NTNU

Norwegian University of
Science and Technology

Human Activity Recognition With Two Body-Worn Accelerometer Sensors

Hans-Olav Hessen
Astrid Johnsen Tessem

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Helge Langseth, IDI
Co-supervisor: Kerstin Bach, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

Data used in studies about physical activity is primarily collected from questionnaires and other subjective methods, which may lead to biased and inaccurate data. As subjective data collection methods have shown to be unreliable, enhancing or even replacing these methods with objective methods like the use of wearable technology and human activity recognition (HAR) systems, will lead to more accurate data. HAR systems are systems that can recognize what kind of activity a subject is performing based on the monitoring of data streams from sensors on, or close to, the subject. One way this can be achieved is by constructing a classification system that can recognize human activities from body-worn sensor readings.

HUNT4 is an upcoming health study with about 60000 participants that will make use of objective measurements of their participants' physical activity to provide precise summaries about each participant's activity level. To be able to create these summaries, there is a need for a system that can recognize physical activities based on sensor readings. The main objective of our research is to design and construct such a HAR system.

In this thesis, we have reviewed related work performed in the field of HAR, and identified potentials for further improvements of current HAR systems. We experimented with deep learning, semi-supervised learning, dynamic classification and dynamic windowing. Through an iterative process of adding and removing components, we propose a system that is able to distinguish between daily activities with a high level of precision. The final HAR system consists of a Convolutional Neural Network followed by a Hidden Markov model, reaching an accuracy for classifying activities of 97.9% for adults and 96.6% for adolescents.

Sammendrag

Studier som omhandler fysisk aktivitet bruker data som i all hovedsak er samlet inn ved hjelp av spørreskjemaer eller andre subjektive innsamlingsmetoder. Da subjektive innsamlingsmetoder har vist seg å være upålitelige, er det viktig å erstatte disse metodene med mer objektive og nøyaktige metoder. En måte å oppnå dette på er å konstruere et klassifiseringssystem som er i stand til å gjenkjenne menneskelige aktiviteter basert på sensordata.

HUNT4 er en stor kommende helseundersøkelse med omtrent 60000 deltagere som vil bruke objektive metoder til å måle deltakerenes fysiske aktivitet for å tilby presise sammendrag om aktivitetsnivået til hver enkelt deltager. For å kunne lage disse sammendragene, er det behov for et system som kan gjenkjenne fysiske aktiviteter basert på sensoravlesninger. Hovedmålet for vår forskning er å designe og konstruere dette aktivitetsgjennkjenningssystemet.

Vi har satt oss inn i tidligere arbeid utført innen aktivitetsgjennkjenning, og på denne måten identifisert muligheter for ytterligere forbedringer av eksisterende aktivititetsgjennkjenningssystemer. Vi har eksperimentert med dyp læring, halvovervåket læring, dynamisk klassifisering og dynamiske datavindu. Gjennom en iterativ prosess av å fjerne og legge til komponenter til systemet, har vi kommet fram til et endelig system som er i stand til å skille mellom aktiviteter med høy presisjon. Dette systemet består av et konvolverende nevralt nettverk etterfulgt av en Hidden Markov modell, og har resultert i en nøyaktighet for klassifisering av aktiviteter på 97,9% for voksne og 96,6% for ungdom.

Preface

This thesis was written spring 2016 for the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU).

The subject for this thesis was defined in cooperation with our supervisors, Helge Langseth and Kerstin Bach, and Paul Jarle Mork from the Department of Public Health and General Practice at NTNU.

Several persons have contributed with inspiration and ideas for this project. We would first of all like to express our graditude to our supervisors Helge Langseth and Kerstin Bach for valuable input, assistance and feedback throughout the project. Furthermore, we would like to thank Alan Bourke, Paul Jarle Mork, Espen Alexander F. Ihlen, Hilde Bårdstu, Astrid Ustad and Atle Melleby Kongsvold for the collaboration we have had during this spring.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Background

Epidemiological studies indicate that there is a strong prospective association between the level of physical inactivity and risk of several non-communicable diseases such as cardiovascular disease, type 2 diabetes, obesity, cancer and musculoskeletal disorders [Lee et al., 2012, US Department of Health, 1996]. Moreover, physical inactivity has been identified as the 4th leading risk factor for global mortality, resulting in approximately 3.2 million deaths per year[1]. More than 23% of the world's adult population do not meet the recommended amount of physical activity [Alwan, 2014]. The all-cause mortality is 20-30% higher among those who do not meet the recommended amount of physical activity than for those who do [Alwan, 2014].

A limitation of the abovementioned studies is that physical activity data is primarily collected by questionnaires and other subjective methods which may lead to biased and inaccurate data [Kwak et al., 2011]. As subjective data collection methods have shown to be unreliable, enhancing or even replacing these methods with objective methods like Human Activity Recognition (HAR) systems, will lead to more valid data. Further, this will enable development of more accurate models for the association between physical activity and risk of disease. More precisely, HAR systems are systems that can recognize what kind of activity a subject is performing based on the monitoring of data streams from sensors on, or close to, the subject. One way this can be achieved is to generate a classification system that can categorize human activities from body-worn sensor readings.

When monitoring a subject's overall health, it is important to get to know, not only the overall amount of physical activity the subject is performing, but also what kind of activities the subject is carrying out (e.g. how long is a subject sitting, standing and walking on a daily basis) [Patel AV1, 2010]. Today's commercial HAR systems, aimed at supporting this monitoring, are limited to simple variables such as

[1] http://www.who.int/dietphysicalactivity/factsheet_inactivity/en/

counting steps, stairs and measuring distance. Current HAR systems are, therefore, not an ideal tool when collecting activity variables relevant to an overall health status. Moreover, sedentary time has been identified as an independent risk factor also among persons performing recommended amount of physical activity [Gupta et al., 2016]. This underline the importance of developing objective methods that differentiate between sedentary activities (sitting, lying down) and moderate and vigorous activities (walking, running, cycling).

The Nord-Trøndelag Health Study (HUNT study)[2] is one of the largest health studies ever performed. The study comprise a large range of health-related data (e.g. questionaries data, blood samples, clinical test data) collected during three intensive sub-studies since 1984. The HUNT4 study is the fourth sub-study in HUNT. It will start in fall 2017 and offer objective measurements of physical activity to their participants. A HAR system will be developed to classify sensor data from the participants. Two wearable accelerometer sensors placed at each participant's thigh and back, will be used to obtain sensor data. The sensors will be worn for a week before they get returned, and an analysis will be performed to measure the participant's activity level.

The main objective of this study is to design and construct a HAR system that can be used in the HUNT4 study. It is important that this system fits with the sensors that are going to be used in HUNT4, and that the system can classify daily activities with a high level of precision. The data set used to construct this HAR system was created during fall 2015 in collaboration with Hilde Bårdstu and Atle Melleby Kongsvold [Tessem and Hessen, 2015].

1.2 Goals and Research Questions

As mentioned, the main objective of this research project is to design and construct a HAR system of sufficient quality for the upcoming HUNT4 study. To reach this objective, we need to accomplish the goals presented below.

To be able to construct a HAR system of sufficient quality, we need to obtain an overview of previous work performed in the field of human activity recognition and get an understanding of the techniques underlying such systems.

Goal 1: To get in-depth knowledge of the state-of-the-art in the field of human activity recognition.

Research Question 1: What methods and which architecture does state-of-the-art HAR systems use?

[2] http://www.ntnu.edu/hunt

Based on the knowledge obtained from RQ1, we will identify potentials for improvements and suggest and evaluate methods that can improve state-of-the-art HAR systems.

Goal 2: To improve state-of-the-art HAR systems.

Research Question 2: How will different methods, or combinations of methods, affect the overall performance of a HAR system?

To evaluate our second goal we will compare our proposed HAR system with a state-of-the-art HAR system called Acti4 [Skotte et al., 2014]. Acti4 is used in several Scandinavian human activity studies [Danquah et al., 2016, Hallman et al., 2015, Munch Nielsen et al., 2016].

1.3 Research Method

March and Smith [1995] have stated that both design science [Simon, 1996] and natural science are needed to ensure that IT research is both relevant and effective. They propose a framework for IT research consisting of four steps which are *Build*, *Evaluate*, *Theorize* and *Justify*. Building an artifact and evaluating this artifact are often viewed as the two basic steps from design science, while generating a theory and justifying this theory are often viewed as the two basic steps from natural science. In this study, we aim at improving state-of-the-art HAR systems by using the framework proposed by March and Smith. We will run the four steps in an iterative process, where we in each iteration will in some manner modify a component in the HAR system, perhaps adding or removing functionality. The modification will be based on existing knowledge and experience gained from the previous iterations. The final result of the iterative process will be a functional HAR system, as well as knowledge about the tested approaches and solutions in such a system. Thus, the iterations in our project can be seen as:

1. **Building:** Building a HAR system based on our existing theories and knowledge.
2. **Evaluating:** Evaluating this HAR system. Does the system work, and if it did, how well did it work? Did we make any progress?
3. **Theorizing:** Contributing with theoretical knowledge as to why the tried solutions in new HAR system works or fails.
4. **Justifying:** Verifying to some extent the different explanations from the *theorizing* step. This step contributes to a more certain scientific understanding about why our HAR system performed as it did.

1.4 Thesis Structure

The remainder of this thesis is structured as follows.

Chapter 2: Background Theory and Motivation provides our motivation for this research as well as an overview of current research in the field of Human Activity Recognition.
Chapter 3: Beyond State-of-the-art presents the methods we experiment with to improve HAR systems.
Chapter 4: Collection of the Data Set presents how our data set was collected.
Chapter 5: System Design presents the architectural structure of our proposed HAR system.
Chapter 6: Experiments presents the experiments performed and the final evaluation of each part of the proposed system. It also presents a precise explanation of the final system architecture.
Chapter 7: Results and Discussion presents and discusses the results provided by our final HAR system.
Chapter 8: Conclusion and Future Work evaluates and concludes the presented work, and presents a summary of further hypothetical research in human activity recognition based on the work presented in previous chapters.

Chapter 2
Background Theory and Motivation

This chapter presents previous research in the field of HAR. Section 2.1 presents an overview of the common steps in HAR system development, where each step is described in detail from Section 2.2 through 2.6. These sections are based on a project we had during fall 2015 [Tessem and Hessen, 2015], and address the first goal of our research by providing a deeper overview of state-of-the-art HAR systems. To get a deeper understanding of how these steps work together, Section 2.7 presents Acti4. Acti4 is a state-of-the-art HAR system that we compare against our proposed HAR system in Chapter 7. The last section presents our motivation for the work undertaken to achieve the remaining research goals.

2.1 Overview of Human Activity Recognition Systems

A typical process of creating a HAR system is divided into the following steps:

1. **Data Collection:** Activity data is obtained from sensors worn by subjects.
2. **Data Pre-processing:** If several sensors are used, the data from each sensor need to be synchronized. The data is then labeled so that each data point has a corresponding activity. This step can also include resampling, noise removal and balancing of the data set.
3. **Data Segmentation:** Instead of evaluating individual data points, the data stream is divided into smaller segments (windows), where each window has a corresponding activity.
4. **Feature Generation and Selection:** Characteristics from the data, called features, are extracted from each window.
5. **Classification:** Features are used to train a classification algorithm, enabling it to distinguish between different activities.

This process is what Bulling et al. [2014] referred to as the *Activity Recognition Chain* and is presented in Figure 2.1. The following sections provide a deeper understanding of each step in this pipeline.

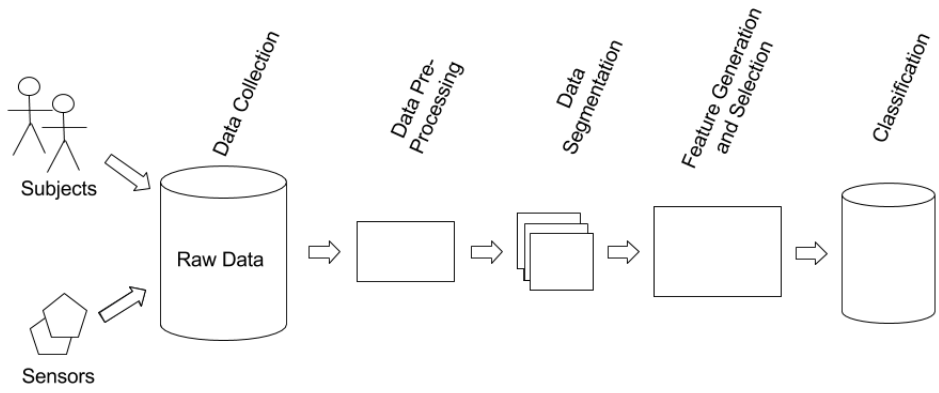


Figure 2.1: The process of creating a HAR system. Called the *Activity Recognition Chain* by Bulling et al. [2014]

2.2 Data Collection

This section presents the process of collecting human activity data using body-worn sensors.

2.2.1 Sensors

A wide range of sensor types have been used in earlier HAR systems, including accelerometers [Ravi et al., 2005], gyroscopes [Leutheuser et al., 2013] and electrocardiograms [Li et al., 2010]. To improve performance, it is not uncommon to combine multiple sensor types [Leutheuser et al., 2013]. Accelerometer sensors usually lead to accurate classification results as well as being small, cheap, and require a small amount of processing power. Accelerometers measure the acceleration of a moving or vibrating body and are important components when analyzing human movement. Gyroscopes measure the rate of rotation around a particular axis and are therefore able to calculate orientation (from a reference orientation). By knowing the orientation of a body, it is possible to distinguish between a number of activities, especially when several gyroscopes are used together. Electrocardiograms, which monitors the electrical activity of the heart, have been used in HAR systems to identify movement intensity from heart rate measurements.

The number of sensors and their ideal placement are debated. Atallah et al. [2011] states that classification results improve when more accelerometers are worn. On the other hand Cleland et al. [2013] did not find any significant improvement when combining more than two sensors. However, both studies agree that using many sensors can feel intrusive and unnatural. Minimizing the number of sensors while maintaining the classification rate is, therefore, important.

Karantonis et al. [2006] states that essentially all measured body movement are contained within a frequency component below 20 Hz. To enable the sensors to detect frequencies below 20 Hz, Nyquist–Shannon Sampling Theorem [Lüke, 1999] states that it is necessary with a sampling rate of minimum 40 Hz (2 times 20 Hz).

In the data set used in this study, it was chosen to use two accelerometer sensors, one at the back and one at the thigh. According to Cleland et al. [2013] two sensors are enough to obtain satisfying classification results while minimizing the constraints brought on the subject's physical behavior. The sensor's sampling rate was set to 100 Hz, which is more than enough to measure human activities.

2.2.2 Subjects

The number of subjects used when developing HAR systems differs between studies. A higher number of subjects leads to a more realistic representation of movement patterns in a population [Bartlett, 2007]. However, as the number of subjects increase, the data collecting and labeling process gets more demanding. It is, therefore, important to balance the trade-off between the number of subjects and the time spent collecting and labeling. In earlier research the number of subjects typically lies between 1 and 25.

Movement patterns differ between different age groups and genders [Bartlett, 2007]. To ensure representativity, it is important to choose subjects based on the application area of the HAR system. HUNT4 is divided into two studies. The first study is called the Adult HUNT4, and is focusing on participants older than 20 years. The second study is called the Young HUNT4, and is focusing on participants between 13 and 19 years. Since our research is a preparation for HUNT4, we will concentrate on the physical movement of adolescents and adults. The data set was therefore collected data from one group of 12 healthy adolescents (age 13-16 years) and one group of 23 healthy adults (age 28-52 years).

2.2.3 Methods to Collect Human Activity Data

In previous HAR studies, it has been common to collect sensor data by requesting the subjects to do a standardized sequence of activities. However, the number and type of activities vary between studies. The majority of data is collected in a recorded lab

environment with supervision by a researcher. Activities performed in an unnatural setting (e.g. a treadmill) can influence the subject's activity pattern [Bartlett, 2007]. Therefore, several studies try to collect activity data from more natural settings. One way this could be achieved is by conducting the experiments out of the lab, without supervision. Bao and Intille [2004] tried to provoke more natural movement by giving the subjects a goal, instead of requesting it to do an activity. For example, instead of requesting the subject to work with a computer, the subject could be asked to find the world's largest city by using a computer. In the data set we used, one part of the data was collected in a recorded lab environment, and the other part was collected in a more natural setting without supervision.

2.3 Data Pre-processing

Before further analysis, the collected data need to go through a pre-processing stage. This stage can include methods such as synchronization, resampling, noise removal and labeling. In this section, we will describe the process of synchronization and labeling, in addition to methods used to balance skewed data sets.

2.3.1 Synchronization

The use of multiple sensors during the data collection can lead to unsynchronized sensor measurements. Even though a sensor has been set to a given sampling rate, the actual sampling rate may deviate from this. Synchronization of the raw data is, therefore, an important step of the HAR chain. A common way to get a coordinated start and end point of the sensor signal is to shake them together in a fixed movement pattern, and mark peaks in the corresponding data made by this shake movement as start and end point [Leutheuser et al., 2013, Bao and Intille, 2004]. The start and end points of the data used in this project are marked with heel-drops (lift heel and then drop).

2.3.2 Labeling

After the data is collected and synchronized, the data needs to be annotated as different activities. The annotation process tends to be both time-consuming and demanding. To save time, the annotation could be performed during the acquisition process [Ravi et al., 2005, Bao and Intille, 2004], however, this may lead to a higher amount of mislabeling. Human activities are very complex, and the same activity can be performed in different ways, depending on the context. Clear and detailed activity definitions are therefore important to prevent subjective labeling. Low-level agreement regarding the start and end time of activities can be avoided by removing several seconds of the beginning and end of each activity [Ravi et al., 2005, Leutheuser et al., 2013]. However, this may lead to a loss of valuable information. The data set

used in this study was labeled by watching recorded videos from the data acquisition. These recordings provided the opportunity to observe the movements multiple times and resulted in a more precise labeling process.

2.3.3 Balancing Data Set

Imbalanced class distribution among the different classes can be a problem when training a classification algorithm. When monitoring human activities, some activities occur frequently, such as sitting, standing, walking and lying, while other activities occur less frequently, such as running, going stairs, bending. A challenge dealing with classes with few instances is that the classifier would not be able to learn this class as well as the classes with more instances.

One way to resolve the challenge of having an imbalanced data set, is to resample the data set so that every activity have the same amount of data instances. There are two main resampling methods:

- Oversampling: Add copies of instances from the minority classes, so that each class has the same amount of instances as the largest class (in our case *sitting*).
- Undersampling: Delete instances from the majority classes, so that each class has the same amount of instances as the smallest class (in our case *cycling(stand)*).

A weakness when balancing the data set is that the distribution of the classes would no longer be representative of the population.

2.4 Data Segmentation

Signal segmentation is an important step in the activity recognition chain. Instead of evaluating each data point, a common approach is to look at longer segments of data, called windows. An optimal segmentation of human activity data would be to generate a new segment for each consecutive activity performed. This would result in segments, referred to as windows, with varying size dependent on the length of the underlying activity. This segmenting approach is called dynamic windowing. However, generating dynamic windows can be a difficult task as consecutive activities often blur into each other rather than being clearly separated by pauses. In the literature, it is more common to segment the data stream with fixed sized windows. Both dynamic sized and fixed sized windowing will be described below.

2.4.1 Dynamic Window Size

A dynamic windowing approach aims to segment the data signal in such a way that a new window is generated whenever a new activity is performed. This is often performed by locating changes in the data signal as changes in the signal often indicates transitions between the underlying activities. There are several ways to implement a dynamic window approach. Some of these approaches are described below.

Event based signal segmentation has shown promising results in earlier research [Krishnan and Cook, 2014], but is limited to data sets with binary-valued (on and off) sensors (e.g. the door in the house is open vs. closed). A new window is created whenever a new *event* occur, i.e. a sensor shifts from one state to another. The data set we use consist of continuous data, not binary, which makes it more difficult to detect new *events*. One way to solve this could be to introduce a new *event* whenever some change in the signal exceeds a given threshold. This was performed by Kozina et al. [2011] who generated a new window whenever they found a significant decreasing change between consecutive data segments. This method showed promising results and increased their classification accuracy by 2%. As different activities are often performed with different intensities, Plötz et al. [2012] and Guenterberg et al. [2009] segmented the data signal by looking for changes in the energy level instead of changes in the data values (e.g. minimum values or maximum values).

2.4.2 Fixed Window Size

Fixed sized windowing has been the state-of-the-art segmenting approach for current HAR systems as generating dynamically sized windows has shown to be problematic. Using a fixed sized window approach, a window is moved along the signal which divides it into windows of equal length (see Figure 2.2). The window size and overlap between adjacent windows has to be decided. Windows used in previous HAR systems vary in size from a split second [Huynh and Schiele, 2005] to several seconds [Ravi et al., 2005, Bao and Intille, 2004, Kwapisz et al., 2011]. The most common degree of overlap is 50% [Ravi et al., 2005, Bao and Intille, 2004]).

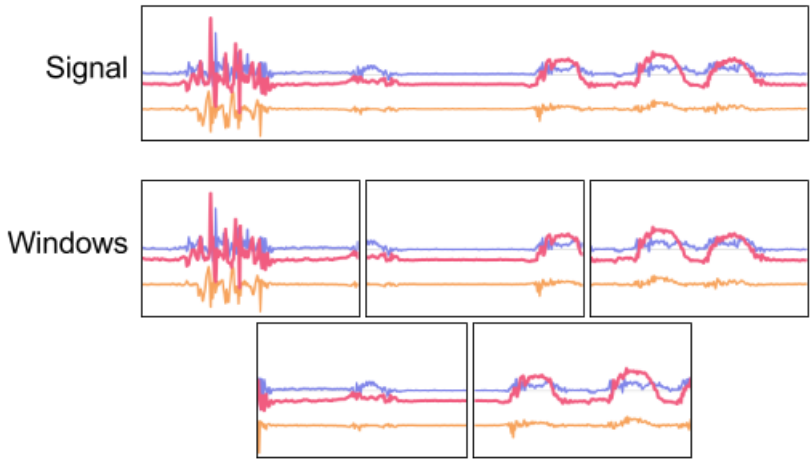


Figure 2.2: Segmenting the data signal into fixed sized windows with 50% overlap between adjacent windows

2.5 Feature Generation and Selection

Generating and selecting features is an important step in the activity recognition chain (Figure 2.1). Features are expected to contain relevant information about the raw data signal and are used as a reduced representation of the initial data. Feature generation is performed by discovering each window's characteristics and representing these as a set of abstractions. Features are usually divided into *time domain* features and *frequency domain* features.

