

Report

Health Analytics

Kamal Nayan Reddy Challa (904943469)

Shreyas Lakhe(105026650)

Sudharsan Krishnaswamy(204944740)

Venkata Sasank Pagolu(804945548)

Table of Contents

Introduction	2
Related Work	3
Feature Engineering	4
Battery Optimization for Remote Health Monitoring	4
Deep Learning:	5
Convolutional Neural Networks:	6
Recurrent Neural Network	7
Review: Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables	8
Methodology	9
Dataset	9
Feature Extraction	10
Mean	12
Standard Deviation	12
Correlation	12
Fast Fourier Transform	13
Discrete Wavelet Transform	13
Classifiers	14
Random Forest	14
SVM	14
J48	15
Logistic Classifier	15
Convolutional Neural Networks:	15
Recurrent Neural Networks:	17
Results and Discussion	17
Conclusion	22
References	24



Introduction

Human Activity monitoring has become a vital area of research in the health care domain. The rise in popularity of smart wearable devices like smart watches, with embedded sensors, has facilitated the process of collecting high quality data both easily and effectively. This area of research is highly intriguing as it finds applications across a wide range of domains. Some of the interesting application include, monitoring the physical activity and health condition of geriatric population, predicting the motion of a robot using sensors, and to develop systems that help the elderly people walk etc.

The primary objective of this project is to come up with an innovative and robust system to monitor the human activity and to classify the positioning of a user into one of the 4 classes, Sitting, Walking, Standing, and Laying down, using a smartwatch. The idea is to model this as a learning problem given the quality data of human activity belonging to the four classes mentioned above. The data has been collected for training the models and to build inference systems for predicting unobserved data sets. The experiments are based on the smartwatch sensor data collected by four different users. The smartwatch users contributed a couple of hours of data for each activity. The smartwatch is embedded with highly precise sensors like Accelerometer, Gyroscope, sensors for measuring the orientation, recording gravity, step count, rotation motion etc. The signals captured by these sensors are well indicative of the hand motion and enables us to predict the activity of the user. For this task, each sensor has a sampling frequency, that is the number of samples it produces per second, of 250Hz. In this setting, the problem can be stated as we are trying to predict a label to a series of samples. Each series of samples of size s , a hyperparameter, forms an instance to this problem.

Throughout the project, various features were extracted from the raw signal data offered by the smart watch. The experiments were conducted on several machine learning models like logistic regression and random forest etc and ran on multiple settings of time windows to build a precise inference model. Deep Learning techniques like Convolutional Neural Networks and Recurrent Neural Networks (LSTM) have also been employed to test the performance of automatic feature extraction techniques. We have found some interesting observations such as, increasing number of sensors for tracking the activity and considering larger time windows improve the accuracy of the model.


Weka tool is used for running machine learning classifiers. The data collected for various human activities and the trained models are open source and available for download at <https://github.com/kamalnayanch/healthanalytics>.

Related Work

In this section, we review some of the research advances in the area of activity prediction. Tapia et al.[1] have focused on activity recognition in a home setting by using simple sensors installed in home environments. The motivation behind this work is to provide a system for medical professionals to monitor the changes in the daily activities of elderly patients which would be useful to identify and diagnose the tumor at an early stage. Most works in this domain monitor human activity by attaching sensors to any of the body parts. There are a class of activities like grooming, brushing, cooking etc. which may be more easily recognized not by watching for patterns in how people move but instead by watching for patterns in how people move things. So, placing sensors on the things that humans interact and using it for activity monitoring was the main focus. They extracted features from the existing environmental state change sensors and recognized the activity by applying naive bayes classifiers on top of the features. Though this is a new direction, the activities of our task walking, sitting etc. does not generalize well to this domain.

Liu et al.[2] studied computational methods for estimating energy expenditure in human physical activities. They observed that accelerometer sensors are not sufficient for computing the energy expenditure as the activities with similar accelerations may have different energy profiles. So, they found the works that use physiological sensors such as heart rate sensors, skin temperature sensors have a better performance. They observed the following as important features to be extracted from the data. 1) Time domain features from the sensors like the mean, standard deviation, median statistics along with the signal power etc. 2) Frequency domain features like FFT to calculate the dominant frequency corresponding to the highest amplitude in the signal. 3) They also found that demographic features like age, location to be important for this task. They stated that linear and non linear regressors and models like Support vector regressors perform at a high accuracy for estimating the energy expenditures.

Rosenberger et al.[3] focused on estimating activity and sedentary behavior and compared the effectiveness of placing the accelerometer at wrist to placing it at hip for this task. Healthy adults wore triaxial accelerometers on the hip and dominant wrist along with a portable metabolic unit to measure energy expenditure during various activities. They used area under the ROC to differentiate the activity from the sedentary behavior. Their experiments concluded that placing accelerometer at hip is better to estimate energy expenditure than to place it on a wrist. But the problem with this approach is that, usual smartwatches won't be suitable for this approach. We need to use special sensors that gives us this ability of performing this task. So far, we have



described different directions of works in this domain. Now, we will discuss some of the important works that used significant features from raw data for classification tasks.

Feature Engineering

Bao et al.[4] used rich set of features for human activity prediction in their work. They developed algorithms to detect physical activities from data acquired using five small biaxial accelerometers worn simultaneously on different parts of the body. Mean, energy, frequency-domain entropy, and correlation of acceleration data were extracted as features from the samples. They calculated energy feature as the sum of the squared discrete FFT component magnitudes of the signal. The sum was divided by the window length for normalization. Frequency-domain entropy is calculated as the normalized information entropy of the discrete FFT component magnitudes of the signal. Correlation is calculated between the two axes of each accelerometer and between all pairwise combinations of axes. These features are very useful for capturing even the slight variations in the signal providing better classification results.


Mannini et al.[5] focused on devising machine learning methods for classifying human physical activity from On-Body accelerometers. The mean, the energy, the frequency-domain entropy, and the correlation coefficients were calculated for inclusion in the feature vector as described in the previous approach. They have used hidden markov models and observed that they perform better for such kind of features.

