AT UMEÅ UNIVERSITET

Cross-Platform Modelling for Human Activity Recognition System

Author
Dan ARVIDSSON

Supervisor
Priyantha WIJAYATUNGA

Abstract

Human activity recognition (HAR) systems have a large set of potential applications in healthcare, e.g. fall detection and tracking physical activities. HAR systems based on wearable sensors have gained the most attraction, due to smartphones having these sensors embedded in them. This makes them a great candidate for collecting human activity sensor data. By utilizing the smartphone sensors, no other sensors need to be supplied and instead only a mobile application needs to be supplied. However, this comes with a trade-off, sensors embedded in smartphones display specific heterogeneity and biases, depending on platform and price range. Normally in such a scenario, multiple HAR systems have to be built and trained for each device. This is both a time consuming effort and gives no guarantees that the different systems will have similar activity recognition accuracy. Therefore, in this thesis, a HAR system is constructed, where classification methods and filtering techniques are explored and evaluated, in an effort to give some guidelines for how to construct a HAR system, that can be embedded in multiple platforms.

This study shows that when considering a few common activities, this HAR system performs well even when sensor data is collected from multiple sources. Ensemble method AdaBoost, in combination with decision trees, gives the overall best performance. Filtering techniques, such as Butterworth and Chebyshev performs better than constant- and linear detrending. This is primarily due to their ability to distinguish between low frequency activities, such as standing and sitting. The best result in this study was given when combining Chebyshev filtering and AdaBoosted decision trees, with a F-score of 0.9877.

Sammanfattning

Klassificering av fysika aktiviteter för multipla plattformar

Klassificering av fysiska aktiviteter har många potentiella applikationer inom sjukvårdssektorn. Exempelvis kan det användas för att summera fysika aktiviter över en längre tid, vilket sedan kan utnyttjas till att kartlägga samband mellan fysisk aktivitet och risk för olika sjukdomar. Klassificeringssystemet av fysiska aktiviteter i den här uppsatsen använder sig av data från bärbara sensorer. Dessa sensorer har fått ett ökat intresse framförallt på grund av att de även finns tillgängliga i de flesta mobiltelefoner. Om man kan använda mobiltelefonernas sensorer skulle inga andra specifika sensorenheter behöva tillhandages, utan endast en nedladdningsbar applikation behöver tillhandahållas. Däremot försvåras klassificeringen av att olika plattformar och prisklasser på telefonerna ger upphov till heterogenitet och andra enhets specifika skillnader i sensordata. Normallt sett i det fallet, har även många olika klassificeringssystem behövt tränats för enskilda enheter. Det är både tidskrävande att genomföra och ger ingen garanti att de olika systemens resultat är jämförbara. I den här uppsatsen byggs därför ett system där ett antal filtrerings tekniker och klassificeringsmetoder utvärderas, för att ge grundläggande riktlinjer om hur de olika metoderna i systemets subprocesser påverkar klassificeringsprecisionen, när sensordata kommer från många olika plattformar.

I den här studien visar vi att när man betraktar ett fåtal vanliga aktiviteter så håller klassificeringssystemet en hög precision, även när sensordata kommer från olika plattformar och enheter. Bäst resultat gavs med ensemble metoden AdaBoost tillsammans med beslutsträd. Filtreringsmetoder som Butterworth och Chebyshev ger bättre klassificering precision än de mer simplistiska filtreringsmetoderna konstant- och linjärtfilter. Det beror framförallt på systemets förmåga att urskilja på lågfrekvens aktiviteterna, sitta och stå vid användning av de två förstnämnda filteringsmetoderna. De bästa resultatet för det här klassificeringssystemet gavs med kombinationen av Chebyshevfiltrering och AdaBoost-klassificeringsmethod, med ett F-värde på 0.9877.

Popular scientific summary

Being physically active helps individuals to stay healthy, so having a way to keep track of an individual's physical activity can have many practical applications. For example, an application that keeps track of physical activity over time, can be used to displaying graphical summaries and deliver personalized recommendations.

This can be accomplished by training a system to distinguish between different physical activities, such as running, walking and standing for example. Wearable sensors are attached to the individual, which interprets the acceleration being applied to it and collects that information. That information is then used by the activity recognition system, to recognize what activity is being performed.

Nowadays most smartphones have these types of sensors embedded in them, which makes them both an easy and cheap way of collecting the necessary sensor data needed to distinguish between physical activities. By using smartphones as a collection method, also means that no other sensor units needs to be supplied. However, smartphones do come with other challenges, the sensor readings will differ depending on brand and price range. That is, during the exact same activity and time period, differences in sensor readings will occur among devices.

Therefore, different techniques of processing the sensor readings, and methods for distinguishing between physical activities is evaluated. The results show that when using proper techniques of processing the sensor readings and methods for recognition, it is possible to preserve a high accuracy in human activity recognition, even when sensor readings have been collected from multiple platforms.

Contents

List of Figures

List of Tables

1 Introduction

Physical inactivity has been well documented to be a critical risk factor for various diseases and disabilities, especially as we grow older [2]. In addition to that, our average life expectancy have also steadily been increasing and is projected to continue in that same direction [3]. This will inevitably lead to a growing need for efficient elderly care and health promotion. Therefore within the healthcare sector, there is an interest in the study of Human Activity Recognition (HAR) systems as a tool for various applications, such as pedometers, calorie burning estimation, anomaly detection (e.g. fall detection) and activity tracking over time (e.g. summarizing activity and trend detection). The question this thesis will try to answer, is how to model such a classification system, that can be embedded in applications across multiple platforms. The rest of this chapter will give an introduction to HAR systems and the objective and motivation for modelling one classification system for multiple platforms.

1.1 HAR systems

HAR system is a collection of sequential processes, that in some manner transform and analyses incoming sensor signals, to distinguish between a set of physical activities. HAR systems can be categorized into three major types, video based (e.g. [4], [5]), environmental based (stationary sensors, often in relation to smart home/living setting, e.g. [6], [7]) or systems that utilizes wearable sensors, which is the type of HAR system this thesis is based on.

1.2 HAR based on wearable sensors

This type of system has gained the most attraction, due to the increasing quality of lightweight wearable sensors and the fact that the majority of smartphones now have such sensors embedded in them. Further argument for smartphones as a data collection unit is how well integrated they are in our daily lives nowadays. This makes the data collection not only easily accessible and efficient, but also greatly reduces the cost of acquiring the sensor data. With smartphones as a data collection unit, no other sensor units need to be supplied to the subjects either, i.e. all necessary data needed to distinguish between physical activities can be collected with the smartphone's sensor.

However, lets first ignore smartphones as a data collection unit and only focus on wearable sensors, whose only objective is to collect sensor readings and nothing else.

Then there still are some key factors that will impact the sensor readings and can be summarized with four points:

- Positioning of the sensor in relation to the individual's body. In most studies so far people have either tried to determine the impact of changing the position or trying to determine some optimal position, depending on the physical activities considered, e.g. [8], [9], [10].
- Number of sensors attached to the individual, e.g. [9], [11].
- Which type of sensor readings and how many types are incorporated in the system, e.g. [8], [10]. By for example, determining which type of sensor reading is the most impactful for distinguishing between activities.
- Individual movement patterns. Previous studies have reported that factors such as gender, height and weight of an individual will have an effect on the signal readings, e.g. [8]. From [12] we can see further such arguments and also find recommendations for collecting sensor data. It should preferably be made from multiple subjects with a large set of diverse characteristics, to ensure that the system can handle a wide set of individuals movement patterns after the system have been trained and implemented.

