

Human Activity Recognition

CS 205

Group 6

Bhargav Parsi (804945591)

Devanshi Patel (504945601)

Shraddha Manchekar (004945217)

Aashna Agarwal (404943216)

March 19th, 2018

Table of Contents

Introduction	2
Literature Review	2
Implementation details	5
Data Collection	5
Data Analysis and Preprocessing	5
Workflow	8
Cropping	8
Smoothing	8
Windowing and Overlapping	10
Data Cleansing	10
Feature Extraction	11
Feature Selection	12
Data Transformation	13
Standard Scalar Transformation	13
Label Encoding	13
Shuffling	13
Models applied	14
Deep Learning	14
Normal Machine Learning Models	15
Results	16
Grid search	16
Experiments and Observations	16
Conclusion and Future Work	22
References	23

Introduction


During our recent past, there has been a burgeoning development in the area of sensors for smartphones, smartwatches etc. having myriad of applications. Their key features are – high computational power, inexpensive and portable size; making them feasible enough for the humans to interact with them and include them in daily routine. These devices have a wide range of applications, ranging from medical to security to military purposes. They are very useful in detecting the type of human activities. Some instances are the times when one needs to monitor workout routine, or maybe when a patient needs to record his different daily activities suggested by physician. Moreover, by keeping track of these activities one can apply different statistical or Machine Learning algorithms to personalise diets and exercises which cater best to their lifestyle.

Additionally, a lot of challenges are associated with this. There have been issues in improving the accuracy in real-world, obtaining clean data during data collection, deciding which attributes to consider and also how much importance to be given to each, providing the flexibility where new (unseen) users are added without the need to re-train the whole system, building the design model for feature engineering and analysis; lastly, we also face problems in implementing these in mobile phone devices where energy and processing requirements are the key constraints.

Mainly, two kinds of sensors are available for human activity recognition purposes. First, external sensors, these are fixed in predetermined points of interest, so the inference of activities entirely depends on the voluntary interaction of users with the sensors. Intelligent homes and external cameras are the classic examples of external sensing. Second, wearable sensors, which are worn by the user i.e. attached to the user, for instance, smartphones and smartwatches. These wearable sensors record many signals such as accelerometer, gyroscope, magnetometer and even light intensity. Combinations of these sensor data allow us to determine which activity is being performed at any given time.

Literature Review

In the paper “*Activity Recognition in Youth Using Single Accelerometer Placed at Wrist or Ankle*” by Andrea Mannini et al, it proposed an algorithm for detecting activity (sedentary, cycling, other activities) from wrist-worn accelerometers, originally developed using data from 33 adults, and then tested on a dataset of 20 youth. The algorithm introduces new features (range, activation interval) required to improve performance on the youth dataset. Subsequent tests on both the adult and youth data were performed using crossed tests. The new feature set improved overall recognition using wrist data by 2.3% for adults and 5.1% for youth.



In the paper “*Machine Learning Methods for Classifying Human Physical Activity from On-Body Accelerometers*” written by Mannini A et al., it uses HMM on a dataset of accelerometer time series. For evaluation on the trained model, other ML algorithms were employed such as KNN, SVM and ANN. Using classifier, they practiced feature selection method by identifying and then ignoring the features which proved to minimally help in the prediction of correct response. For position/velocity and altitude navigation data, it uses GPS/magnetometers. The specific force additively combines the linear acceleration component a , due to body motion, and the gravitational acceleration component, g —both projected along the sensitive axis of the accelerometer.

The author of the paper “*Combining Smartphone and Smartwatch Sensor Data in Activity Recognition Approaches*” experimentally proposes a model that combines the data from the sensors of smartwatches and smartphones. The evaluation metric used here is how accurately the model recognizes the different activities. Readers can observe that this combined usage of smartwatch and smartphone can indeed aid in incrementing accuracy. Data was collected from 10 different participants performing 7 different activities by holding smart phones at different positions. The analysis demonstrates that all the sensors, but magnetometer are capable of taking the lead roles individually, depending on the type of activity being recognized, the body position, the used data features and the classification method employed (personalized or generalized). Furthermore, this hybridization combination only improves the overall recognition performance when their individual performances are not very high, so that there is room for performance improvement.

In the paper entitled “*Computational Methods for Estimating Energy Expenditure in Human Physical Activities*” authored by Shaopeng Liu et al., we see the various types of body-worn sensors and different computational techniques have been applied to the PA(Physical Activity) assessment based on recent publications. Particularly, PAEE (Physical activity energy expenditure) has been given more importance.

PAEE detection and computation takes place in 3 steps:

1. Body-worn sensors are used to capture body motion and other physiological variables.
2. After data collection, data preprocessing is carried out by filtering, artifact removal, etc. Not just time-domain signals but frequency-domains are also calculated and harnessed.
3. These extracted features are then used as inputs to data fusion algorithms for estimating/modeling the PAEE.

Accelerometer can detect the quantity and intensity of human body movements, making it most commonly used wearable sensors for estimating the PAEE. These movements include detection of both static and dynamic accelerations caused by changes in posture, body motion, or

transition in motion patterns. Static acceleration responds to gravity, and has been used to measure tilt angles or posture changes. Dynamic acceleration captures the intensity of body movements, and thus has been extensively applied to estimate PAEE.

Shaopeng Liu, further, observed that activities involving mostly upper-body motion would result in inaccurate estimation of the PAEE when only using a hip accelerometer. In the paper, *“Multi-sensor data fusion for physical activity assessment”* he mentions that placing additional accelerometers at various locations on the body improves the estimation of PAEE. A five-accelerometer device (IDEEA) with accelerometers located on the chest, frontal area of each thigh, and bottom of each foot achieved a 56% higher prediction accuracy for estimating energy expenditure than a single hip-mounted accelerometer device (ActiGraph).

In the paper *“Multisensor Data Fusion for Human Activity Recognition”*, we find that the biggest drawback of fitness sensors in smartphones and smartwatches is that they are very susceptible to motion often leading to wrong results. The idea of using an egocentric camera is to prevent such false sudden changes by including the visual data in the decision making process. While the phone sensors prove to be efficient in detecting static motion, the egocentric camera proved to be superior in classifying the various motion activities.