2.5.1 Time Domain Features

Features from the time domain describes how the signal in the data segment changes with time. Time domain features are inexpensive to calculate and are therefore often used in HAR systems. Table 2.1 describes some common time domain features.

It is also possible to discriminate between body postures by calculating the direction of acceleration provided through Earth's gravitational pull. A posture can, for example, be classified as sitting if the angle between the sensor's z-axis and the gravitational vector is 90 degrees for the thigh sensor and 0 degrees for the back sensor (see Figure 2.3). The gravitational component of an accelerometer signal is the part of the signal with frequencies lower than 0.2 Hz. We can extract this component by filtering the raw signal (x, y, z) with a low-pass filter, and then use

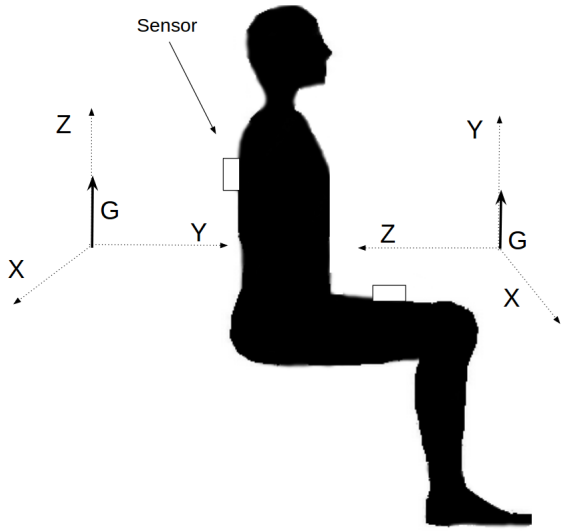


Figure 2.3: Orientation of thigh and back sensor compared to the gravitational force when sitting

the three resulting signals (x', y', z') to calculate the direction and the Euclidean norm ($\sqrt{(x')^2 + (y')^2 + (z')^2}$) of the gravity force [Van Hees et al., 2013].

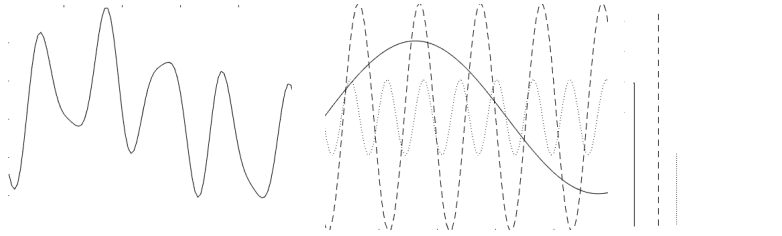
2.5.2 Frequency Domain Features

The theory of Fourier Series [Kreyszig, 2006] states that any signal in the time domain can be represented as a sum of sinusoidal waves at different frequencies with different amplitudes. Each sinusoidal waves can be represented in the frequency domain as a spike (vertical line). The spike's height is determined by the wave's amplitude, while the position of the spike is a result of the wave's frequency. By analyzing the frequency domain of the signal, it is possible to identify and analyze the different frequencies that are present in the original signal. The relationship between time domain and frequency domain is demonstrated in Figure 2.4. The signal in Figure 2.4a can be represented as the three different sinusoidal waves presented in Figure 2.4b (Fourier's theorem), while Figure 2.4c represents these three waves as spikes in the frequency domain.

To create frequency domain features, each data segments of the signal needs to be transformed into frequency domain. This transformation makes frequency domain features more expensive to compute than time domain features [Dargie, 2009, Muhammad et al., 2011]. Table 2.2 presents some common frequency domain features.

Table 2.1: Common time domain features

Feature	Description	Formula	Clarification
Mean	The average or central value of the signal sequence.	$\frac{1}{N}\sum_{i=1}^{N} x_i$	N is the length of the signal sequence and x_i is the value at position i.
Standard Deviation	Measure of the the amount of variance in a sequence of data values	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}$	N is the lenght of the signal sequence, x_i is the value at position i, and μ is the mean value of the signal
Maximum and Minimum	The highest and lowest value of a signal sequence.	$max(x_i)$ $min(x_i)$	x_i is the value at position i.
Zero-Crossing Rate	The zero-crossing rate is the rate of sign-changes along the signal sequence.	$\frac{\sum_{i=2}^{N} \lvert sgn(x_i)-sgn(x_{i-1})\rvert}{2(N-1)}$	N is the lenght of the signal sequence, x_i is the value at position i. sgn() returns +1 for positive inputs, and -1 for negative inputs.
Mean-Crossing Rate	The mean-crossing rate is the rate of how often the signal crosses the mean value.	$\frac{\sum_{i=2}^{N} \lvert sgn(x_i-\mu)-sgn(x_{i-1}-\mu)\rvert}{2(N-1)}$	N is the lenght of the signal sequence, x_i is the value at position i. sgn() returns +1 for positive inputs, and -1 for negative inputs. μ is the mean value of the signal.
Root Square Mean	The square root of the averaged square values of a signal sequence	$\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$	N is the lenght of the signal sequence, x_i is the value at position i
Energy	A signals energy is a measure of the signals strength	$E_x = \sqrt{\sum_{i=1}^{N}(x_i-\mu)^2}$ $Energy = \frac{1}{3N}(E_x + E_y + E_z)$	N is the lenght of the signal sequence, x_i is the value at position i on the x-axis signal. E_x, E_y and E_z are the energy for the different axises
Median	The number seperating the higher and lower half of the signal sequence	$Median_{odd} = x_{\frac{n+1}{2}}$ $Median_{even} = \frac{1}{2}(x_{\frac{n}{2}} + x_{\frac{n}{2}+1})$	x_i is number i in a sorted signal sequence. n is the length of the sequence
Cross-Correlation	Measure of similarity between two signals.	$d = \sqrt{\sum_{i=1}^{N}(x_i-\mu_x)^2 \sum_{i=1}^{N}(y_i-\mu_y)^2}$ $r_{xy} = \frac{1}{d}\sum_{i=1}^{N}(x_i-\mu_x)(y_i-\mu_y)$	x_i and y_i are the values at position i at the x and y signal. μ_x and μ_y are the signals mean value. r_{xy} is the resulting cross-correlation value.



(a) (b) (c)

Figure 2.4: This figure demonstrates that any signal in the time domain can be represented as a sum of sinusoidal waves, and how these waves can be represented in the frequency domain. a) A signal represented in the time domain. b) The signal in a) represented as three sinusoidal waves. c) Each sinusoidal wave consists of a frequency and an amplitude, and can be presented as a spike in the frequency domain.

2.5.3 Feature Selection

Feature selection, also called dimensionality reduction, is the process of reducing the number of features before feeding them into a classification algorithm. Feature selection serves two main purposes. First, it makes training a classifier more efficient by decreasing the size of the training set. This can be done by removing redundant or irrelevant features. Strongly correlated features could be redundant in the presence of each other. Second, feature selection can increase the classification accuracy by identifying and removing noisy features. Noisy features are features that give incorrect information. For example, if subjects in a data set are moving their legs when sitting, the classifier could misclassify activities with moving legs to the activity sitting. Such an incorrect generalization from an accidental property in the data set is called overfitting. Feature selection methods can be divided into filtering and wrapper methods.

Filtering methods apply statistical measures to assign a score for each feature. Based on the feature's score, the feature will either get selected or discarded from the data set. Filtering methods are faster than the wrapper approach, but they tend to include redundant features because they do not consider the relationships between features.

Wrapper methods select features based on a heuristic search. A common strategy is a greedy search, where features are added as long as the accuracy of the classification algorithm improves. This could also be done the other way around, where we start off with the entire feature space and keep removing features. Wrapper methods may obtain better performances than filtering methods but requires greater computational resources.

Table 2.2: Common frequency domain features

Feature	Description	Formula	Clarification
Mean	The average or central value of the magnitudes of all frequencies	$\frac{1}{N}\sum_{i=1}^{N} x_i$	N is the different frequencies, and x_i is the magnitude of frequency i.
Standard Deviation	Measure of the the amount of variance in the frequency spectrum	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}$	N is the different frequencies, and x_i is the magnitude of frequency i. μ is the mean magnitude.
Maximum	The highest magnitude value in the spectrum.	$max(x)$	x is the frequency spectrum
Median	The number separating the higher and lower half of the spectrum	$Median_{odd} = x_{\frac{n+1}{2}}$ $Median_{even} = \frac{1}{2}(x_{\frac{n}{2}} + x_{\frac{n}{2}+1})$	x_i is the magnitude of frequency i in a sorted spectrum
Spectral Centroid	Indicates where the "center of mass" of the spectrum is. It is calculated as the weighted mean of the frequencies present in the signal.	$\frac{\sum_{i=0}^{N-1} x_i \cdot i}{\sum_{i=0}^{N-1} x_i}$	x_i represents the magnitude of frequency i.
Dominant Frequency	Extracts the frequency that carries the maximum energy among all frequencies found in the spectrum.	maximum = max(x) frequency= find_index(maximum)	max(x) finds the maximum magnitude in the spectrum. find_index(maximum) finds the frequency with maximum magnitude.
Spectral Entropy	Measure of randomness or disorderness of the spectrum.	$p_i = \frac{\frac{1}{N}x_i^2}{\sum_{i=1}^{N}\frac{1}{N}x_i 2}$ $H = -\sum_{i=1}^{N} p_i ln(p_i)$	N is number of frequencies. $x_i is$ the magnitude of frequency number i. p_i is the normalized Power Spectral Density. H is the Entropy

2.6 Classification

Classification is the last step of the activity recognition chain, and is the problem of categorizing new observations. Several classification algorithms have been used in current HAR systems, where the majority of these algorithms are supervised learning algorithms. Supervised learning algorithms use labeled training data to produce a classification model that can be used to determine unseen data instances. In the case of activity recognition, features generated from labeled sensor data will be used to train a classification model. This model will then be able to classify activities based on new sensor data. Decision Tree, Random Forest, Support Vector Machine, K-Nearest Neighbour, Artificial Neural Networks and Naive Bayes classifier are all examples of supervised learning algorithms that have demonstrated success

in classifying human activities. During fall 2015 [Tessem and Hessen, 2015], we experimented with these classification algorithms. Random Forest and Artificial Neural Networks provided the best results and are classifiers we will proceed with in this study. Random Forest and Neural Networks are described below.

2.6.1 Random Forest

The Random Forest algorithm [Pang-Ning Tan, 2006] works as a large collection of decorrelated decision trees. We, therefore, start by describing how one individual decision tree is created.

A decision tree [Pang-Ning Tan, 2006] is a predictive model that maps observation of an item to conclusions about the item's class. The decision tree, illustrated in Figure 2.5, has three types of nodes: a root node at the top of the tree with no incoming edges and two or more outgoing edges, internal nodes with one incoming edge and two or more outgoing edges, and finally, leaf nodes with one incoming edge and no outgoing edges. The root node and the internal nodes include an attribute test conditions to separate instances depending on their values. The leaf nodes have an assigned class label to it. Each recursive step of the tree-growing process must split on an attribute. This attribute is selected using different methods, such as Gini, Information Gain or Variance reduction. This procedure is continued until a stopping criterion is met (see Algorithm 2.1). An advantage of using decision trees is that it enables us to identify the most important attributes (features).

Algorithm 2.1 A skeleton decision tree algorithm

```
E-training set, F-attribute set
function DECISION_TREE(E, F)
    if stopping_confition(E,F) = true then
        create leaf node
        set leaf class as majority number of records in E
        return leaf
    else
        create new node T(either root or internal)
        find attribute A that splits the training records best
        for each possible value a in A do
            let E_a be the subset of training records in E that has value a for A
            child = Decision_tree(E_a, A)
            add child as descendent of T and label the edge (T → child) as a
        end for
    end if
    return node
```

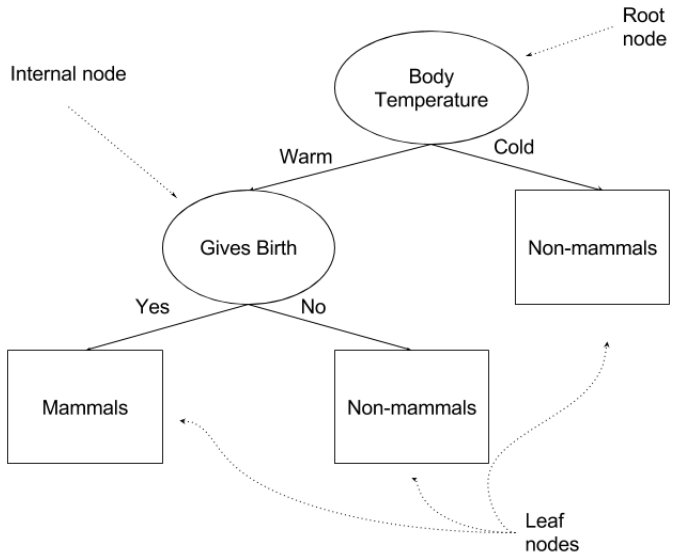


Figure 2.5: Illustation of a decision tree that can classify between mammals and non-mammals

As mentioned above, the Random Forest algorithm [Pang-Ning Tan, 2006] works as a large collection of decorrelated decision trees. Random Forest starts off by dividing the data set into N random subsets. For each subset, it creates a decision tree. Instances are classified by each decision tree, and the final class prediction is the majority prediction of the trees (see Algorithm 2.2). Supported by the law of large numbers (the average of results from a large number of experiments is close to the expected value), Random Forest produces accurate results and is to some extent shielded against overfitting [Breiman, 2001].

Algorithm 2.2 Random forest - Tree bagging

for b = 1 to B **do**

Sample, with replacement, n training examples X, Y; call these X_b,Y_b

Train a decision tree f_b on X_b,Y_b

end for

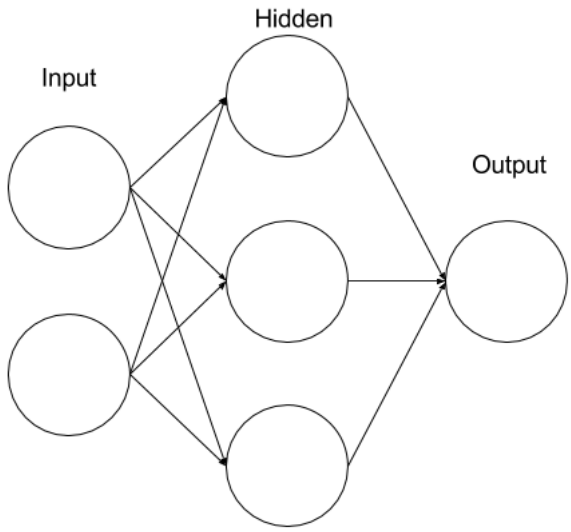


Figure 2.6: ANN Structure

2.6.2 Artifical Neural Network

An Artificial Neural Network (ANN) [Pang-Ning Tan, 2006] is inspired by the biological neural system. The human brain consists of neurons linked together by axons. In an ANN, these neurons are called nodes and axons are called weights, and are arranged in a layered structure (see Figure 2.6).

The first layer is the input layer. The different input values are placed in the different input nodes (e.g. if we are classifying a 4x4 picture, these values would be the 16 pixel values). The second layer is the hidden layer (there could be multiple hidden layers). Nodes in one layer are fully connected to nodes in the subsequent layer. These connections are weighted. The resulting value in a node is the sum of all node values in the previous layer multiplied by the weight between them. The result is then run through a function (typically the Sigmoid function), and the procedure is continued until we reach the last layer, the output layer. An ANN learns by altering the weight between the nodes after each training iteration. This learning process could be done by using a backpropagation algorithm. ANNs have proven to be effective in several domains, but they also have some drawbacks. One of the main drawbacks is that the network can be viewed as a black box because it provides a limited explanation of how the classification is performed. Another weakness is that training deep neural networks take a large amount of time.

2.7 Acti4

Acti4 [Skotte et al., 2014] is a state-of-the-art HAR system which follows the activity recognition chain described in this chapter. Acti4 is used in several Scandinavian human activity studies [Danquah et al., 2016, Hallman et al., 2015, Munch Nielsen et al., 2016] and is trained to classify everyday physical activity. These activities are *walking*, *running*, *cycling*, *walking stairs*, *sitting*, *standing* and *moving* (*moving* is a standing posture mixted with small leg movements).

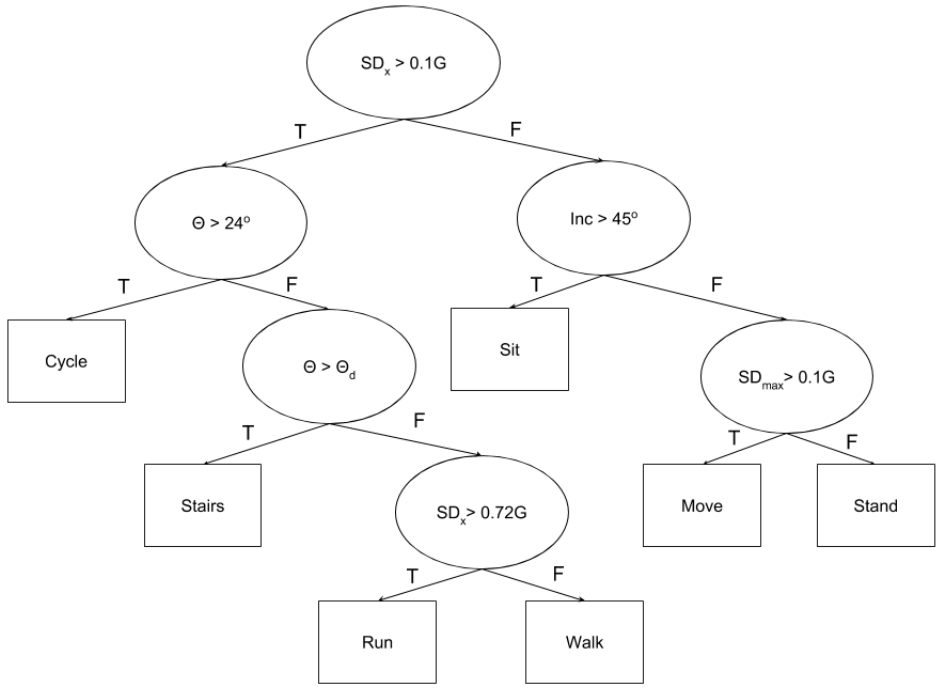


Figure 2.7: The decision aree used to classify instances in Acti4. SD_x = standard deviation of accelerometers x-axis. SD_{max} = is the maximum standard deviation of the three axes. Inc = inclination of x-axis (only positive values, range: 0-180°). θ = backward/forward thigh angle (range: ±90°) and θ_d = individual threshold angle for each subject. $G = 9.81\text{m/s}^2$

The data used when developing Acti4 was collected from 17 healthy adults (10 females and 7 males). The collection process was divided into two protocols, one standardized protocol, and one free-living protocol. In the standardized in-lab protocol the subjects were asked to perform the activities *walking*, *running*, *cycling*, *walking stairs*, *sitting* and *standing*. Each activity was performed for 5 minutes, resulting in 30 minutes of data for each subject. The second protocol lasted for 9 hours and was performed to validate the criteria for detecting seated postures during everyday life. During the

free-living protocol the subjects had a pressure sensor placed in their back pocket, enabling the detection of periods spent sitting.

During both protocols, a triaxial accelerometer (actiGraph GT3X+ [1]) was located on the subject's thigh. An additional accelerometer was located on the subject's hip during the free-living protocol.

The data was divided into 2-second windows with 50% overlap. The standard deviation and inclination of the sensor were calculated and used as features. The final classification system works as a decision tree and is depicted in Figure 2.7.

2.8 Motivation

State-of-the-art HAR systems classify activities with reasonable successful rates. However, in this study, we will try to find methods that can further improve these systems. We will look into several machine learning approaches not commonly tried before in this context. This includes deep learning, semi-supervised learning, and dynamic classification. We will also look into different dynamic windowing approaches to see if this can lead to more accurate HAR systems. We are interested in seeing how different methods and combinations of methods will affect the overall performance of a HAR system. Hopefully, these approaches will contribute to even better HAR systems.

Most HAR systems are not released to the public, making it difficult for researchers to work together towards the common goal of generating an optimal HAR system. This motivates us to make our system open-source, so that other researchers can both contribute to our system and apply components from our system into their systems.

HUNT4 will offer objective measurements of both adults and adolescents. In this study, we will generate a HAR system based on a data set from adults only. Therefore, we are eager to see how well our HAR system will work on data from adolescents. If we do not get satisfactory results it may be necessary to generate separate HAR classifiers for these two groups.

[1] http://actigraphcorp.com/support/activity-monitors/gt3xplus/

Chapter 3
Beyond State-of-the-art

3.1 Introduction

In this chapter, we will describe deep learning, semi-supervised learning and dynamic classification, and how these methods can improve state-of-the-art HAR systems.

3.2 Deep Learning

Most previous research into HAR has used well-established classification algorithms that rely on hand-crafted feature generation. Hand-crafted feature generation is a weak point in the HAR process, as the choice of features is driven by human intuition, and using the right features is crucial for obtaining a high accuracy. Traditional HAR systems generate hand-crafted features by using prior knowledge about the data, whereas in deep learning features are generated by discovering patterns in the data. Deep learning has been proven to outperform traditional machine learning in domains such as speech recognition [Hinton et al., 2012] and image recognition [Farabet et al., 2013]. Moreover, researchers believe deep learning will succeed in several domains in the near future due to the absence of hand-crafted features and the increasing amount of available data [LeCun et al., 2015].