The efficiency of using wavelet features is depicted in the work by He et al[6]. They have used Wavelet Transform to decompose the raw accelerometer signals and obtain the decomposed signals that can efficiently discriminate the different activities. Wavelet transform decomposes a signal into a set of detailed signals and approximate signals using a set of basis functions. They found that using wavelet features improves the accuracy of classification.

Battery Optimization for Remote Health Monitoring

Battery optimization in remote health monitoring systems is an important area of research. It has significant role to play in user adherence to the experiment. If the device fitted with sensor has to be charged very frequently, then the user may not be interested to participate in the experiment. Here, we review some of the important works that contributed to this intriguing area.

Alshurafa et al.[7] studied battery optimization techniques for Smartphones for Remote Health Monitoring Systems to enhance user adherence. In this work, they presented a new battery optimization technique to increase the battery life of the smartphones used for remote health monitoring. When the phone is connected to a charger, it enters an initial state, where the accelerometer can be turned off. Once the phone is disconnected, it enters an active state,




where the accelerometer is turned on and the sampling rate is set at 10Hz. If the user becomes stationary, where they are sitting on a couch or at the dinner table, little physical activity is captured and the smartphone enters an inactive state, where the sampling rate decreases to 1Hz. They also proposed to reduce the frequency of data uploads to the server.

Alma et al.[8] have studied Bluetooth low energy driven data transfer to improve the battery life of the smartphone for remote health monitoring. They have conducted a preliminary study of the distribution of resources in remote monitoring scenario, regarding power consumption and processing capabilities of smartphone and wireless sensor node. In order to minimize the transmission cost, which leads to higher power consumption, event-driven Bluetooth Low Energy (BLE) communication standard is employed between Wireless Sensor and smartphone. Specific signal processing operations have been implemented in Wireless Sensor so they can be used as a trigger for data transmission between sensor and smartphone, where additional processing can be further conducted.

Deep Learning:

Deep Neural Networks have revolutionized many areas, including but not limited to, health analytics, computer vision and Natural Language Processing. The entire field of neural networks is based on the idea of multi-layer perceptrons, which is made up of a few layers of neuron-like units which does a linear transformation followed by a non-linear activation. Each such layer is called a fully-connected layer, due to the fact a hidden unit in each layer is connected to all the units in the previous layer. To introduce more expressive power, Deep Learning practitioners tend to experiment with really deep neural networks. It all began with AlexNet in 2012, which beat the then state-of-the-art in IMAGENET, an annual image classification challenge, by a huge margin. Since then, Deep Learning has seen an exponential amount of success in various fields. In a way, it has impacted the entire machine learning community as a whole.

The two main types of Deep Neural Networks are Convolutional Neural Networks and Recurrent Neural Networks. The main reason for the success stories of DNNs is that they need very little or no feature engineering. They are very good at learning features themselves, thus eliminating the need for domain experts in coming up with sophisticated features. In other words, neural networks perform efficient representation learning, namely learning essential representations of the input data which can be used for classification of them. Though DNNs have made commendable success in many domains, the main problem with them is that they are data-hungry. The amount of data they need is proportional to the number of parameters they have. This is in contrast to the scenario in many real-world applications, where one cannot collect more high quality data due to various restrictions in cost and effort. Though Deep Learning community has been able to address these limitations using techniques like one-shot learning, they have not been completely successful. In our work, we wish to validate many hypotheses on



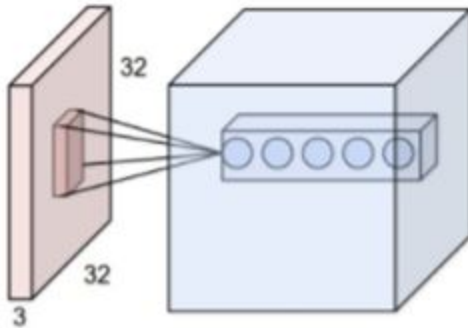
using Deep Neural Networks for our task at hand. First, let us look at the most famous deep learning models and get to know their working principles.

Convolutional Neural Networks:

Convolutional Neural Networks, as the name suggests, are deep neural networks which work on the principle of applying repeated convolutions on various parts of the inputs before learning complex functions on them. The first few layers of CNNs apply repeated convolutions using filters of various sizes followed by subsequent pooling to extract important features from the input data. This is analogous to the famous signal processing technique of obtaining locally distributed correlation coefficients, which convey the presence of strongly distinguishing features for the problem at hand. While in a typical signal processing model, these filters are manually designed, a convolutional neural network learns them by itself. This is often achieved by minimizing a loss function which quantitatively measures the performance of the CNN. The process of loss minimization is achieved using the technique of gradient descent on the loss curve. To perform gradient descent, one must calculate the gradients of the loss function with respect to all the inputs and the parameters of the model. This is often achieved with the help of back propagation. The problem with back propagation in deep networks is that it often leads to vanishing and exploding gradients. This is often counter-acted with techniques like gradient clipping and residual connections. One can refer to literature for more on this.

Now, let us try to understand the mechanism of the working of a typical CNN in a detailed manner. The two main layers in a CNN architecture are Convolutional Layer and Pooling Layer which are shown in the below figure. The convolutional layer is the building block of a CNN. It consists of a set of learnable filters or kernel which are typically small spatially, but extend through the full depth of the input volume. They perform a convolution operation over the input image and output a filter response. Intuitively, a convolutional layer learns different types of filters that activate the some type of visual features. A pooling layer is generally inserted periodically after convolutional layers. The main task of a pooling layer is to progressively reduce the spatial dimensionality of intermediate vectors. This facilitates in decreasing the number of parameters and thus controlling overfitting, Different types of pooling layers have been used in the literature such as Max Pooling or Average Pooling.