Sazonova N has proposed another sensor kind for estimating PAEE. In the paper *“Accurate Prediction of Energy Expenditure Using a ShoeBased Activity Monitor”* he mentions about a pressure device. The placement of such a device is at the bottom of the feet that is useful in detecting the contact pressure during any human body movement. The results and analysis in the paper demonstrates that by using the combination of accelerometer and pressure sensor signals, better PAEE prediction with a RMSE of 0.69 METs was achieved than its single sensor counterpart.

In the paper, *“An investigation of a novel three-dimensional activity monitor to predict free-living energy expenditure”*, Carter et al. mentions findings pertaining to the fact that the body mass, age and gender were not significant predictors for the estimation of total energy expenditure (TEE) in young adults, and explained 0% of the estimation variance. Height, on the other hand, was found to be an important predictor of TEE, and explained 46% of the estimation variance. For different types of activities that produce similar acceleration profiles but have different energy expenditure, accelerometer-based activity monitors cannot provide accurate estimates of PAEE.

Under such circumstances, sensors that measure human physiological responses, such as heart-rate, respiration, or an armband consisting of heat flux, galvanic skin response, and skin temperature sensors, have been studied to determine if these additional sensors improve the estimation of the energy expenditure.

Implementation details

Data Collection

For data collection, firstly we installed the app “Decent Logger” on the Sony SmartWatch. Once the app has been installed, all we need to do is start the app whenever we want to collect data related to any activity and select appropriate label. We used the ADB pull command to extract the collected data from the watch. We then examined all the files pulled from the Sessions folder. Each session file has a “data” folder that contains raw data collected from the following sensors,

- Accelerometer
- Magnetic field
- Orientation
- Gyroscope
- Light
- Gravity
- Linear acceleration
- Rotation vector
- Significant motion
- Step counter

We have only used Accelerometer and Gyroscope data for our analysis. Secondly, we used the data from 3 people in our group for training our model and used the remaining person’s data for the testing purpose. This was done to collect insights about the performance of the model on unseen data.

Data Analysis and Preprocessing

Our next step was to analyze the collected data. We started out by calculating the frequency of the sensors i.e. the number of rows recorded by the sensors in one second. We found that the frequency of accelerometer was 200 Hz whereas the frequency of gyroscope was 225 Hz. Moreover, we also plotted various plots to observe the distribution of data. For instance, Fig. 1. shows the histogram for number of training examples per activity label/type.

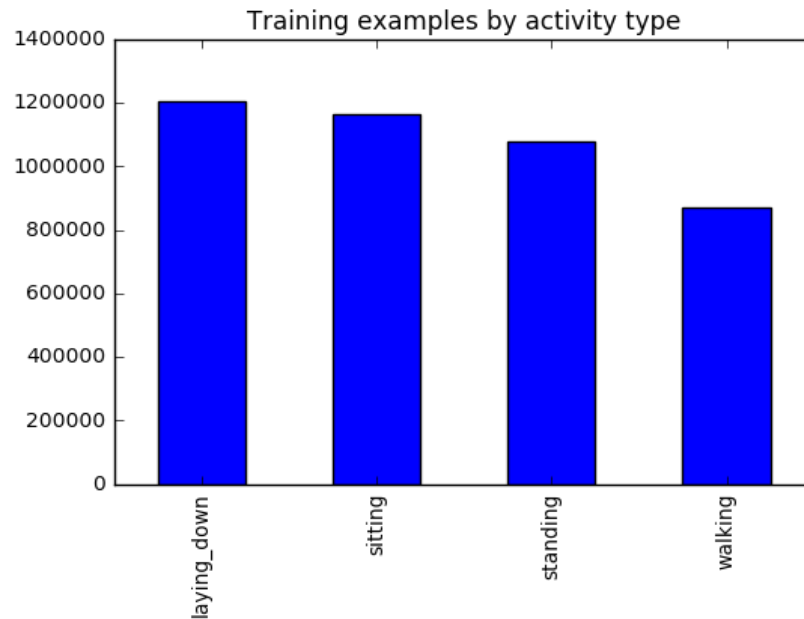


Fig. 1: Number of training examples per activity label/type

From this plot, we can observe that our training dataset is almost balanced. In a classification problem, distribution of the training dataset is very important. The data should be evenly distributed. If not, proper steps (such as down-sampling the majority classes) should be taken to handle them imbalance in the data size.

We also plotted the signal forms of the accelerometer data for different activities and noticed that the walking signal has a lot of spikes but the sitting, laying down and standing signals were almost similar. This would make the activity recognition process more challenging.

Fig. 2, Fig. 3, Fig. 4 and Fig. 5 show the plots for x, y, z signals for walking, sitting, standing and laying down respectively.