3.2.1 Convolutional Neural Network

A Convolutional Neural Network (CNN) is an extended Artificial Neural Network (see Section 2.6.2) inspired by the biology of the visual cortex in animals. In the visual cortex, complex arrangements of cells cover the respective field (visual field). These groups of cells act as filters over the visual field and are able to detect (filter out) different types of features. The same concepts are transferred into the field of artificial neural networks, and will be explained in the following sections.

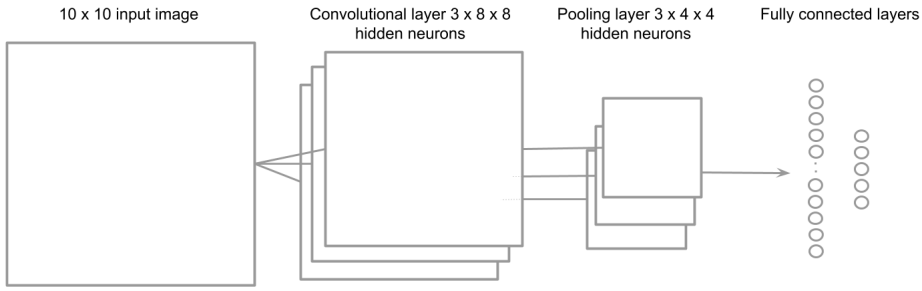


Figure 3.1: CNN Structure

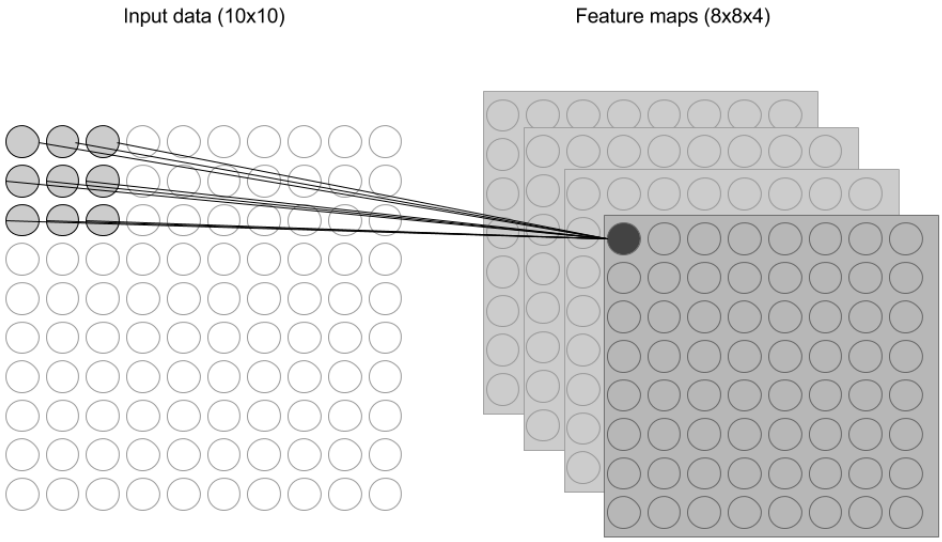


Figure 3.2: Convolutional step in a CNN

A CNN is structured in layers, where the first N layers consist of a convolutional layer followed by a pooling layer. The convolutional layer and the pooling layer will be explained in the following sections. After N layers, we attach a fully connected Neural Network. The depth and layer complexity of the CNN depends strongly on the problem domain. A one layer CNN is depicted in Figure 3.1.

The objective of the convolutional layer is to extract features from the previous layer (the *input layer* or a *hidden layer*). Features are extracted using filters, called *kernels* (e.g. a 3x3 matrix), that are convolved over the previous layer, resulting in a *feature map*. The kernel detects the feature, which results in a feature map that represents the locations of each feature in the previous layer. Using multiple kernels,

the convolutional step can detect multiple features, resulting in several feature maps. Figure 3.2 shows this process using four kernels over the input data, outputting four different feature maps.

The feature map is calculated using a non-linear function. An example of such a function is presented below.

$$h_{ij}^{f} = \tanh((W^{f} * x)_{ij} + b_{f})$$

Where h^f is the f-th feature map at layer h, and x is the previous layer. The filters are determined by the weights W^f and bias b_f, and $*$ is the mathematical symbol for convolution.

After each convolution step there is a pooling layer, also called the subsampling step. The objective of the pooling layer is to reduce the network complexity and control overfitting. The output of a pooling layer is the maximum (other methods could also be used, such as the mean value) value of each NxN region in the previous feature map, see Figure 3.3.

The *pooling matrix* is the result of subsampling the feature maps.

$$p^{f} = \max(h^{f})$$

Where h^f is the f-th feature map at layer h.

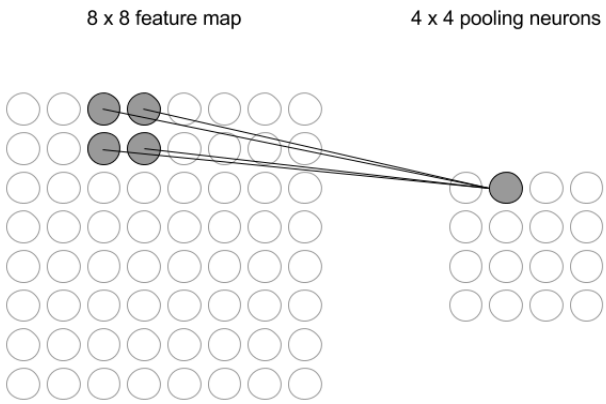


Figure 3.3: Pooling step in a CNN

After L convolutional and pooling layers, we feed the output from the final pooling layer into a fully connected neural network.

Every kernel in the convolutional layer and the weights in the fully connected layers are initialized with random values. When training, data is fed through the network and the network produces a vector of scores, representing the score for each class. An error is calculated by comparing the score vector against the desired output vector using an objective function. The weights throughout the network are then tuned to reduce this error. The learning algorithm first calculates the gradient for each of its weights, learning how much the error would decrease or increase by altering the weights. It then adjusts the weights in the opposite direction of the gradient. Using the backpropagation algorithm, it is possible to propagate the error from the output layer backward, throughout the hidden layers [LeCun et al., 2015, Krizhevsky et al., 2012].

The features in the lower layers in a CNN capture basic features from the signal while the higher layers detect a combination of the features obtained from the lower layers. Basically, the deeper the CNN is, the more fine-grained activities can be classified [Yang et al., 2015]. In image recognition, the first layers typically detect the presence or absence of edges, the second layers often detect arrangements of these edges, while the third layers may represent patterns that resemble parts of objects [LeCun et al., 2015]. These concepts are transferable to the problem of detecting activities in a raw accelerometer signal.

Dropout is a technique that deals with overfitting in a neural network [Srivastava et al., 2014]. During training, each hidden neuron is either dropped out of the network with the probability $1 - p$, or kept with the probability p. If the neuron is dropped, the weights connected in and out of it are canceled. The thinned network is then trained, forcing the overall network to have multiple networks (i.e. multiple paths) that are able to detect the same classes. Dropout improves the training speed (i.e. fewer neurons to train) and creates more robust features (i.e. by having multiple paths in the network). The alternative is to train several networks and have a majority voting at the end, but this approach is very time-costly. Dropout can be applied to one or several layers in the fully connected neural network.

3.3 Dynamic Classifiers

Humans perform activities in a natural order. For example, it is very normal to shift back and forth between walking and standing, but not between walking and laying. In other words, the recent history of activities a person has performed can help to predict the activities that will be performed in the near future. To ensure temporal smoothness of activities, it is possible to combine results obtained by a baseline classifier (e.g. CNN or Random Forest) with dynamic classification models, like Hidden Markov model (HMM).

3.3.1 Hidden Markov Model

The HMM [Rabiner and Juang, 1986] is based on the theory of Markov chains. A Markov chain is a stochastic process characterized by a sequence of states over time. The Markov chain assumes the Markov property, which is that the future states depend only upon the present state, not on the sequence of former states. For example, if the current state is known, then future states are independent of the previous states. A sequence of human activities can be modeled as a Markov model, where a state is the activity performed at time t, and each single activity can be assumed to be dependent on only the previous performed activity.

HMM is a Markov model with unobserved (hidden) states. Even though the states are not directly visible, the output of each state (observation) is visible. Since these observations are related to the states, a sequence of observations will give information about the sequence of states. In the field of HAR, the underlying activity will be the hidden state, and the observations will either be the sensor readings, or the classification results based on the sensor readings.

For example, if you are in a house without windows, and you observe a person entering the house with an umbrella, the probability for it raining outside increases. In this case, the *umbrella* is the observation, and the *rain* is the underlying state. If we assume that the weather on a given day is only dependent on the weather on the previous day, then we can say that this weather model satisfies the Markov property. Figure 3.4 is a representation of three timesteps of a HMM. The top nodes, S, represent the hidden states, and the bottom nodes, O, represent the observations. Even though the HMM assumes the Markov property, the underlying states in an HMM are hidden (unknown) and we can therefore not expect any interdependencies between nodes.

There are three types of probabilities in an HMM. Firstly, transition probabilities, which are the probabilities of going from one state to another. Secondly, emission probabilities, which are the probabilities of making an observation given a state. Thirdly, an initial probability distribution for the first state. Figure 3.5 shows the

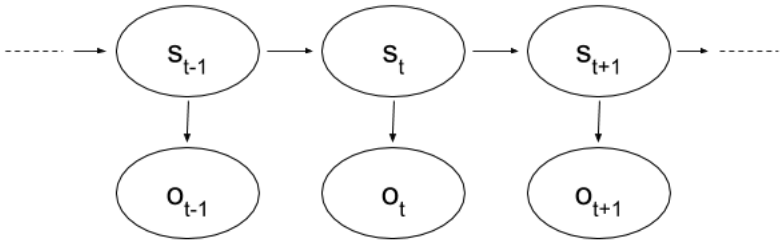


Figure 3.4: Illustration of three steps in a HMM

transition and emission probabilities of an HMM with the hidden states *rain* and *sunny*, and observable states *umbrella* and *no umbrella*. The arrows between the hidden states show the probability of going from the first state to the next state (transition probabilities), meaning that if it is raining there is a 30% probability of it being sunny the following day, and 70% of it still raining. The arrows from the hidden states to the observable states show the probability of making an observation given the hidden state (emission probabilities), meaning that the likelihood of observing an umbrella if it is raining is 90%, and not observing an umbrella if it is raining is 10%.

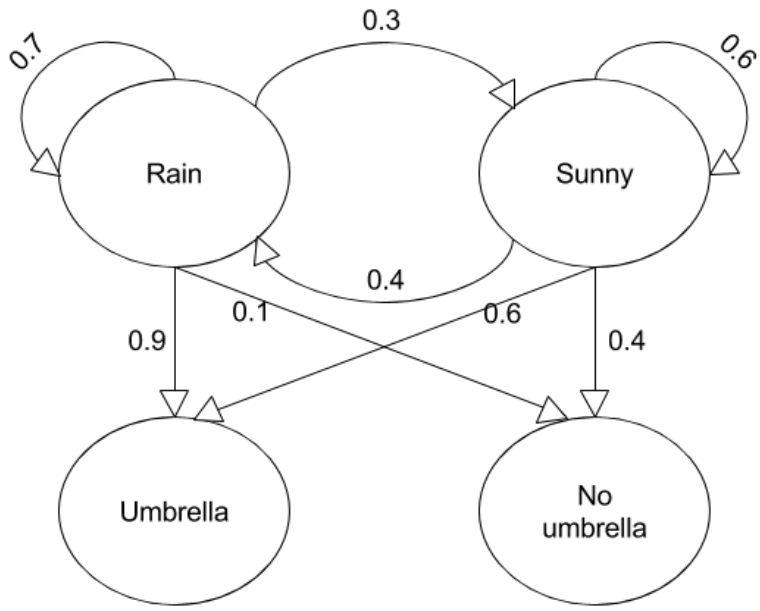


Figure 3.5: HMM - Transition and Emission Probabilities

3.3.2 Estimate Transition Matrix by Counting Transitions

A common way to estimate a transition matrix from a data sequence is to count the number of transitions between states. However, this method requires that the hidden states are known, which is not always the case. For example, if you have a sequence of states S = $\{s_1, s_2, ..., s_T\}$, let k_{ij} be the number of transitions from state i to state j. Compute $k_{i,j}$ for all the different states and estimate the transition matrix A using the estimation $a_{ij} = \frac{k_{ij}}{\sum_{j=1}^{N} k_{ij}}$, where N is the number of different states.

3.3.3 Estimate Transition Matrix Using Baum-Welch

Contrary to the previous method where the transition matrix is estimated by counting transitions, the Baum-Welch algorithm is not dependent on known hidden states. Given an HMM model with an unknown transition matrix A and an observation history $O = \{o_1, o_2, \ldots, o_T\}$, this algorithm calculates the transition matrix A that best explains the observation history. The initial transition matrix can be selected in different ways (e.g. uniform distribution or random). Baum-Welch is based on an iterative process where a new transition matrix A' is generated at every iteration, where A' explains the observation history O better than the old transition matrix A. The probability of observing O given A' is greater than the probability of observing O given A, $P(O|A') > P(O|A)$. We can say that Baum-Welch maximizes $P(O|A)$. The main steps of the Baum-Welch algorithm are:

1. Start with an initial transition matrix A
2. Iterate until convergence
    - Compute the expected number of occurences of state i
    - Compute the expected number of transitions from state i to state j
    - Adjust transition matrix to maximize $P(O|A')$

The expected number of occurences of state i in sequence O is calculated by summarizing the probabilities of being in state i at time t, $\gamma_t(i)$, over $t = 1, 2, ..., T-1$.

$$\sum_{t=1}^{T-1} \gamma_t(i)$$

The probability of being in state i at time t, $\gamma_t(i)$, is calculated using the formula below and is demonstated in Figure 3.6.

$$\gamma_t(i) = \frac{\alpha_t(i) \cdot \beta_t(i)}{P(O|\lambda)}$$

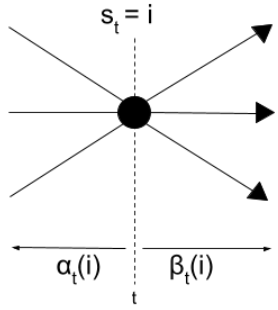


Figure 3.6: Illustration of the probability of being in state i at time t. $\alpha_t(i)$ is the forward probability and $\beta_t(j)$ is the backward probability.

$P(O|\lambda)$ is the overall probability to generate the observations O given HMM λ. $\alpha_t(i)$ is the forward probability. $\beta_t(i)$ is the backward probability.

The expected number of transitions from state i to state j is calculated by summarizing the probabilities of going from state i to state j, $\xi_t(i,j)$, over $t = 1, 2, ..., T-1$.

$$\sum_{t=1}^{T-1} \xi_t(i,j)$$

The probability of going from state i at time t to state j at time $t+1$, $\xi_t(i,j)$, is calculated using the formula below and is demonstated in Figure 3.7:

$$\xi_t(i,j) = \frac{\alpha_t(i) \cdot a_{ij} \cdot p(o_{t+1}|j) \cdot \beta_{t+1}(j)}{P(O|\lambda)}$$

$P(O|\lambda)$ is the overall probability to generate the observations O given HMM λ. $\alpha_t(i)$ is the forward probability. $\beta_{t+1}(j)$ is the backward probability. $p(o_{t+1}|j)$ is the probability of observing o_{t+1} given state j (emission probability), and a_{ij} is the transition probability from state i to state j.

The new transition matrix A' is calculated by dividing the expectation for the number of transitions from state i to state j by the expectation of the number of occurrences of state i. The formula is presented below.

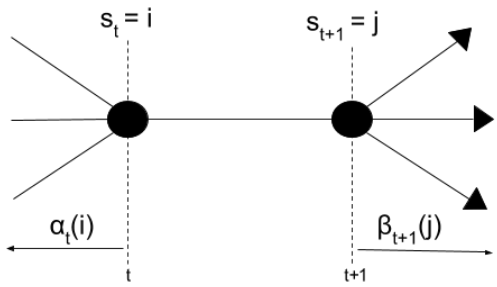


Figure 3.7: Illustration of the probability of being in state i and j at time t and $t+1$. $\alpha_t(i)$ is the forward probability, $\beta_{t+1}(j)$ is the backward probability, and the line between t and $t+1$ illustrates the transition probability of going from S_t to S_{t+1}

$$a'_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

These steps will be repeated until the transition matrix converges.

3.3.4 The Viterbi Algorithm

The Viterbi algorithm is an algorithm that uses the HMM to find the most likely sequence of hidden states based on the observations made, the emission probabilities, the transition probabilities, and the initial probabilities.

$$v_i(1) = w_i \cdot P(o_t|i) \quad \text{for } 1 < i < N$$

where $v_i(1)$ is the probability for state i at time $t = 1$, N is the number of states, w_i is the start probability for state i, and $P(o_t|i)$ is the emission probability for observing o_t given state i.

Let $v_i(t)$ be the probability of the most likely path ending in state i at time t.

$$v_i(t) = \max_{s_1,s_2,...s_{t-1}} P(s_1, s_2...s_{t-1}, s_t = i, o_1, o_2, ..., o_t|\lambda)$$

where λ is the HMM with initial, emission and transition probabilities.

$v_j(t)$, the probability of the most likely path ending in state j, can then be calculated by induction.

$$v_j(t) = \max_{1 \leq i \leq N} [v_i(t-1)a_{ij}]P(o_t|j)$$

where N is the number of states, a_{ij} is the transition probability from state i to state j, and $P(o_t|j)$ is the emission probability for observing o_t given state j.

Finally, the endpoint with the highest probability is chosen, and backtracing from this endpoint to the start is used to find the most likely path.

3.4 Semi-supervised Classifiers

In supervised learning, we have a training set that contains both input data $(x_1, x_2, ..., x_n)$ and output data $(y_1, y_2, .., y_n)$. The main goal is to create a function that approximates the output given the input. In unsupervised learning, the training set only consists of the input data $(x_1, x_2, .., x_n)$, and the goal is to find meaningful structures in the data.

Semi-supervised learning is a learning method that combines the properties of both supervised and unsupervised learning, where we have both labeled and unlabeled data. This method is motivated by the fact that labeled data is often costly and time-consuming to obtain. In HAR systems, semi-supervised learning can potentially reduce the amount of manual labeling of data and adapt a classifier to a specific individual or demographic group.

Semi-supervised learning algorithms can be divided into two subgroups, single-view, and multi-view. The number of views represents the different ways we can interpret the data. Single-view is where the algorithm uses a single set of features, and multi-view is where the algorithm uses multiple sets of features. A multi-view example is the task of detecting junk email, where the one view is the features obtained from the email body, while the second view is features obtained from the email header.

3.4.1 Self-training

Self-training is a single-view semi-supervised learning algorithm. Self-training starts by creating a static classifier based on labeled data (supervised learning). The classifier is then used to classify unlabeled data. Instances that the algorithm classifies with a high enough confidence are added to the training set. The classifier is then retrained (or further trained, depending on the type of classifier) on the updated training set. These steps are repeated until the remaining unlabeled data is labeled or a stopping criterion is met (see Algorithm 3.1 and Figure 3.8).

Algorithm 3.1 Self-training

```
Classifier.build(trainingData)
while unlabelledData.size > 0 do
    testData = unlabelledData.subset()
    for sample in testData do
        prediction = Classifier.predict(sample)
        if prediction.confident > threshold then
            trainingData.add(sample, prediction)
            unlabelledData.remove(sample)
        end if
    end for
    Classifier.rebuild(trainingData)
end while
```

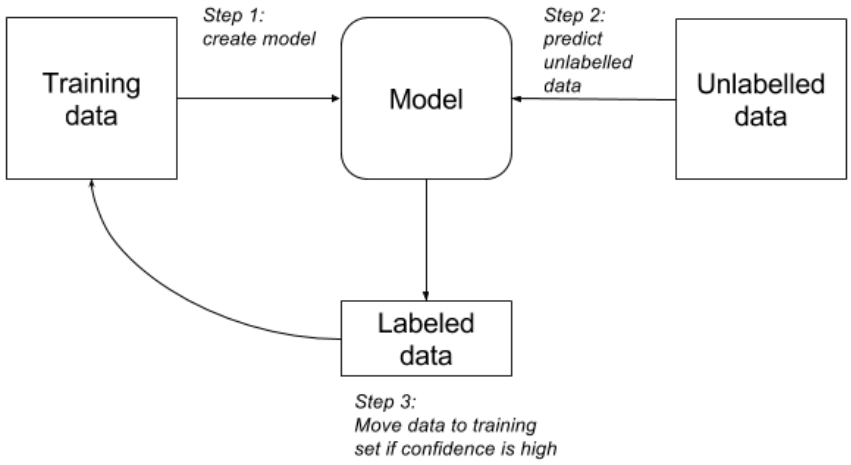


Figure 3.8: The different steps in a semi-supervised learning process

3.4.2 Active Learning

Active learning is a single-view learning algorithm, that, unlike self-training algorithms, does not automatically label new data. It analyses how confident the classifier is when classifying instances, and then requests that the user manually labels data instances that will benefit the classifier. This will typically be data segments where the classifier is unsure of the data segment's class, i.e. the confidence is low. This learning process is not necessarily suitable for human activity classification, as it requires additional equipment (smart-phone or similar) for the HAR system, and the user must actively respond to the system throughout the learning phase.

3.4.3 Co-training

Co-training is a multi-view semi-supervised learning algorithm [Blum and Mitchell, 1998]. Co-training is an extension of self-training, but instead of using one classifier, Co-training uses multiple classifiers. These classifiers are trained on different feature sets (views). Like in self-training, the newly labeled data is added to the training set if the confidence is above a threshold.

Co-training is based on two assumptions; the first is that the two different views have features that are conditionally independent given the class (e.g. each instance could be described by two different feature sets), the second is that each view is sufficient to classify an instance alone. Not all real-world classification problems have multiple conditionally independent views, especially when analyzing human activities using accelerometer sensors. However, this could be solved by adding more sensors to the HAR system, such as a gyroscope or GPS, in which case, each sensor could be a different view.

