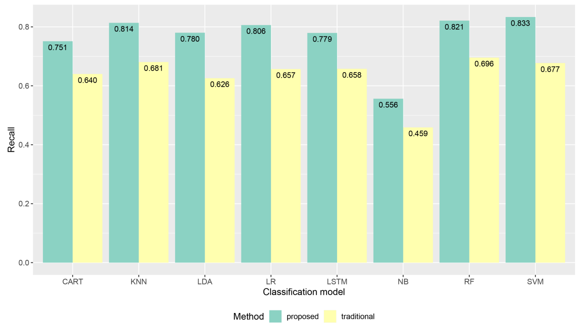


Figure 9 The recall performance results of several machine learning models

Table 4 shows all users improve average F1-score, average precision, and average recall performances of all machine learning models with our proposed method compared to the traditional method.

4.2 Quantity of collected activity data

Figure 10 shows the number of collected activity labels was increased with our proposed method. The number of activity labels was increased from 311 to 402 **(+91)** compared to traditional method. Table 5 shows the number of labels of each activity class by comparing the proposed and traditional. While some activity classes have more labels with the proposed method, only one class has fewer labels with the proposed method. The number of walking labels

Table 4 The average classification performance of all models for each user

User	Method	F1-score	Recall	Precision
98	proposed	0.7778	0.7756	0.7973
98	traditional	0.7127	0.7139	0.7391
98	**Improvement**	**+0.0651**	**+0.0616**	**+0.0582**
99	proposed	0.7009	0.7119	0.7156
99	traditional	0.4442	0.4830	0.4605
99	**Improvement**	**+0.2567**	**+0.2289**	**+0.2551**
101	proposed	0.8700	0.8774	0.8727
101	traditional	0.6449	0.6619	0.6701
101	**Improvement**	**+0.225**	**+0.215**	**+0.203**
103	proposed	0.7693	0.7663	0.7950
103	traditional	0.6490	0.6685	0.6584
103	**Improvement**	**+0.120**	**+0.098**	**+0.137**
104	proposed	0.7881	0.7954	0.8120
104	traditional	0.6333	0.6223	0.6705
104	**Improvement**	**+0.155**	**+0.173**	**+0.142**
105	proposed	0.6658	0.6794	0.6888
105	traditional	0.6600	0.6702	0.6654
105	**Improvement**	**+0.006**	**+0.01**	**+0.023**

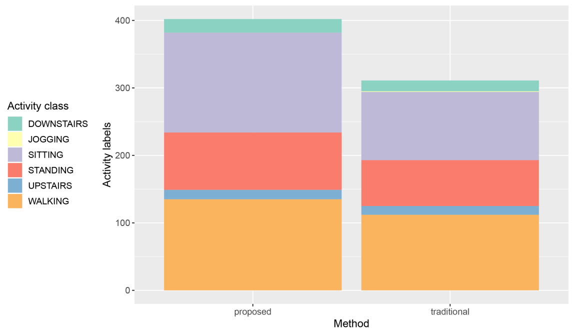


Figure 10 The number of activity labels for each method

was increased from 112 to 135 **(+23)**. The number of upstairs labels was increased from 13 to 14 **(+1)**. The number of standing labels was increased from 68 to 85 **(+17)**. The number of sitting labels was increased from 101 to 148 **(+47)**. The number of downstairs labels was increased from 16 to 20 **(+4)**. The number of jogging labels was decreased from 1 to 0 **(-1)**.

Table 5 The number of activity labels of each activity class for each method

Activity class	Proposed	Traditional	Improvement
Walking	135	112	+23
Upstairs	14	13	+1
Standing	85	68	+17
Sitting	148	101	+47
Downstairs	20	16	+4
Jogging	0	1	**-1**
Total	**402**	**311**	**+91**

5. Discussion and Future directions

By evaluating with the dataset and comparing with the traditional method, the results reflect that our proposed method has improvements in data quality for all machine learning models evaluated and data quantity that indicate improvements in activity data collection. What we have found most interesting is that all users improve quality of activity data collection with the proposed method, as shown in Table 4. While this study enabled us to improve activity data collection effectively, there are some limitations that we would like to point out and reference in the future. While RF achieves the highest F1-score at 81.5%, LDA has the most improvements by 16.2%. RF achieves highest the precision at 81.3%, NB has the most improvements by 16.4%. SVM achieves highest the recall at 83.3% and also has the most improvements by 15.6%. While this study enabled us to improve activity data collection effectively, there are some limitations that we would like to point out and reference in the future.

First, while we notified information about estimated activity when the user is currently doing the activity, it might be necessary to design both our mobile app and our recognition model to identify when a user starts or stops a particular activity, such as walking, biking, or driving (e.g., detect when users start and end an activity). For activity recognition systems, it is crucial to collect correct segments data. In other words, we need a labeled sequence of activities (i.e., the start and finish times of the events). Hence, if the app can be used to detect changes in the user's activity, we can also deliver this information as feedback to the user for better activity data collection. Researchers may consider this idea for other purposes, for example, an app subscribes to a transition in activities of interest and notifies the user only when needed (e.g., the app notifies driving when a user starts driving and mute until the user stops driving).

