Fitness Monitoring Using Smartphone Accelerometers

Niraj Parajuli

Earlham College

Background

Fitness trackers like Fitbit and smart watches have been steadily gaining popularity over the past few years. However, instead of investing money on purchasing additional devices, we can harness the capability of our smartphones which come with sensors like accelerometers, gyroscopes and magnetometers that can be used to detect user activity.

The field of human activity recognition is concerned with predicting what activity an user might be engaged in based on sensor readings. Thus, human activity recognition presents us with a marvelous way to monitor fitness using smartphones.

Project Description and Goals

This project aims to take advantage of recent advancements in machine learning and human activity recognition to build an Android application that can help users monitor their fitness without incurring additional cost.

Methodology

Figure 1 below shows the workflow of this project. Firstly, a Convolutional Neural Network (CNN) based classifier was trained by using data made available by Wireless Sensor Data Mining (WISDM). Figure 2 depicts a bar graph of number of examples for each labelled activity. Similarly, Figure 3 and 4 show a plot of accelerometer readings recorded in the dataset when the user is walking and sitting respectively.

Accelerometer readings is gathered by the application at regular intervals. It is then preprocessed and fed into the trained classifier. The predicted activity outputted by the classifier is added to an SQLite database stored on the phone. When the user requests to view his statistics, the database is queried and a pie chart is generated.

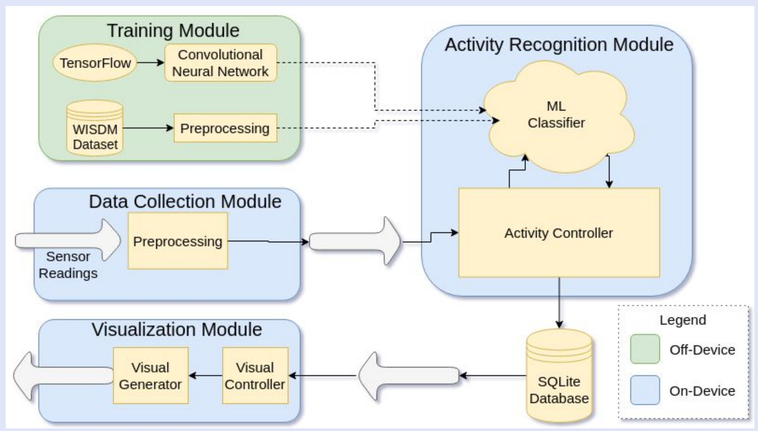

Figure 1. Data Flow Diagram

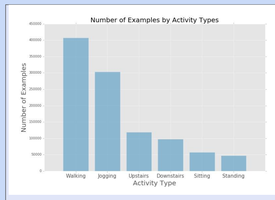

Figure 2. Bar graph showing number of examples per activity type

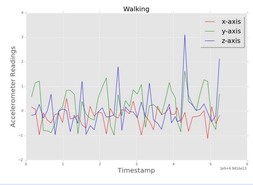

Figure 3. A plot of accelerometer readings for walking

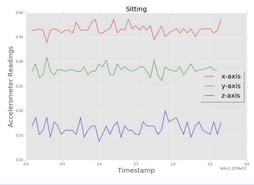

Figure 4. A plot of accelerometer readings for sitting

Results

The CNN classifier used in this project was trained for 50 epochs. Figure 5 shows how test and training accuracy increased over training epochs. It also shows how test and training losses decreased over training epochs.

Figure 5. A plot of training and test losses and accuracy against training epoch

The loss function chosen for this classifier was cross-entropy loss, which can bring logarithmic decrease in losses by inferring new rules from the training dataset. The final training accuracy (over the entire training dataset) was 97.6% and the final test accuracy was 94.9%.

To the right, Figure 6 shows an example of a pie chart generated by the Android application. The user has the option to view data from different time spans (for example, past day, past 7 days and past month).

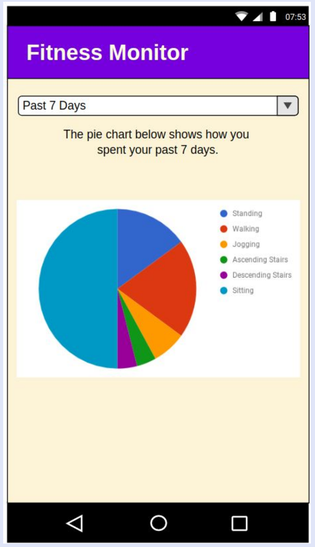

Figure 6. A screenshot showing fitness statistics

Discussion

The training accuracy of the classifier is significantly higher than that reported by Erdaş et al (2016), who extracted hand-crafted features to train classifiers based on Random Forest, kNN and SVM algorithms. The reason behind this is that CNN-based classifiers are particularly good at determining features that discriminate between labels (Yang et al).

Furthermore, it can be determined from Figure 5 that the classifier is not overfitting because training and test losses are constantly decreasing while training and test accuracy are constantly increasing.

Figure 7 shows the confusion matrix for the classifier's predictions. It can be seen from the figure that activities like sitting and standing can be predicted with very high accuracy. The classifier particularly tends to misclassify descending stairs as ascending stairs or walking.

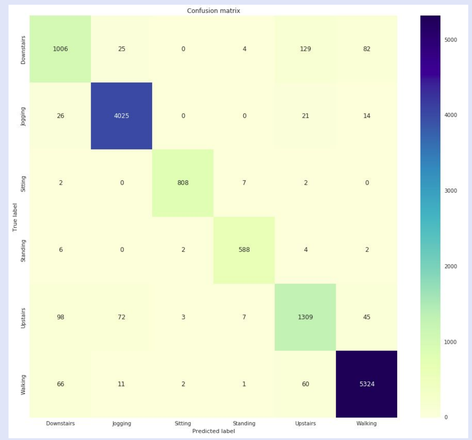

Figure 7. A confusion matrix for classifier's predictions

References

1. Erdaş, Ç. Berke, et al. "Integrating features for accelerometer-based activity recognition." Procedia Computer Science 98 (2016): 522-527.
2. Yang, Jianbo, et al. "Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition." IJCAI. 2015.

Acknowledgements

I would like to thank Ajit Chavan, Charlie Peck and David Barbella for providing guidance and invaluable feedback throughout this project. I would also like to thank developers of Android Studio and TensorFlow and contributors in Android forums for providing proper documentation and code debugging.