Human Activity Recognition

Team: Chethan S V,Dhananjaya,Harish S,Kiran Nadig A M,Navaz Pasha, Mohammad Sadrulhuda Quadri.

Abstract:

We intend to classify the physical activities performed by a user based on accelerometer and gyroscope sensor data collected by a smartphone in the user's pocket. To implement the above, we will be using a number of machine learning concepts and use a variety of classifying techniques to figure out which methods will best classify our data.

Phone applications nowadays can show you how many steps you have walked, ran, flights of stairs you have climbed, calories burnt, etc. On a similar line, we also intend to build such a classifier from scratch which based on data from sensors already present in smartphones is able to identify the user activity. By doing this project we intend to gain practical knowledge of building classifiers and learn & implement machine learning concepts.

Currently the accuracy of these human activity classifiers are about 85% and improving them has many hurdles.

Some of these being:

● High sampling rate of data is required, so more data needs to be processed every second.

● Also since the physical attributes of users, the sensors used in the smartphones vary greatly, a bias creeps in which decreases accuracy.

If we can discover a novel method to handle this data or to decrease user bias or something else which can potentially increase accuracy, it would be a great step forward.

Literature Survey:

1**] Smart Devices are Different: Assessing and Mitigating Mobile Sensing Heterogeneities for Activity** -Research paper (http://userpages.umbc.edu/~nroy/courses/spring2016/cmisr/papers/Smart_Devices_Different_Sen Sys15.pdf)

This research was conducted on the same dataset we are using. They have implemented different types of classifiers and cross validation techniques and also tried to minimize the device and user bias.

2] **Using Machine Learning on Sensor Data** -Research Paper (Journal of Computing and Information Technology - CIT 18, 2010, 4, 341–347 doi:10.2498/cit.1001913)

This research is unrelated to smartphone activity classification. But it gave us an idea as to how we can use sensor data and train our neural network with it.

Problem Statement:

Human Activity Recognition (HAR) using smartphones dataset and an LSTM RNN. Classifying the type of movement amongst six categories:

- WALKING,
- WALKING_UPSTAIRS,
- WALKING_DOWNSTAIRS,
- SITTING,
- STANDING,
- LAYING.

Our Solution/Project:

Compared to a classical approach, using a Recurrent Neural Networks (RNN) with Long Short-Term Memory cells (LSTMs) require no or almost no feature engineering. Data can be fed directly into the neural network who acts like a black box, modeling the problem correctly. Other research on the activity recognition dataset can use a big amount of feature engineering, which is rather a signal processing approach combined with classical data science techniques. The approach here is rather very simple in terms of how much was the data preprocessed.

We used Google's neat Deep Learning library, TensorFlow, demonstrating the usage of an LSTM, a type of Artificial Neural Network that can process sequential data / time series.

Our Dataset:

We have used the "UCI Human Activity Recognition Dataset" which can be obtained from:

https://archive.ics.uci.edu/ml/machine-learning-databases/00240/UCI%20HAR%20Dataset.zip

The dataset includes the following files:

- 'README.txt'

- 'features_info.txt': Shows information about the variables used on the feature vector.

- 'features.txt': List of all features.

- 'activity_labels.txt': Links the class labels with their activity name.

- 'train/X_train.txt': Training set.

- 'train/y_train.txt': Training labels.

- 'test/X_test.txt': Test set.

- 'test/y_test.txt': Test labels.

The following files are available for the train and test data. Their descriptions are equivalent.

- 'train/subject_train.txt': Each row identifies the subject who performed the activity for each window sample. Its range is from 1 to 30.

- 'train/Inertial Signals/total_acc_x_train.txt': The acceleration signal from the smartphone accelerometer X axis in standard gravity units 'g'. Every row shows a 128 element vector. The same description applies for the 'total_acc_x_train.txt' and 'total_acc_z_train.txt' files for the Y and Z axis.