Since it is often problematic to obtain multiple views of the data, there have been studies that focus on co-training with use of single-views. An example of this is Nigam and Ghani [2000], who accomplish this by creating artificial views. They introduced a feature splitting algorithm which creates two feature sets that are maximally independent of each other. Another example is Zhou and Goldman [2004], where they use different learning algorithms in the same view. Their methods build on the assumption that different learning algorithms have different learning biases (the set of assumptions the algorithm makes about the target function). The final class prediction is then decided using majority voting among the different classifiers.

3.4.4 Other Strategies

Both single view and multi-view semi-supervised learning approaches can be combined with other machine learning techniques.

Bagging [Breiman, 1996] is a method used to improve stability and accuracy in machine learning problems. This method creates N training sets with size S', by selecting random samples, with replacement, from the original training set S. It then trains N classifiers on each of the N data sets and performs a majority voting classification to predict the final class.

Boosting [Freund et al., 1999] is an iterative learning process where weak learners (classifiers that are able to solve simple classification problems) are combined to create a strong learner. For each iteration, the algorithm increases the importance of the instances that are misclassified by the strong learner. In this way, a new weak learner that focuses on instances misclassified by the strong learner is added to the system.

Random Subspace Method [Ho, 1998] is a learning method where the feature space is divided into different subspaces. Each classifier is then trained on the different feature subspaces, and a majority voting is performed to obtain the overall prediction.

3.5 Summary

Deep learning, dynamic classification, and semi-supervised learning are machine learning techniques that we believe have the potential to improve current HAR systems. Chapter 5 outlines a proposed HAR system design where each of these methods are included, while Chapter 6 will present an iterative process to see how each of these methods, and the combination of methods, will affect the overall performance of the HAR system.

Chapter 4
Collection of the Data Set

The data set used in this study was collected in collaboration with Hilde Bårdstu and Atle Kongsvold as a part of a project which took place fall 2015 [Tessem and Hessen, 2015]. The data set consists of daily activities and was collected from body-worn accelerometer sensors. This chapter presents the different aspects of the data collection process.

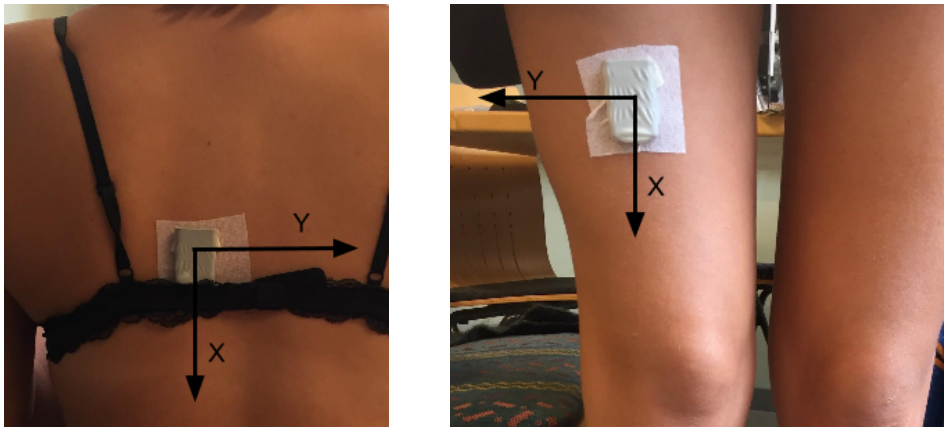
4.1 Sensors

The data was collected using two AX3 Axivity sensors[1]. These sensors feature a state-of-the-art accelerometer that is used to detect movement, vibrations and orientation changes in all three axes. The sensors sampling frequency was set to 100 Hz (100 measurements each second). The dimensions of the sensors are 23 mm x 32.5 mm x 7.6 mm and the weight is 11 g. The small size and light weight of the sensor makes it suitable for being worn for a long period of time without discomfort. The sensors were placed at the front thigh and the upper back. The sensor at the upper back was placed 2 cm to the left of the spine for comforting reasons and to avoid the sensor to loosen. Figure 4.1 shows the sensors placement and axes.

4.2 Subjects

The subjects consisted of one group of 12 healthy adolescents (age 13-16 years) and one group of 23 healthy adults (age 28-52 years). The adolescents were recruited from Charlottenlund secondary school, and adults were recruited from staff at NTNU. All participants were informed about the purpose of the study and their legal rights as participants. They were also informed that they could withdraw from the study at any given time without further questions. The adolescents that were participating needed parental approval. All participants were treated according to a legal and ethical perspective. The data was anonymized by giving each participant a serial

[1] http://axivity.com/product/ax3



(a) Back sensor (b) Thigh sensor

Figure 4.1: Sensor Placement

Table 4.1: Information about the subjects contained in the data collection process

	Subjects	Males	Females	Average Age	Average Height	Average Weight
Adults	23	9	14	40,17	172,58 cm	71,89 kg
Adolescents	12	6	6	14,67	167,35 cm	58,27 kg

number. The project obtained ethical clearance from NTNU's REK (Regional Committees for Medical and Health Research Ethics) and was carried out according to the Declaration of Helsinki.

Table 4.1 presents information about the subjects contained in the data collection process. More detailed information about each subject is presented in Section A.1 in Appendix A.

4.3 Data Collection Process

The process of collecting the data differed between adults and adolescents.

The data collection process for the adults were divided into two protocols. The first protocol was carried out at St. Olavs Hospital (Trondheim, Norway). The subjects were first asked to perform a standardized sequence of typical everyday activities, including lying down, sitting, standing and picking up objects. This sequence was carried out in a lab. A GoPro camera[2] was then attached to the subjects chests. The camera was pointing down towards their legs, making it possible to identify the activities the subjects were performing. The subjects were then asked to do a

[2]http://gopro.com

sequence of activities including walking stairs, walking uphill, jogging/running, and cycling. The uphill walking, jogging/running, and some of the regular walking was performed on a treadmill. The cycling was carried out on an ergometer cycle. No instructions regarding pace or movement style were given. The total duration of the in-lab protocol was around 45 minutes. The second protocol was carried out in the subject's working environment. In this part, the subjects also had a GoPro camera attached to their chest. The subjects got a list of activities they were asked to carry out within the next couple of hours. These activities included sitting, standing still, shuffling (standing with small leg movement), walking flat, walking stairs, laying down and running.

Since adults were wearing a GoPro camera in their work environment, it was made sure the camera did not interfere with the performance of their scheduled activities.

The data collection process for the adolescents were also divided into two protocols. Both protocols were carried out at St. Olavs Hospital (Trondheim, Norway), and were performed with the GoPro camera attached to the subject's chest the same manner as described above. During the first protocol, the subjects were asked to carry out a sequence of rapid movement activities with a lot of changes in directions. These activities included running with direction changes, running sideways, and running zig-zag between cones. This protocol lasted for around 10 minutes. During the second protocol, the adolescents were asked to perform different tasks. These tasks included running one round on a running track, sitting in three different chairs, walking to the store and buy lunch, and finding a bench to lie down on. No instructions were provided regarding how to perform the different tasks.

To get a coordinated start and end point of the sensor signal, the subjects had to do three repetitions of *heel-drops* at the beginning and end of each protocol in the data collecting process. This *heel-drop* made a very distinct pattern in the sensor data stream, and the peaks in this pattern were marked as start and end points.

The data stream was annotated using a video annotation tool called ANVIL [3]. The activities used in the annotation process were

1. *Walking*
2. *Running*
3. *Shuffling*
4. *Stairs (ascending)*
5. *Stairs (descending)*
6. *Standing*
7. *Sitting*

[3] http://anvil-software.org/

8. *Lying*
9. *Transition*
10. *Bending*
11. *Picking*
12. *Undefined*
13. *Cycling (sitting)*
14. *Cycling (standing)*
15. *Heel-Drop*
16. *Vigorous Activities*
17. *Non-Vigorous Activities*

The activity definitions utilized in the annotation process can be found in Table A.4 and A.5 in Appendix A. The inter-rater reliability was 0.96 for the adult data set and 0.91 for the adolescent data set, and was calculated using Fleiss' Kappa [Fleiss, 1971].

4.4 Post-processing of Activities

Before generating the HAR system we decided to delete 3 activities activities from the data set. These were *undefined*, *heel-drop* and *non-vigorous* activities.

Undefined are all activities that can not be defined during the labeling process. It is impossible for a classifier to recognize this activity as it is a collection of all types of bodily movements. *Undefined* was therefore excluded from the data set.

Each subject performed a *heel-drop* to mark the start and end point of the data collection protocol, making it possible to synchronize the thigh and back sensor. *Heel-drop* is not defined as an activity, and was, therefore, deleted from the data set.

Non-vigorous activities is a collective term and are defined as "All non-cyclic movements that do not classify according to the definitions". However, the movement that are labeled as *non-vigorous* activities in this data set are very similar to *standing* and *walking*. It is also very few samples of *non-vigorous* activities in the data set, and it was, therefore, deleted as an activity.

We also ended up relabel *picking* as *bending*. *Picking* is the point at the bottom of the *bending* activity. There are very few data samples of this activity, and we had an agreement with medical experts at ISM at NTNU that we could relabel this activity as *bending*.

Chapter 5
System Design

5.1 Introduction

The main objective of this study is to create a precise and reliable HAR system for the upcoming HUNT4 study. In Chapter 3 we presented deep learning, semi-supervised learning, and dynamic classification, and will in this chapter introduce an HAR system architecture that combines these machine learning methods. The system is divided into two parts; a training phase, described in Section 5.2, and a classification and validation phase, described in Section 5.3. The architecture presented in this chapter demonstrates the different parts of the system, while Chapter 6 provides a deeper description and evaluation of each part, before presenting the final instantiation of the system.

5.2 Training Phase

The training phase involves training the classification system using labeled activity data. The proposed training pipeline is illustrated in Figure 5.1 and the different steps in the pipeline are described below.

1. **Data Segmentation**: Instead of classifying each data point, the raw training data is segmented into data segments, called windows. Each data window will correspond to a single activity.

2. **Pre-processing**: Data windows that reduce the classification system's ability to differentiate between activities will be removed or adjusted. This could be activities that either overlap with other activities or activities with no specific movement pattern, making it difficult for the classifier to recognize them. This step also provides the opportunity to remove undesirable or unimportant activities from the HAR system.

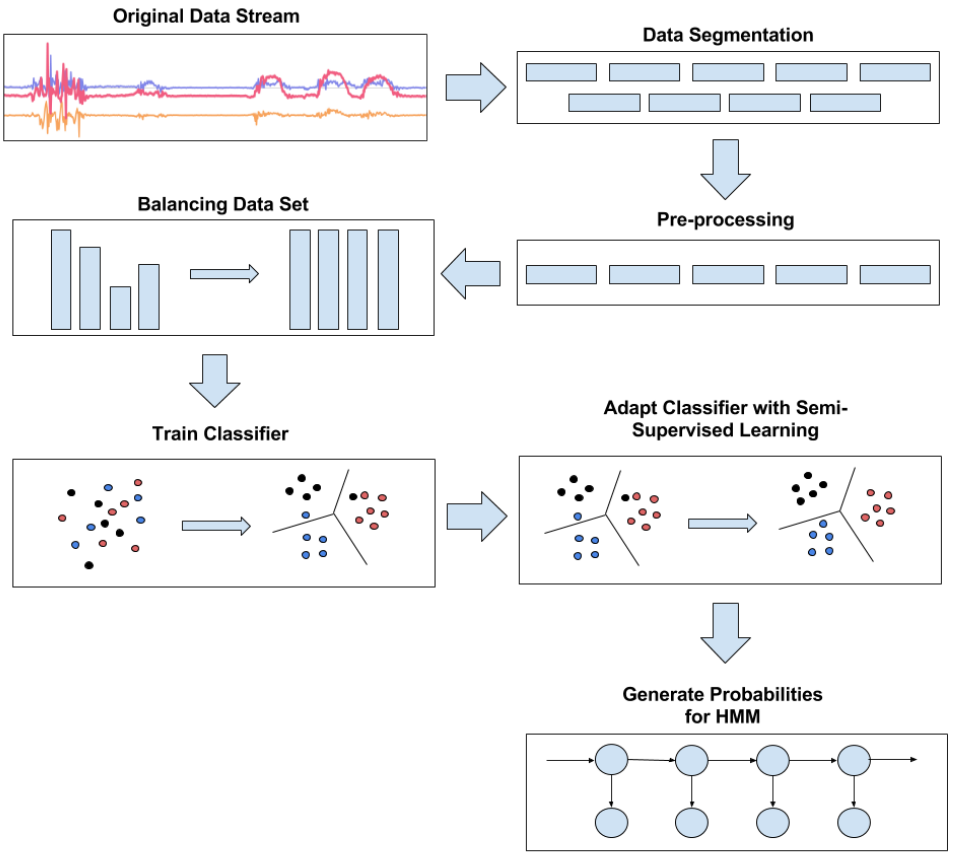


Figure 5.1: Pipeline of the system's training phase

3. **Balancing Data Set**: A skewed distribution of activities can influence the classification algorithm's ability to recognize minority activities. This problem could be addressed by balancing the data set, either by oversampling or undersampling.

4. **Train Classifier**: A baseline classification algorithm is chosen. Data windows are used to train this classifier, either as features or as raw data (depending on the classification algorithm).

5. **Adapt Classifier with Semi-Supervised Learning**: The classifier trained in step 4 is adapted to subjects, or groups of subjects, using semi-supervised

learning methods. This results in a new classification model that can capture more fine-grained movement patterns for one specific subject or group of subjects.

6. **Generate Probabilities for HMM**: To ensure temporal smoothness of activities, it is possible to combine classification results with HMM. This can be achieved by using the posterior probabilities, generated by the classifier from step 5, as emission probabilities. The initial and transition probabilities for the HMM is generated in this step.

5.3 Classification and Validation Phase

In the classification and validation phase of the system, new data instances are classified using the classification system generated in the training phase. The proposed classification and validation pipeline is illustrated in Figure 5.2. The different steps in the pipeline are described below.

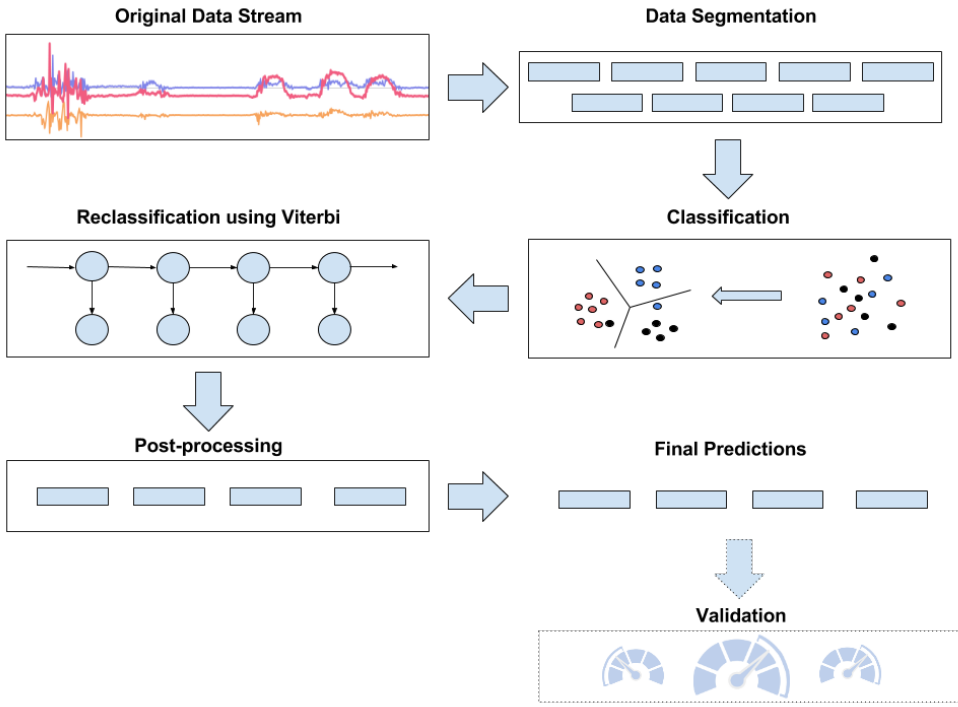


Figure 5.2: Pipeline of the system's classification and validation phase

1. **Data Segmentation**: Instead of classifying each data point, the data is segmented into data segments, called windows. This data could either be labeled test data or new unlabeled data.

2. **Classification**: Segmented windows are classified by the classifier from the training phase. Depending on the type of classifier, these windows are either represented as features or as raw data. The classifier will return a distribution of posterior probabilities, i.e. a vector for each classified instance, where each entry in the vector represents the probability for belonging to the corresponding activity.

3. **Reclassification using Viterbi**: Reclassification of the predictions provided by the classifier in step 2 by taking advantage of the temporal information. The posterior probabilities generated by the classifier in step 2 are used as emission probabilities, while transition probabilities and initial probabilities were generated in step 6 in the training phase. The Viterbi algorithm uses this HMM to calculate the most likely sequence of activities (hidden states).

4. **Post-processing**: Introduces activities that the system's classification algorithm is not trained to detect. This could either be new activities, or activities that were removed in the pre-processing step in the training phase. This could be obtained by relabeling activities in the activity sequence calculated in step 3 (e.g. *walking* less than three seconds can be relabeled as *shuffling*).

5. **Validation:** Validating and measuring the system's performance. This step is optional and requires that the labels of the classified data are known.

Chapter 6
Experiments

This chapter presents an iterative process of adding and removing components in the proposed pipeline from Chapter 5. In Section 6.2 through 6.8 we will discuss and evaluate the different steps of the pipeline. Section 6.9 presents the final HAR system that is based on the experiences we gained throughout this iterative process.

6.1 Metrics

When evaluating our system, we use four different metrics: accuracy, sensitivity, precision, and specificity. A detailed explanation of these metrics is presented below. Figure 6.1 visualize the four different outcomes of a prediction, and is useful when understanding the different metrics.

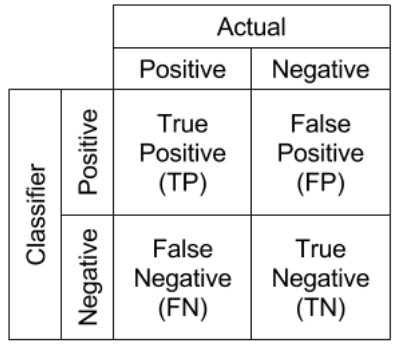


Figure 6.1: A matrix describing the four different outcomes of a prediction. *True Positive (TP)* and *True Negative (TN)* are correctly classified instances, *False Positive (FP)* is when an instance is classified as positive when it is actually negative, and *False Negative (FN)* is when an instance is classified as negative when it is actually positive.

- *Accuracy* is the portion of correctly classified instances.

$$\frac{\text{Number of correct predictions}}{\text{Number of predictions}}$$

- *Sensitivity* measures the portion of positive instances that are correctly classified.

$$\frac{TP}{TP \cap FN}$$

- Out of all instances that are classified as positives, *precision* is the portion of instances that are correctly classified.

$$\frac{TP}{FP \cap TP}$$

- *Specificity* measures the portion of negative instances that are correctly classified.

$$\frac{TN}{FP \cap TN}$$

In addition to the metrics described above, we make use of confusion matrices when evaluating our system. A confusion matrix visualizes the classification algorithms performance. The matrix rows represents the true class, while the matrix columns represents the predicted class. Figure 6.2 is an example of a confusion matrix for *sitting* and *standing*. *Sitting* is classified as *sitting* 80% of the time, while misclassified as *standing* 20% of the time. *Standing* is classified as *standing* 85% of the time, while misclassified as *sitting* 15% of the time.

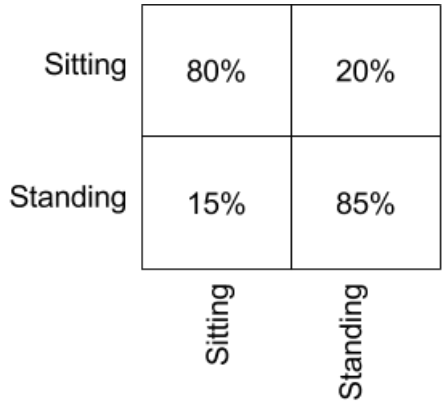


Figure 6.2: Simple confusion matrix

6.2 Data Segmentation

Signal segmentation is a crucial step in the HAR pipeline. Instead of evaluating individual data points, the data stream is divided into segments. As described in Section 2.4, there are two main methods to segment a data stream. The most frequently used method is fixed sized windows with an overlap between adjacent windows. The second method is to use dynamic windows, where the goal is to create a new window in the data stream whenever the underlying activity change. Dynamic windowing can be performed in several ways. We experimented with two different methods, first, by looking at energy changes in the signal, and second, by doing parallel classification with decision fusion. The results obtained from these methods are presented below.

6.2.1 Dynamic Windowing Based on Energy Changes

Energy changes in the signal often indicate changes in the underlying activity, and could, therefore, be used to decide where to divide the signal into windows. We calculated the energy of 3-second long data segments with 33.3% overlap. We calculated the difference in energy between adjacent (non-overlapping) segments and divided the data stream wherever the energy difference between two segments was above 0.01 for both sensors. We used the formulas below when calculating the energy.

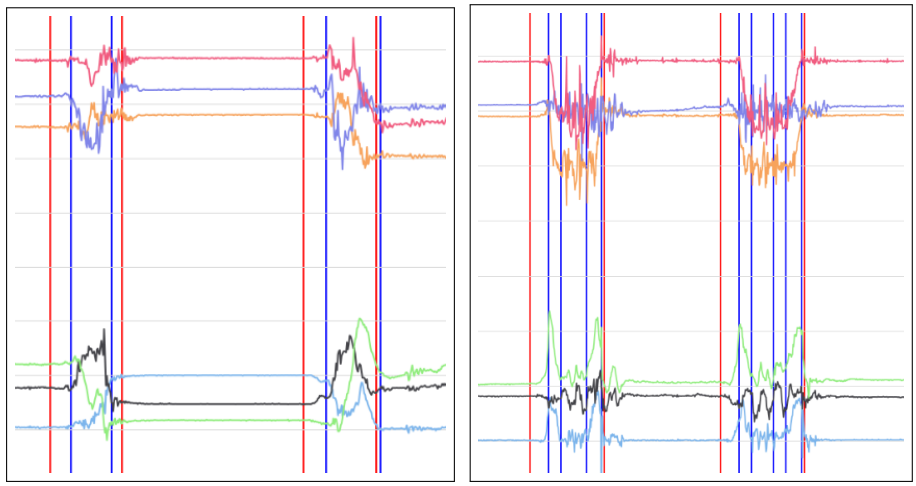
$$E_x = \sqrt{\sum_{i=1}^{N}(x_i - \mu)^2}$$

Where E_x is the energy of the x-axis, N is the length of the data segment (300 in this case), x_i is the value at position i on the x-axis signal, and μ is the mean value of the data segment. E_y and E_z are calculated likewise. The energy from each axis was combined using the formula below.

$$\mathrm{E} = \frac{1}{3N}(E_x + E_y + E_z)$$

Where E is the energy for the data segment.