Convolution Layer



Pooling Layer

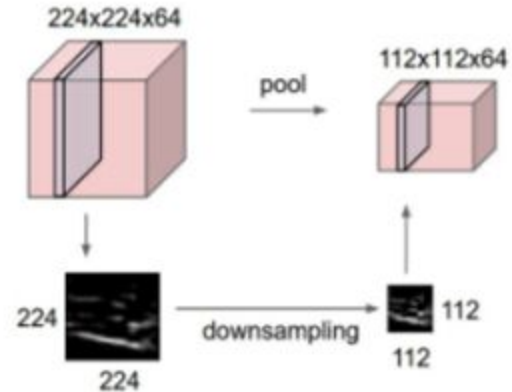
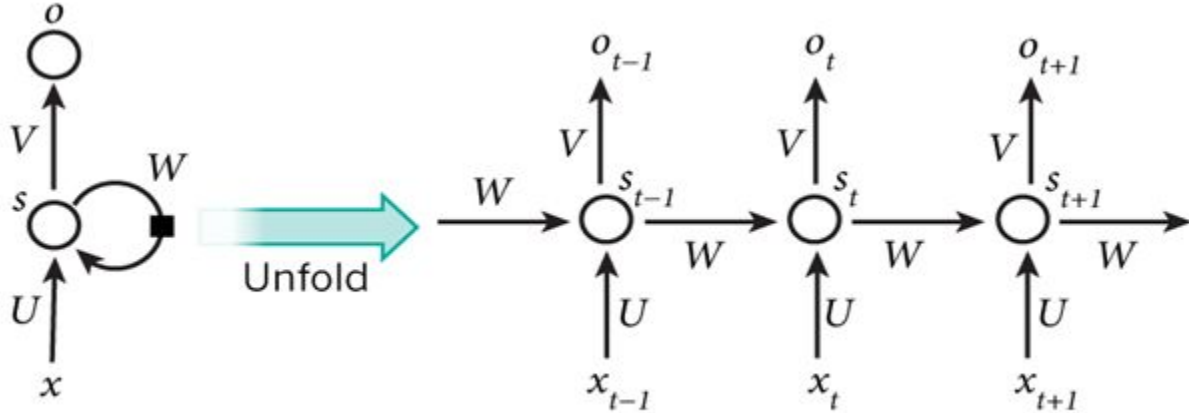


Figure: Different Types of Layers in a Convolutional Neural Network

Recurrent Neural Network

Both feed forward and convolutional neural network consider the input to be independent of each other which may not be the case in applications such as natural language processing and for modelling time series signals. In such cases, we use Recurrent Neural Network (RNN) for understanding the temporal and sequential nature of the input. RNNs are called recurrent mainly because they perform the same task for each element of the sequence, with the current output being dependent upon previous outputs. Another way of visualizing a RNN, is to consider a RNN cell to have a memory which stores the results of previous computations. Now, this memory which captures the information processed until now is used for the calculation of the current output. As we can see in the below figure, a RNN typically consists of one neuron with a feedback loop. However, on unfolding the architecture we observe that the *current output at time t (o_t)*, is dependent upon both the *current input (x_t)* and the *previous output (o_{t-1})*. RNNs have been successfully applied in various fields such as Language Modelling, Machine Translation, Modelling Time Series Data and Speech Recognition Systems.

More common RNN architectures involve stacking multiple layers of RNN cells, to learn complex mathematical functions of the input data. A more recent work in this line of research was Google's NMT system, which achieved state-of-the-art performance in Machine Translation. To alleviate the problem of vanishing gradients, typically RNN architectures involve using stacks of LSTM/GRU cells for modeling time-dependent sequences.



Coming back to the problem at hand, we review the recent work by *Hammerla et al.* as below.

Review: Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables

In this study (*Hammerla et al., 2016*), the authors propose deep, convolutional and recurrent models to tackle the problem of human activity recognition. The authors work on three different dataset that model different types of human activities. The Opportunity dataset contains data from subjects performing manipulative gestures like opening and closing doors. The PAMAP2 dataset models prolonged and repetitive activities such as walking and running. The Daphnet Gait dataset consists of Parkinson's disease (PD) subjects who are asked to perform tasks which result in freezing of gait.

The authors study various deep learning models such as Deep LSTMs, Bidirectional LSTM, CNN, and Deep Neural Network for the three representative datasets. They conclude that Bidirectional LSTM outperforms the state-of-the-art accuracy for Opportunity dataset. They also provide some interesting insights of the deep learning architectures. They state that RNNs outperform CNNs for activities that are short in duration but have natural ordering, whereas CNNs are recommended for prolonged and repetitive activities such as walking or running. They perform an extensive fine tuning of hyperparameters and present how the accuracies vary with different parameters. Finally, the authors conclude that models which have low variability of performance with respect to hyperparameters can be more practically used by a practitioner.