In most cases at least some combination of these are considered when building a HAR system. More over, all these key factors is equally true when adding smartphones as a collection unit of sensor data. But when collecting sensor data from smartphones, we also need to add two more considerations. First, there will be variation in sensor readings, due to hardware differences between different platforms and price range. Secondly, it is not only hardware that differ among platforms, so does the software and this can also lead to differences in sensor readings. So by adding smartphones as a data collection method, we need to add one more key factor that will impact the sensor readings:

- With both hardware and software differences, we have to expect units to display both heterogeneity and unit specific biases [13].

1.3 Objective & Motivation

In this paper, an experimental HAR system is built to primarily handle two of these factors, individual movement pattern in combination with sensor readings collected from multiple platforms. This HAR system is built to distinguish between five physical activities:

$$activity_{set} = \{Cycling, Running, Sitting, Standing, Walking\},$$

in first hand for the targeted population of elderly persons. Other than age group of the individuals, no differentiation is being made on factors, such as height, weight, gender or other individual traits that may ultimately affect the sensor readings. Differences in sensor reading from these type of individual traits are instead put on the HAR system to handle and it should be capable of doing so without having them explicitly defined.

The objective here is to fill the gap in research about HAR systems, by evaluating the proposed system's ability to distinguishing between physical activities when sensor data is collected from multiple subjects and platforms. Investigating choices of filtering methods (the second step in the HAR systems workflow Figure 1) and evaluating three machine learning algorithms for classifying human activity (the fourth step in the workflow Figure 1). The main motivation for this research, can be summarized with two key points:

- Extracting sensor readings and storing them can easily be done, but stored data does not directly translate it to any usable information. These sensors can run and collect readings over long periods. This gives rise to a quickly increasing size of stored data, data for which could have been collected from many different platforms and devices. Labeling such sensor reading, when no prior knowledge exist of what activity was performed, will be riddled with errors. But more importantly, as the database explodes in size, manually labeling such sensor readings becomes unfeasible.

- Building a HAR system, that can be embedded across multiple platforms. Normally when considering multiple platforms and different devices, multiple HAR systems have to be built and trained independently for each platform and device. This is not only time consuming, but has no guarantee that they will give comparable performances to each other. Hence, investigating the possibility for

cross-platform modelling, would simplify the implementation process.

From this study, some preliminary analysis and guidelines will be presented for building a HAR system, when setting individual movement pattern and multiple platforms in focus. To accomplish this, data from multiple subjects were collected with three different devices (Actigraph, Iphone and Android) and together they satisfy the cross-platform requirements, both in terms of software and hardware specifications.

1.4 Modelling scheme

This HAR system will utilized a sliding window approach, the most widely used segmentation technique in activity recognition [14]. First, the HAR system receives raw accelerometer data (described in Chapter 2.1) as input, with an input size equal to the size of the window. In the next step, that raw accelerometer data is filtered to remove unwanted noise (using one of the four filtering methods described in Chapter 2.5.1). After the filtering process, features are calculated and extracted (described in Chapter 2.5.2). Finally in the last step, the trained classification methods is used to predict what physical activity was performed duration that time period, inside the sliding window (described in Chapter 3).

The workflow of this system is illustrated in Figure 1, where all signal processing, feature calculations and human activity prediction, is done directly inside the sliding window. This enables the system to both predict already stored full length sensor sequences and has the ability to give in real time predictions when embedded in a mobile application.

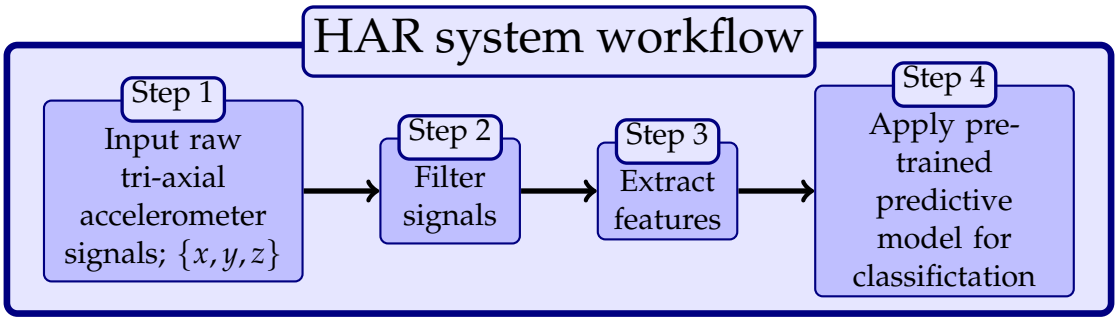

Figure 1: Workflow for the HAR system. Input raw accelerometer data to the system for which it filters the tri-axial signals, then calculates and extract features and feed those values to the predictive model for actvity classification.

1.5 Rest of this thesis

The rest of this thesis is structured to first give an overview of the sensor data. How the data was collected, methods for processing the data and features extracted from the raw sensor data. A brief introduction is given to the algorithmic operations used to train the models. Then results are evaluated, by comparing classification accuracy between combinations of filtering- and classification methods in the HAR system. Finally these results are discussed and compared with some other similar studies.

2 Data

In this chapter a full presentation of the sensor data will be given. First presenting how the raw sensor data was collected and give an overview of what accelerometer signal data is. Then describing the signal processing performed to the raw sensor data and the features extracted from the processed sensor data.

2.1 Accelerometer signal data

There are many different types of sensor data that can be collected from lightweight wearable sensors, such as gyroscope, accelerometer, magnitude field data and so on. In this paper the only sensor type considered was accelerometer signal data. The sensor-framework used to collect the sensor data, uses a tri-axial coordinate system to express the acceleration and direction for which the device is being pushed. We denote these three axes as x, y and z. In this framework, the acceleration being applied to the device is measured in m/s^2. For the two smartphones (Android and Iphone device), the tri-axial coordinate system is defined relative to the screen. That is, the x-axis is horizontally, y-axis vertically and z-axis is pointing in the same direction as the screen. The same tri-axial coordinate system is also used for the Actigraph device, however the Actigraph is a sensor device without any screen. These wearable sensors collects all acceleration being applied to them, this includes the force of gravity. In other words, when a device in our perspective is motionless, the device reads an acceleration of approximate $g = 9.81 m/s^2$ [15].

2.2 Data collection

The raw accelerometer data was collected by Healthy Ageing Initiative research group for behavioral medicine in Umeå Västerbottens län. The data collection was done through an in-house supervised examinations and during the examination each subject performed a set of activities, while wearing these sensor devices attached to their bodies. In total the sensor data sequences consist of raw accelerometer data, collect for a time period of about 20 min for each subject.

Table 1: Each subject started off by standing still then walked in an increasingly faster pace, before transition into running. Then sat down and afterwards performed the last two activities, walking uphill and cycling.

Order	Activity	Levels
1.	Standing	
2.	Walking	$(2/3/4\ km/h)$
3.	Running	$(5/6/7\ km/h)$
4.	Sitting	
5.	Walking Uphill	$3km/h$
6.	Cycling	$(20/30\ km/h)$

During the data collection process, each subject performed the activities in Table 1 and in that order with corresponding levels for the activities where the subjects weren't motionless. Some mild deviations from the above mentioned pattern did occur for various reasons. During the collection process of accelerometer data each subject were attached with the following three devices:

Notation	Brand/Model
Actigraph	Actigraph wGT3X-bt
Android	Samsung Galaxy S8
Iphone	Iphone 5

where the device notation is followed by the actual brand and model. In total, the accelerometer signal data was collected from 14 subjects. Along side the collection of accelerometer data, some meta data about device rotation and placement were also

noted for each device and subject (found in Table 2). In this study, the placement of devices varied among subjects, where they were either positioned on the subjects left or right side. Further the devices were either attached to the subjects belt or loose in their pocket. However the devices were always placed with the same orientation and screen direction, i.e. facing downwards and inwards towards the subject.