Second, we used the WISDM dataset to train our deep learning model. Hence, the smartphone's position is limited for activity data collection in our experiment as we have to put the smartphone in the same position. If the smartphone's position and/or orientation is discrepant from theirs, the on-device recognition will not be correct. Consequently, considering to collect training dataset by ourselves will be vital. Also, we can collect more data to make the samples balance among activity classes. Furthermore, while we applied 3-axis accelerometer for training the recognition model and inferring on a smartphone device, other smartphone sensors would be useful for more accurate recognition. For example, adding gyroscope can help indicate orientation.

Third, we run the trained model on a device without retraining. When designing activity recognition (machine learning) systems, it is crucial to understand how our data is going to change over time. A well-architected system should take this into account, and a plan should be put in place for keeping our models updated. There are several ways to retain the model, for example, manual retraining by training and deploying your models with fresh data using the same process you used to build your models in the first place or continuous learning by using an automated system to evaluate and retrain your models continuously (e.g., hosting a model on the cloud). However, retraining the model to maintain machine learning systems would be challenging for research questions in future work, for example, *how do you ensure our predictions continue to be accurate? how do you keep your models up-to-date with new training data?*

Fourth, as our proposed method can be applied for several algorithms, but the main on-device recognition model that drove our work – that LSTM-based deep learning model. If a training model were evaluating using other deep learning methods, such as CNN, then there would be value in expanding – *why LSTM? What are the challenges that are different from other methods? Which method is best?*

Finally, we plan to evaluate the method with long-term data collection and more diverse samples, find data insights as well as find out the correlations between accuracy, the number of activity labels and classes to show whether and how strongly pairs of variables are related. For example, *do notifications affect the number of activity labels or do notifications affect the number of activity classes?* Answering these questions, it would also be helpful to understand user motivations and support activity data collection further. Likewise, we have seen that although the number of activity labels is increased with our method, not all activity classes (see in Table 5). Therefore, analyzing the data collected more deeply will be useful to understand correlation and causation.

6. Conclusion

We have proposed on-device deep learning recognition using LSTM for providing estimated activities as feedback for activity data collection in smartphone-based activity recognition. By evaluating with 713 activity labels and mobile sensor data collected, the results indicate that our proposed method has improvements in F1-score, precision, and recall for all machine learning classifiers compared to the traditional method. Moreover, the proposed method has an increment in the number of activity labels compared to the traditional method. There are several challenging areas that we see as ripe for next steps, for instance, exploiting on-device deep learning recognition for detecting changes in the user ' s activity, collecting own training data for on-recognition recognition model, adding more sensor types for training activity recognition models, retraining an on-device model, showing and comparing with other deep learning methods as well as collecting more data and analyzing it deeply. We will leave this for future work.

References

[1] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *International conference on pervasive computing*, pages 1–17. Springer, 2004.

[2] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):33, 2014.

[3] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[4] Paul Föckler, Thomas Zeidler, Benjamin Brombach, Erich Bruns, and Oliver Bimber. Phoneguide: museum guidance supported by on-device object recognition on mobile phones. In *Proceedings of the 4th international conference on Mobile and ubiquitous multimedia*, pages 3–10. ACM, 2005.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[6] Tao Guan, Yunfeng He, Juan Gao, Jianzhong Yang, and Junqing Yu. On-device mobile visual location recognition by integrating vision and inertial sensors. *IEEE transactions on multimedia*, 15(7):1688–1699, 2013.

[7] Sozo Inoue, Naonori Ueda, Yasunobu Nohara, and Naoki Nakashima. Mobile activity recognition for a whole day: recognizing real nursing activities with big dataset. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 1269–1280. ACM, 2015.

[8] Shian-Ru Ke, Hoang Thuc, Yong-Jin Lee, Jenq-Neng Hwang, Jang-Hee Yoo, and Kyoung-Ho Choi. A review on video-based human activity recognition. *Computers*, 2(2):88–131, 2013.

[9] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[10] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.

[11] Joe Lemley, Shabab Bazrafkan, and Peter Corcoran. Deep learning for consumer devices and services: Pushing the limits for machine learning, artificial intelligence, and computer vision. *IEEE Consumer Electronics Magazine*, 6(2):48–56, 2017.

[12] He Li, Kaoru Ota, and Mianxiong Dong. Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE Network*, 32(1):96–101, 2018.

[13] TensorFlow Lite. Accessed: Feb. 28, 2019.

[14] Nattaya Mairittha and Sozo Inoue. Gamification for high-quality dataset in mobile activity recognition. In *International Conference on Mobile Computing, Applications, and Services*, pages 216–222. Springer, 2018.

[15] Nattaya Mairittha, Tittaya Mairittha, and Sozo Inoue. A mobile app for nursing activity recognition. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pages 400–403. ACM, 2018.

[16] Francisco Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[17] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[18] Reza Rawassizadeh, Elaheh Momeni, Chelsea Dobbins, Joobin Gharibshah, and Michael Pazzani. Scalable daily human behavioral pattern mining from multivariate temporal data. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):3098–3112, 2016.

[19] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[20] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.