Methodology

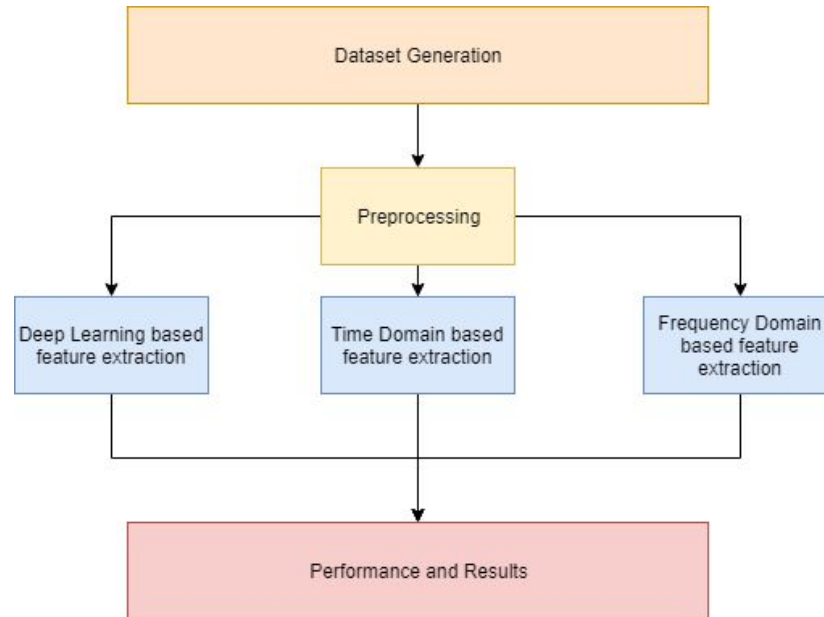


Figure: Flowchart of the system and the processes involved

In the figure above we have given the flowchart of the process that we followed while doing the project. Initially, we began with dataset generation, which involved collecting data for various activities from the smartwatch sensors. The next step was preprocessing the data by using windowing techniques. The third step was to perform feature extraction, which involved extracting deep learning based features, time domain based features and frequency domain based features. Finally we trained few classifiers on these extracted features to see their performance and to get the best model. Now, we will see the detailed explanation of each of the steps mentioned above.

Dataset

For generating the dataset we collected the data from all the sensors from the smartwatch. This data was labeled with the labels laying down, sitting, standing and walking. As an example of how the information is collected consider the accelerometer sensor, which collects the acceleration information for each of the X, Y and Z axis. In addition to this for every instance of the data, we had a timestamp indicating when the data was collected. The sensor in this watch collects data with a sampling frequency of 250Hz. Initially to train our model we collected 2 hours

of data with 30 mins collected by four different persons each, and this we did for all the four activities.

Feature Extraction

Overall, we extracted a total of 16(check) features from the dataset. This features can be mainly categorized into five categories: mean, standard deviation, correlation, features from Fast Fourier Transform and features from Discrete Wavelet Transform. Now, since our data is three dimensional, if we extracted 16 features for each of the axis, the total number of features would be 48, which is too high. Too reduce the dimensionality, we converted the three dimensional data into a single dimensional signal magnitude vector(SMV). This was done for all the features except correlation which was extracted by using combinations of each of the axis. The formula of SMV is given below:

$$SMV = \sqrt{x^2 + y^2 + z^2}$$

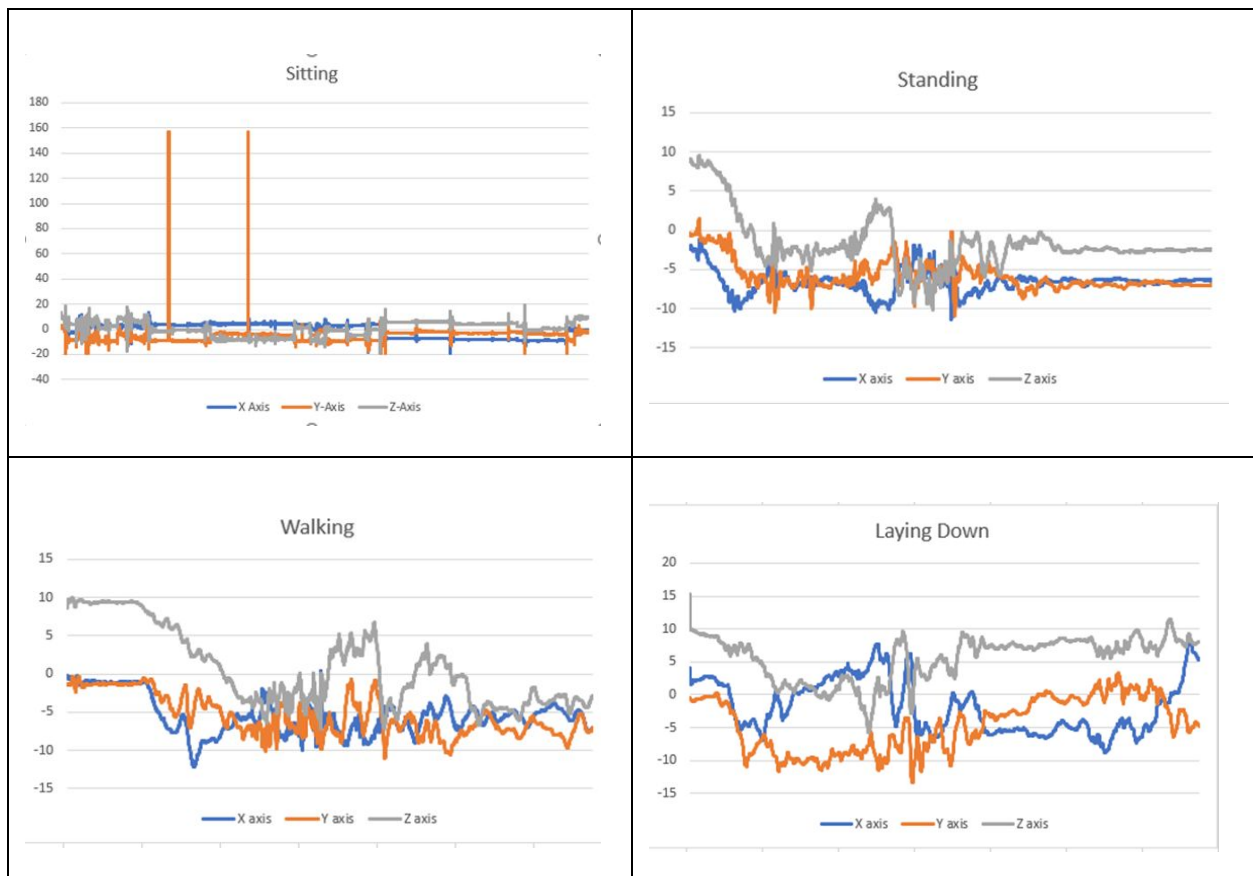



Figure: Graphs for accelerometer data collected in a period of 4 sec for sitting, standing, walking and laying down



In the figure above we have plotted graphs for the data collected from the accelerometer sensor for all the three axis X, Y and Z and for each of the four activities. Each of the graph contains data samples collected within a period of 4 seconds. As can be seen in the figure above, just by looking at the graph we are unable to distinguish between the four activities. Hence we need to extract features from the dataset so that we are better able to distinguish between the activities.