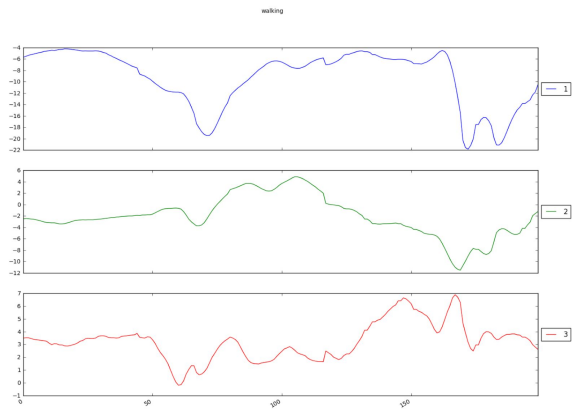


Fig. 2: Walking

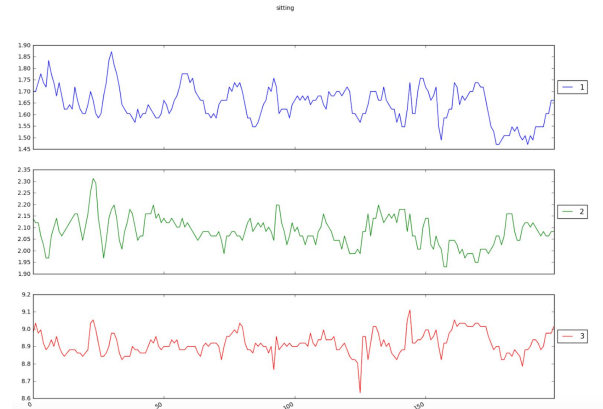


Fig. 3: Sitting

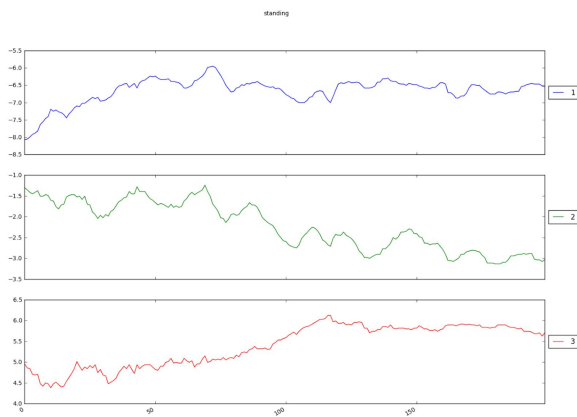


Fig. 4: Standing

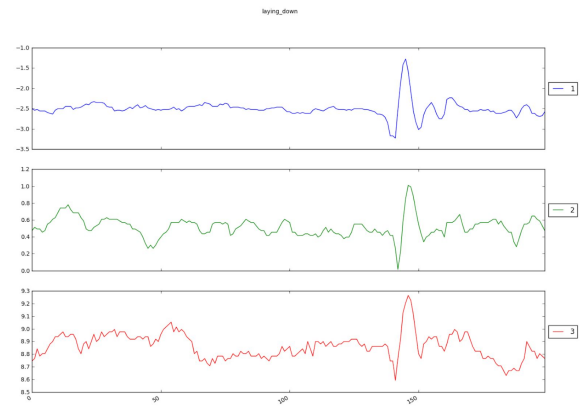
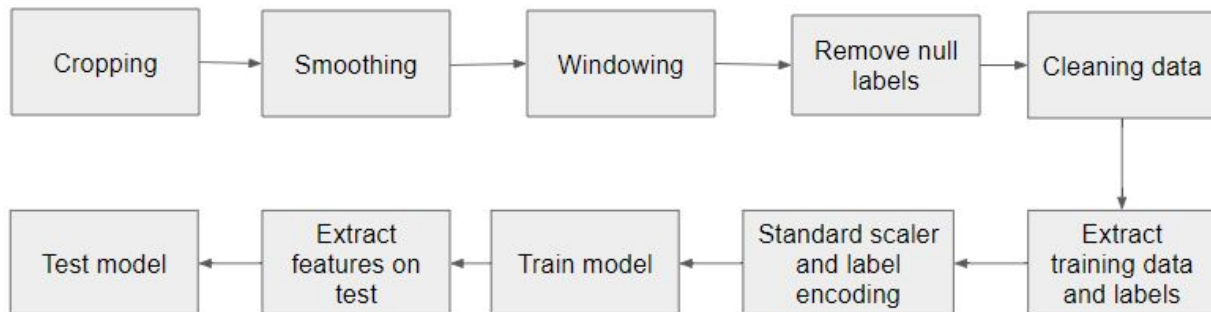


Fig. 5: Laying down

From these plots, it can also be observed that the signals had a lot of jitters. These are caused by noisy data. We used the process of smoothing to remove these jitters.

Workflow



Cropping

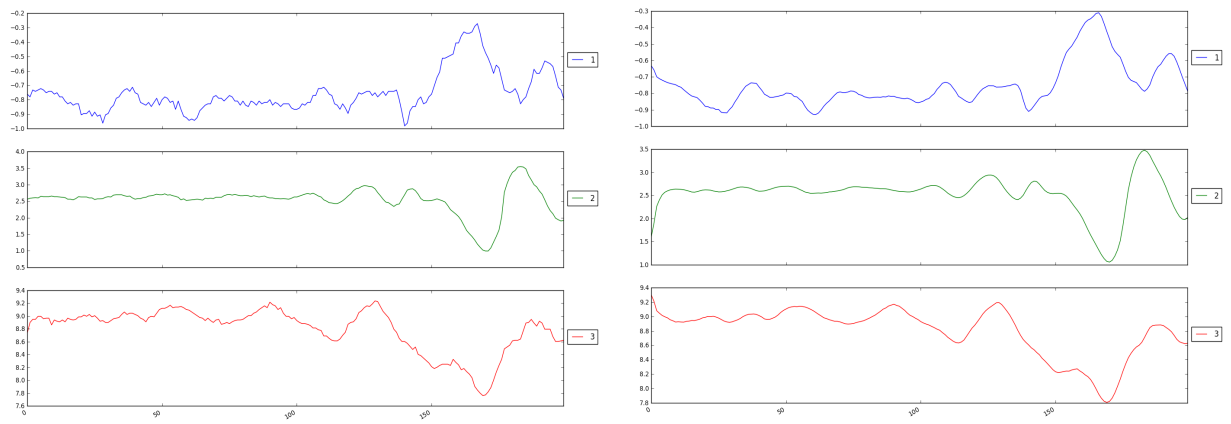
While analyzing our dataset, we also noticed that unusually high accelerometer values were recorded for the first and last four seconds for every activity. This is because the accelerometer sensor is highly sensitive and pressing the button to start and stop recording the activity label while collecting the data causes a sudden jerk and unusually high values get recorded. Hence, we dropped the first and last four seconds data from the accelerometer dataset to handle this problem. We observed a slight improvement in the performance after cropping.