This method enabled us to detect the changes between static and dynamic activities (Figure 6.3a), but could not detect changes between consecutive dynamic activities (Figure 6.3b). Dynamic activities are activities that include movement (e.g. *walking*, *running*, *cycling*), while static activities are activities with a fixed posture (e.g *lying*, *sitting*, *standing*). We tried to lower the threshold, but even though more activity



(a) Detects changes between static and dynamic activities

(b) Do not detect changes between dynamic activities

Figure 6.3: The vertical red lines are where energy changes are detected, and the vertical blue lines are where there is a change in the activity. The datastreams on the top are from the x ,y and z-axis on the thigh sensor, and the datastreams on the bottom are from the x, y and z-axis from the back sensor.

changes were detected, the number of false positives also increased. In general, this dynamic windowing approach did not work in our study.

6.2.2 Dynamic Windowing with Parallel Classification and Decision Fusion

Another dynamic windowing approach is to use different classification models for different window sizes [Banos et al., 2015]. Let each classification model classify the data, and select the final prediction based on a majority voting or highest confidence (see Figure 6.4). In majority voting, each classifier predicts a class, and the class that is predicted by most classifiers is selected. In highest confidence, each classifier predicts a class, and the prediction with the highest confidence (probability) is selected. Highest confidence is only applicable for probabilistic classifiers, where the classifier returns a vector of probabilities, and each entry in the vector represents the probability for belonging to the respective class. Using several classification models could be computationally expensive. However, in our study, we do not focus on real-time systems, and a high computational expense would not be critical.

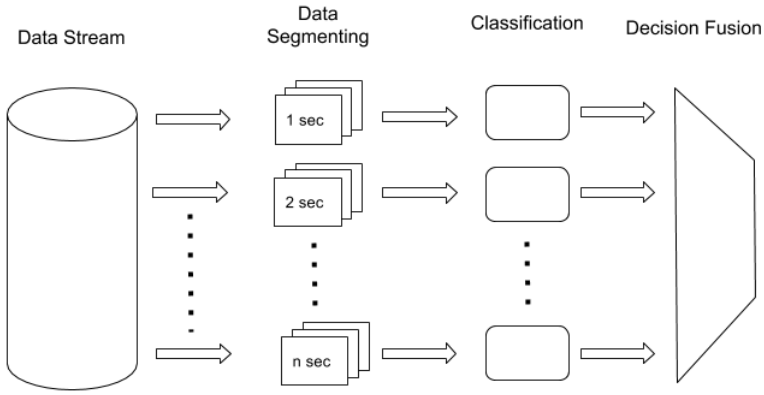


Figure 6.4: Dynamic windowing using multiple classifiers and decision fusion

When experimenting with parallel classification, we selected CNN as our baseline classifier. We created five classification models, each with different window size (0.5, 0.75, 1.0, 1.25 and 1.5 second). The data was classified by each classifier, resulting in five predictions at each data point. The prediction with the highest confidence was chosen as the final prediction for the corresponding data point.

We expected the accuracy to increase when combining the five classification models. However, the accuracy obtained using the most confident classifier turned out to be lower than the accuracy obtained using the best individual classifier. Majority voting was also tried without success. If the results from the five classifiers were combined in a different way, the accuracy could increase. However, such a combination was not found in this study.

6.2.3 Fixed Sized Windowing

Since the dynamic windowing methods did not produce satisfying results, a fixed sized windowing approach was chosen for our system. The main decisions when using fixed windows are the size of the windows and the amount of overlap between adjacent windows. We used 1-second windows as this showed promising results in our previous HAR project [Tessem and Hessen, 2015]. We experimented with 0%, 50%, and 80% overlap when training the CNN. The results obtained using different overlaps is presented in Table 6.1

80% overlap resulted in the highest accuracy and average sensitivity. A reason for this could be that a large overlap between adjacent windows results in more data instances for the classification algorithms to learn from. Another reason could be

Table 6.1: Comparing the overlap between adjecent windows

	Accuracy (%)	Average Sensitivity (%)
0% overlap	97.08	90.62
50% overlap	97.24	91.74
80% overlap	97.51	92.60

that large overlaps reduce information loss at the edges of the windows because all part of the signal will be included.

For the final system, we chose to use 1-second windows. In the training phase, we used 80% overlap between windows, as this resulted in the most accurate classifier. In the testing and classification phase, we used no overlap between windows, because no overlap results in only one prediction for each specific data point.

6.3 Pre-processing

The activities *shuffling*, *vigorous* and *transition* were removed from the data set before training. These activities were removed because they partially or completely overlapped other activities. *Shuffling* is defined as either short *walking* segments or *standing* with some leg movements, and is therefore difficult to recognize. *Vigorous* activities are defined as high-intensity activities but share characteristics that are similar to *walking* or *walking stairs*. In addition to this, *vigorous* activities consist of a limited amount of instances, making this activity difficult for the classifier to learn. *Transition* is defined as the movement between activities. Instances of this class have few similarities, and are therefore difficult to recognize. A large amount of the *transition* instances occurred when subjects shifted lying positions during the data collection process, and is therefore often recognized as *lying*. See confusion matrix in Figure 6.5 for an overview over what *shuffling*, *vigorous* and *transition* were classified as.

These overlapping activities can be reintroduced in the post-processing phase. This can be achieved by defining rules, for example, reclassify *walking*-segments with a duration below 3 seconds as *shuffling*.

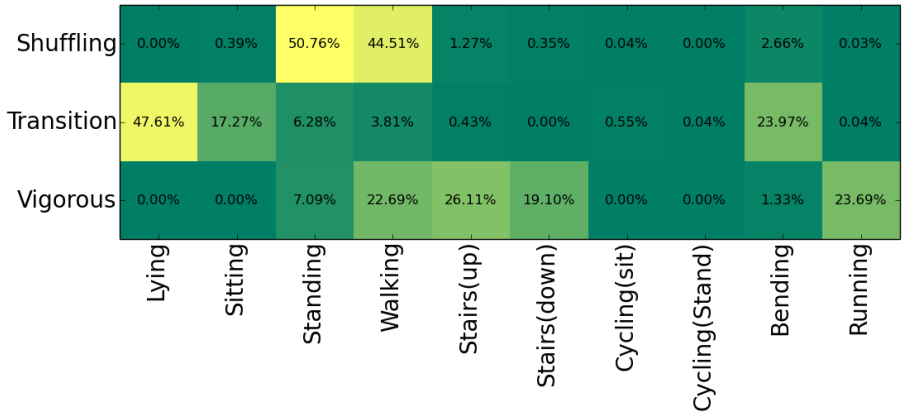


Figure 6.5: Overview of which activities *shuffling*, *transition* and *vigorous* activities are classified as

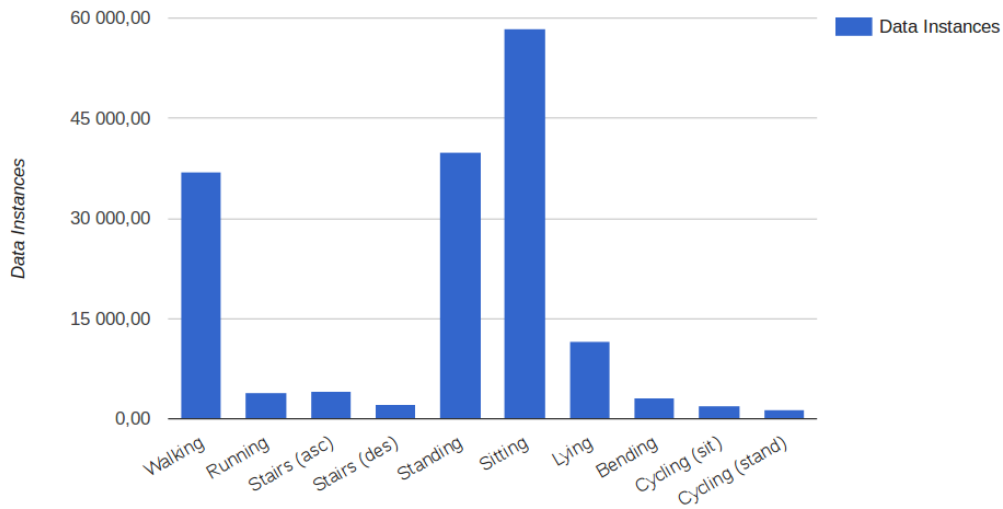


Figure 6.6: Class distribution in the adult data set

6.4 Balancing Data Set

As mentioned in Section 2.3.3, unbalanced class distributions is a common problem in machine learning. Figure 6.6 shows the distributions of the data set used in this study. *Sitting*, which is the most represented activity, has as much as 40 times more instances as *cycling(stand)*. A challenge dealing with classes consisting of few instances is that the classifier will not be able to learn this class as well as the classes consisting of more instances.

In this study, we decided to balance the data set with oversampling for two reasons. Firstly, because the smallest class only consists of 1415 instances, and undersampling would, therefore, result in a very small data set. Secondly, because removing instances in the larger classes could result in a loss of valuable information.

We trained two CNN models, one with the original data set and another with the balanced data set. They were both tested on the original, unbalanced data set. The results are listed in Table 6.2 and Table 6.3.

Table 6.2: Comparison of the sensitivity (%) for each activity for the original and balanced data set

	Original Data Set	Balanced Data Set	Difference
Walking	97.671	96.957	-0.714
Running	96.745	98.456	1.711
Stairs (up)	91.186	94.153	2.966
Stairs (down)	87.052	91.792	0.871
Standing	97.002	95.284	-1.718
Sitting	99.227	99.074	0.153
Lying	99.737	99.848	0.111
Bending	85.632	91.3	5.668
Cycling (sit)	78.554	86.205	7.651
Cycling (stand)	93.193	95.882	2.689
Average	92.6	94.895	2.295

Table 6.3: Comparison of the accuracy (%) for the original and balanced data set

	Original Data Set	Balanced Data Set	Difference
Accuracy	97.506	97.330	-0.175

The accuracy decreased slightly when training the classifier on the balanced data set, but the sensitivity of most of the activities increased by a good amount. Due to these results, we ended up using the balanced data set when training the classifier in our system.

Table 6.4: Features used when training the classifiers. The *domain* column tells whether the feature is generated for the time domain, frequency domain or both. The *axis* column tells which axis, or the combination of axes, the feature is extracted from.

Feature	Domain	Axis
Mean	Time, Frequency	x, y, z
Standard Deviation	Time, Frequency	x, y, z
Max	Time, Frequency	x, y, z
Min	Time	x, y, z
Mean-Crossing Rate	Time	x, y, z
Root Square Mean	Time	x, y, z
Median	Time, Frequency	x, y, z
Correlation	Time	xy, xz, yz
Energy	Time	xyz
Spectral Centroid	Frequency	x, y, z
Spectral Entropy	Frequency	x, y, z
Dominant Frequency	Frequency	x, y, z
DC-Angle	Time	xyz

6.5 Train Classifier

As mentioned in Chapter 2, several classification algorithms have been used in previous HAR systems. The majority of these classification algorithms are supervised learning algorithms, which are algorithms that use labeled data to create classification models that can classify unlabeled data instances. In Section 6.5.1, our baseline classification algorithm is selected by comparing traditional machine learning algorithms against a deep learning algorithm. The architecture of the selected classifier is described in Section 6.5.2.

6.5.1 Comparing Classifiers

In a previous HAR project [Tessem and Hessen, 2015] we compared different classifiers. These classification algorithms were Decision Tree (J48), Random Forest, Support Vector Machine, Artificial Neural Network and Naive Bayes classifier. All of them were trained on the features listed in Table 6.4. The *domain* column tells whether the feature is generated for the time domain, frequency domain or both. The *axis* column tells which axis, or the combination of axes, the feature is extracted from. Each feature was generated from both sensors. All features are described in Table 2.1 or Table 2.2. Since Random Forest performed best in our earlier HAR project, we considered using a Random Forest classifier in this research project as well. We were also interested in how CNN would work in HAR systems as this classifier is not dependent on handcrafted features. We ended up comparing Random Forest with CNN to decide on a final classifier for our system. Both classifiers were trained with a balanced data set.

Table 6.5: Comparison of the sensitivity (%) for each activity using a CNN and Random Forest

	CNN	Random Forest	Difference
Walking	96.957	97.003	-0.046
Running	98.456	93.791	4.664
Stairs (up)	94.152	78.494	15.657
Stairs (down)	91.791	57.934	33.857
Standing	95.283	96.529	-1.246
Sitting	99.074	99.821	-0.747
Lying	99.848	98.401	14.465
Bending	91.299	87.157	41.423
Cycling (sit)	86.204	68.411	17.793
Cycling (stand)	95.882	77.898	17.984
Average	94.895	85.544	9.350

A comparison between the two classifiers is presented in Table 6.5 and Table 6.6, where Table 6.5 presents the sensitivity for each activity, while Table 6.6 presents the accuracy. The CNN provided approximately 1% higher accuracy and 9% higher average sensitivity than Random Forest. Additionally, it is not necessary to generate handcrafted features when using CNN, which can be a drawback with Random Forest. CNN was therefore chosen as our classifier.

Table 6.6: Comparison of the accuracy (%) using a CNN and Random Forest

	CNN	Random Forest	Difference
Accuracy	97.330	96.462	0.867

6.5.2 Final Classifier - Convolutional Neural Network

When selecting the final structure of the CNN, we experimented with the size and number of convolutional layers, and the size and number of neural network layers. The results of these experiments are presented in Table B.2 and B.3 in Appendix B, and the final network architecture was based on a combination of the accuracy and average sensitivity from these experiments. The final network architecture will be presented in detail below.

Representation of Input Data

In this research, we represent the acceleration signal in two dimensions. The first dimension represents time and the second dimension represents the X, Y and Z axes of the two sensors. The size of the first dimension was 100, as the data windows were set to 1 second, and the data was collected with a sampling rate of 100 Hz. The input is illustrated in Figure 6.7, and has the dimensions 6x100.

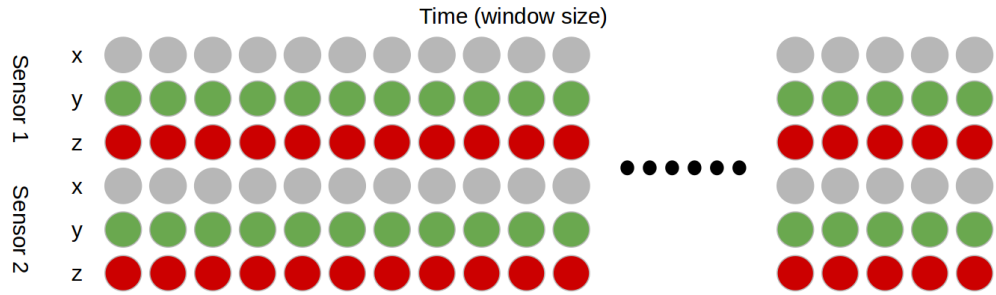


Figure 6.7: CNN input with dimension 6x100. The first dimension represents 2 sensors with 3 axes each, and the second dimension represent a 1 second window with sampling rate of 100 Hz

First Convolutional Layer
The first convolutional layer consists of 20 different kernels. These kernels are one-dimensional filters with size 1x30 that are convolved over each axis in the time domain (see Figure 6.8). The filters are convolved only within the valid input field (i.e. no zero padding), resulting in output with dimension 6x71x20 (input has dimension 6x100).

Second Convolutional Layer
The second convolutional layer consists of 40 kernels. These kernels have the same shape as in the previous layer and are convolved over the 20 feature maps produced in the previous layer. This will then again produce 40 feature maps, each with dimension 6x42.

Neural Network
The second convolutional layer is attached to a fully connected neural network with 1500 hidden nodes. The output from the second convolutional network is transformed into a one-dimensional array and connected to each neuron in the neural network. During training, we applied the dropout technique (described in Section 3.2.1) to the 1500 hidden nodes with a keep probability of 50%. Finally, the fully connected layer is connected to 10 output nodes, each representing one of the ten activities.

The CNN architecture is illustrated in Figure 6.9. Technical details can be found in Table B.1 in Appendix B.

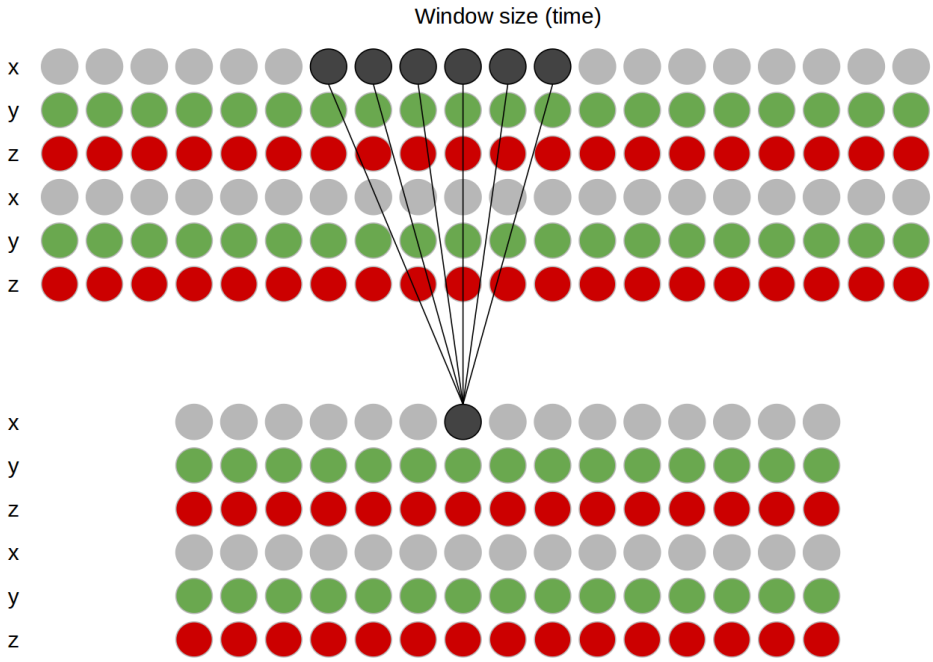


Figure 6.8: Illustation of how the CNN input is convolved with a 1x6 filter

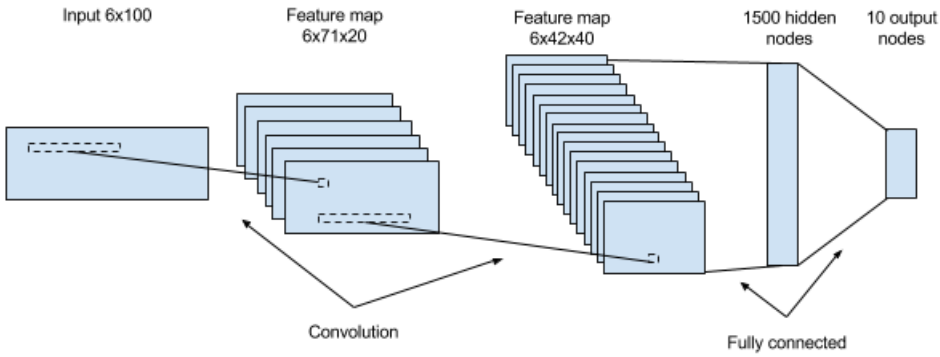


Figure 6.9: CNN architecture

We have implemented all our networks using TensorFlow [1] (Version 0.6.0). TensorFlow is an open source library for numerical computation using data flow graphs. Each node in the graphs represents mathematical operations, while the graph edges represent the multidimensional data arrays communicated between them. TensorFlow was developed by the Google Brain Team and is designed to facilitate research in machine learning. TensorFlow is used by Google in several products, such as speech recognition and search in Google Photos[2]. We chose to use this library due to the optimistic response from the machine learning community and its specialization in deep neural networks.

6.6 Adapt Classifier with Semi-Supervised Learning

In this section, we experiment with semi-supervised learning methods by adapting the classification model developed in the previous section to specific subjects. We start by describing the methods and experimental setup in Section 6.6.1, followed by Section 6.6.2, where we evaluate the results of these experiments.

6.6.1 Semi-supervised setup

A CNN was used as our baseline classification algorithm and is described in detail in Section 6.5.2. The network was first trained on 14 adult subjects where the class distribution was balanced by oversampling before training. The same baseline classification model was used for all semi-supervised experiments. The classifier returns a 10-element vector for each classified instance, where each entry in the vector represents the confidence for belonging to that specific activity.

Each experiment was performed three times, each on different subjects from the adult data set. The data for the subjects were individually divided into three equal sized sets, one for training, one for validating and one for testing. The different experiments are described below.

1. Experiment 1: The most confident instances (self-training): Classify 1000 randomly chosen instances from the test set. Select the 400 instances with the highest confidence, independent on class. Label these instances as the class with the highest confidence.

2. Experiment 2: The most confident instances for all activities, equally distributed: Classify 1000 randomly chosen instances from the test set. For all 10 activities, select the 40 most confident instances belonging to that specific class. Label these instances as the class with the highest confidence.

[1] https://www.tensorflow.org/

[2] https://www.youtube.com/watch?v=oZikw5k$_2FM$

3. Experiment 3: The least confident instances (active learning): Classify 1000 randomly chosen instances from the test set. Select the 400 instances with the lowest confidence, independent on class. Manually label these instances as the actual (correct) class.