For extracting the features from the dataset, signal segmentation is considered to be an important step. This is because usually the human activities are performed during relatively long periods of time (in the order of seconds or minutes) compared to the sensors' sampling rate which in our case is 250Hz. Additionally just taking a single sample on a specific time instant provides insufficient information to describe the performed activity. Thus, activities need to be segmented and detected in a time window basis rather than in a sample basis.

This segmentation is usually done by employing various windowing approaches such as activity-defined windows, event-defined windows and sliding windows. In activity-defined windowing, the data is partitioned based on detection of activity changes by determining initial and end points for each activity based on the analysis of variations in frequency characteristics. In event driven approach, specific events are identified and they are further used to define successive data partitioning. The window size is not fixed in this approach as the events may not be evenly distributed in time. The last approach is the sliding window approach. This is one of the most used approach in activity recognition and the one that we have used for this project. This approach has proven to be beneficial for the recognition of periodic activities like walking and static activities like standing and sitting. In this approach the data is split into windows of a fixed size with no gap between two windows. Additionally within the sliding window approach one can use an overlapping or a non-overlapping sliding window. Here we tested by using window sizes of 4s, 8s and 12s. We observed that the results for shorter window sizes 4s and 8s were not so good and window size of 12s gave the best results. Even though the shorter window sizes are preferable as they reduce the latency in providing feedback in real-time implementations but the time period is too less for any meaningful activity detection, hence there is a decrease in accuracy with window size. For a given overlap, the choice of window length affects two aspects namely: temporal resolution and coefficient quality. If the window increases in length then we have fewer windows spread across the signal and therefore a greater time interval between each - i.e. lower temporal resolution. The coefficient quality, however, increases with window length. If we perform a Fourier transform over a longer sample we get a higher frequency resolution and each coefficient is therefore more 'representative' of its frequency. Thus, shorter windows leads to wider frequency bins and smoothing/smearing of the coefficients, whereas longer windows leads to narrower, more 'distinct' bins. Hence, we have a tradeoff in

window length here - longer windows give less temporal resolution and better coefficient quality; shorter windows give higher temporal resolution and lower coefficient quality.

Additionally for window sizes of 4s, 8s and 12s we have used a window overlap of 2s, 4s and 4s respectively. If overlapping of windows is not used, its possible that we might miss some information present in the boundary of the windows this could be useful while considering while extracting features from Fourier transforms where if non overlapping windows are not used we might miss frequency jumps. Now, we will give a brief overview of all the extracted features.

Mean

In this we calculated the mean of the SMV for each window. Given the accelerometer data the mean of the data could be useful in separating between standing and non - standing activities such as sitting and laying down. For standing activities the mean will be higher as compared to the non-standing activities. The mean value is calculated by the formula given below:

$$\bar{x} = \frac{1}{n} \left(\sum_{i=1}^n x_i \right)$$

Standard Deviation

In this we calculated the standard deviation of the SMV for each window. Standard deviation could be used to distinguish between activities with lots of movements. So it could be used to distinguish between walking which will have a higher standard deviation and standing which will comparatively have a lower deviation. The standard deviation value is calculated by the formula given below:

$$s = \sqrt{\frac{\sum_{i=1}^N (x_i - \bar{x})^2}{N - 1}}$$

Correlation

For calculation of the correlation we used three dimensional data instead of converted into a single dimension using SMV. Correlations help in determining relationship between two variables by determining the extent to which the two variables fluctuate together. We extracted correlation for each combination of pair of axis. So a total of three features were obtained from this.

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

Fast Fourier Transform

Fast fourier transform uses frequency spectral analysis to distinguish between different types of physical activity. Fast fourier transform takes a signal over a period of time (or space) and divides it into its frequency components. The frequency component with the highest frequency obtained from FFT is chosen as the dominant frequency. This dominant frequency is also known as the frequency of repetition and could be used for activities like walking. In addition to the dominant frequency we extracted the second dominant component to increase the feature descriptiveness. A disadvantage of this fast fourier transform is that it does not provide the time at which this frequency components happened. To overcome this we apply fourier transform repeatedly to different time localized windows. From the fourier transform we obtained the following features:


1. f1 - dominant frequency in each of the window segment
2. p1 - power of the dominant frequency f1
3. f2 - second dominant frequency
4. p2 - power of f2
5. f₆₂₅ - dominant frequency between 0.6 and 2.5 Hz
6. p₆₂₅ - the power corresponding with f₆₂₅
7. p1 / total power
8. f_{1s} / f_{1s-1} - the ratio between dominant frequency at the current segment (f_{1s}) and the segment before (f_{1s-1})

Discrete Wavelet Transform

In addition to the frequency domain features that we obtained from fast fourier transform, we used two time domain features obtained from Discrete Wavelet Transform. Discrete wavelet transform is a transform for which direct sampling of wavelets is done. The idea behind wavelet transforms is that the transformation should allow only changes in time dimension but not in shape. This is achieved by choosing a suitable basis function. The two features that we extracted from DWT are given below:

$$DWT_{SMV} = \sum_{j=\alpha}^{\beta} d_j^2 / SMV^2$$

$$DWT_{SMV1} = \sum_{j=\alpha}^{\beta} d_j^2 / \sum_{j=1}^J d_j^2$$

- 
1. DWT_{SMV} represents the ratio of detail signals between levels α and β to the total power of SMV.
 2. DWT_{SMV1} represents the ratio detail signals between levels α and β to the total power of details between levels 1 and J (we choose $J = 8$)

Classifiers

In the section above, we saw the features that we extracted from the dataset so that we could feed them to our classifiers so that their performance in our classification task improves. For training out models, for this classification task we experimented with Random Forest, SVM, J48, Logistic Classifier. Additionally we also tried the deep learning methods like Convolutional Neural Networks and Recurrent Neural Networks. An overview of how each of the classifiers functions is given below.