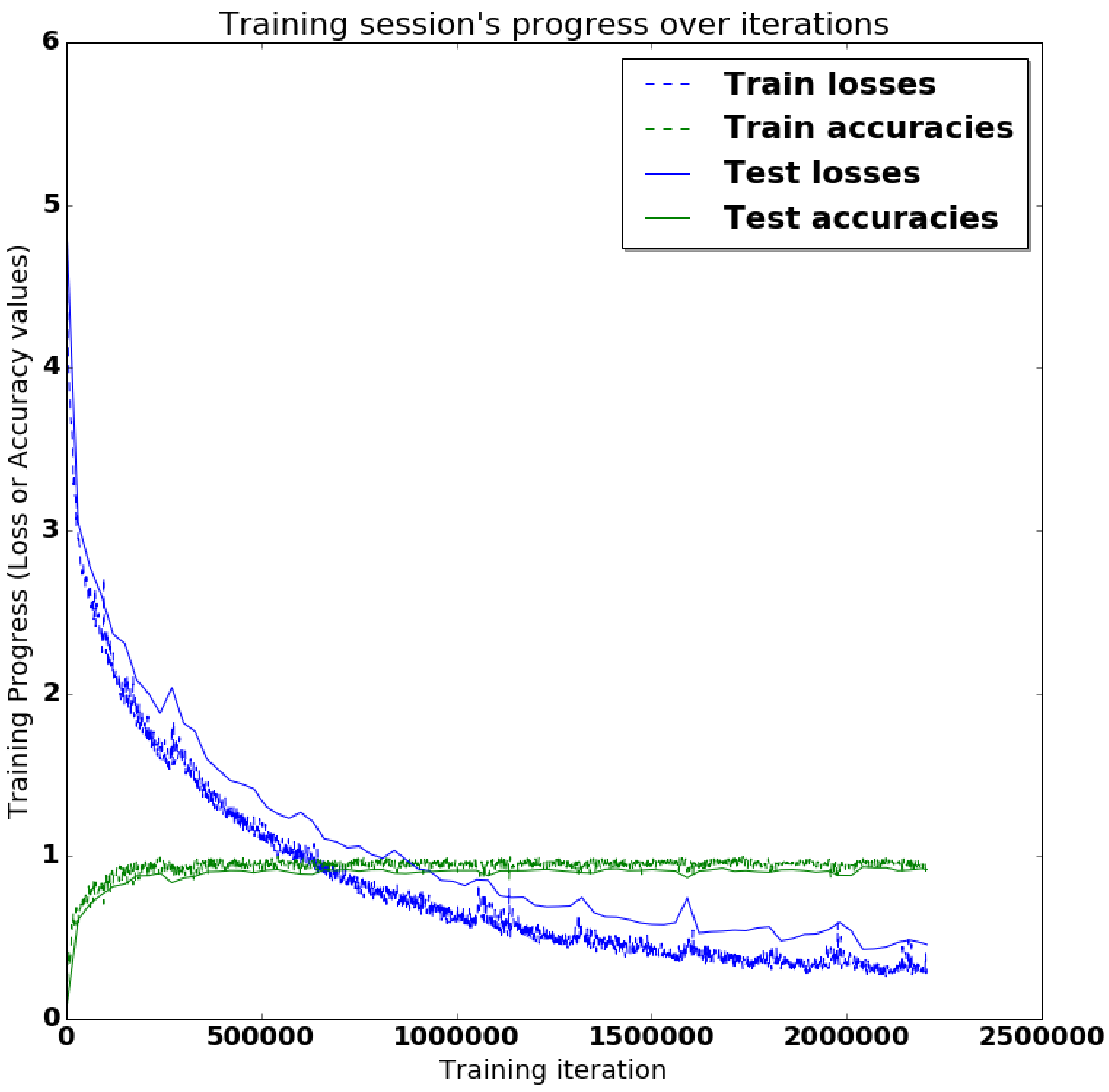
- 'train/Inertial Signals/body_acc_x_train.txt': The body acceleration signal obtained by subtracting the gravity from the total acceleration.

- 'train/Inertial Signals/body_gyro_x_train.txt': The angular velocity vector measured by the gyroscope for each window sample. The units are radians/second.

- Since the number of instances was extremely large , we faced a lot of issues in running code on such a large dataset. In order to overcome this we took every tenth instance reducing the sampling rate to one tenth of original sampling rate.
- Initially instead of feeding a window of data points to a neural network, we fed individual data points to the network.
- However, since this was conceptually incorrect it led to very low training and test accuracy and led to some activities getting 90%+ false negatives.
- After having realized that a number of things could be improved upon, we started by doing the most important thing first: Found a way to correctly merge the accelerometer and gyroscope files.
- We now wanted to implement a neural network which is affected by previous inputs along with the current inputs.
- So, we switched from scikit_learn library to keras and used keras Sequential model to implement LSTM.