Table 2: Notations of how each device was attached to the subject.

Notation	Summary
Position:	on the individual, either left or right side
Orientation:	facing downwards or upwards
Screen:	facing inwards or outwards
Attachment:	attached on the belt or loose in a pocket

2.3 Sequence overview

Below in Figures 2 and 3, are full length raw accelerometer sequences from the three devices and for two different subjects. In each subfigure, the x-axis is on top in green, y-axis in the middle and red, then on the bottom in blue is the $z - axis$. From left to right, the first subfigures represents accelerometer data collected with the Android device, then Iphone in the middle and on the right Actigraph. Both the Iphone and Actigraph device had a sampling frequency of $30Hz$, while the Android device was set to $50Hz$. This can easily be seen in the figures, where the Android device have significantly more sampling points for the same time period then the other two devices. In Figure 2, the Android and Actigraph devices were attached on the subjects left side and belt, while the Iphone was attached on the right. In the middle section of these sequences during the sitting interval (interval 3), the device specific noises are quite different between the devices. Both the Android and Actigraph device, have significantly more irregularities and noise than Iphone. Even the type of irregularities differs between Android and Actigraph. Android has recurring sharp spikes during the transition from sitting to walking uphill (second interval denoted as 5), while the noise for Actigraph is more in form of a moving trend. This is despite them being attached right next to each other, on the subjects belt and right side. Another examples of sensor differences occurring between devices, can be found in Figure 3. In the end of the running interval (interval 2), the x-axis values for the Android device goes from being pushed towards negative

values to positive values. This is due the device being loose in the subjects pocket and slightly changed its orientation. These types of differences are important to note, since they can have a drastic effect on extracted features.

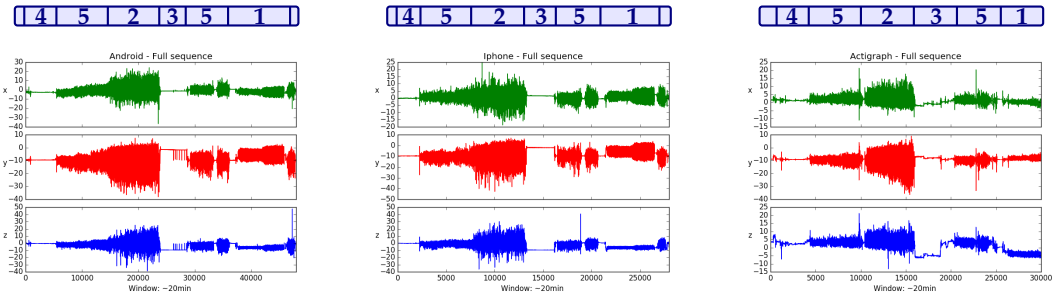

Figure 2: Full length raw accelerometer signals, where each device have been plotted for one subject. Numerical labels: 1 = *Cycling*, 2 = *Running*, 3 = *Sitting*, 4 = *Standing* and 5 = *Walking*.

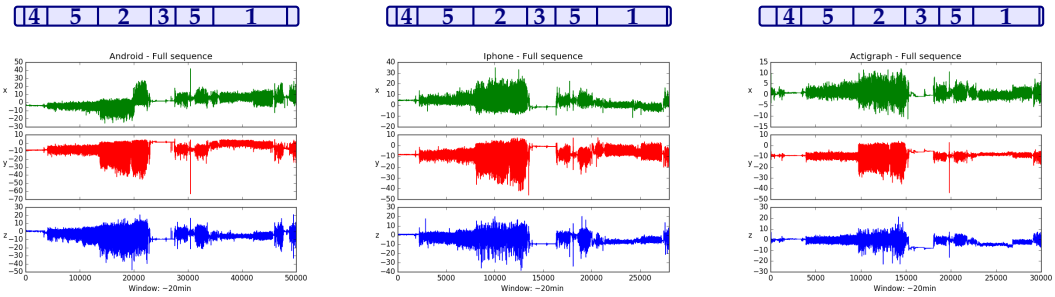

Figure 3: Full length raw accelerometer signals, where each device have been plotted for one subject. Numerical labels: 1 = *Cycling*, 2 = *Running*, 3 = *Sitting*, 4 = *Standing* and 5 = *Walking*.

2.4 Window Size

When utilizing a sliding window approach, the size of the window will affect the overall performance and computational time [16]. However, in this paper no personal testing for different window sizes have been performed, nor the effect of changing it. The window size was chosen and afterwards considered fixed. First consideration for choosing the window size, were primarily to have a sufficiently large window. By sufficiently large in this case refers to the fact that the devices have different sampling rates, so the devices with the lowest sampling rate should have enough sampling points inside the window for stable feature calculations. Besides having a sufficiently

large window size, results from [14] were used as a guideline for what a good choose of window size might be. The recommendations are quite wide, depending on activities and whether one is considering recognition speed or performance. In this thesis, performance is considered more important then recognition speed. By combining the need for a sufficiently large window and the relevant window size recommendations, the window size was set to 2.5 seconds.

2.5 Processing data

The original data was collected and stored as raw accelerometer signals. During the collection process only approximate handwritten notations were made, where they wrote down the time period and which activity was performed. Therefore, the first step of processing the raw accelerometer data was to manually adding activity labels to the data points in the collected data sequences. The different activity intervals of the accelerometer sequence for each subject and device, were labeled into one of the following classes:

$$activity_{set} = \{Cycling, Running, Sitting, Standing, Walking\},$$

where the two intervals, walking and walking uphill in Table 1, were concatenated into one common group, walking.

During the activity labeling process of the raw accelerometer data, each sequence were plotted and checked against the handwritten time notations and the system time of that device. Parts that matched and had no extreme time deviations between the notations, were labeled into the corresponding activity. Observations as those made in Chapter 2.3 regarding Figures 2 and 3, were not removed and is considered to be perfectly reasonable occurrences. After the data labeling process was completed, the raw accelerometer signal sequences were segmented into non overlapping windows. Each segmented window represents a 2.5 second time period, where the subject performed one of the activities. All further signal processing and feature calculations were performed independently for each segmented window.

2.5.1 Filtering

The raw accelerometer signal data includes acceleration from both the subjects movement and gravity. Besides that, the signals also include other noise, which preferably

should be segmented from the part that represents the subjects movement. It is important to note, that the acceleration from gravity would not have been a problem if devices can be considered as a fixed object. Under those conditions, gravity could instead be a helping hand to distinguish between motionless activities, such as sitting and standing. This can be seen in both Figure 2 and 3, by comparing the intervals marked as 4 (standing) and 3 (sitting). In interval 4, where the subjects are standing, the acceleration in the frequency domain is approximately around 0, for both the x-axis and z-axis, while the acceleration from gravity is picked up by the y-axis. However for both these subject, during the sitting interval, that acceleration is instead logged under the z-axis. That is, if the device has a fixed and known orientation and screen direction (as described in section 2.2), then for any period where the subject is standing, we would expect values in the frequency domain to be, approximately:

$$standing = \begin{cases} 0, & x-axis \\ -9.81, & y-axis \\ 0, & z-axis \end{cases}$$

and during a sitting time period:

$$sitting = \begin{cases} 0, & x-axis \\ 0, & y-axis \\ -9.81, & z-axis \end{cases}$$

This would make it exceptionally easy to distinguish between the two activities. But such rigorous setting for a smartphone is not reasonable in practice and the notion of using that information is predicated on the device being both fixed and known. Therefore the acceleration from gravity, should also preferably be removed from the acceleration that represents the subjects movement. In this study, four different filtering methods were considered and evaluated. All filtering of raw accelerometer data is done directly on the sampling points inside the size of the sliding window, where each axis in the coordinate system were filtered separately. The four filtering method considered is denoted and characterized as:

Constant- & Linear filtering: Constant filtering of raw accelerometer data, fits a simple least squares fit, with only the intercept term. That is, it removes only the mean trend along each axis. Linear filtering, fits a ordinary least squares fit, which also removes any linear trend inside the window.