Random Forest

Random Forest is an ensemble technique in which every tree in the ensemble is built from a sample drawn with replacement from the training set. The splitting of nodes during the decision tree construction is based among the random subset of features and not chosen from all features. The main parameters to adjust in random forest regression models are `n_estimators` (Number of estimators) which represents the number of trees in the forest. A large value produces better results but also takes a longer computation time. It should also be noted that the results will stop getting significantly better beyond a certain number of estimators. Another parameter which could be adjusted is the `max_features` (Maximum features) which represents the size of the random subsets of features to be considered while splitting the nodes.

SVM

Linear Support Vector Machines aim to learn a vector of feature weights and an intercept, given the training data set. Once the weights are learned, the label of a data point is determined by thresholding $W^T x + b$ with 0, i.e. $\text{sign}(W^T x + b)$. Alternatively, one produce probabilities that the data point belongs to either class, by applying a logistic function instead of hard thresholding, i.e. calculating $\sigma(W^T x + b)$. For extending these SVM's for multiclass classification techniques we perform one vs one classification on all pairs of classes. Eventually an instance is assigned to the class which gets the majority vote.

J48

Decision tree J48 is the implementation of ID3 (Iterative Dichotomiser 3) developed by the WEKA project team. The ID3 algorithm starts with the original set as the root node. On every iteration of the algorithm, it iterates through every unused attribute of the set and calculates the entropy or information gain and then selects the attribute which has the smallest entropy (or largest information gain) value. The algorithm stops when it is either out of attributes or out of examples or every instance in the subset belongs to the same class.

Logistic Classifier

Here we used the Logistic Classifier from Weka which uses a multinomial logistic regression model with a ridge estimator. If there are k classes for n instances with m attributes, the parameter matrix B to be calculated will be an $m \times (k-1)$ matrix. In order to find the matrix B for which negative log likelihood is minimised, a Quasi-Newton Method is used to search for the optimized values of the $m \times (k-1)$ variables.

Convolutional Neural Networks:

Following the trend of using Deep Neural Networks, especially Convolutional Neural Networks, we extend the works of (Hammerla et al. 2016), to develop strong CNN based classifiers for our task. The idea is to not extract specific features from our time-series data and instead, allow our CNN to learn the most important features which are best suited for classification. The input data is of the shape $(N_{ex} \times N_{feat})$, where N_{ex} is the number of time series inputs we have, and N_{feat} is the number of features or input signals we have per time unit. This is '3' as we are just using accelerometer readings and more if we are using gyroscope signals too. To better understand the network we experimented on, we summarize below the architecture specification.

Layer name	Output shape	Number of parameters
Input layer	(None, 2500, 3)	0
Conv Layer 1	(None, 2500, 64)	640
Pooling layer 1	(None, 1250, 64)	0

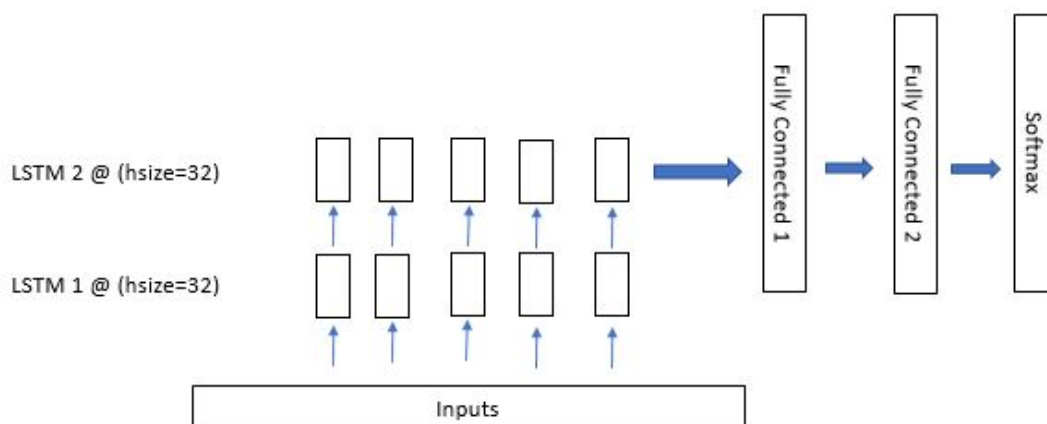
Conv Layer 2	(None, 1250, 32)	6176
Pooling layer 2	(None, 625, 32)	0
Conv Layer 3	(None, 625, 8)	776
Pooling layer 3	(None, 315, 8)	0
Flatten layer	(None, 2496)	0
Dense or Fully Connected	(None, 4)	9988

In summary, the total number of parameters we have is 17580, all of which are trainable. We train using categorical cross entropy as the loss function, which tries to minimize the difference in the cross entropy between the model distribution and the true data distribution. As we can see, we use 3 layers of convolutions and pooling interleaved to gradually reduce the input dimensions while extracting the most important features. Since our inputs are 2 dimensional, in the first layer, we apply 1D convolutional filters of sizes (3x3) to extract features of all '3' (the second '3' in the shape expression) accelerometer values for consecutive time steps of size '3' (the first 3 in the shape expression). Since the number of consecutive time units we can consider is more of a hyper-parameter, we experimented on filters of sizes (3x3), (10x3) and (100x3), and found that (3x3) works the best. We also tried experimenting with 2D convolutions of sizes (50x1), (100x1) and (500x1), but they performed poorer when compared to their 1D counter-parts. The last layer, which is the Fully Connected layer, is the classification layer, which tries to learn a classifier from the important features extracted by the earlier convolution layers.

Our preliminary experiments with CNN gave us an understanding of how CNNs perform when compared to more traditional machine learning algorithms like random forests and logistic regression classifiers. Our results show that, in the presence of limited data, it's always better to use more traditional models than to apply deep neural networks. Being data-hungry models, they were not able to generalize well to the unseen data, thus leading to poor performance on the test set. We report their performances in the results section. Please note that all our deep learning models are trained on limited data, and a possible future work would be to extend them to learn from more labelled data, if available.