Smoothing

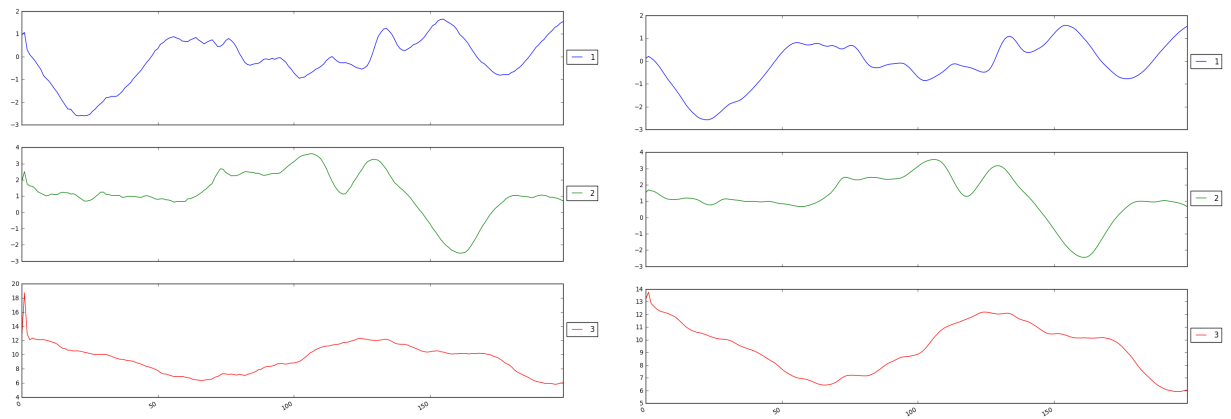
In case of most signals, true signal amplitudes i.e. the y-axis values change rather smoothly as a function of the x-axis values. However, noisy data can cause rapid, random changes in the amplitude from point to point, known as jitters. We handled the jitters in our training data by using the process of smoothing. Smoothing reduces the jitters and hence smoothens the signal by reducing the value of a data point if its adjacent points are higher or increasing the value of a data point if its adjacent points are lower. This process doesn't distort the signal but reduces the high frequency noise by acting as a low-pass filter.

We used a smoothing technique of the fifth-order to filter the spikes. In this technique, the mean value upto five points is set as the point signal and hence, the shape of waveform is retained and spikes get smoothened. One can clearly see the difference between the signals before and after smoothing in the following figures. Also, since the jitters were removed, our performance improved a lot after smoothing.

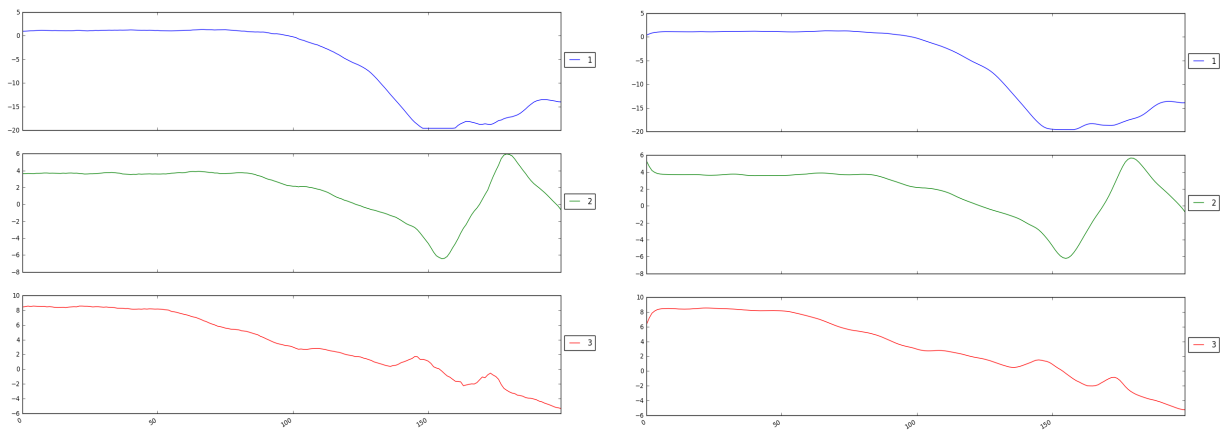
Sitting signal



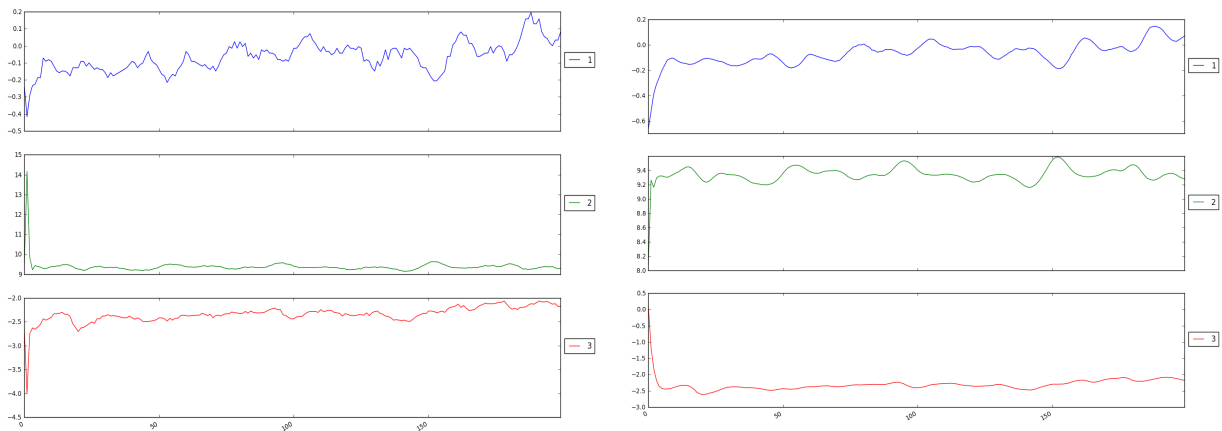
Walking signal



Standing signal



Laying down signal




Windowing and Overlapping

A typical procedure for extracting features is to first divide the sensor data into small fixed-length windows. We used a sliding window of 5 and 10 seconds and a 50% overlap. Different results are obtained by changing the window size. We observed that a 10-second windows with an overlap of 50% performs the best.

Data Cleansing

During the data analysis, we observed that some of the data points had 'null' labels and did not contain any associated activity label. It is important to filter out these data points to make sure that our model gets trained properly. We performed this filtering during the windowing phase of



the workflow. After extracting the feature dataset, we also observed it contained some NaN and infinite values. To get rid of these values, we cleaned our dataset to remove rows that contained any of those values.

Feature Extraction

The raw signals obtained from the sensors are first sampled over periods of time by using the fixed-length sliding window technique. Each window contains considerable number of data points and features are extracted from the original sensor signal for each window or frame and these features are used as inputs to more advanced post-processing computation. For each frame, we calculate a feature vector, which serves as an input to our models. We compute features for both, time domain and frequency domain.