Chebyshev- & Butterworth filtering [17, pp. 333–342]: This filtering methods do not only remove any mean trend along the axes, but they also removes low frequency noise. One high-pass was used to segment signal noise from the body acceleration, with a cut-off frequency set to $0.2Hz$. That is, any frequency response inside the window below the cut-off, is segmented away from the rest of the accelerometer signal and discarded.

2.5.2 Feature Extraction

After filtering the raw accelerometer signals one time with each filtering method, features were calculate and extract from the segmented windows. Extracted features are presented in Table 3 and where peak based features are illustrated in Figure 4.

Table 3: Summary of features included in the final feature sets with corresponding calculations and notations.

Feature	Calculation	Notation summary
F_{sma}	$\frac{1}{n_w}\sum_{i=1}^{n_w}(\lvert x(i)\rvert + \lvert y(i)\rvert + \lvert z(i)\rvert)$	Signal magnitude area where n_w is the number of time points inside the 2.5s window.
F_e	$\frac{1}{n_w}\sum_{i=1}^{n_w}\lvert x(i)\rvert^2$	Mean energy inside the window n_w (Same calculations for $\{y,z\}$-axis)
F_g	$\frac{1}{n_w}\sum_{i=1}^{n_w-1}\lvert \Delta x_{(i+1,i)}\rvert$	Mean absolute gradient inside the window n_w (same calculations for $\{y,z\}$-axis)
F_{cor}	$\frac{cov(x,y)}{\sigma_x\sigma_y}$	Pairwise correlation inside the window n_w (same calculations for all pairs $\{x,y,z\}$)

Besides those features in Table 3, the peak based features were computed in an attempt of capturing the rhythmic differences among the activities. In previous studies of HAR systems utilizing wearable sensors, other approaches have been tried to capture

the rhythmic nature of human motion, e.g. those in [8], [18], [19]. In this paper, the rhythmic features utilizes both the time and frequency domain. Averaging the peak feature measurements between the closest connected low to high peak and high to low peak. This is done for all three axes in the coordinate system. An example of high to low peaks inside a window are marked in the first subfigure in Figure 4, where dots marks the high and low peaks and a line have been drawn connecting the peaks. The peak based feature and their corresponding domain can be defined as:

- **Time-domain**: Average time between $low \rightarrow high$ peak, and $high \rightarrow low$ peak.
- **Frequency-domain**: Average frequency response between $low \rightarrow high$ peak, and $high \rightarrow low$ peak.
- **Frequency-domain**: The distance between the average frequency of high and low peaks.
- **Both-domains**: Average gradient between; $low \rightarrow high$ peak, and $high \rightarrow low$ peak.

Due to devices having varying sampling rate, distances between frames in the time-domain is not directly comparable to each other, so the average time distance between peaks have therefore been weighted such that:

$$average\ time\ between\ peaks = \frac{100}{n_w},$$

where n_w is the number of frames inside the window for that specific device. In an attempt to correct the time measurement and to better represents the same temporal distance between peaks.

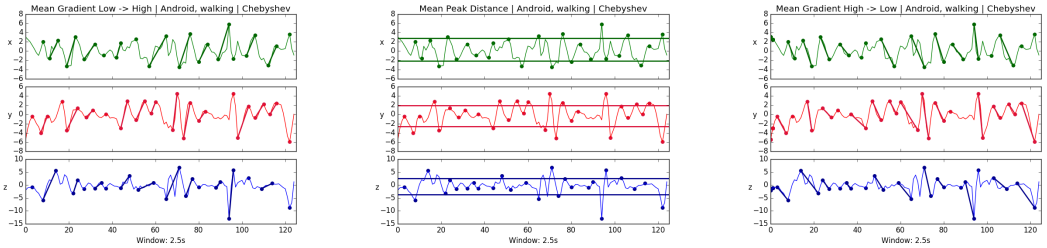

Figure 4: Peak feature calculations are based on the marked high and low peaks. The accelerometer signal data comes from the android device after Chebyshev filtering in a 2.5 second interval while walking.

2.6 Final processing of dataframe structure

The final processed data consist of 8808 observations from the non-overlapping segmented windows. Data from the different devices were split into a training, validation and test set, separately from each other and only afterwards were each set concatenated. It was done in this manner to ensure that each dataset would contain a similar number of observations from each device. After all data processing was done, the number of observations from each devices, activity and in each dataset is summarized in Table 4.

Table 4: In this table the number of observations for each device, the number of observations for the five activity labels and the number of observations in each dataset are summarized.

Device		Activity class-label		Dataset	
Device	Number of observations	Label	Number of observations	Set	Number of observations
Android	3979	Bicycke	1781	Train	4623
Actigraoh	2444	Running	2001	Validation	1541
Iphone	2385	Sitting	845	Test	2644
		Standing	1556		
		Walking	2625		

3 Methods

In this chapter an overview of the models and their algorithmic operations used to train and fit the sesnor data will be given. Describing decision trees, ensemble methods and the basic structure of neural nets.

3.1 Decision Trees

Let $\boldsymbol{X}$ denote a features space of n observations and p predictor variables, then the decision tree (DT) classifier is a way of partitioning $\boldsymbol{X}$ into a set of nodes. The process of growing a tree (Table 5) is done to separate a set of K categorical labels, with the p predictor variables in the feature space $\boldsymbol{X}$. A tree is grown recursively, starting with one single node (the seed), where one splitting rule is determined for each one of the p predictors. The feature with its corresponding splitting rule, used to split the feature space into two daughter nodes (regions) is then removed. The tree growing process is now repeated for the two new nodes, while only considering $p - 1$ features. This continues for each new node until they are all either pure (i.e., only one class label

is represented in that node) or when some stop criterion is meet [20]. To find good splitting rules, methods such as "*best-split*" is usually used. It searches for the splitting rule that produces the least amount of misclassifications in each recursive step in the process. The most common choices used to quantify "*best-split*", are *gini-index*, *cross-entropy* and *missclassification error* [1, pp. 308–310].

By growing a tree very large, we are likely to overfit the training data. Methods such as *cost-complexity pruning* is one example of how to counteract the overfitting of our training data, by pruning the tree back down [1, pp. 307–308]. However the algorithm used to grow decision trees in this thesis do not have any pruning mechanism built in, so instead stop criteria such as:

- *max depth*: maximum depth of the tree before stopping the recursive growth,
- *min sample split*: minimum number of observations in a node required for splitting that node,

is two tuning parameters used to control the growth of the decision trees during the training phase. This algorithm is based on the original CART-algorithm [20] proposed by Leo Breiman and part of the scikit-learn [21] libraries for machine learning.

Table 5: The recursive steps of growing a single decision tree. Based on the original CART-algorithm.

CART-algorithm

1. **For each** feature in $\boldsymbol{X}$ find the best splitting rule.
2. Choose the feature with corresponding splitting rule which minimizes the error and remove that feature from the set $\boldsymbol{X}$ {according to the chosen splitting criterion}.
   (a) **For each** of the new daughter nodes (regions) **repeat** 1 & 2.
3. **Stop** when some constraint is meet or when all nodes are pure.