Recurrent Neural Networks:

Now, we experiment on recurrent neural networks to see if we can improve the accuracy of our Convolutional neural networks. The intuition behind using RNNs is that they are good at modeling time-dependent sequences. Since our problem at hand consists of input signals which are inter-related, we try to validate our RNNs for this task. Our experiments included using simple RNN cells, LSTM units and GRUs as the cells of our RNN network. We also experimented on various depths of our RNN network (i.e.) stacking multiple RNN layers above each other to see, if we can improve results with more expressive power. Since we had limited data, as per our expectations, our LSTM network did not perform as good as the traditional approaches like Logistics regression and Random Forests. Again, similar to CNNs, even in RNNs, we don't do any feature engineering to see, if our model can learn strong features for classification, as is the case with any strong Deep Neural Network based classifiers. We got a menial performance of around 45% test accuracy with such a network. We illustrate below the architectures we used



Results and Discussion

In this section, we cover the various details of the experiments we have done and the results for the same. Firstly, we present the experimentation settings for the statistical models. To analyse the effect of window size, we repeat the experiments for **three different window settings**. To model the effect of small windows, we consider a window size of 4 seconds with an overlap of 2 second, whereas to model the influence of large windows we use a window size of 12 seconds with an overlap of 4 seconds. Finally, to model windows of medium time interval we use window size of 8 seconds with an overlap of 4 seconds.



In Table 1 and Table 2, we present the performance measures of various classifiers for the different window sizes and on different type of dataset. In Table 1, we present the results obtained by doing a **cross person evaluation**, where we trained the model on the data of three different individuals and tested on the smart watch data from the fourth individual. In Table 2, we present the results obtained by doing a **same-person analysis**. In this analysis, we train the machine learning classifier on the data from four individuals across different sessions and then test on different sessions but from one of the individuals involved in the training. To show the effect of data from different sensors, we present models that are trained only on **accelerometer** and models that are trained both on sensor data from **both the accelerometer and gyroscope**. In the current experimentation, we used **J48, Random Forest, Logistic Regression and Support Vector Machines** for the task of classification. In addition to accuracy, we report the **precision and recall** for each of the models to provide a better insight of the performance.

In Table 1, we have bolded the models that have the best accuracy for each setting for cross-person evaluation. We can observe that on **combining sensor data from gyroscope provides better performance. We also observe that in general the accuracy of the models increases with the size of the window.** This corresponds to our intuition, as given more time it may become easier to predict the human activity. In summary, by using only sensor data from accelerometer we find that J48 Trees perform best with an accuracy of 54.43% on a window size of 12 seconds with an overlap of 4 seconds. By using sensor data from gyroscope in conjunction with data from accelerometer we are able to achieve better results. We observe that **Support Vector Machine with sliding window 8 seconds** performs best for this case. It attains an **accuracy of 68.35%** on the test set.

In Table 2, we have bolded the models that have the best accuracy for each setting for same-person evaluation. We can observe that on combining sensor data from gyroscope provides better performance. **We also observe that the same-person models perform significantly better as compared to cross-person evaluation.** This is understood as the model is able to learn the behaviour of the individuals in training data. We also observe that for all the cases **decision trees perform very well** as compared to other machine learning models. The main reason for this is that decision trees are capable of remembering the training set very well leading to overfitting. As our test set is similar to the training set, decision trees are able to give superior accuracies. In summary, by using only sensor data from accelerometer we find that **Random Forest perform best with an accuracy of 92.57% on a window size of 8 seconds** with an overlap of 4 seconds. By using sensor data from **gyroscope in conjunction with data from accelerometer we are able to achieve better results.** We observe that Random Forest with sliding window 12 seconds performs best for this case. It attains an accuracy of 93.25% on the test set.

Window Size, Overlap Size		Only Accelerometer			Accelerometer + Gyroscope		
		Precision	Recall	Accuracy	Precision	Recall	Accuracy
4 Sec, 2 Sec	J48	0.481	0.506	0.5063	0.543	0.545	0.545
	Logistic Regression	0.474	0.472	0.472	0.491	0.507	0.507
	SVM	0.425	0.522	0.521	0.643	0.648	0.6475
	Random Forest	0.463	0.465	0.46519	0.439	0.553	0.554
8sec, 4sec	J48	0.489	0.522	0.522	0.559	0.573	0.573
	Logistic Regression	0.485	0.462	0.462	0.569	0.576	0.5759
	SVM	0.433	0.516	0.5158	0.683	0.684	0.6835
	Random Forest	0.302	0.421	0.421	0.425	0.528	0.528
12sec, 4sec	J48	0.49	0.544	0.5443	0.507	0.525	0.525
	Logistic Regression	0.474	0.532	0.5316	0.605	0.611	0.611
	SVM	0.445	0.506	0.5068	0.677	0.674	0.674
	Random Forest	0.282	0.43	0.431	0.421	0.532	0.5316

Table 1: Performance Measure of Various Settings using Cross-Person Evaluation

Window Size, Overlap Size		Only Accelerometer			Accelerometer + Gyroscope		
		Precision	Recall	Accuracy	Precision	Recall	Accuracy
4 Sec, 2 Sec	J48	0.906	0.905	0.9045	0.927	0.923	0.9226
	Logistic Regression	0.753	0.745	0.745	0.821	0.821	0.8206
	SVM	0.712	0.723	0.717	0.769	0.758	0.758
	Random Forest	0.923	0.92	0.9204	0.934	0.932	0.9324
8sec, 4sec	J48	0.906	0.9	0.903	0.924	0.92	0.92
	Logistic Regression	0.692	0.682	0.682	0.822	0.82	0.8203
	SVM	0.752	0.727	0.7266	0.754	0.743	0.7473
	Random Forest	0.929	0.926	0.9257	0.933	0.929	0.9285
12sec, 4sec	J48	0.915	0.911	0.913	0.909	0.905	0.9045
	Logistic Regression	0.689	0.678	0.678	0.818	0.815	0.815
	SVM	0.806	0.78	0.78	0.752	0.746	0.7457
	Random Forest	0.928	0.924	0.9239	0.936	0.932	0.932

Table 2: Performance Measure of Various Settings using Same-Person Evaluation

In the below Figure, we report the **confusion matrix** for the best model in case of cross-person evaluation i.e. SVM on accelerometer and gyroscope data with a window size of 12 seconds and overlap of 4 seconds. We also present the confusion matrix for the best model in case of same-person evaluation i.e. Random Forest on accelerometer and gyroscope data with a window size of 12 seconds and overlap of 4 seconds.