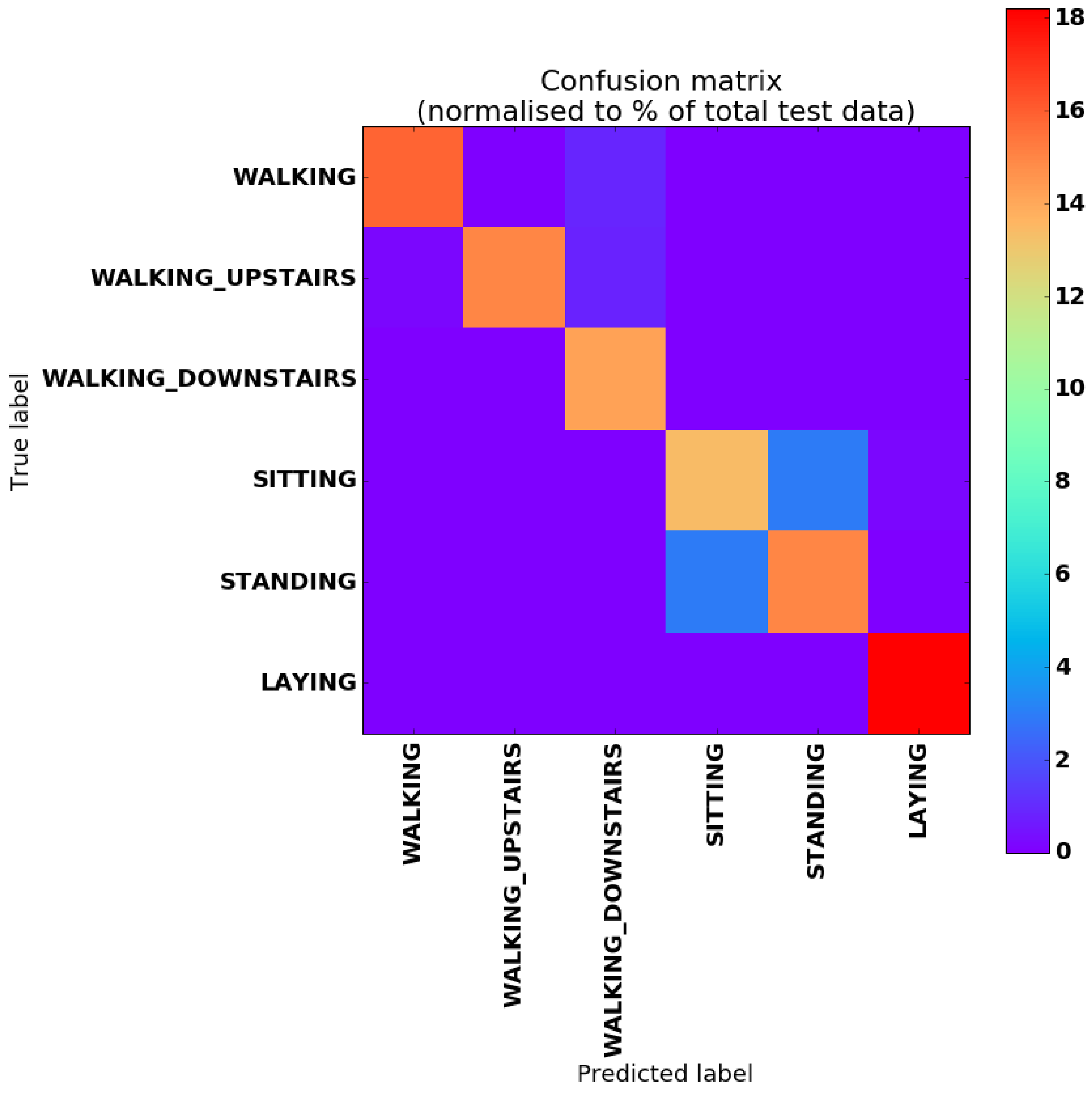
- We got a good accuracy of 78% using LSTM in the first attempt itself. To further improve the accuracy, we normalized the accelerometer and gyroscope readings and applied one-hot encoding for different devices.
- In order to make our model better, we added some LSTM layers to our Sequential model which increased the number of parameters and led to higher scores.
- We later tried out a number of models by varying the batch size, loss function, number of neurons, number of epochs, optimizer, activation function and came up with a model achieving more than 85.3% accuracy.

Our Plot1:



Training session's progress over iterations
Train losses
Train accuracies
Test losses
Test accuracies
Training Progress (Loss or Accuracy values)
Training iteration
0
1
2
3
4
5
6
0
500000
1000000
1500000
2000000
2500000

Our Plot2:



Benefit of your project to the society:

Healthcare monitoring applications

The development of medical science and technology considerably increased the life quality of patients.

Security and surveillance applications
Traditional surveillance systems are monitored by human operators. They should be continuously aware of the human activities that are observed via the camera views.

Conclusion:

Outstandingly, **the final accuracy is of 91%**! And it can peak to values such as 92.73%, at some moments of luck during the training, depending on how the neural network's weights got initialized at the start of the training, randomly.

This means that the neural networks is almost always able to correctly identify the movement type! Remember, the phone is attached on the waist and each series to classify has just a 128 sample window of two internal sensors (a.k.a. 2.56 seconds at 50 FPS), so it amazes me how those predictions are extremely accurate given this small window of context and raw data. I've validated and re-validated that there is no important bug, and the community used and tried this code a lot. (Note: be sure to report something in the issue tab if you find bugs, otherwise Quora, StackOverflow, and other StackExchange sites are the places for asking questions.)

We specially did not expect such good results for guessing between the labels "SITTING" and "STANDING". Those are seemingly almost the same thing from the point of view of a device placed at waist level according to how the dataset was originally gathered. Thought, it is still possible to see a little cluster on the matrix between those classes, which drifts away just a bit from the identity. This is great.