3.2 AdaBoost

The core principle of boosting is to turn a ”weak” classifier that performs just slightly better than random guessing, into a strong classifier by combining many of them together. Freund and Schapire [22] showed that their AdaBoost-algorithm, could do just that for any so called PAC (probably approximate correct) algorithm. The AdaBoost-algorithm repeatedly calls the weak base classifier many times, the CART-algorithm (DT classifier) in this thesis. The CART-algorithm grows a DT on the feature space $\boldsymbol{X}$ (as described in Chapter 3.1) and predicts the outcome, for which the initial DT classifier will get many cases wrong. The feature space $\boldsymbol{X}$ is then reweighted, to ”force” the next call to the CART-algorithm to focus more on the observations in $\boldsymbol{X}$ its predecessor got wrong. When a number of consecutive DT classifier (voting member) have been trained, each voting member gets a weighted vote. Their respective votes are then summarized and used as the final prediction.

However the original AdaBoost-algorithm do suffer in accuracy when considering a multi-class problem, i.e. when $K > 2$, such as in this thesis. This algorithm was originally designed to solve the multi-class problem, by subdividing them into multiple two class problems. The SAMME-algorithm [23] addresses this problem directly and extended on the original AdaBoost-algorithm to solve the multi-class problem, without having to subdivide them into multiple two class problems.

Now let $T(\boldsymbol{X})$ denote the DT classifier that assigns (predict) a class label q to each observation in $\boldsymbol{X}$ and let $Q(\boldsymbol{X})$ denote the final prediction from the SAMME-algorithm, then the algorithmic operations for training the AdaBoost classifier can be found in Table 6. Further note that the function $\mathbb{1}(\cdot)$ in this algorithm is an indicator function, i.e. if the argument inside is true then the function assigns the value 1 to the i^{th} observations and otherwise 0. Finally let k denote the i^{th} observations true class label. To give an example of how the indicator function is used to compute the error of the m^{th} DT classifier in step 2(b), we multiply the value of the indicator function with the corresponding weight w_i. That is, we are effectively summarizing the weights for all observations that the m^{th} classifier got wrong and dividing it by the total weight sum.

Table 6: The SAMME-algorithm and the algorithmic steps of calling a base classifier and then manipulating data points to force it to address the harder cases in a recursive manner.

SAMME-algorithm

1. **Initialize** observation weights $w_i = \frac{1}{n}$, $\{i\}_{i=1}^{n}$
2. **For** $m = 1$ to M:
   (a) Call the weak classifier $T^{(m)}(\boldsymbol{X})$ and fit the training data using weights w_i.
   (b) Compute $err^{(m)} = \frac{\sum_{i=1}^{n} w_i \mathbb{1}(k_i \neq q_i)}{\sum_{i=1}^{n} w_i}$.
   (c) Compute $\alpha^{(m)} = log\left(\frac{1-err^{(m)}}{err^{(m)}}\right) + log(K-1)$.
   (d) Set $w_i \leftarrow w_i * exp\left(\alpha^{(m)} * \mathbb{1}\left(k_i \neq q_i\right)\right), i = 1, 2, \ldots, n$.
   (e) Re-normalize w_i.
3. **Output** $Q(\boldsymbol{X}) = \arg\max_{q} \sum_{m=1}^{M} \alpha^{(m)} * \mathbb{1}\left(T^{(m)}(\boldsymbol{X}) = q\right)$.

The SAMME-algorithm follows the same structure as the original AdaBoost-algorithm with one major difference, their added constant $log(K-1)$ in step $2(c)$ in Table 6, for computing the α coefficient used to update observation weights. The α coefficient is also used in step 3, as a measurement of how "trustworthy" the m^{th} voting member's vote is and weighted accordingly. In the case of binary labels, i.e. when $K = 2$, it is easy to see that the SAMME-algorithm is then reduced back to the original AdaBoost-algorithm.

3.3 Random Forest

Random forest (RF) [24] is another ensemble method, that binds many weak classifiers together. The RF ensemble method grows a number of DT classifiers (i.e. the same CART-algorithm as for AdaBoost), thus growing a forest, then we average over all trees in that forest to improve the overall prediction. The core idea of RF is to improve the variance reduction of bagging (bootstrap aggregating), by reducing the correlation be-

tween trees without increasing the variance too much [1, pp. 588]. This can be achieved by only considering a random subsets of $v < p$ features from the feature space X [1, pp. 588–589]. That is, for each node when growing a tree, only a subset of v features is considered when searching for a splitting rule. During the process of growing a forest, besides only considering a random subset of the feature space when searching for splitting rules, we can also grow each tree on a different bootstrap sample $\boldsymbol{X}^*$ with replacement of size n. This method is known as growing a forest on random patches [25], where the algorithmic operations to grow a forest follows the process in Table 7.

Table 7: The process of growing a forest on random patches for classification[1, pp. 588].

RF-algorithm

1. **For** $m = 1$ to M:
   (a) Draw a bootstrap sample $\boldsymbol{X}^*$ of size n from the training data $\boldsymbol{X}$.
   (b) Grow a tree $T^{(m)}(\boldsymbol{X}^*)$ on the bootstrap sample by recursively repeating the following steps.
      i. Select $v = \sqrt{p}$ predictor variables from the p features in X.
      ii. Pick the *best split* among the features in the subset v.
      iii. Split the node into two daughter nodes.
2. **Output** the ensemble of trees $\{T^{(m)}(\boldsymbol{X}^*)\}_{i=1}^{M}$
3. Let $C^{(m)}(\boldsymbol{X})$ be the assigned (predicted) class label from the m^{th} tree, then let $C^{RF}(\boldsymbol{X})$ be the final prediction for each observation i in $\boldsymbol{X}$, by setting $C^{RF}(\boldsymbol{X}) = majority\, vote\{C^{(m)}(\boldsymbol{X})\}_{m=1}^{M}$.

3.4 Neural Network

All neural network (NN) modeling and training was done under the Keras [26] framework, with tensorflow [27] backend. Keras is a high-level neural network API (Application programming interface) written in Python. The basic structure for a NN, can be represented by the equations:

$$\mathbf{a}^{(l)}_{(n_l,1)} = \sigma_l(\mathbf{W}^{(l)}_{(n_l,p)}\mathbf{x}_{(p,1)} + \mathbf{b}^{(l)}_{(n_l,1)})$$

where $\mathbf{x}$ denotes the input vector containing p features, $\mathbf{W}$ denotes the weight matrix for layer l of the size n_l neurons and p features. Vector $\mathbf{b}$ is the so called bias vector, containing one weight for each neuron in layer l. Then some user defined activation function $\sigma_l(.)$ (where index l in $\sigma_l(.)$ is to denote that the activation function is layer specific) is applied yielding the hidden layers activation $\mathbf{a}$. This vector is then treated as the input vector for the next consecutive layer in the network:

$$\mathbf{o}^{(L)}_{(K,1)} = \sigma_l(\mathbf{W}^{(l+1)}_{(K,n_l)}\mathbf{a}^{(l)}_{(n_l,1)} + \mathbf{b}^{(l+1)}_{(K,1)})$$

where $\mathbf{o}$ is the output vector (often represented in form of softmax probabilities for multi-class classification problems, such as in this thesis), L denotes the last layer in the network and K is the total number of distinct class labels. This structure of connected layers is then used to feed data back and forth between the input vector $\boldsymbol{x}$ and output vector $\boldsymbol{o}$, to adjust the weights in the weight matrices $\mathbf{W}$ during the training process to successively improve the predictions. All network weights were trained following the same procedure found in Table 8.

Table 8: Algorithmic operation used to train a NN in this thesis.