Time-domain features

Time-domain features are directly extracted from the time series of the original, raw sensor signals. For time-domain, we considered the following signals:

- **XYZ (3 signals):** Original signals
- **Magnitude (1 signal):** Magnitude signal computed from the previous three signals. This magnitude is computed as the square root of the sum of squared components
- **JerkXYZ (3 signals):** Jerk signals (derivative of the accelerometer signals) obtained from the original signals
- **JerkMag (1 signal):** Magnitude signal computed from the previous jerk signals (square root of the sum of squared components).

The set of features that we computed for the time-domain signals are as follows:

- Mean value, standard deviation, median absolute deviation, minimum and maximum values of the samples in a frame
- Signal Magnitude Area: The normalized integral of the samples in a frame
- Energy measure: Sum of the squares of samples divided by the number of samples in a frame
- Interquartile range: Variability measure obtained by dividing a data set into quartiles

Frequency-domain features

Frequency-domain features are extracted from the coefficients obtained by performing spectral analysis, usually Fast Fourier Transform (FFT), on the original signal data. The values of the coefficients represent the amplitudes of the corresponding frequency components. For frequency domain, we considered the following signals:

- **fXYZ (3 signals):** Fast Fourier transforms (FFTs) from XYZ
- **fMag (1 signal):** FFT from Mag
- **fJerk-XYZ (3 signals):** FFTs from Jerk-XYZ
- **fJerkMag (1 signal):** FFTs from JerkMag

The set of features that we computed for the frequency-domain signals include those calculated for the time-domain as well as the following:

- Skewness and Kurtosis of the frequency-domain signal
- Index of the frequency component with largest magnitude

Feature Selection

Initially we trained our model for accelerometer and gyroscope using all the features mentioned earlier for each of the signals. This gave a total of 144 features for each sensor and it was taking a long time to train the model as well as to extract the features for the testset. Even after using so many features, our model was not performing well on unseen data. Soon we discovered that the model is trying to overfit since our training dataset is small.

To overcome this issue, we used ExtraTreesClassifier method available in the sklearn package to extract the best features out of all the extracted features. This classifier implements a meta estimator that fits a number of randomized decision trees on various sub-samples of the dataset and thus helps in controlling overfitting. We sort the feature importance values returned by the classifier and selected 12 best features that are most significant.

For all the further analysis, we consider only the features for the magnitude signals corresponding to accelerometer. This is because computing features for both the sensors was really time consuming and also, the signals corresponding to magnitude had a major contribution to improving the performance of the model. The following features were assigned highest importance by the classifier:

- **Mad_macc :** Median Absolute Deviation of the magnitude signal
- **Min_macc :** Minimum of the magnitude signal
- **Max_macc :** Maximum of the magnitude signal
- **Em_macc :** Energy Measure of the magnitude signal
- **Sma_macc :** Signal Magnitude Area of the magnitude signal
- **Mad_macc_jerk :** Median Absolute Deviation of the jerk magnitude signal
- **Min_macc_jerk :** Minimum of the jerk magnitude signal
- **Iqr_macc_jerk :** Interquartile range of the jerk magnitude signal
- **Sma_macc_jerk :** Signal Magnitude Area of the jerk magnitude signal

- **Std_mfacc_jerk** : Standard Deviation of fJerk magnitude signal
- **Mad_mfacc_jerk** : Median Absolute Deviation of fJerk magnitude signal
- **Iqr_mfacc_jerk** : Interquartile range of fJerk magnitude signal

Data Transformation

Standard Scalar Transformation

Standardization of datasets is a common requirement for many machine learning estimators. They might behave badly if the individual features do not more or less look like standard normally distributed data with zero mean and unit variance. After feature extraction, we discovered that some features were in a range of 0.01-0.1 and some with range 100-1000. To ensure that the most of the features share equal variance, we standardized features using StandardScaler that is provided by sklearn library of Python.

Label Encoding

Scikit learn only handles real numbers. If we just pass in string labels, they'll get cast to floats in unpredictable ways. Most methods can be made to work directly on categorical values. It is also convenient to use integers to represent the categories because an array of integers is much nicer to work with than an array of strings on a computational level. Hence, we encoded our labels to represent integers 0 to 4 for laying down, sitting, walking and standing respectively.

Shuffling

The data obtained after preprocessing the original sensor data and computing the feature vectors is shuffled before feeding it to the models. We observed that shuffling the data results in a slightly better performance than the non-shuffled data. Shuffling the data prevents same labels from being together and thereby making sure that the model remains general and overfits less. It also ensures that each data point creates an independent change on the model, without being biased by the same points before it. The following table shows our results before and after shuffling the data.

Model	CV Score before shuffling	CV Score after shuffling
Decision Tree Classifier	0.8677	0.9613
Random Forest Classifier	0.8894	0.9833
SVM	0.9100	0.9600
MLP Classifier (Neural Network)	0.9045	0.9831

XGBoost	0.9072	0.9855
Logistic Regression	0.9044	0.9613
KNN	0.8979	0.9607

Table 1

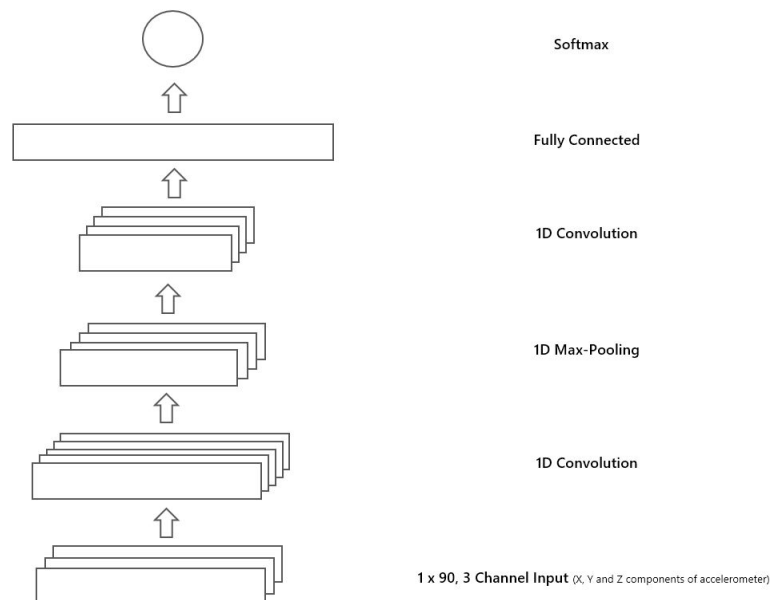
Models applied

Deep Learning

CNN

In this type of model, we do not need feature extraction. This is because, CNN is a deep learning model which learns the features by itself. We need to however, define some functions which would help transform the dataset in a particular format, and then the CNN would read it accordingly.

