Android Integration of a Machine Learning Pipeline for Human Activity Recognition

Viktor Srbinoski, Daniel Denkovski, Emilija Kizhevska, Hristijan Gjoreski
Faculty of Electrical Engineering and Information Technologies,
Ss. Cyril and Methodius University in Skopje, N. Macedonia, Jozef Stefan Institute, Slovenia
viktor_srbinoski@hotmail.com, danield@feit.ukim.edu.mk, emilija.kizhevska@ijs.si, hristijang@feit.ukim.edu.mk

ABSTRACT

In the last decade, smartphones have seen a serious growth in the processing power. Coupled with greater affordability this has led to a worldwide smartphone ubiquity. Alongside the advances in processing and battery technology, there are great advances in sensor technology as well, and every smartphone today comes equipped with multiple sensors: accelerometer, gyroscope, magnetometer etc. The sensory data is already being used in a variety of applications, among which several focus on the human activity recognition. In this paper, we propose a smartphone Android integration of a machine learning pipeline for recognizing human activities. The proposed approach uses the 3-axis accelerometer in the smartphone, processes the data in real time, and then a machine learning model recognizes the user's activities in real time: walking, running, jumping, cycling and standing still. The proposed Recurrent Neural Network model and its machine learning pipeline are developed on a publicly open activity dataset, which are then implemented into the Android application and once again validated on a dataset recorded with a smartphone itself.

KEYWORDS

Human activity recognition, machine learning, Android integration, Tensorflow Light, recurrent neural network, accelerometer, magnitude.

1 INTRODUCTION

Human Activity Recognition (HAR) is the process of examining data from one or multiple sensors and determining which (if any) activity is being performed. The sensors are traditionally placed on key points on the human body and contain composite data (accelerometer, gyroscope, magnetometer data, etc.). Advances in sensor technology have made sensors more compact and precise over the years, but most importantly more affordable. Today these sensors can be found in the standard package of any smartphone.

The purpose of this paper is to leverage these smartphone sensors to perform HAR in real time, by utilizing an Android application which continuously reads its own sensor data, instead of using the traditional dedicated wearable sensors. The premise is that the smartphone sensors have reached the required quality to be comparable to the wearable sensors in accuracy [1]. The benefit of this approach is that it is much more convenient to use smartphone sensors for the common user, as smartphones have become ubiquitous.

Human activity recognition is a popular topic, which has been worked on extensively in the recent years [2]. Practical applications for HAR are mainly in improvement of the quality of life and medicine. A great example of HAR models being used in medicine can be found in paper [3], which focuses on fall detection mainly for the elderly population.

Using dedicated wearable sensors to recognize activities is the most common approach. Smartwatch is usually equipped with the same sensors as the smartphones and has a much more fixed position on the body (tightly around the wrist). The drawback is that the arms are more prone to random movement which introduces noise into the system and makes HAR more difficult. A detailed analysis on these issues can be found in paper [4].

Using data from smartphone sensors to train models for HAR has also been explored recently in [5], where a deep neural network is trained on the data from multiple sensors on the smartphone. In our study we go a step further and analyze and compare a simplified subset of the sensor data (only accelerometer magnitude) - which allows us to have a model that will work regardless of the smartphone orientation and to have a simple yet effective method of integrating a model into an Android application.

We propose an Android integration of a Machine Learning (ML) pipeline for recognizing human activities in real time on a smartphone. In particular, the proposed approach uses the 3-axis accelerometer in the smartphone, processes its data in real time, and then the ML model recognizes the user's activities: walking, running, jumping, cycling and standing still. The proposed Recurrent Neural Network (RNN) model and its machine learning pipeline are developed on a publicly open activity dataset, then implemented into an Android application, which finally, is once again evaluated on a dataset recorded with a smartphone itself. Additionally, as part of this study we release an Android application [6], which can be used by other researchers to easily gather data with a smartphone and as a practical demonstration of how to integrate an ML model with an Android application and use the built-in accelerometer data.

2 DATASET

The models were trained on a publicly available dataset which was originally used to evaluate the impact of sensor placement in activity recognition [7]. The dataset consists of

wearable sensor readings from 17 healthy subjects which perform any of 33 different activities. There are a total of 9 wearables placed on the body: two on each arm and leg, and one on the back. Each wearable sensor reads 13 values with a frequency of 50Hz: three for acceleration, three for rotation, three for magnet flux vector and four for orientation in quaternion format. This brings the total amount of readings to 117 per frame (9 wearable sensors with 13 values each). Out of all these measurements only six are used: the **3 accelerometer values** from each of the two upper leg sensors (left and right). These sensors are chosen as they are approximately at the location where a smartphone would be (in a side pocket). Additionally, the magnitude of each sensor is added as an additional feature, calculated as:

$$magnitude = \sqrt{acc_x{}^2 + acc_y{}^2 + acc_z{}^2} \quad (1)$$

Due to the position of the sensors, recognizing motion mainly expressed with the upper torso and arms is impossible, so the dataset is truncated to only activities that are dependent on the legs: walking, running, jumping, cycling and standing still.