NN weight training algorithm

1. **Initialize** all weight matrices through simulation from some distribution.
2. **For** $epoch = 1$ to $Epoch$:
   (a) Split training set into mini-batches.
   (b) **For each** mini-batch:
      i. Feed batch forward through the network.
      ii. Calculate training error.
      iii. Backpropagate the error and update weights according to some gradient descent optimization algorithm.
   (c) **End of each** $epoch$ calculate error on validation set.
      i. **If** validation accuracy has not improved in some user defined number of $epochs$ stop the process and roll back weights corresponding to best validation accuracy.
3. **Output** prediction accuracy on test set from the saved network.

If the reader is interested in a concise overview of what constitute mini-batch training and examples of gradient descent optimization algorithms, I suggest starting at [28].

4 Evaluation

In this chapter choices made during the training phase for tuning parameters are presented, for each classification algorithm. The evaluation is done by comparing combinations of filtering techniques and classification algorithms used in the consecutive steps in the system, as illustrated in Figure 5. The final comparison and evaluation of the system's activity recognition accuracy, is done by cross referencing error matrices and F-scores depending on the methods used, so finally a clarification is given on how they are structured and should be interpreted.

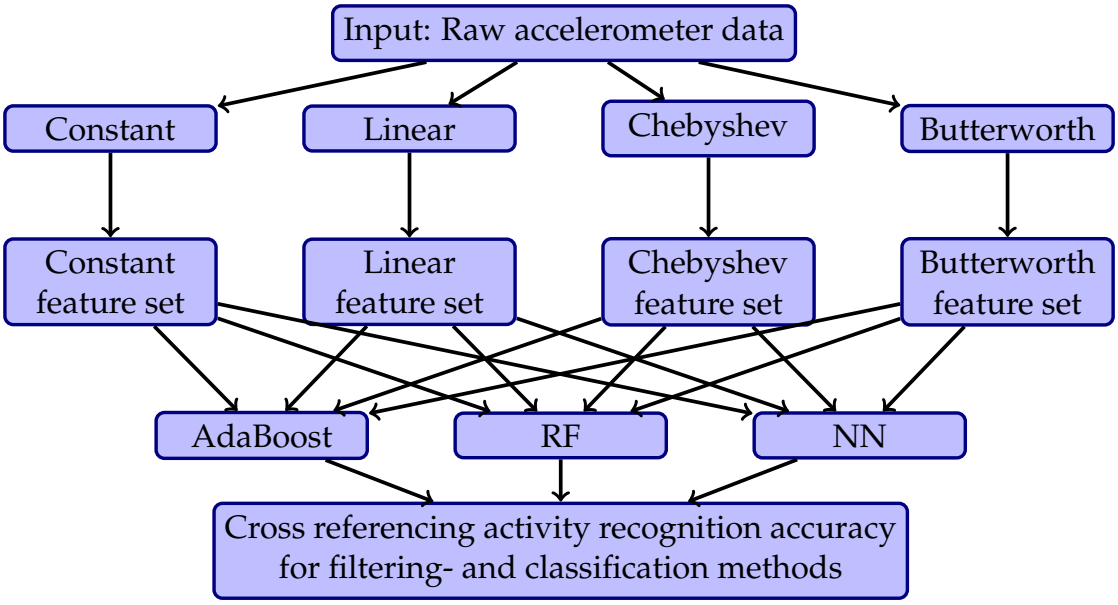

Figure 5: The process of cross referencing the different filtering- and classification methods.

4.1 Tuning

All classification algorithms were trained and tuned using the training and validation sets, with a sample size of 4623 and 1541 respectively. The settings used during the training phase of the classification algorithms can be found in Table 9, with their corresponding grids and range for the tuning parameters. The last value inside the $\{\cdot\}$-

bracket, corresponds to the final settings used to predict the observations in the test set. That is, these values notes the optimal settings for the tuning parameters, in terms of validation accuracy during the training phase.

Table 9: The set up and range for tuning parameters during the training process.

AdaBoosted DT		RF	
Set up			
Split criterion (DT):	*gini-index*	Split criterion (DT):	*gini-index*
Number of features considered when splitting the feature space into nodes using CART			
Features:	p	Features:	$v = \sqrt{p}$
Tuning parameters			
Max-depth:	$[2, 3, \ldots, 16]$, $\{6\}$	*Max-depth*:	$[2, 3, \ldots, 16]$, $\{9\}$
Min-sample-split:	$[2, 3, 4, 5]$, $\{2\}$	*Min-sample-split:*	$[2, 3, 4, 5]$, $\{2\}$
Number of classifiers	$[80, 84, \ldots, 800]$, $\{240\}$	*Number of classifiers*	$[80, 84, \ldots, 800]$, $\{296\}$
NN Set up			
Initial weights:			$\boldsymbol{W} \sim \mathcal{N}(0, 0.01)$
Initial bias:			$\boldsymbol{b} = 0$
Optimization algorithm:			*Adam*
Early stopping (step 2(c)i in Table 8):			20
Tuning parameters			
Dropout [29]:			$[0.25, 0.3, \ldots, 0.5]$, $\{0.35\}$
$l_2 - regularization$:			$[0, 0.00025, 0.0005]$, $\{0.00025\}$

4.2 Error matrices

In a multi-class classification problem, it is usually not only of interest to see the overall accuracy, but also the association between true labels and their classified labels. Error matrices (or commonly referred to as confusion matrices) will be used to present F-scores, precision, recall and the association between the predicted and true label. Error matrices presented in the result chapter, have the following structure:

- Each row is the sum of all observation in class k.
- Each column is the sum of all observation predicted to be in class k.
- The diagonal element are observations that have been correctly classified, denoted as true positive (TP).

- Elements in each row not on the diagonal are predictions denoted as false negative (FN) for class k.
- Elements in each column not on the diagonal are predictions denoted as false positive (FP) for class k.
- Last value in each row represents, $recall = \frac{TP}{(TP+FN)}$ for label k. That is among the observations in the actual true set of k, what proportion have been correctly identified.
- Last value in each column represents, $precision = \frac{TP}{(TP+FP)}$ for label k. That is among the observations predicted to be in class k, what proportion of these have been correctly classified.
- Last value on the diagonal represents, weighted $F-score = 2 * \frac{precision_{mean} * recall_{mean}}{precision_{mean} + recall_{mean}}$, where $precision_{mean} = \frac{1}{K} \sum_{i=1}^{K} preision_i$, $recall_{mean} = \frac{1}{K} \sum_{i=1}^{K} recall_i$ and $F-score \in [0,1]$.

5 Results

In this chapter results is presented in form of the error matrices with an overall F-score and individual class accuracy, in form of recall and precision. The tables are structured by classification method, where each table shows results for the four different filtering methods.

5.1 Error matrices

Error matrices in Tables 10-12, shows that AdaBoosted DT gives the best overall accuracy in terms of F-score, for all filtering methods. Likewise, when comparing filtering methods, Chebyshev filtering gives the best accuracy, for all classification methods.

The two filtering types constant- and linear detrending have similar F-scores, but where linear detrending have a slightly better classification accuracy than constant detrending, for all classification methods. Regardless of classification algorithm, the system have difficulty distinguishing between the motionless activities, when the accelerometer signals have been filtered with these two methods. For Butterworth- and Chebyshev filtering, the same classification algorithms performs much better, primarily due to the increase in accuracy for the low frequency activities. The majority of

missclassifications made for the motionless activities is now removed, when the accelerometer signals have been filtered with Butterworth and Chebyshev instead.

Table 10: Confusion matrices for the different filtering methods for AdaBoosted DT.