Foe each of the three experiments, the newly labeled instances were used to continue training the baseline classifier (see Algorithm 6.1). This process was repeated in three iterations, and the learning rate and the number of training iterations were altered in each experiment. The result of each experiment is the average result obtained by running it three times, each on different subjects.

Algorithm 6.1 Self-training

```
Classifier.build()
while testData.size > 0 do
    testDataSubset = testData.subset(1000)
    for sample in testDataSubset do
        prediction = Classifier.predict(sample)
        if prediction.satisfies(criteria) then
            trainingData.add(sample, prediction)
            testData.remove(sample)
        end if
    end for
    Classifier.continueTrain(trainingData)
end while=0
```

6.6.2 Results and Discussion

When adding the most confident instances to the training set (experiment 1), the algorithm tries to fine-tune the classification algorithm to recognize these instances even better. Our baseline classifier already provides reasonable results, and for some classes, the sensitivity is already over 95%. The most confident instances will often belong to one of the classes with an already high sensitivity, resulting in only instances from a subset of the classes are selected. By continuing training the algorithm on this subset of classes, the sensitivity of the remaining classes decreased (i.e. the algorithm forgets the remaining classes). In these experiments, we observe that the accuracy has a small tendency to increase, but the average sensitivity decreases.

By selecting an equal number of instances from each class (experiment 2), the algorithm will fine tune its weights with respect to all classes, not just the ones the algorithm are already comfortable with. We observed that the average sensitivity had a tendency to increase, but not significantly. Dominating classes, such as walking, had a slightly decreasing sensitivity, resulting in a drop in the accuracy.

With active-learning (experiment 3), data instances with low confidence were automatically labeled. We observed that both the accuracy and average sensitivity increased.

Active learning resulted in the highest accuracy and average sensitivity gain in our experiments. It is intuitive that Active learning will improve its performance when fine-tuning the parts where the classifier struggles. The problem with active learning, is of course, that the subject will have to manually annotate activities which the system proposes. If implemented using a smartphone, this could be handled in an acceptable manner. Both self-training (experiment 1) and equal distribution of classes (experiment 2) did not result in a significant improvement of our system, and we will therefore not use these methods in our pipeline. These findings are supported by Guan et al. [2007], where the improvement of semi-supervised learning decreased as the amount of labeled data increased. They showed that going from an average improvement of 14.5% using 90% unlabeled data, to an average improvement of 4.82% using 50% unlabeled data. Since our system is trained on data collected from 14 subjects, the demand for labeled data is satisfied and the amount of unlabeled data is significantly lower than the amount of labeled data. In the HUNT4 study, the amount of unlabeled data will exceed the amount of labeled data, and these adaptation methods may be beneficial to reintroduce when the data is collected. See Figure B.1 through B.6 in Appendix B for more details about how these methods affected the system's accuracy and average sensitivity.

The same experiments provided approximately the same results for the adolescents data set. The reason for this is that adolescents and adults have similar movement patterns. We believed that semi-supervised learning methods would provide better results when adapting the classifier to subjects with significantly different movement patterns. We, therefore, performed the same experiments on a data set consisting of children between 6-12 years (see more details in Table A.3 in Appendix A). The classifier was first trained on labeled data from adults, and then continued trained and tested on the children data. The result of each experiment is the average result obtained by running it three times, each on different subjects.

The accuracy increased on average 3.15% when performing experiment 1. In the same experiment, the average sensitivity decreases with 3.8%. We, therefore, performed more experiments, altering the variables where these findings occurred. Instead of selecting 1000 random instances for testing, we used all available test data from each subject. After two iterations, we got an increase in accuracy of 4.59%, while the average sensitivity increased by 0.91% (Figure 6.10). These findings indicate that semi-supervised learning could be used in HAR system, and we will discuss this further in Section 8.3.5.

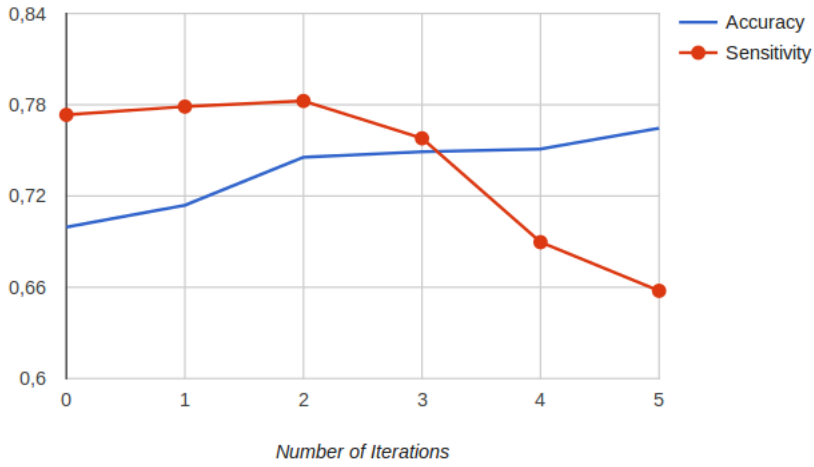


Figure 6.10: The effect of semi-supervised learning on children data

6.7 Experimenting with HMM and Viterbi

To exploit the temporal information of activities, it is possible to combine our baseline classifier with dynamic classification models. In this system, we chose to use a HMM and the Viterbi algorithm to ensure temporal smoothness of activities. When using HMMs on human activities, the hidden states are the activities performed by the subjects, while the observations and emission probabilities are calculated using the posterior probabilities provided by the baseline classifier. The posterior probabilities is a 10-element vector for each classified instance, where each entry in the vector represents the probability of belonging to a specific activity. An uniform distribution was used as our initial probabilities, while the calculation of transition probabilities are described in the section below.

6.7.1 Generating Transition Probabilities

We experimented with three different methods when calculating the transition probabilities. These are described below.

- **Method 1**: Calculated using the Baum-Welch algorithm (described in Section 3.3.3). We started with a uniform transition matrix and ran five iterations of the Baum-Welch algorithm.
- **Method 2**: Calculated by counting the number of transitions from each activity to the other activities (described in Section 3.3.2).

Table 6.7: Transition Probabilities (%). Summarizes how likely it is to change between the different activities. The rows show the activity a subject is changing from, while the columns show the activity a subject is changing to.

	Walking	Running	Stairs (up)	Stairs (down)	Standing	Sitting	Lying	Bending	Cycling (sit)	Cycling (stand)
Walking	98.679	0.079	0.337	0.233	0.589	≈ 0	≈ 0	0.084	≈ 0	≈ 0
Running	0.542	98.917	0.252	0.290	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
Stairs (up)	1.592	0.081	97.779	0.379	0.168	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
Stairs (down)	1.942	0.165	0.264	97.450	0.178	≈ 0	≈ 0	0.001	≈ 0	≈ 0
Standing	0.781	0.011	≈ 0	0.010	98.850	0.008	≈ 0	0.340	≈ 0	≈ 0
Sitting	≈ 0	≈ 0	≈ 0	≈ 0	0.007	99.755	0.050	0.127	0.0599	≈ 0
Lying	≈ 0	≈ 0	≈ 0	≈ 0	0.044	0.122	99.832	0.002	≈ 0	≈ 0
Bending	0.655	≈ 0	0.017	≈ 0	3.167	1.107	0.180	94.806	0.069	≈ 0
Cycling (sit)	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	0.919	≈ 0	≈ 0	98.068	1.013
Cycling (stand)	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	1.609	98.391

- **Method 3**: Average between method 1 and method 2.

The final transition matrix was created using method 3. A version of this transition matrix, with rounded values, is presented in Table 6.7, while the complete transition matrix is presented in Table B.4 in Appendix B. The transition matrix summarizes how likely it is to change between the different activities. The rows of the matrix are the present activity, while the columns are the future activity. For example, the probability that a subject that is walking at time t will be walking at time $t + 1$ is 98.678% (first entry in Table 6.7).

When calculating the transition probabilities, we used all data instances, including instances removed in the pre-processing step (*shuffling*, *vigorous* and *transition*). The reason for this is that HMM uses temporal information, and it is, therefore, important to include all parts of the data sequence.

Since the transition matrix is generated using the adult data set presented in Chapter 4, the matrix will not necessarily reflect the real word and how people normally shift between activities. Suggestion on how to improve the transition matrix is discussed in Section 8.3.2.

6.7.2 Reclassification using Viterbi

Table 6.8 and Table 6.9 presents the results given by the CNN, the Viterbi algorithm, and the difference between the two. The Viterbi algorithm increased the accuracy by 0.6%. The sensitivity increased for all activities except *Cycling(sit)* which decreased by 0.12%.

Figure 6.11 presents a part of the data stream where CNN alternates between classifying *cycling(sit)* as *sitting* and *cycling(sit)*, and the Viterbi algorithm reclassified the correctly classified *Cycling(sit)* as *Sitting*. We believe this is the reason for the decreased sensitivity for *Cycling(sit)*. However, in most cases the Viterbi algorithm lead to increased results, and one of these cases are depicted in Figure 6.12.

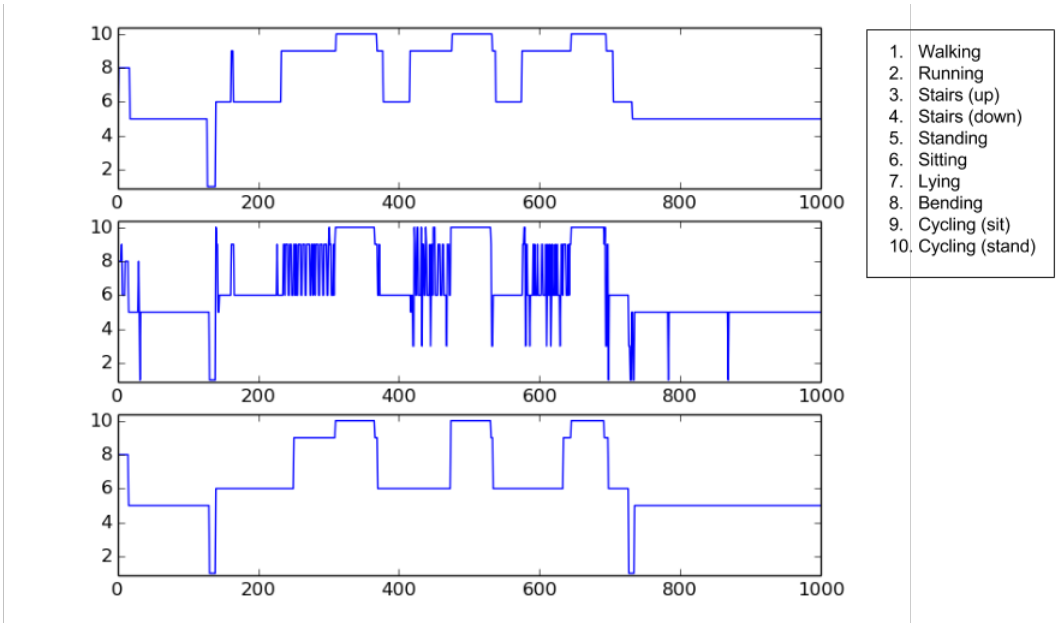


Figure 6.11: Example where Viterbi decrease the result produced by the CNN. The top graph presents the real activity, the middle graph presents the CNN result, and the bottom graph presents the Viterbi results.

Table 6.8: Accuracy (%) after reclassification with Viterbi algorithm on the adult data set

	CNN	Viterbi	Difference
Accuracy	97.325	97.928	0.603

Table 6.9: Sensitivity (%) for each activity after reclassification with the Viterbi algorithm on the adult data set

	CNN	Viterbi	Difference
Walking	96.958	98.260	1.302
Running	98.456	98.624	0.168
Stairs (up)	93.968	94.758	0.789
Stairs (down)	91.748	92.960	0.212
Standing	95.284	95.956	0.672
Sitting	99.074	99.238	0.163
Lying	99.848	99.980	0.132
Bending	91.306	94.625	3.319
Cycling (sit)	86.101	85.981	-0.120
Cycling (stand)	95.913	96.747	0.834

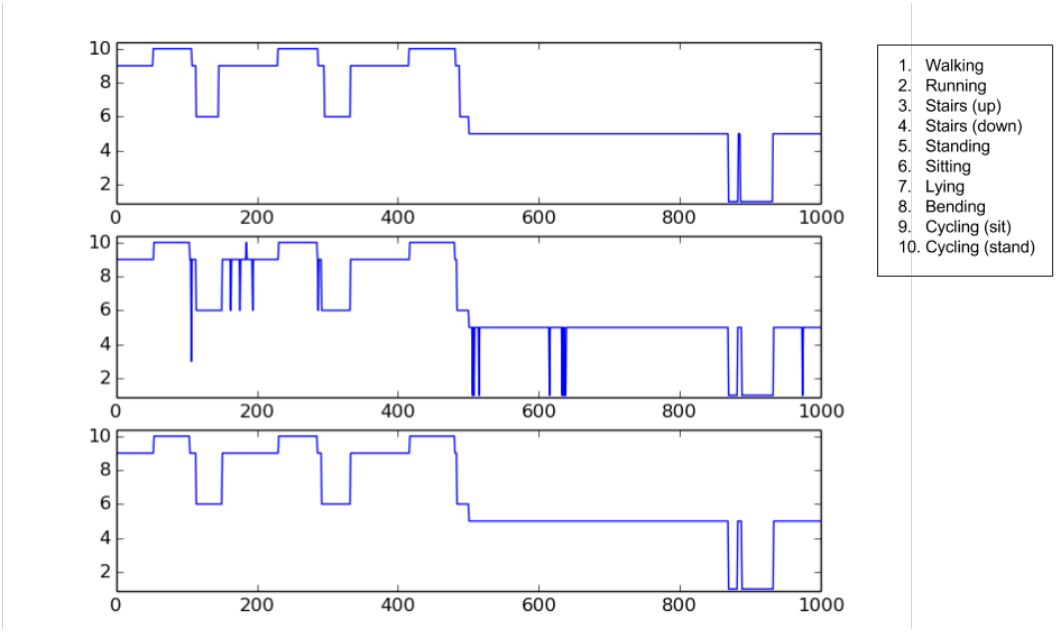


Figure 6.12: Example where Viterbi increase the result produced by the CNN. The top graph presents the real activity, the middle graph presents the predictions made by the CNN, and the bottom graph presents the Viterbi results.

We performed the same experiments with the adolescents data set. These results are presented in Table 6.11. The sensitivity of all activities increased, and the accuracy increased by 1.7%.

The Viterbi algorithm gave positive results for both data from adults and adolescent, and was therefore chosen to be included in our final HAR system.

Table 6.10: Accuracy (%) after reclassification with the Viterbi algorithm on the adolescent data set

	CNN	Viterbi	Difference
Accuracy	94.908	96.622	1.714

Table 6.11: Sensitivity (%) for each activity after reclassification with the Viterbi algorithm on the adolescent data set

	CNN	Viterbi	Difference
Walking	95.059	97.383	2.324
Running	94.507	95.836	1.329
Stairs (up)	91.559	94.651	3.092
Stairs (down)	76.637	82.280	5.643
Standing	94.397	95.091	0.694
Sitting	97.901	98.666	0.765
Lying	100.0	100.0	0.0
Bending	79.512	80.976	1.463

6.8 Post-processing

Since we removed several activities in the pipeline's pre-processing step, we will discuss how we can reintroduce them in this section. These activities were *transition*, *shuffling* and *vigorous*.

Transition is defined as the movement between certain activities (e.g. movement between *standing* and *sitting*). This activity is important to include in the labeling process, as we do not want the other activities to include these movements. (e.g. *sitting* and *standing* should only contain their respective activities, not the movement between them). As mentioned earlier, *transition* was removed in the pre-processing step since it does not contain any specific movement patterns, and would, therefore, be difficult for a classifier to learn. We could use hand-crafted rules to reintroduce *transition*, for example by relabel segments between *sitting* and *standing* as *transition*. However, from a medical perspective, *transitions* are not an important aspect of a subject's health, and it was therefore excluded from all further analyses.

After talking with the medical staff at NTNU, we concluded that *shuffling* could be redefined as *walking* segments with a duration below three seconds. However, this definition did not fit our data set, as 50% of all activity labeled as walking had a duration below three seconds.

Vigorous activities is a composition of several short duration high-intensity activities. An example of this could be playing soccer. *Vigorous* activities could be detected in a post-processing step by detecting segments with a longer duration that consist of a variation of several short duration high-intensity activities, and relabel this as *vigorous*. However, the data labeled as *vigorous* activities in our data set were not always high-intensity (see Figure 6.5).

Clear definitions are needed to enable rule-based reintroduction of activities. The definitions for *shuffling* and *vigorous* are vague, making the labeling process of these activities inconsistent. Therefore, we did not find any good way to reintroduce these activities, and the post-processing step was not included in our system.

6.9 Final HAR system

This section summarizes the training phase and classification and validation phase of our final HAR system. The decisions regarding the system design were made due to the experiments described earlier in this chapter and the arguments put forth there.

The system's classification part was learned using the adult data set. We divided the data set into two parts, 2/3 for training, and 1/3 for testing.

6.9.1 Training Phase

The pipeline for the final training phase of the system is presented in Figure 6.13. Each step of the pipeline is described below.

1. **Data Segmentation**: We chose 1 second windows with 80% overlap.
2. **Pre-processing**: We removed instances with *Transition*, *Shuffling* and *Vigorous* activities.
3. **Balancing Data Set**: The training data were balanced using oversampling.
4. **Train Classifier**: We chose to use a CNN with an architecture as described in Section 6.5.2.
5. **Generate Probabilities for HMM**: The initial probabilities are uniformly distributed. The posterior probabilities returned by the CNN were used as emission probabilities. The transition probabilities were generated using Baum-Welch and are presented in Table 6.7

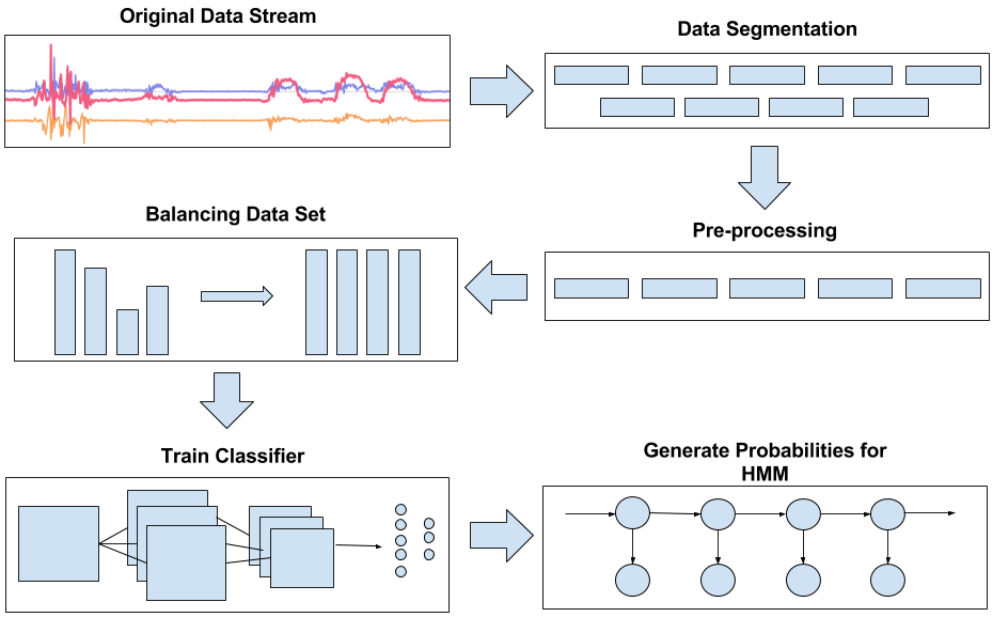


Figure 6.13: Final training pipeline

6.9.2 Classification and Validation Phase

The pipeline for the final classification and validation phase of the system is presented in Figure 6.14. Each step of the pipeline is described below.

1. **Data Segmentation**: We chose to use 1-second windows without overlap.
2. **Classification**: The system classifies data instances using the CNN trained in the training phase.
3. **Reclassification using Viterbi**: The classification results from step 2 were re-evaluated by the Viterbi algorithm, producing the final classification of the activities. The posterior probabilities generated by the classifier are used as emission probabilities, while transition probabilities and initial probabilities were generated in step 6 in the training phase.
4. **Validation:** Validating and measuring the system's performance using sensitivity, precision, specificity, and accuracy. This step is optional and requires that the labels of the classified data are known.

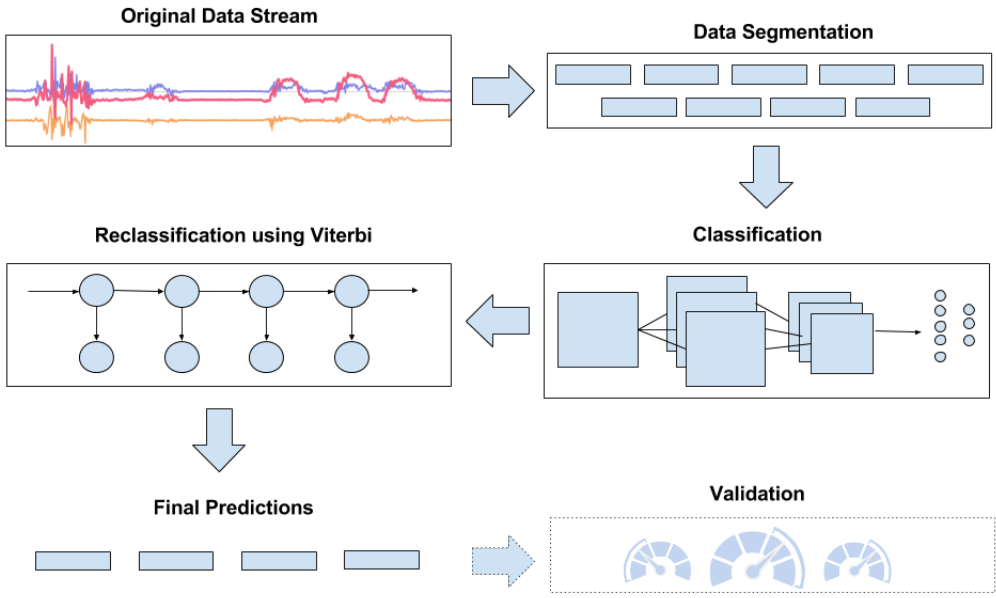


Figure 6.14: Final classification and validation pipeline

Chapter 7
Results and Discussion

This chapter presents the final results of the proposed HAR system. As mentioned in Chapter 6, the system was trained using the adult data set, however, we tested the system using data from both adults and adolescents. The results using the adult and adolescent data sets will be discussed in Section 7.1 and Section 7.2 respectively. Section 7.3 presents a comparison between the state-of-the-art HAR system Acti4 (described in Section 2.7) and our proposed HAR system.