We created windows of particular sizes for example 90, and it can move over the signal specified by the step size. Each segment is associated with a label, and it would be the label of the most frequent class of that segment. Using one hot encoding, the labels were converted from strings to one hot encoded fashion. We need to perform a 1D convolution so have to reshape the output of the windows accordingly. The figure below would give an overview of the type of input the CNN would get.





LSTM

We perform the similar process as in the CNN model, which is dividing the dataset into segments. We similarly associate each segment to a label which occurs most frequently in it. We also apply one hot encoding to the labels. We then split this dataset to train and test.

We generate the model to contain, 2 fully-connected and 2 LSTM layers that are stacked on one another.

Normal Machine Learning Models

Decision Tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Random Forest

In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features.

Support Vector Machine

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. It is very effective in high dimensional spaces; even in cases where number of dimensions is greater than the number of samples.

Multi Layer Perceptron

It is a simple neural network model in which we can configure different activation functions, optimisers and hidden layer units.

K nearest neighbors

Neighbors-based classification is a type of instance-based learning or non-generalizing learning; it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point.



XGboost

XGBoost is short for “Extreme Gradient Boosting”. It is developed with both deep consideration in terms of systems optimization and principles of machine learning. The goal of this library is to push the extreme of the computation limits of machines to provide a scalable, portable and accurate library.

Logistic Regression

Logistic regression is a linear model for classification rather than regression. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

Results

In this project, we used two methods to evaluate the models’ performance. The first one is 10-fold cross validation evaluation. However, this method suffers the problem that the data in both training and testing could contain information from the same subject, so the machine learning algorithm can learn not only physical activity characteristics but also some subject-dependent ones. This aspect makes it hard to evaluate the system performance when facing a new subject. Hence, we also used a subject-wise cross-validation. In this case, the same kind of cross-validation is done but on different subjects rather than on automatically split parts: all data from the same user is considered for testing and the data from the remaining subjects for training. So, in a group of four, data collected by three people was used for training whereas the fourth person’s data was used for testing.

Grid search

After finding the model which gives the best validation accuracy, we will search for optimal parameters so as to maximise that accuracy. This process of finding the optimal parameters is known as grid search.

Experiments and Observations

We tested various combinations of the preprocessing techniques and features mentioned above to evaluate the performance for each combination.

Initially, we used 288 features on the time and frequency-domain signals and observed that the models did not perform well. The accuracy of prediction on unseen data was very low. The reason for this was that our training dataset was small and the number of features were very high

for such a small dataset. In such cases, the model can identify single data points by single features and build a special case just for a single data point, which leads to overfitting. Hence, the models weren't really performing well with 288 features. Therefore, we decided to gradually reduce the number of features. Following were our observations:

Features: 76 (magnitude only features for Accelerometer and Gyroscope)

Model	CV Score	Accuracy (Cross-subject)
Decision Tree Classifier	0.9438	0.5281
Random Forest Classifier	0.9743	0.4886
SVM	0.5009	0.3095
MLP Classifier (Neural Network)	0.4994	0.4112
XGBoost	0.9910	0.5569
Logistic Regression	0.7009	0.1684
KNN	0.7281	0.2489

Features: 38 (magnitude only features for Accelerometer)

Model	CV Score	Accuracy (Cross-subject)
Decision Tree Classifier	0.9362	0.5621
Random Forest Classifier	0.9638	0.5394
SVM	0.5016	0.3091
MLP Classifier (Neural Network)	0.4837	0.1227
Logistic Regression	0.5001	0.2894
KNN	0.5863	0.2288

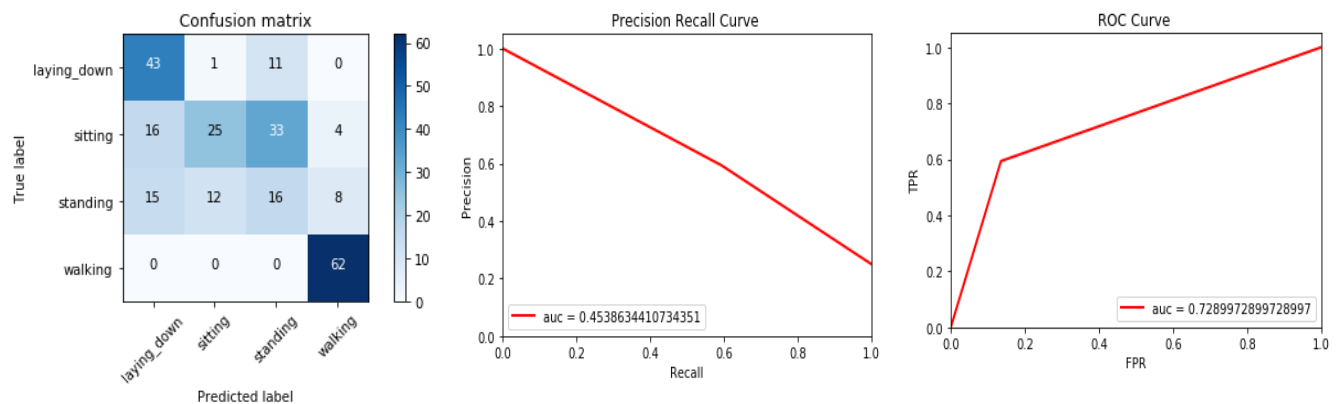
We can see that the cross-subject accuracy increases as the number of features decrease. To select the best features, we used ExtraTreesClassifier method in the sklearn package and selected 12 features with the highest importance values. The best selected features are: 'mad_macc_jerk', 'min_macc_jerk', 'iqr_macc_jerk', 'sma_macc_jerk', 'std_mfacc_jerk', 'mad_mfacc_jerk', 'iqr_mfacc_jerk', 'mad_macc', 'min_macc', 'max_macc', 'em_macc', 'sma_macc'. Our observation for these features is as shown below:

Features: 12 (For accelerometer only)

Model	CV Score	Accuracy (Cross-subject)
Decision Tree Classifier	0.7895	0.5935
Random Forest Classifier	0.8100	0.6341
SVM	0.7494	0.6260
MLP Classifier (Neural Network)	0.7729	0.6057
XGBoost	0.8984	0.5569
Logistic Regression	0.7352	0.5691
KNN	0.7930	0.5854

We can see that almost all the model performed much better than the previous ones; with Random Forest classifier giving the highest accuracy. However, accuracy alone is not a good indicator for making predictions. Hence, we also use other metrics such as confusion matrix, ROC curve, precision-recall curves, etc. to decide which model performs the best. The metrics for each of the above models are as shown below:

Decision Tree:

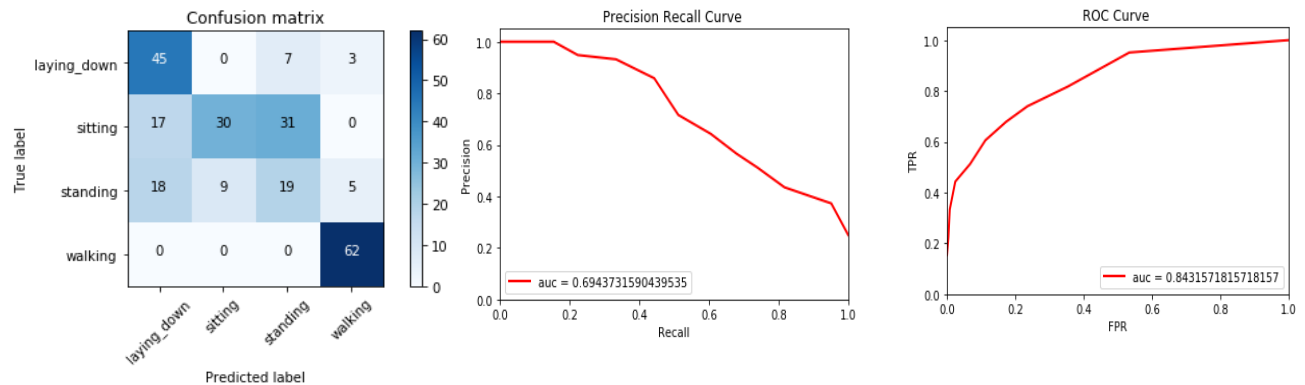


Label	Precision
Laying down	0.581

Sitting	0.658
Standing	0.267
Walking	0.838

From the confusion matrix and the precision value, it is quite clear that this model does not perform well on standing label.

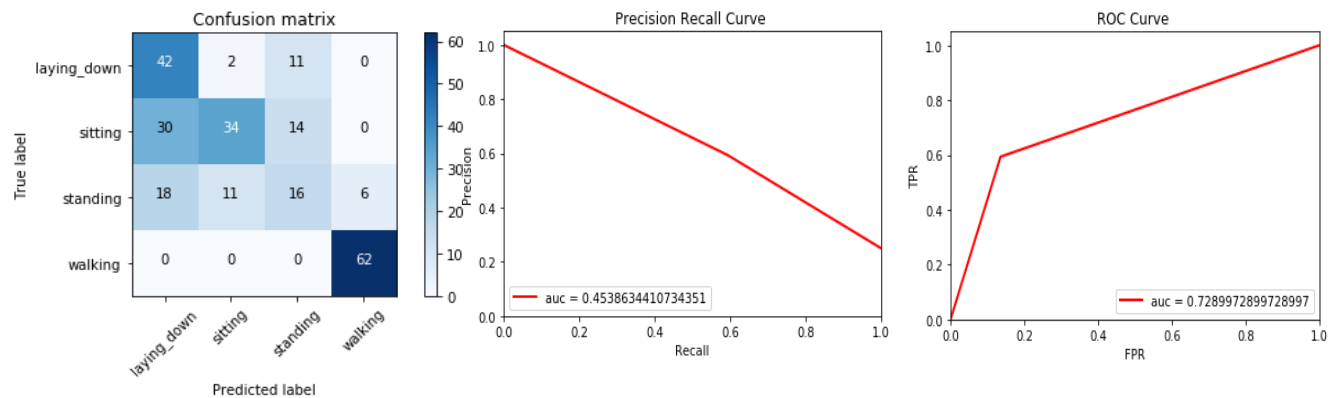
Random forest:



Label	Precision
Laying down	0.562
Sitting	0.769
Standing	0.333
Walking	0.886

The model does a good job at predicting walking and laying down positions. Although this classifier gives the best accuracy, it fails to differentiate between sitting and standing activities. The precision value for standing is also quite low.