AdaBoosted Decision Trees

Constant filter

True label	Prediction					
	Cycling	Running	Sitting	Standing	Walking	Recall
Cycling	510	0	4	2	9	0.9714
Running	0	635	0	0	1	0.9984
Sitting	0	0	187	83	0	0.6880
Standing	3	0	27	428	1	0.9325
Walking	1	0	1	0	752	0.9974
Precision	0.9922	1	0.8512	0.8343	0.9856	**0.9250**

Linear filter

True label	Prediction					
	Cycling	Running	Sitting	Standing	Walking	Recall
Cycling	512	0	3	3	7	0.9752
Running	0	635	0	0	1	0.9985
Sitting	0	0	179	91	0	0.6630
Standing	3	0	19	435	2	0.9477
Walking	0	0	2	0	752	0.9974
Precision	0.9942	1	0.8818	0.8223	0.9869	**0.9265**

Butterworth filter

True label	Prediction					
	Cycling	Running	Sitting	Standing	Walking	Recall
Cycling	512	0	1	0	12	0.9752
Running	2	631	0	0	3	0.9921
Sitting	2	0	263	5	0	0.9813
Standing	5	0	0	453	1	0.9869
Walking	2	2	0	0	750	0.9946
Precision	0.9827	0.9965	0.9962	0.9891	0.9791	**0.9874**

Chebyshev filter

True label	Prediction					
	Cycling	Running	Sitting	Standing	Walking	Recall
Cycling	513	0	1	1	10	0.9771
Running	0	634	0	0	2	0.9969
Sitting	2	0	263	6	0	0.9741
Standing	4	0	1	453	1	0.9869
Walking	1	2	0	0	751	0.9960
Precision	0.9865	0.9969	0.9925	0.9869	0.9830	**0.9877**

Table 11: Confusion matrices for the different filtering methods for RF ensemble method.

Random Forest

Constant filter

True label	Prediction					
	Cycling	Running	Sitting	Standing	Walking	Recall
Cycling	512	0	2	3	8	0.9754
Running	0	636	0	0	0	1
Sitting	0	0	180	89	0	0.6692
Standing	6	0	21	429	2	0.9367
Walking	3	2	1	1	747	0.9907
Precision	0.9828	0.9969	0.8824	0.8218	0.9868	**0.9242**

Linear filter

True label	Prediction					
	Cycling	Running	Sitting	Standing	Walking	Recall
Cycling	515	0	2	1	7	0.9754
Running	0	636	0	0	0	1
Sitting	1	0	177	92	0	0.6556
Standing	5	0	19	433	2	0.9434
Walking	2	2	2	1	747	0.9907
Precision	0.9828	0.9969	0.8895	0.8216	0.9881	**0.9243**

Butterworth filter

True label	Prediction					
	Cycling	Running	Sitting	Standing	Walking	Recall
Cycling	505	0	1	1	18	0.9619
Running	0	629	0	0	7	0.9889
Sitting	4	0	259	7	0	0.9593
Standing	8	0	1	450	0	0.9804
Walking	17	5	0	0	732	0.9708
Precision	0.9457	0.9921	0.9423	0.9825	0.9669	**0.9691**

Chebyshev filter

True label	Prediction					
	Cycling	Running	Sitting	Standing	Walking	Recall
Cycling	508	0	1	1	15	0.9676
Running	0	634	0	0	2	0.9969
Sitting	4	0	260	6	0	0.9629
Standing	7	0	1	448	3	0.9760
Walking	8	5	0	0	741	0.9828
Precision	0.9639	0.9922	0.9924	0.9846	0.9737	**0.9793**

Table 12: Confusion matrices for the different filtering methods for NN.

Neural Network

Constant filter

True label	Prediction					
	Cycling	Running	Sitting	Standing	Walking	Recall
Cycling	489	0	3	22	11	0.9314
Running	0	636	0	0	0	1
Sitting	1	0	51	218	0	0.1889
Standing	1	0	31	424	3	0.9237
Walking	0	0	4	11	739	0.9801
Precision	0.9960	1	0.5730	0.6282	0.9841	**0.8202**

Linear filter

True label	Prediction					
	Cycling	Running	Sitting	Standing	Walking	Recall
Cycling	488	0	10	17	10	0.9296
Running	1	629	0	0	6	0.9889
Sitting	0	0	98	172	0	0.3630
Standing	1	0	53	402	3	0.8758
Walking	0	0	12	5	737	0.9775
Precision	0.9959	1	0.5665	0.6745	0.9749	**0.8345**

Butterworth filter

True label	Prediction					
	Cycling	Running	Sitting	Standing	Walking	Recall
Cycling	484	1	8	6	26	0.9214
Running	0	635	0	0	1	0.9984
Sitting	3	0	263	4	0	0.9741
Standing	7	0	22	429	1	0.9346
Walking	133	8	0	5	608	0.8064
Precision	0.7719	0.9860	0.8976	0.9662	0.9560	**0.9212**

Chebyshev filter

True label	Prediction					
	Cycling	Running	Sitting	Standing	Walking	Recall
Cycling	494	1	4	12	14	0.9401
Running	0	636	0	0	0	1
Sitting	1	0	257	12	0	0.9519
Standing	3	0	8	447	1	0.9739
Walking	30	3	1	8	712	0.9443
Precision	0.9356	0.9938	0.9519	0.9332	0.9794	**0.9604**

6 Discussion

From this thesis, we have shown that it is sufficient to construct and train one HAR system, that can be embedded in multiple platforms, while also considering it being used by multiple subjects. That is, at least when considering this set of physical activities. The best accuracy was given when utilizing Chebyshev filtering (in the second step of the HAR system) and AdaBoosted DT (the fourth step in the HAR system) with a F-score of 0.9877. This result can be considered sufficiently good for most general purposes of any HAR system, such as calorie burning applications, tracking activities over time, to summarize and spotting changes in an individual's activity levels.

Table 13: Results compared with other similar studies, for HAR systems when utilizing a sliding window approach and wearable sensors.

Summary	From [30]	From [31]	From [32]	This study
Labels	Sitting,Standing, Ascending stairs, Descending stairs, Walking,Running	Stationary,Walking, Ascending stairs, Descending stairs, Running, Driving	Dancing,Running, Ascending stairs, Descending stairs, Slow-walk,Fast-walk	Cycling,Running, Sitting, Standing Walking
Sensors	eWatch sensor accelerometer and light sensors embedded	Accelerometer and magnitude sensors embedded in two Nokia N97	Accelerometer sensors embedded in an Android device	Accelerometer sensors embedded, Android, Iphone Actigraph
Window Size	4s window buffer classifying 0.5s increments	1-6s window	1.28s window	2.5s window
Subjects	6	7	4	14
Classification Method	DT Naive-Bayes	SVM with and without magnitude field	NN RF SVM	NN RF AdaBoosted DT
Classification Accuracy	Accelerometer only DT Pocket: 85.2% Belt: 87.0%	Accelerometer only SVM window size: 5s Pocket: 91.5%	Accelerometer only in pocket NN: 89.72% RF: 85.15%	Accelerometer only pocket or belt AdaBoosted DT: 98.87% RF: 97.93% NN: 96.04%

Some key factors with corresponding classification accuracy from similar studies are summarized in Table 13. The activity accuracy for this HAR system is surprising good, when considering that the system handles both individual movement patterns from 14 different subjects and platform differences. In [31], a HAR system was built with a generic support vector machine classifier (SVM), where data was collected with two Nokia N97 from six different pocket placements, two of which were coat pockets. This resulted in an classification accuracy of 0.915, when only using accelerometer data and the position of the device was unknown. To give another example of the impact

between classification accuracy and device placement comes from [8], where they evaluated 5 different positions:

$$positions = \{upper\,arm, hand, jack\,pocket, trouser\,pocket, waist\},$$

where data was collected with a MEXZU MX3 (Android device). Their system was evaluated on five physical activities, ascending- and descending stairs, walking, jogging and jumping. In their study, they used both gyroscope- and accelerometer data. In the case where their only utilized the accelerometer data, their best result for the RF classifier was given when the device was placed in the trouser pockets with an accuracy of 71.3%, while for example when being attached to the arm going down to an accuracy of 60.4%. This give some indication that by extending the number of positions considered for this HAR system, the activity recognition accuracy will likely drop. For the same classification method when utilizing both accelerometer data and gyroscope data, the activity recognition accuracy was increased to 80.9%, for the trouser pocket placement and increased to 80.4% for the arm placement. This also gives some indication that incorporating gyroscope sensor data could potentially help to counteract that drop in activity recognition accuracy when extending to more positions. In my opinion a HAR system utilizing smartphone sensors, should besides handling individual movement patterns and platform differences, should also be capable of handling multiple positions and orientations. The user should not constantly have to remember to put the smartphone in a certain rotation and placement for the system to work. Therefore further analysis and investigation is needed for this system, by extending the number of positions and orientations. More data needs to be collected when including other common positions, such as in the hand, in a bag and jack-pocket. More activity data is also need when the devices have other orientations and screen directions.