7.1 Adults

The accuracy for the adults data set is 97.9%. Table 7.1 presents recall, precision, and specificity for the different activities. The confusion matrix for adults is presented in Figure 7.1.

Overall, the HAR system produces satisfying results for the adult data set. Looking at the results in Table 7.1, only two table values have a score lower than 90%. These are the sensitivity of *cycling(sit)* (85.981%) and the precision of *bending* (86.682%).

The sensitivity of *cycling(sit)* measures the portion of *cycling(sit)* instances that are correctly classified. The confusion matrix for adult data (Figure 7.1) shows that

Table 7.1: Measurements of the system's performance on adult data set

	Sensitivity(%)	Precision(%)	Specificity(%)
Lying	99.980	100.000	100.000
Sitting	99.238	99.161	99.417
Standing	95.956	97.768	99.423
Walking	98.260	96.830	99.124
Stairs (up)	94.758	95.566	99.887
Stairs (down)	92.960	96.990	99.964
Cycling (sit)	85.981	91.019	99.900
Cycling (stand)	96.747	98.976	99.991
Bending	94.625	86.682	99.711
Running	98.624	97.836	99.953

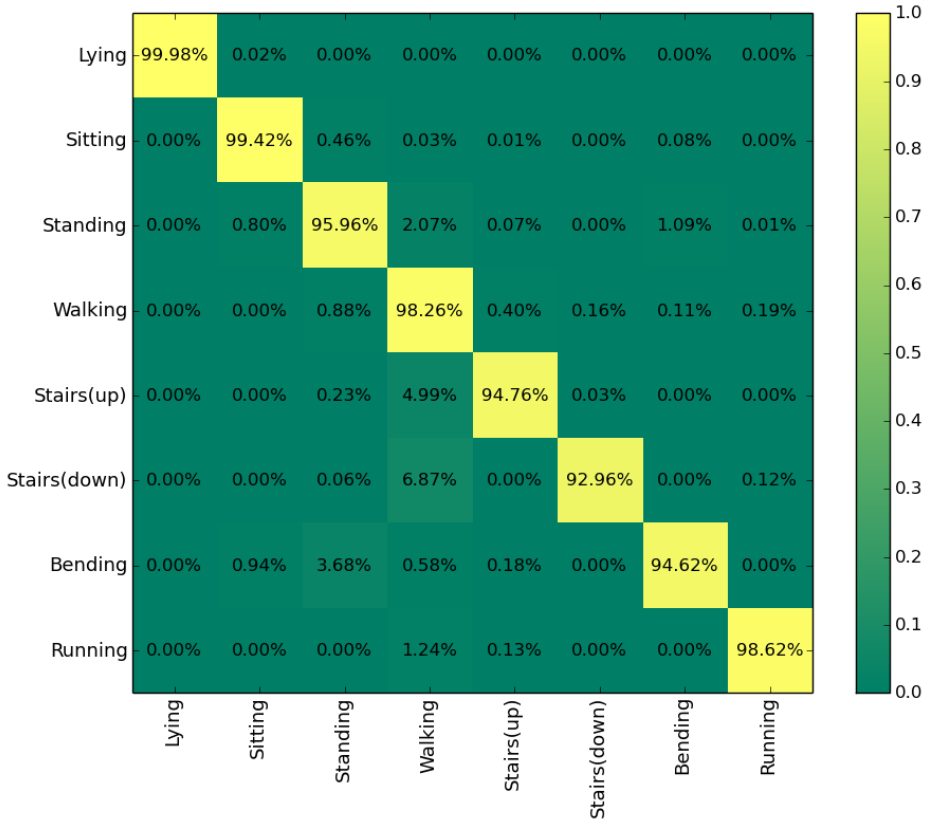


Figure 7.1: Confusion matrix for the adult data set

13.5% of *cycling(sit)* instances are wrongly classified as *sitting*. A reason for this is that *cycling(sit)* and *sitting* are very similar. This was especially the case for one of the subjects (see Figure 6.11), where the CNN decreased the average sensitivity by alternating between classifying *cycling(sit)* as *sitting* and *cycling(sit)*. The Viterbi algorithm decreased the results even more by reclassifying the correctly classified *cycling(sit)* instances as *sitting*. Since there are few instances of *cycling(sit)* in the test data (1.2% of the entire test data set), these wrongly classified instances influenced the specificity of *cycling(sit)* a great deal.

Out of all instances that are classified as *bending*, the precision is the portion that is correctly classified as *bending*. In the adult confusion matrix (Figure 7.1), we see that 1% out of all *standing* instances are wrongly classified as *bending*. The skewed distribution in the test data affects the results. Since *standing* represents

Table 7.2: Measurements of the system's performance on adolescents data set

	Sensitivity(%)	Precision(%)	Specificity(%)
Lying	100.000	97.324	99.954
Sitting	98.666	98.320	99.877
Standing	95.091	95.983	98.410
Walking	97.383	97.334	96.426
Stairs (up)	94.651	92.790	99.874
Stairs (down)	82.280	76.576	99.842
Bending	80.976	76.498	99.928
Running	95.836	93.739	99.790

around 24.4% of all instances in the test data, the 1% misclassification of *standing* equals 13.6% of the total amount of *bending* instances, and will therefore affect the precision of *bending*. On the other hand, 3.68% of *bending* instances are misclassified as *standing*, but will not affect the precision of *standing* to the same extent, due to the skewed test set. *Bending* is typically surrounded by *standing*, and data windows that are labeled as either of these activities have a chance of containing portions of the other activity (i.e. the transition between these activities may be less than a second long and thus containing portions of both activities), and are likely to be the main reason for the misclassifications between them.

The confusion matrix shows that *stairs(up)* is misclassified as *walking* 4.99% of the time, and *stairs(down)* is misclassified as *walking* 6.87% of the time. This is not very surprising, since these activities share many of the same movements. However, *stairs(up)* and *stairs(down)* have a low probability of being misclassified as each other (0.3% at the highest). The reason behind this can be that both *stairs(up)* and *stairs(down)* are more similar to *walking* than to each other.

7.2 Adolescents

The accuracy for adolescents data set is 96.6%, only 1.3% lower than the adult data set. Table 7.2 presents recall, precision, and specificity for the different activities. The confusion matrix for adolescents is presented in Figure 7.2.

Even though the HAR system is trained on the adult data set, the system provides promising results on data from adolescents. Looking at the results in Table 7.2, only two activities, *stairs(down)* and *bending*, have scores lower than 90%.

Bending is misclassified as *sitting* 13.17% of the time. This may be because adolescents tend to bend their knees when *bending* more often than adults do, resulting in a position that is very similar to *sitting*. Opposed to the adult data set, where *bending* was mainly misclassified as *standing*, this part of the error seems to have shifted over to *walking*. *Bending* is misclassified as *walking* 5.61% of the time. This is probably because the adolescents data set contains more transition between *bending*

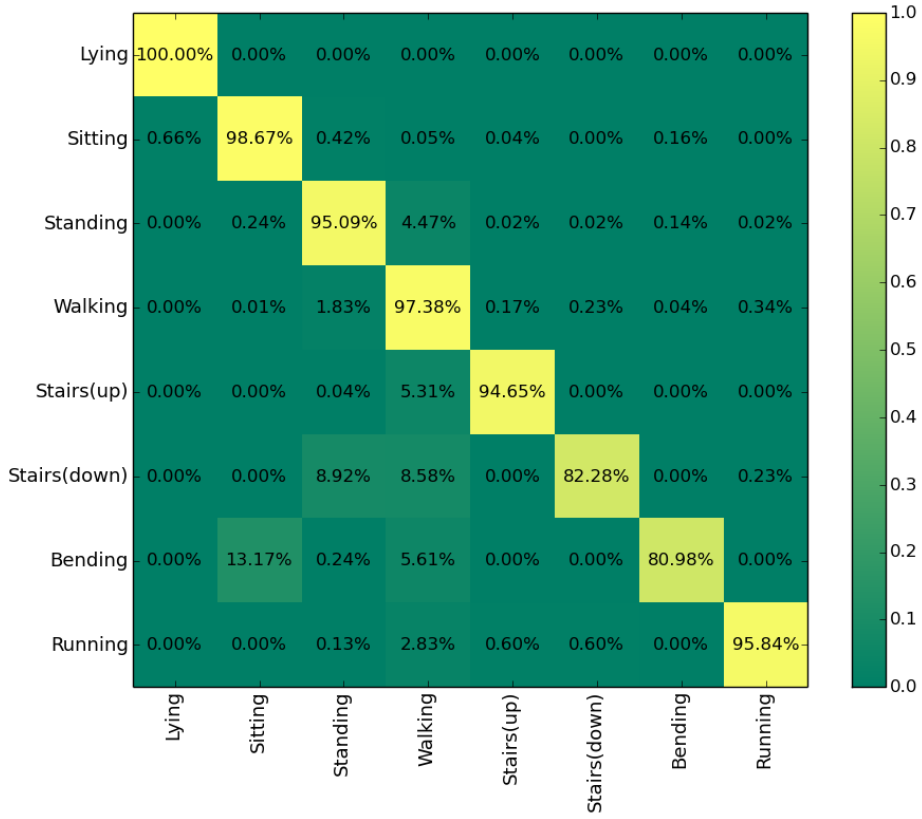


Figure 7.2: Confusion matrix for adolescents data set

and *walking* than between *bending* and *standing*, and therefore windows that were labeled as *bending* could include portions of *walking*.

Stairs(down) is misclassified as *standing* 8.92% of the time and *walking* 8.58% of the time. *Stairs(down)* and *walking* are similar activities, and is a misclassification we also see when classifying adult data. During the data collection process for adolescents, there were many transitions between a standing position and *stairs(down)*. This can result in data windows containing both activities, which could be the reason for the misclassification between *Stairs(down)* and *standing*.

The confusion matrix shows that *running* and *stairs(up)* are often misclassified as *walking*, which is not surprising as these activities are similar. More surprising is it, that *standing* is misclassified as *walking* 4.47% of the time. We believe the reason

behind this is that *standing* and *walking* often occurs after each other, so that data windows that are labeled either of these activities have a chance of containing portions of the other activity, which could lead to misclassification.

7.3 Comparing with Acti4

As mentioned in Section 2.7, Acti4 is able to recognize *walking*, *running*, *stairs*, *standing*, *sitting*, *lying*, *cycling* and *moving*. *Moving* is defined as periods that match a standing posture that includes small movements without ordinary walking. Since we do not have the activity *moving* in our HAR system, we will not evaluate it. In addition to classifying the remaining Acti4 activities, our system can classify *bending*, distinguish between walking up and down stairs (*stair(up)*, *stairs(down)*), and between cycling in a seated position versus standing position (*cycling(sit)*, *cycling(stand)*). Figure 7.3 presents the confusion matrix generated when classifying the adult data set with Acti4, while Table 7.3 presents a comparison between the our system and Acti4.

Table 7.3: Comparing sensitivity (%) between Acti4 and the proposed HAR system

	Acti4	Proposed HAR system	Difference
Walking	94.76	98.26	3.5
Running	94.11	98.62	4.51
Stairs	76.87	93.86	16.99
Standing	91.48	95.96	4.48
Sitting	98.13	99.24	1.11
Lying	99.00	99.98	0.98
Cycling	37.25	91.37	54.12

When classifying our data with Acti4, we obtain an of 88.85%. This is almost 10% lower than the accuracy obtained when classifying with the system generated in this research. One of the main reasons for this may be that Acti4 recognizes activities using a fixed decision tree (see Figure 2.7), which could make it fragile when used on a slightly different data set. Acti4 was generated based on data obtained from sensors on the subject's thigh and hip, while our data set is collected from sensors on the subject's thigh and back. This could be a reason for the low accuracy obtained by Acti4. An advantage of our system is that it contains a training phase, making it adaptable to new types of data (e.g. new sensor locations or subjects from other age groups).

Figure 7.3: Confusion matrix for adults using the Acti4 system

Chapter 8
Conclusion and Future Work

Section 8.1 presents the conclusions drawn from the results presented and discussed in Chapter 6 and Chapter 7, while Section 8.2 presents our research's contributions. Throughout this research period, we have gathered ideas on possible improvements that could lead to better results, these will be presented in Section 8.3.

8.1 Conclusion

As a result of our research, we have developed a Human Activity Recognition system that can be used in the upcoming HUNT4 study. The system can distinguish between 10 different daily activities based on raw acceleration data captured from two accelerometer sensors located on a subject's thigh and back. To improve state-of-the-art HAR systems, we have used a both deep learning and dynamic classification, reaching an accuracy of 97.9% for the adult data set and 96.6% for the adolescent data set. These results are almost 10% higher than the accuracy obtained by the state-of-the-art HAR system Acti4.

Segmenting data windows using a dynamic windowing approach did not improve our system's ability to distinguish between activities, and a fixed sizes windowing approach was therefore used. However, fixed size windowing could affect the results for activities with short duration (e.g. *bending* and *stairs*), since a significant portion of their instances will contain data from more than one activity.

Deep learning algorithms have a potential in the field of HAR, with its automatic feature generation and classification of raw data. Our Convolutional Neural Network produced promising results, as it performed better than Random Forest in both accuracy and average sensitivity (Section 6.5.1). The Convolutional Neural Network architecture have still many properties and aspects that can be further explored. These will be discussed in Section 8.3.2.

Dynamic classification, using Hidden Markov model and the Viterbi algorithm, worked to a certain extent. For some activity sequences, the algorithm improved the predictions, while for others sequences, the algorithm failed. However, the Viterbi algorithm improved the accuracy and is therefore included in the final HAR system. The Viterbi algorithm transition matrix has potential for improvements, and will be discussed in Section 8.3.2.

Adapting the classification model for individual adults and adolescents using semi-supervised learning methods did not improve our HAR system's performance. However, adapting the classification model to individual children showed promising results, where the accuracy increased 4.59%, and average sensitivity increased 0.91%. These findings suggest that subjects with activity patterns that differs from the majority group may benefit from semi-supervised learning methods. Active learning did improve the results for adults, adolescents, and children, but will not be applicable in HUNT4, due to the continuous labeling process.

8.2 Contributions

Based on our research goals and questions presented in Chapter 1, we can summarize our contributions in the following way:

Our first, and main contribution is a HAR system that can distinguish between 10 different activities with an accuracy of 97.9%. The system is open sourced[1], both to allow other researchers to get a deeper understanding of our work, and to enable others to benefit from a HAR system of sufficient quality. To be able to develop this system, we needed an overview of the typical components in state-of-the-art HAR system. Our research's second contribution has therefore been an architectural overview of state-of-the-art HAR systems. This overview is presented in Chapter 2. Based on our second research goal, our research provided insight into how different methods, and a combination of methods, influence HAR systems performance. These methods include dynamic windowing, deep learning, dynamic classification and semi-supervised learning. By implementing the HAR system iteratively, we obtained insight on how different methods interplay.

[1] https://github.com/hessenh/HAR-Pipeline

8.3 Future Work

This section presents possible improvements of our proposed HAR system, followed by some future research directions that could be the next steps along the path to improved HAR systems.

8.3.1 Data set and protocol improvements

As discussed earlier, some of the activities in the data set have a very limited amount of instances. In combination with this, these activities are often performed over a short duration. As a result, a large portion of these data windows contains other activities. It could, therefore, be beneficial to have longer segments of activities in the data recording protocol. This would increase the amount of data for each activity, and decrease the portion of data that include multiple activities. Having short segments of activities may also affect the subject's ability to perform the activity naturally (e.g. *walking* for 3 seconds may not always be enough to reflect *walking*).

In this research, we are segmenting the raw data into windows in the same order as the data was recorded, resulting in data windows that could contain more than one activity (unclean windows). To remove unclean windows, the raw data could first be separated into individual activity pools based on their label, and then divided into data windows. A second approach could be to remove each window with a purity lower than a given threshold (e.g. remove a window that contains 40% *walking* and 60% *standing*), but this would then again reduce the number of instances in the data set.

The proposed HAR system were not able to classify *shuffling* and *vigorous*. The definitions of these activities are presented below.

- *Shuffling*: Stepping in place by non-cyclical and non-directional movement of the feet. Includes turning on the spot with feet movement not as part of walking bout.
- *Vigorous*: All non-cyclic rapid leg movements that do not classify as running. This includes sport like activities such as rapid change in direction and jumping. Can occur in all directions

These definitions are vague, which could result in subjective labeling. The activities are also similar to other activities in the data set (e.g. *shuffling* is very similar to both *standing* and *walking*). If these definitions are clearer, it may be easier to recognize these activities. An example of a clearer definition is "all *walking* with a duration shorter than 2 seconds is defined as *shuffling*".

8.3.2 System improvements

Our selected CNN architecture was chosen after experimenting with the number of kernels in each convolutional layer, the number of convolutional layers, the number of fully connected layers and the kernel size. Since training and testing a CNN architecture is time-consuming, we did not have the time or the resources to test every possible CNN configuration systematically. The following aspects of the CNN architecture would, therefore, be interesting to explore further:

- Input representation
- Kernel dimensions
- Number of kernels in each convolutional layer
- Number of convolutional layers
- Number of nodes in each neural network layer
- Number of neural network layers

The transition matrix used in the Viterbi algorithm have potential for improvements. As mentioned, this matrix consists of probabilities on of how often people shift between the different activities. In this system, these probabilities were calculated based on the sequence of activities in the data collection protocol, even though this protocol does not reflect how people normally shift between activities. A better way to calculate transition probabilities would be to either make use of expert knowledge or by observing how people normally shift between activities.

8.3.3 Reccurent Neural Network

In this research, we use a CNN as our baseline classification algorithm. A shortcoming with CNN is that they do not make use of the temporal information of the data stream. Data windows are seen as independent events, even though this is not the case with human activities (e.g. if a subject was running for the last 10 seconds, the probability of it running the next second is increased). Recurrent Neural Network (RNN) make use of this temporal information, and may, therefore, be more suitable for HAR systems than a CNN.

Traditional RNN uses temporal information by mapping the input sequences to a sequence of hidden states, and the sequence of hidden states to the output sequence. The hidden layers are recurrently connected to itself [Mikolov et al., 2010]. Figure 8.1 demonstrates the structure of a traditional RNN.

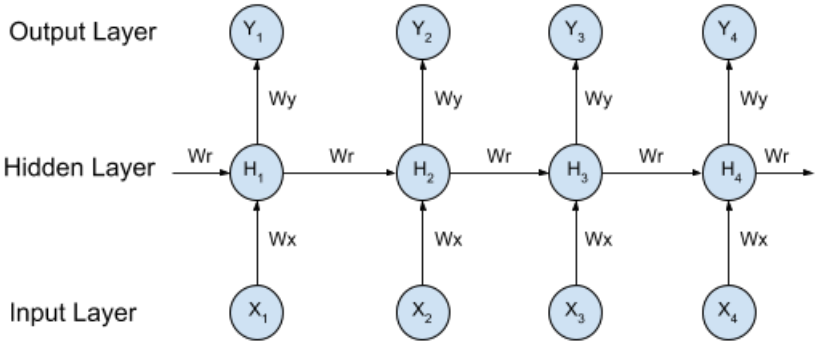


Figure 8.1: Reccurent Neural Network. X_t is the input unit at timestep t, H_t is the hidden unit at timestep t, Y_t is the final prediction at timestep t. W are the weights between the different units.

RNN has already provided promising results for activity recognition in smart homes [Fang et al., 2013], in video-based activity recognition [Baccouche et al., 2011, Donahue et al., 2015], and very recently in wearable sensor-based activity recognition [Ordóñez and Roggen, 2016].

8.3.4 Sleep Monitoring

According to Centers for Disease Control and Prevention (CDC), insufficient sleep is a public health problem[2]. Risk of several chronic diseases, such as depression, diabetes, stroke, and obesity, is associated with sleep problems [Altevogt et al., 2006]. According to Altevogt et al. [2006], 50-70 million of adults in the USA have some type of sleeping disorders.

Sleep tracking can provide valuable information about people's sleep, such as sleep cycles. There exist several sleep tracking products, however, most of these systems do not track activities during the day. The use of accelerometers alone may not be sufficient to measure overall sleep quality, but can provide valuable information about the subject's sleep-wake cycle.

The use of several sensors could provide a more accurate characterization of a subject's sleep. However, sleeping with multiple sensors can interfere with the subject's normal sleep. It is, therefore, like in HAR, important to minimize the interference of sensors while maximizing the amount of information provided by the sensors. Examples of sensors that has been used to track sleep are listed below.

[2]http://www.cdc.gov/features/dssleep/

- Accelerometer - Monitors body movement.
- Electroencephalogram (EEG) - Electrodes attached to the scalp (often a headband) to detect brain waves and activity. This sensor is the gold standard for measuring sleep stages and sleep-wake cycle.
- Electrooculogram (EOG) - Electrodes attached above and below the eye to detect eye movement. This sensor can give an indication of when a subject is in REM (rapid eye movement) sleep.
- Electrocardiogram (ECG) - Electrodes attached to the chest to measure the heart's electrical activity.
- Heart Rate Monitor - Measure of subject's pulse.
- Microphone – Record frequency and volume of snoring.