3 METHODOLOGY

In order to adapt the dataset to fit the needs of this application, certain preprocessing and feature extraction is performed, described in detail in the following subsections.

3.1 Preprocessing and segmentation

The dataset contains a disproportionate number of readings for standing still in comparison to all other activities. To correct this a random under-sampling is performed (only 5% of the standing still data is used). Additionally, similar activities are grouped together, namely jogging and running are grouped together as running, and jumping upwards, jumping front and back, jumping side to side, and jump rope are grouped as jumping. The resulting distribution of data is illustrated on Figure 1, with running having the most amount of data (1760s), and cycling having the least (860s).

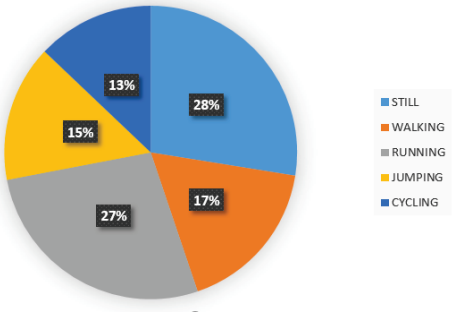

Figure 1 Activity distribution after selection

Once selection has been performed, the data is grouped into **3-second windows**. Since the data is collected at a frequency of 50Hz, each window contains 150 records.

3.2 Feature extraction

After the data has been split into 3-second windows, five statistical features are calculated per window. The first two are the **mean** and the **standard deviation** of the 150 values in the window. The three additional statistical features are:

- **Mean first-order difference**: average difference between consecutive values in the window. Computed by first creating a list of first-order differences between consecutive values in the window and then calculating the mean of this list.
- **Mean second-order difference**: average difference between consecutive elements in the list of first-order differences.
- **Min-max difference**: difference between the minimum and maximum value in the window.

The feature extraction is performed on every sensor (x, y, z axis and magnitude on both accelerometers, left and right), which gives a total of 40 features. The features are then separated into three datasets: *left accelerometer, right accelerometer* with 20 features each, and *combined accelerometers* which contains the data from both the left accelerometer and right accelerometer datasets, by matching the respective features (e.g., x-axis on the left accelerometer and x-axis on the right accelerometer are treated as the same feature: x-axis), thus the combined accelerometers dataset also contains 20 features, but it is twice as long.

To compare the effectiveness of a simplified version of the model that is orientation independent, a second version of the dataset is created. This dataset uses only the features extracted from the **magnitudes** of both accelerometers (5 features each). It is further split into three parts: *magnitude-only left*, *magnitude-only right* and *magnitude-only combined*, each containing five features.

3.3 ML Models

Multiple ML models were evaluated, such as K-NN, Linear SVM, Random Forest, Naïve Bayes and Neural Networks (DNN and RNN).

Ultimately the **RNN model** had the best performance. A simple RNN was chosen as the ML model for this application. The model is created using Keras and contains two RNN layers with 512 nodes each and tanh activation function. The final decision layer is a Dense layer with 5 nodes and a softmax activation function. It is trained for 100 epochs with a sparse categorical cross entropy activation function.

4 EXPERIMENTS

With the dataset prepared, the following experiments were conducted:

- Accuracy comparison between magnitude-only and full-featured versions of the dataset.
- Evaluation of models trained on data from the left accelerometer and evaluated on data from the right, and vice-versa.

4.1 Evaluation and metrics

The models were evaluated using **K-fold Cross-Validation**, where K is equal to the number of subjects, and in each iteration a different subject's data is used as the validation set. Splitting the data this way ensures that the test

data and train data do not both contain windows from the same subject (as consecutive windows from the same subject are very similar). Instead, when using the data from a separate subject as a validation set, a good estimate can be made of how the model will behave when a never seen before person's data needs to be evaluated.

In every iteration of the K-fold Cross-Validation a confusion matrix is generated from the predicted values. From there the precision and recall are calculated for every activity as well as the overall accuracy. These metrics are compiled for every iteration and the average values across all iterations form the overall evaluation of the model.

4.2 Results

Initially nine models were considered and evaluated on both the full-featured dataset and the magnitude-only dataset (for combined accelerometers). The results are illustrated on Figure 2, sorted by accuracy.