7 Future Work

In the systems current state the hand-crafted features were only constructed and used to evaluate the systems accuracy, while their individual importance to distinguish between the physical activities were not. So in future work, analysis of features importance and a feature selection will be performed. As mentioned in the discussion, it is also intended to extend the data collection to include less restrictions in both orientation and positioning.

8 Conclusion

The two ensemble methods AdaBoosted DT and RF outperforms NN, i.e. for a HAR system utilizing a sliding window and the four step process described in Figure 1. RF and Adaboosted DT have similar performance, but where Adaboosted DT have a slightly better F-score over RF for all filtering types.

Both Chebyshev- and Butterworth filtering have higher activity recognition accuracy than the constant- and linear filtering. It can be seen from the error matrices in Tables 10-12, that their primary advantage is for low frequency activities. Both these two filtering methods preserve more information to distinguish between the motionless activities, standing and sitting. As in the case with Adaboosted DT, which gave slightly better F-score than RF, Chebyshev filtering gives a slightly better classification accuracy over Butterworth filtering.

Finally, in these tables, further similarities can be found in all three of them. In each table there is a trade-off between a large increase in accuracy for the low frequency activities and a slight decrease in accuracy for the high frequency activities, cycling, walking and running, when switching from constant- and linear filtering to Chebyshev and Butterworth. Both in terms of activity specific precision and recall, linear filtering have comparable or better accuracy, for the three high frequency activities, when comparing with Chebyshev. This leads to the final recommendation, that if there are more than one low frequency activities to distinguish between, Chebyshev filtering is the best option. However, if the system only have high frequency activities, or all low frequency activities can be classified as one stationary activity, then linear filtering may be the better option.

References

[1] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, 2nd ed. New York, NY, USA: Springer series in statistics, 2009.

[2] Geneva: World Health Organization. (2010) Global Recommendations on Physical Activity for Health. [Accessed: May 01, 2018]. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK305051/

[3] Geneva: World Health Organization . (2015) World report on ageing and health. [Accessed: May 01, 2018]. [Online]. Available: http://www.who.int/ageing/events/world-report-2015-launch/en/

[4] A. Jalal, S. Kamal, and D. Kim, "A depth video-based human detection and activity recognition using multi-features and embedded hidden markov models for health care monitoring systems," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, no. 4, pp. 54–62, Jun. 2017.

[5] R. Alazrai, M. Momani, and M. I. Daoud, "Fall detection for elderly from partially observed depth-map video sequences based on view-invariant human activity representation," *Applied Sciences*, vol. 7, no. 4, Art No. 316, Mar. 2017.

[6] A. Fleury, M. Vacher, F. Portet, P. Chahuara, and N. Noury, "A french corpus of audio and multimodal interactions in a health smart home," *Journal on Multimodal User Interfaces*, vol. 7, no. 1, pp. 93–109, Mar. 2013.

[7] A. Wang, G. Chen, C. Shang, M. Zhang, and L. Liu, "Human activity recognition in a smart home environment with stacked denoising autoencoders," in *Web-Age Information Management*, S. Song and Y. Tong, Eds. Cham: Springer International Publishing, Oct. 2016, pp. 29–40.

[8] Y. Chen and C. Shen, "Performance analysis of smartphone-sensor behavior for human activity recognition," *IEEE Access*, vol. 5, pp. 3095–3110, Mar. 2017.

[9] I. Orha and S. Oniga, "Study regarding the optimal sensors placement on the body for human activity recognition," in *2014 IEEE 20th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, Oct. 2014, pp. 203–206.

[10] O. Dobrucalı and B. Barshan, "Sensor-activity relevance in human activity recognition with wearable motion sensors and mutual information criterion," in *Information Sciences and Systems 2013*, E. Gelenbe and R. Lent, Eds. Cham: Springer International Publishing, Sep. 2013, pp. 285–294.

[11] L. Gao, A. Bourke, and J. Nelson, "Evaluation of accelerometer based multi-sensor versus single-sensor activity recognition systems," *Medical Engineering & Physics*, vol. 36, no. 6, pp. 779 – 785, Jun. 2014.

[12] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1192–1209, Oct. 2013.

[13] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen, "Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, Nov. 2015, pp. 127–140.

[14] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas, "Window size impact in human activity recognition," *Sensors*, vol. 14, no. 4, pp. 6474–6499, Apr. 2014.

[15] Documentation. Work with raw data. Android Developers. [Accessed: February 01, 2018]. [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_motion

[16] D. BASHIR S.A., DOOLAN and P. A., "The effect of window length on accuracy of smartphone-based activity recognition." *IAENG International journal of computer science*, vol. 43, no. 1, pp. 126–136, Apr. 2016.

[17] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, 2nd ed. San Diego, CA, USA: California Technical Publishing, 1999.

[18] M. O. Derawi, P. Bours, and K. Holien, "Improved cycle detection for accelerometer based gait authentication," in *2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Oct. 2010, pp. 312–317.

[19] A. Bayat, M. Pomplun, and D. A. Tran, "A study on human activity recognition using accelerometer data from smartphones," *Procedia Computer Science*, vol. 34, pp. 450 – 457, Jul. 2014.

[20] L. Breiman, *Classification and regression trees*, ser. The Wadsworth statistics/probability series. Boca Raton, Fla. : Chapman & Hall, 1984.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[23] J. Zhu, S. Rosset, H. Zou, and T. Hastie, "Multi-class adaboost," *Statistics And Its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[24] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[25] G. Louppe and P. Geurts, "Ensembles on random patches," in *Machine Learning and Knowledge Discovery in Databases*, P. A. Flach, T. De Bie, and N. Cristianini, Eds. Berlin, Heidelberg: Springer, Sep. 2012, pp. 346–361.

[26] F. Chollet *et al.* (2015) Keras. [Accessed: Jan 15, 2018]. [Online]. Available: https://keras.io

[27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[28] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, Jun. 2017, [Accessed: Feb 15, 2018]. [Online]. Available: http://arxiv.org/abs/1609.04747

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[30] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions," in *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, Apr. 2006, pp. 4 pp.–116.

[31] L. Sun, D. Zhang, B. Li, B. Guo, and S. Li, "Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations," in *Ubiquitous Intelligence and Computing*, Z. Yu, R. Liscano, G. Chen, D. Zhang, and X. Zhou, Eds. Berlin, Heidelberg: Springer, Oct. 2010, pp. 548–562.

[32] A. Bayat, M. Pomplun, and D. A. Tran, "A study on human activity recognition using accelerometer data from smartphones," *Procedia Computer Science*, vol. 34, pp. 450 – 457, Jun. 2014.