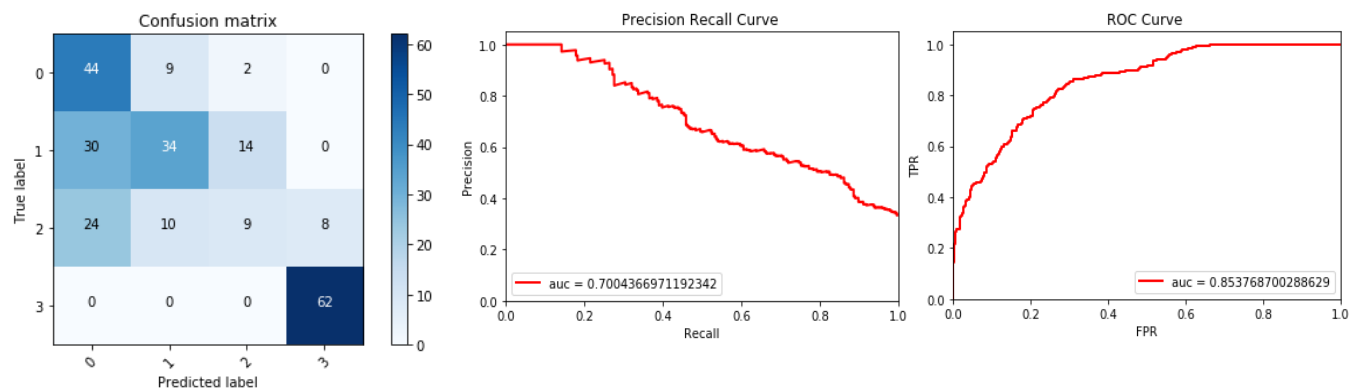
SVM



Label	Precision
Laying down	0.467
Sitting	0.723
Standing	0.390
Walking	0.912

There is not much difference between the accuracies of SVM and Random Forest classifiers. Although SVM has slightly lower accuracy, the confusion matrix indicates that the model actually performs better when it comes to distinguishing between sitting and standing labels.

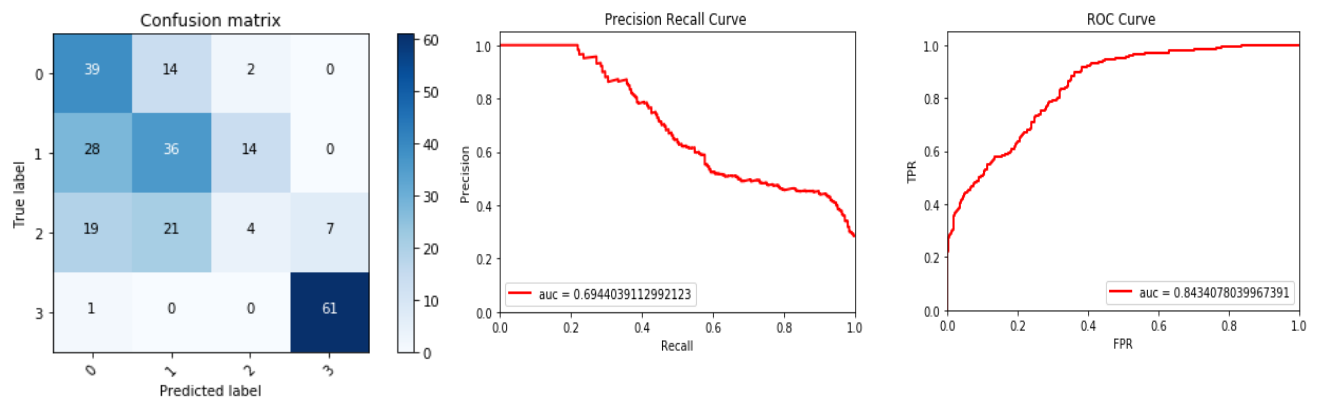
MLP (Neural network):



Label	Precision
Laying down	0.449
Sitting	0.641
Standing	0.360
Walking	0.886

The confusion matrix indicates that this model performs poorly on standing label. Moreover, it classifies all activities apart from walking as laying down.

Logistic Regression



Label	Precision
Laying down	0.448
Sitting	0.507
Standing	0.200
Walking	0.897

From the confusion matrix and the precision value, it is quite clear that this model does not perform well on standing label.

Conclusion and Future Work

In this report, we analysed various data preprocessing and feature extraction techniques as well as different machine learning models to develop a Human Activity Recognition (HAR) system that classifies data collected from smartwatch into four activities. We used both time-based and frequency-based signals for feature extraction. It was observed that time-based features provided better results. Moreover, we also tried using different window sizes while extracting features and it was found that a sliding window of 10 sec with 50% overlap gives best results. Cropping, smoothing and shuffling also contribute to enhancing the performance of the model. During our analysis, we found that extracting 12 best selected features for accelerometer data succeeds in improving the cross-subject accuracy.

Currently we have used data from just one sensor to make the predictions. As future work, this can be extended to incorporate data from the other sensors as well. A study can be carried out to select the sensors that effectively contribute to performance enhancement. Moreover, if there is enough training data available, the behavior of various models can be studied by incrementally adding more features without having to deal with the problem of overfitting.

References

1. Mannini, A., Rosenberger, M., Haskell, W. L., Sabatini, A. M., & Intille, S. S. (2017). Activity Recognition in Youth Using Single Accelerometer Placed at Wrist or Ankle. *Medicine and science in sports and exercise*, 49(4), 801-812.
2. White, T., Westgate, K., Wareham, N. J., & Brage, S. (2016). Estimation of physical activity energy expenditure during free-living from wrist accelerometry in UK adults. *PLoS One*, 11(12), e0167472.
3. Ferscha, A., Resmerita, S., Holzmann, C., & Reichör, M. (2005). Orientation sensing for gesture-based interaction with smart artifacts. *Computer communications*, 28(13), 1552-1563.
4. Rosenberger, M. E., Haskell, W. L., Albinali, F., Mota, S., Nawyn, J., & Intille, S. (2013). Estimating activity and sedentary behavior from an accelerometer on the hip or wrist. *Medicine and science in sports and exercise*, 45(5), 964.
5. Liu, S., Gao, R., & Freedson, P. (2012). Computational methods for estimating energy expenditure in human physical activities. *Medicine and science in sports and exercise*, 44(11), 2138.
6. Mannini, A., Intille, S. S., Rosenberger, M., Sabatini, A. M., & Haskell, W. (2013). Activity recognition using a single accelerometer placed at the wrist or ankle. *Medicine and science in sports and exercise*, 45(11), 2193.
7. Mannini, A., & Sabatini, A. M. (2010). Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, 10(2), 1154-1175.
8. Tapia, E. M., Intille, S. S., & Larson, K. (2004, April). Activity recognition in the home using simple and ubiquitous sensors. In *International conference on pervasive computing* (pp. 158-175). Springer, Berlin, Heidelberg.
9. Mortazavi, B., Nemati, E., VanderWall, K., Flores-Rodriguez, H. G., Cai, J. Y. J., Lucier, J., ... & Sarrafzadeh, M. (2015). Can smartwatches replace smartphones for posture tracking?. *Sensors*, 15(10), 26783-26800.