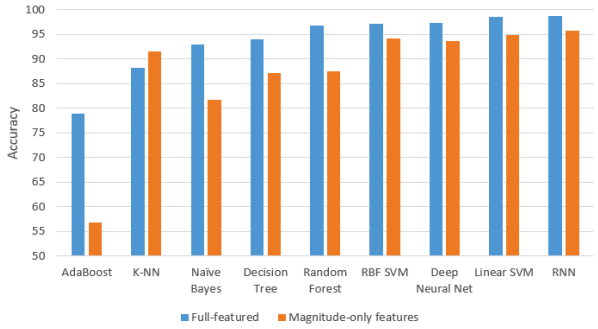

Figure 2 Accuracy comparison of all inspected ML models

The accuracy of the models with full features was expectedly higher than the magnitude-only version, with the drop in accuracy being on average 7% (K-NN being the exception with an increase in accuracy of 2%). The RNN had the highest accuracy in both cases, with 98.8% on the full-featured dataset and 95.8% on the magnitude-only dataset. Therefore, the following results focus on the RNN model.

The comparison in accuracy between the full-featured and magnitude-only versions was made on all three datasets (left, right and combined). The results for the RNN are displayed on Figure 3.

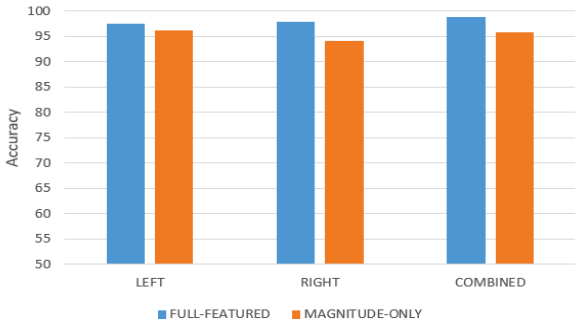

Figure 3 Comparing full-featured and magnitude-only datasets

The average drop in accuracy for the RNN was 3% which is well within acceptable boundaries. As a side note, the right side in general seems to show slightly weaker results, however at most this is 1.5% (when comparing the left simplified and right simplified sets) which could be due to random noise.

In order to evaluate if the model takes in a bias from the side on which it is trained or if the sides carry an intrinsic difference, the model was trained on one side and evaluated on the other. This was done twice, trained on left and evaluated on right, and trained on right and evaluated on left. The results are displayed on Figure 4, along with a control set which was trained and evaluated on the same side.

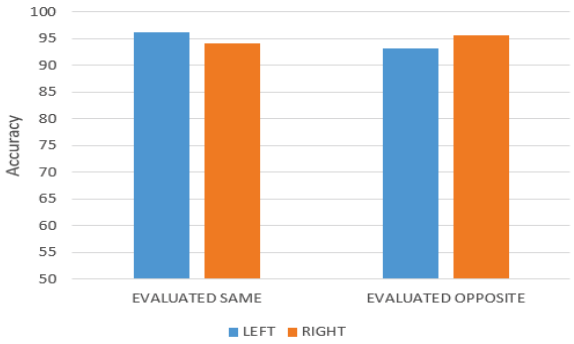

Figure 4 Comparison between same and opposite side evaluation

The accuracy differences are within 2% which is negligible, and in the case of the right accelerometer dataset, evaluating on the left actually increased the overall accuracy. This is due to the slight difference in quality between the left and right sides, and not due to switching sides when evaluating.

These results suggest that there is no significant side bias in the models and thus the activity recognition will work regardless of on which side the smartphone is located. This in addition to the simplified model's independence from orientation make it the ideal choice for integrating with a smartphone.

5 ANDROID INTEGRATION

In order to integrate with an Android smartphone device, the magnitude-only model with combined accelerometers was converted into a tflite format using the Tensorflow Lite library, which is the most commonly used library for artificial intelligence in Android. The converted models are then added in the file structure of an Android application which reads them into memory when it starts up and uses them in real time to recognize activities.

All Android devices come equipped with accelerometers (along with many other sensors) and they can be accessed with the built-in class SensorManager, which is part of the default library: android.hardware. The data read by the SensorManager is on a by-axis basis and in the standard unit of m/s^2. The orientation of the x, y and z axis is illustrated in Figure 5.

The frequency with which the sensor records data is adjustable, with the tradeoff being higher quality data vs lower battery consumption. In our implementation, the sensor delay is set to 20ms between reads (50Hz frequency).

Since there is no way to predict which way the smartphone will be oriented in the pocket, the magnitude of

the accelerometer is the only thing that is used in the feature calculation. The magnitude readings are kept in memory until 150 samples are accumulated (exactly 3s), which is the size of the window used in the training of the models. Then the same statistical features are calculated on the collected window: mean, std. deviation, mean first-order and second-order differences, min-max difference. These values are then placed in a tensor and it is sent as the input into the model, which is also kept in memory (in the form of an object). The output of the model is also a tensor (the output layer which has a softmax activation function), which is then converted into a single result (the node with the highest value) and is displayed on screen.