Walking	Sitting	Laying	Standing	<- Classified as
67	3	7	3	Walking
4	71	12	12	Sitting
2	30	39	0	Laying
8	12	10	36	Standing

Figure: Confusion Matrix for SVM on Cross Person Evaluation

We also try to combine the features from all the sensors, but observe that they have a very poor performance due to overfitting. On combining data from different sensors the number of features had increased drastically, and with the limited amount of data we had collected, the models very overfitting to the training set. We attained an accuracy as high as 99% for SVM on train set and the performance on test set was very poor. Hence, we disregarded this approach for the project.

Walking	Sitting	Laying	Standing	<- Classified as
311	4	0	1	Walking
12	819	9	10	Sitting
16	48	472	7	Laying
34	9	0	458	Standing

Figure: Confusion Matrix for Random Forest on Same Person Evaluation

We then approach the problem using **deep learning approaches**. We present the results for various approaches in the Table 3. We observe that due to the lack of sufficient training data the models fail to provide an impressive accuracy, however as the current state of art methods are based on the deep neural networks, we have implemented them to understand the underlying architectures. We have tried using one-dimension convolutional neural networks(**CNNs**), two-dimensional convolutional neural networks along with **LSTMs and GRUs**. We present the results on cross-person analysis in Table 3. We also implemented **stacked models**, but observed the performance to be poor, due to the huge number of parameters in the architecture and comparatively less data. In future, we would like to collect more data and as well train the models using GPUs to reduce the training time.

Finally, **as per the discussion during the presentation**, we tried to remove the features extracted using correlation and ratio of frequency from previous windows. We observed that the **performance of the models drastically decreased** as compared to the previous models. In case of cross-person evaluation, we observe that Support Vector Machine which gave the best performance with an accuracy of 68.34%, had its **accuracy decreased to 54.1189% for window size of 8sec**. Other classifiers like J48, logistic regression and random forest also had **reduced accuracy of 49.05%, 56.7% and 44.05% respectively**. This analysis helps us understand the removed **features play an important role** in the automated models and hence should be incorporated into the classifiers.

Deep Learning Architecture	Train Accuracy	Test Accuracy
1D Convolutional Neural Network	0.9249	0.4984
2D Convolutional Neural Network	0.8956	0.4512
LSTM Neural Network	0.7932	0.4503
GRU Neural Network	0.8231	0.437
Stacked LSTM	0.7437	0.4536
Stacked GRU	0.7732	0.4340

Table 3: Performance Measure of Various Deep Learning Architectures using Cross-Person Evaluation

Conclusion

The goal of this project was to build a model to predict the user positioning like standing, walking, sitting and laying down given the data collected from smartwatch sensors. The first step to solve this problem was to extract features from the dataset. Based on our literature review, we extracted mean, standard deviation, correlation, frequency domain features and time domain features. After extracting the features we experimented with J48, SVM, Logistic Regression and Random Forest classifiers to select the one which performed the best. Additionally, based on our survey we realized that window size and overlap plays an important role in determining the model performance hence we experimented with different configurations for that. Next we



experimented with deep learning models like CNN's and RNN's, contrary to our expectations the data hungry deep learning models didn't perform as well thereby validating the Occam's razor.

After our experiments we found that the best model in case of cross-person evaluation is SVM on accelerometer and gyroscope data with a window size of 12 seconds and overlap of 4 seconds. The best model in case of same-person evaluation is Random Forest on accelerometer and gyroscope data with a window size of 12 seconds and overlap of 4 seconds. One explanation for impressive performance of Random Forests is that it is capable of remembering the training data perfectly. As our test data in case of same-person analysis is from an individual in the train-set, random forest is able to perform well. However, we can observe that in case of cross-person analysis, its performance decreases as it fails to generalize well. In such cases, we observe that SVM performs better. One possible future work for this project would be to collect more data and use more computing power to model richer deep neural networks, thereby harnessing the power of automatic feature extraction.

References

- 1) 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.
- 2) 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.
- 3) 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.
- 4) Bao, L., & Intille, S. S. (2004, April). Activity recognition from user-annotated acceleration data. In *International Conference on Pervasive Computing* (pp. 1-17). Springer, Berlin, Heidelberg.
- 5) Mannini, A., & Sabatini, A. M. (2010). Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, 10(2), 1154-1175.
- 6) He, Z. (2010, December). Activity recognition from accelerometer signals based on wavelet-ar model. In *Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on* (Vol. 1, pp. 499-502). IEEE.
- 7) Alshurafa, N., Eastwood, J., Nyamathi, S., Xu, W., Liu, J. J., & Sarrafzadeh, M. (2014, May). Battery optimization in smartphones for remote health monitoring systems to enhance user adherence. In *Proceedings of the 7th international conference on PErvasive Technologies Related to Assistive Environments* (p. 8). ACM.
- 8) Alma, S., Jasmin, N., Nermin, S., & Aljo, M. (2016, June). Improving energy-efficiency of real-time health monitoring using wireless sensors and smartphones. In *Telecommunications and Signal Processing (TSP), 2016 39th International Conference on* (pp. 396-399). IEEE.
- 9) Hammerla, N. Y., Halloran, S., & Ploetz, T. (2016). Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880*.