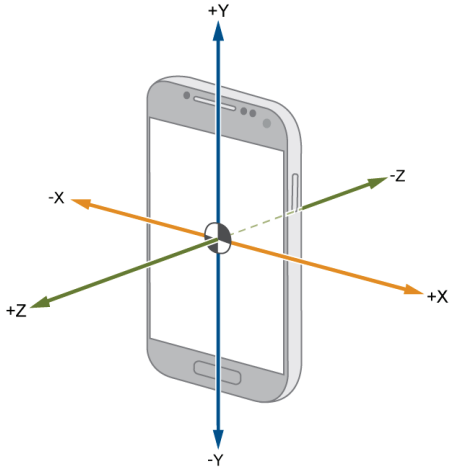

Figure 5 Accelerometer axis orientation in smartphones

Since 150 samples need to be accumulated before the features are calculated and the model is called to make the prediction, there is the side effect that the displayed value on screen is 3s behind (in other words the current activity the user is doing will be displayed in 3s). All the data read by the accelerometer along with the prediction and a timestamp and is kept in memory (a single entry will contain all the calculated features from the 3-second window, the model prediction and a timestamp). The user can choose to export this data to csv and use it as a dataset.

The model was evaluated on a practically collected dataset with a Samsung Galaxy s20 smartphone (5 minutes of each activity). The predicted value was compared to the actual activity by cross-referencing the timestamps (the activities were performed at specific times), and a confusion matrix was created, from which the precision, recall and f1 score, as well as overall accuracy, was calculated. The results are displayed on Figure 6.

	Precision	Recall	F1
still	0.929	0.939	0.931
walking	0.902	0.901	0.893
running	0.891	0.910	0.884
jumping	0.944	0.870	0.885
cycling	0.778	0.622	0.629

Figure 6 Precision, recall and f1 score results on the practically collected dataset on a Galaxy s20 smartphone

The overall accuracy of the model was **90.2%,** which is a noticeable drop from the 95.8% evaluated from the original training dataset. This is expected, as there is a certain amount of noise introduced to the system from the fact that the smartphone is not fixed in place as rigidly as the wearables.

6 CONCLUSION

This paper presented a practical way of training and implementing a HAR model in an Android application, along with solving the practical issues of reading smartphone accelerometer data such as unpredictable orientation and whether it is kept on the left or right side.

To determine whether there is an intrinsic difference between the left and right side or whether the models develop a side bias, an experiment was conducted where models were evaluated on the opposite side of where they were trained, and it was determined that no such bias existed.

To gain independence from orientation, a simplified dataset was created which used only the magnitude readings. Training on this dataset resulted in an expected drop in accuracy, but within an acceptable margin.

An RNN was trained on the magnitude-only dataset and integrated into an Android application which reads the accelerometer data and calculates the features in real time. The calculated features are used as an input for the model, which then outputs the predicted activity, and is subsequently shown on screen.

The sensors in the used smartphone did prove to be of a comparable quality to the wearable sensors as the model successfully recognized activities recorded with smartphone sensors with a solid accuracy of 90.2%, even though it was trained on a dataset from wearable sensors.

ACKNOWLEDGEMENT

This research was partially supported by the WideHealth project - EU Horizon 2020, under grant agreement No 952279.

REFERENCES

[1] Patima Silsupadol, Kunlanan Teja, Vipul Lugade, “Reliability and validity of a smartphone-based assessment of gait parameters across walking speed and smartphone locations: Body, bag, belt, hand, and pocket”, Gait & Posture, Volume 58, 2017,

[2] O. D. Lara and M. A. Labrador, "A Survey on Human Activity Recognition using Wearable Sensors," in IEEE Communications Surveys & Tutorials, vol. 15, no. 3, pp. 1192-1209, Third Quarter 2013

[3] Kozina, S., Gjoreski, H., Gams, M., & Luštrek, M. (2013, September). Efficient activity recognition and fall detection using accelerometers. In International competition on evaluating AAL systems through competitive benchmarking (pp. 13-23). Springer, Berlin, Heidelberg.

[4] Gjoreski, M.; Gjoreski, H.; Luštrek, M.; Gams, M. How Accurately Can Your Wrist Device Recognize Daily Activities and Detect Falls? Sensors 2016, 16, 800. https://doi.org/10.3390/s16060800

[5] Charissa Ann Ronao, Sung-Bae Cho, Human activity recognition with smartphone sensors using deep learning neural networks, Expert Systems with Applications, Volume 59, 2016, ISSN 0957-4174

[6] https://github.com/ViktorSrbinoski/SmartphoneActivityRecognition

[7] Oresti Banos, Miguel Damas, Hctor Pomares, Ignacio Rojas, Mt Attila Toth, and Oliver Amft. A benchmark dataset to evaluate sensor displacement in activity recognition. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing, UbiComp ’12, pages 1026–1035, New York, NY, USA, 2012. ACM.