When constructing a system that combines sleep tracking and activity recognition, the best would be to use sensors that are valuable in both processes. Accelerometers and heart rate sensors have shown to be useful in both HAR systems and sleep tracking system. Microphones are often used in sleep tracking systems to record frequency and volume of snoring. A microphone would not be valuable in activity recognition. However smartphones are often used as sensors in HAR systems, and since these phones include microphones, microphones could be easily added as a sensor. Instead of using the same sensors during sleep tracking and activity recognition, adding extra sensors or removing unnecessary sensors before sleep could be a possibility. EEG are very accurate when monitoring sleep stages and the sleep-wake phases. This sensor would not make sense in regular HAR systems, but could be added before going to sleep.

8.3.5 Adapting classification models to specific subjects using Semi-Supervised learning

When experimenting with semi-supervised learning in this research, we tried to adapt our classification model to work better for individual adults and adolescents. As discussed in Section 6.6.2, we experienced limited improvement with this approach. However, we obtained promising results when adapting the classification model to children. This made us believe that semi-supervised learning methods can be valuable whenever the subject's movement pattern differs from the group of subjects the classifier was trained on. An idea could, therefore, be to use these methods for subjects with movement disorders (e.g. cerebral palsy which can result in poor muscle tone, uncontrolled movements, and tremors). Since people with cerebral palsy often have a varying degree of the disorder, collecting a data set that reflects every

subject could be difficult. It can, therefore, be beneficial to develop a general HAR system, and then adapt this system to each specific subject.

Chapter A
Appendix

This appendix contain information about each subject in the data set and the definition of the activities used in the system.

A.1 Subject Information

Table A.1: Adolescents data set statistics

Subject	Gender	Age	Height	Weight
1	M	15	179	62.9
2	M	15	168	52.2
3	M	15	167	71.2
4	F	16	172	56.5
5	F	15	169	60.2
6	F	16	171.2	67.8
7	M	13	173	68.5
8	M	13	161	44
9	M	13	160	44
10	F	15	159	56.4
11	F	15	169	65.3
12	F	15	160	50.2

Table A.2: Adult data set statistics

Subject	Gender	Age	Height	Weight
1	F	40	176	68.5
2	F	38	171	64.5
3	M	30	191	84.5
4	F	38	169	59.1
5	M	42	182.5	84.3
6	M	46	190	82
7	F	35	172.5	64.9
8	M	36	166.7	74
9	F	47	173.5	74.8
10	M	34	193	99.9
11	M	44	198	98.6
12	F	36	161	62.2
13	F	36	173.7	57.9
14	F	48	178	78.9
15	F	46	155	49.5
16	F	49	171	63.8
17	F	39	164.5	76.9
18	M	28	185	84.5
19	F	52	163.3	79.4
20	F	37	162.5	61.1
21	F	45	172.7	68.5
22	M	34	186.2	78.2
23	M	32	198	101.8

Table A.3: Children data set statistics

	Subjects	Average Age	Average Height	Average Weight
Boys	7	9.7 years	139.8 cm	32.7 kg
Girls	8	9.3 years	136.6 cm	34.3 kg
Total	15	9.5 years	138.1 cm	33.5 kg

A.2 Activity Definitions

Table A.4: Activity definitions

Activities	Description
Sitting	When the person's buttocks is on the seat of the chair, bed or floor. Sitting can include some movement in the upper body and legs; this should not be tagged as a separate transition. Adjustment of sitting position is allowed.
Standing	Upright, feet supporting the person's body weight, with no feet movement, otherwise this could be shuffling/walking. Movement of upper body and arms is allowed until forward tilt and arm movement occurs below knee height. Then this should be inferred as bending.For chest mounted camera: If feet position is equal before and after upper body movement, standing can be inferred. Without being able to see the feet, if upper body and surroundings indicate no feet movement, standing can be inferred.
Walking	Locomotion towards a destination with one stride or more, (one step with both feet, where one foot is placed at the other side of the other). Walking could occur in all directions. Walking along a curved line is allowed.
Shuffling	Stepping in place by non-cyclical and non-directional movement of the feet. Includes turning on the spot with feet movement not as part of walking bout.For chest mounted camera: Without being able to see the feet, if movement of the upper body and surroundings indicate non-directional feet movement, shuffling can be inferred.
Stair ascending/descending	Start: Heel-off of the foot that will land on the first step of the stairs. End: When the heel-strike of the last foot is placed on flat ground. If both feet rests at the same step with no feet movement, standing should be inferred.
Lying down	The person lies down. Adjustment after lying down is allowed if it does not lead to a change between the prone, supine, right and left lying positions. Movement of arms and head is allowed. Movement of the feet is allowed as long as it does not lead to change in posture. Prone: On the stomach. Supine: On the back. Right side: On right shoulder. Left side: On left shoulder.
Sit cycling	Pedaling while the buttocks is placed at the seat. Cycling starts on first pedaling and finishes when pedaling ends. For outdoor bicycling: Cycling starts at first pedaling, or when both feet have left the ground. Cycling ends when the first foot is in contact with the ground. Not pedaling: Sitting without pedaling should be tagged separate as sitting.
Stand cycling	Pedaling while standing. Cycling starts on first pedaling and finishes when pedaling ends. Standing without pedaling should be tagged separate as standing.
Running	Locomotion towards a destination, with at least two steps where both feet leave the ground during each stride. For chest mounted camera: Running can be inferred when trunk moves forward is in a constant upward-downward motion with at least two steps. Running along a curved line is allowed.
Bending	While standing/sitting, bending towards an object placed below knee-height is bending.
Picking	This refers to picking/placing/touching an object from below knee height. Picking occurs when the trunk is at its lowest point and the person has touched/placed/picked an object. When the person starts to rise it's trunk, picking finishes, and bending begins.
Other vigorous activities	All non-cyclic rapid leg movements that do not classify as running. This includes sport like activities such as rapid change in direction and jumping. Can occur in all directions.
Other non-vigorous activities	All non-cyclic movements that do not classify according to the definitions. Can occur in all directions.
Undefined	Until all the sensors are attached, or final adjustment made to position the video can be tagged as undefined. All postures/movements that can not be clearly identified should be tagged as undefined.

Table A.5: Transition definitions

Transition	**Description**
Bending to picking from standing/walking/sitting	As soon as forward/sideways trunk tilt occurs, bending has started. Bending finishes when the person has reached the lowest point of the movement and picking occurs. When the person starts to rise up, picking finishes and bending begins. When the trunk is in an upright and stable position, bending finishes. This should be tagged as "bending-picking-bending". Steps can occur during bending.
Walking to posture	Walking ends when both feet are at rest, or at first evident forward tilt of upper body. Steps can occur during the transition from walking to posture.
Upright to sitting	Can be from walking or standing, as soon as forward trunk tilt occurs, or a lowering of the trunk, the transition has started. Steps can occur during the transition for positioning. Transition ends when buttocks are in contact with the seat of the chair, bed or floor.
Sitting to upright	Transition starts when the person's buttocks leave the chair and ends when the trunk has reached its upright position. Steps and turning can occur during the transition from sitting to upright.
Standing/walking/sitting to lying	When the trunk flexion begins, or a lowering of the center of mass, the transition has started. Transition finishes when the person is lying flat with the trunk in a stable position.
Lying to standing/walking/sitting	While lying, the transition begins with an upward movement of the trunk or leg movement that leads to a stable upright position or continuous walking. The trunk angle should be in a steady posture for the transition to finish. Steps can occur during the transition.
Standing to walking	As soon as heel-off occurs, walking has started.
Standing to shuffling	As soon as one foot moves, shuffling has started.
Shuffling/walking to standing	As soon as the feet stop moving, walking/shuffling has finished and standing has started.
Shuffling to walking	As soon as walking direction is set and heel-off occurs, shuffling has ended and walking starts.
Walking to shuffling	When walking is interrupted by stepping in place, non-cyclical, non-directional movement of the feet or turning on the spot, this should be tagged as shuffling.
Sit cycling to stand cycling / stand cycling to sit cycling	When the buttocks leave the seat, stand cycling can be inferred. When the buttocks is placed at the seat, sit cycling can be inferred.

Chapter B
Appendix

This appendix contain additional information about the different methods used in the process of generating the proposed HAR system.

B.1 Convolutional Neural Network

Table B.1: Convolutional Neural network parameters

Data input size	6x100
Kernels in convolutional layer one	20
Kernels in convolutional layer two	40
Filter size width	30
Filter size height	1
Filter type	Valid
Nodes in neural layer one	1500
Nodes in neural layer two	10
Dropout probability in neural layer one	50%
Optimizer	Adam algorithm
Training iterations	20.000
Batch size	100

Table B.2: Experiment with convolutional layers. Neural layers: 200-100. Training iterations: 3000

Convolutional layers	Accuracy (%)	Average Sensitivity (%)
20-40	0.9551451802	0.9395618379
30-40	0.9554338165	0.9394215703
40-50	0.955821065	0.9376101077
15-30	0.951624926	0.9337377548
10-20-40	0.9545889789	0.9273004651
20-30	0.9576867898	0.9253455102
10-20	0.9462671373	0.9235385954
15-20	0.9383888529	0.9121215522
40-20-10	0.9487383485	0.8947891533
2-10-20	0.9248430131	0.8743480027

Table B.3: Experiment with neural layers. Convolutional layers: 20-40. Training iterations: 3000

Neural layers	Accuracy (%)	Sensitivity (%)
1500	0.9741213316	0.9521336073
1000	0.9636148628	0.9442937613
1000	0.9695710762	0.943088311
1000-500	0.9596440395	0.9424107015
200	0.9594539579	0.9366894364
1500-1000	0.9727863408	0.9361109152
500	0.9559125899	0.9360525131
1000-100	0.9514630203	0.9359237254
400-200	0.9639387199	0.9352448106
400-300-200	0.9585386984	0.9332973599
50	0.9463586749	0.9092043042
100	0.8989833426	0.9019963443

B.2 Semi-supervised Learning

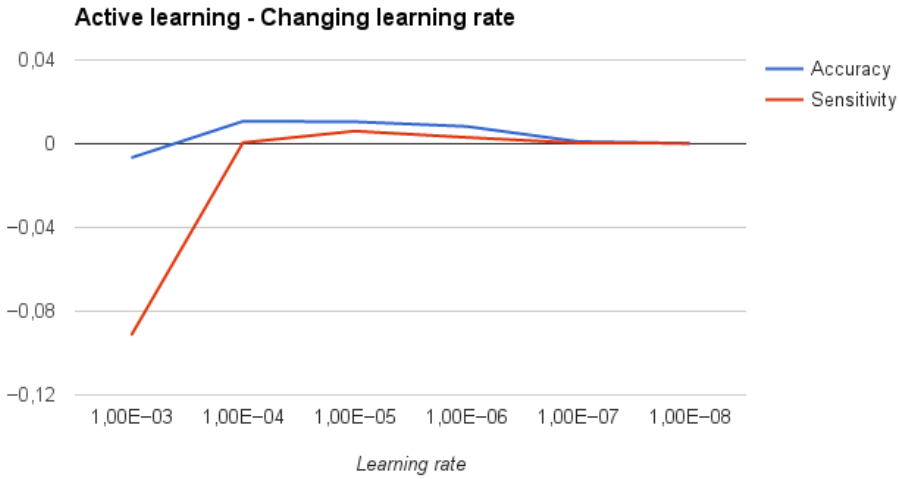


Figure B.1: Active learning - Changing learning rate (Adult data set)

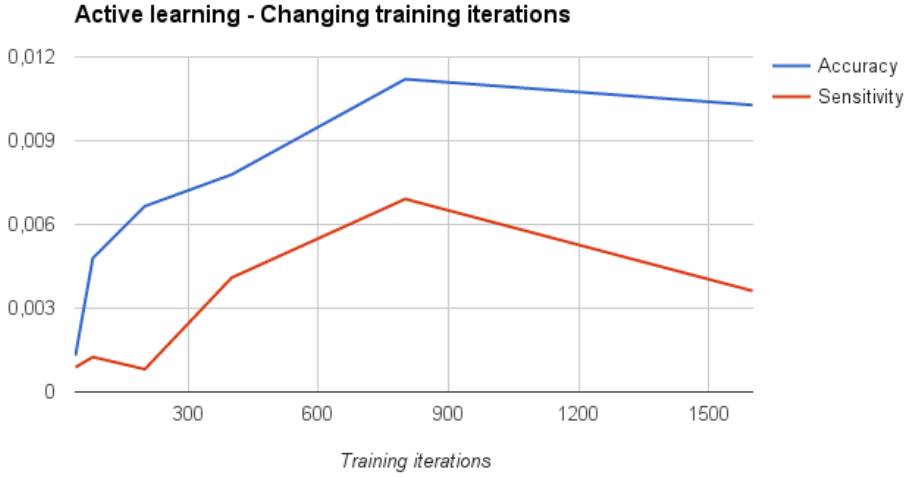


Figure B.2: Active learning - Changing training iterations (Adult data set)

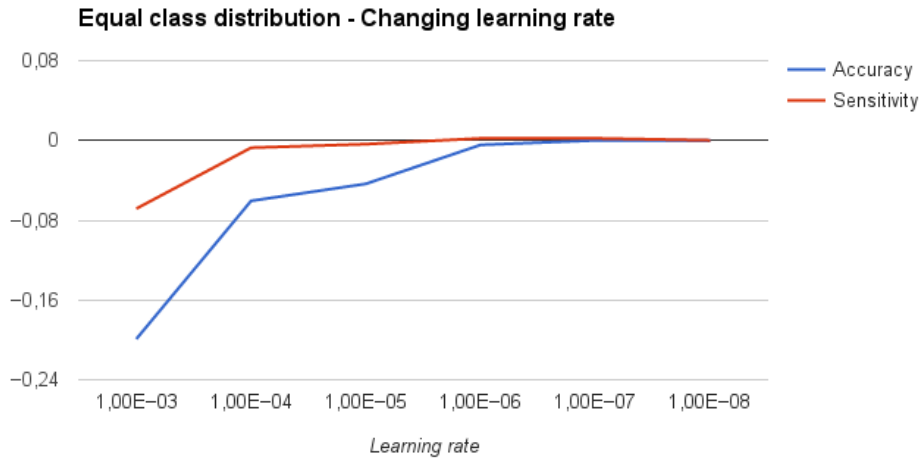


Figure B.3: Equal class distribution - Changing learning rate (Adult data set)

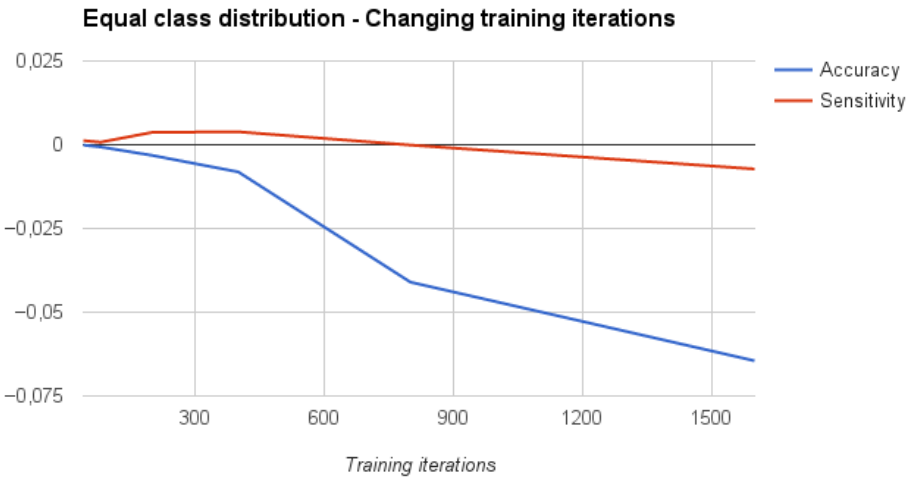


Figure B.4: Equal class distribution - Changing training iterations (Adult data set)

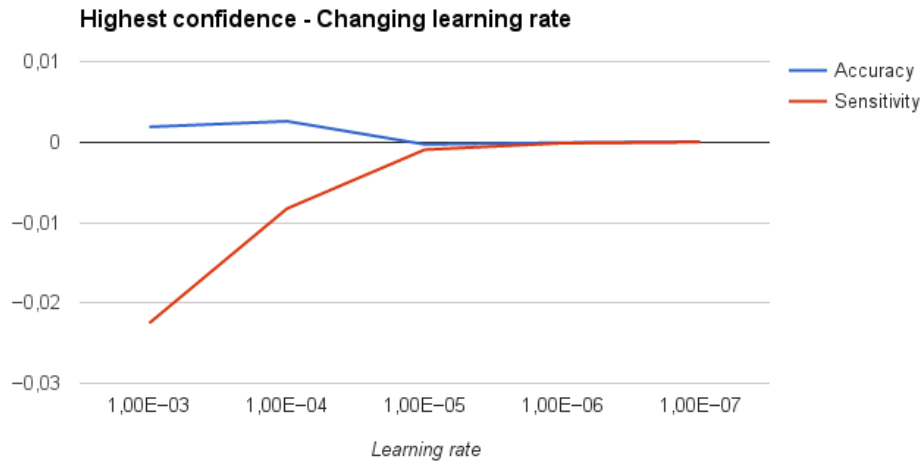


Figure B.5: Highest confidence - Changing learning rate (Adult data set)

Figure B.6: Highest confidence - Changing training iterations (Adult data set)

B.3 Hidden Markov Model

Table B.4: Transition Probabilities (%)

	Walking	Running	Stairs (up)	Stairs (down)	Standing	Sitting	Lying	Bending	Cycling (sit)	Cycling (stand)
Walking	9.86786277e-01	7.89825481e-04	3.36470077e-03	2.33137500e-03	5.88734896e-03	3.44696915e-14	4.04802417e-16	8.40473155e-04	1.00217319e-28	2.42400268e-29
Running	5.42132757e-03	9.89165116e-01	2.51794250e-03	2.89561080e-03	2.90990620e-09	1.43882036e-33	2.40194509e-29	1.99109327e-28	4.11858877e-44	1.68234975e-34
Stairs (up)	1.59231567e-02	8.04915516e-04	9.77794006e-01	3.79441981e-03	1.68350186e-03	2.54770752e-23	3.22473095e-18	6.78926438e-17	1.09426046e-30	2.55732649e-25
Stairs (down)	1.94191102e-02	1.64544548e-03	2.64357989e-03	9.74499488e-01	1.77801621e-03	2.34135836e-11	8.09128705e-21	1.43603784e-05	2.69024480e-40	1.23907917e-15
Standing	7.81106629e-03	1.10023809e-04	5.23407890e-08	9.48411386e-05	9.88495279e-01	8.38995277e-05	9.87118772e-07	3.40385174e-03	5.30037367e-20	1.10320345e-17
Sitting	2.46532053e-15	2.24679650e-30	1.39814035e-13	4.05465817e-24	7.38488581e-05	9.97553923e-01	4.98549156e-04	1.27418945e-03	5.99489154e-04	2.96341912e-18
Lying	7.21809295e-26	2.12675869e-30	5.68191974e-27	4.87465433e-34	4.37927118e-04	1.22078529e-03	9.98321742e-01	1.95456599e-05	1.18876960e-15	5.70354448e-24
Bending	6.54987072e-03	1.57303544e-24	1.65068822e-04	9.14600653e-18	3.16733220e-02	1.10649793e-02	1.80244806e-03	9.48058305e-01	6.86005824e-04	6.53475471e-11
Cycling (sit)	3.26899736e-22	3.60055804e-46	6.23873912e-21	2.08480127e-40	1.78261332e-14	9.18803633e-03	4.10196250e-10	1.50567587e-12	9.80678307e-01	1.01336560e-02
Cycling (stand)	4.35019814e-24	5.92164667e-32	1.13551210e-18	3.27314808e-15	5.50380939e-14	3.51408400e-14	3.05207636e-20	5.89108573e-16	1.60924695e-02	9.83907531e-01

Bibliography

I-Min Lee, Eric J Shiroma, Felipe Lobelo, Pekka Puska, Steven N Blair, Peter T Katzmarzyk, Lancet Physical Activity Series Working Group, et al. Effect of physical inactivity on major non-communicable diseases worldwide: an analysis of burden of disease and life expectancy. *The lancet*, 380(9838):219–229, 2012.

United States US Department of Health. *Physical activity and health: a report of the Surgeon General*. diane Publishing, 1996.

A Alwan. Global status report on non-communicable diseases 2010. geneva: World health organization; 2011. h ttp. *apps. who. int/iris/bitstream/10665/44579/1/9789240686458_eng. pdfA ccessed on*, 22, 2014.

Lydia Kwak, Karin I Proper, Maria Hagströmer, and Michael Sjöström. The repeatability and validity of questionnaires assessing occupational physical activity-a systematic review. *Scandinavian journal of work, environment & health*, pages 6–29, 2011.

Deka A Feigelson HS Campbell PT Gapstur SM Colditz GA Thun MJ. Patel AV1, Bernstein L. Leisure time spent sitting in relation to total mortality in a prospective cohort of us adults. 2010.

Nidhi Gupta, Marina Heiden, Mette Aadahl, Mette Korshøj, Marie Birk Jørgensen, and Andreas Holtermann. What is the effect on obesity indicators from replacing prolonged sedentary time with brief sedentary bouts, standing and different types of physical activity during working days? a cross-sectional accelerometer-based study among blue-collar workers. *PloS one*, 11(5):e0154935, 2016.

Astrid Johnsen Tessem and Hans-Olav Hessen. Human activity recognition with two body-worn accelerometer sensors. 2015.

Jørgen Skotte, Mette Korshøj, Jesper Kristiansen, Christiana Hanisch, and Andreas Holtermann. Detection of physical activity types using triaxial accelerometers. *J Phys Act Health*, 11(1):76–84, 2014.

IH Danquah, S Kloster, A Holtermann, M Aadahl, A Bauman, AK Ersbøll, and JS Tolstrup. Take a stand!–a multi-component intervention aimed at reducing sitting time among office workers–a cluster randomized trial. *International Journal of Epidemiology*, page dyw009, 2016.

David Hallman, Nidhi Gupta, Svend Erik Mathiassen, Mette Korshøj, and Andreas Holtermann. Temporal patterns of physical activity during work and leisure: Exposure variation analysis of accelerometer recordings processed by the acti4 software. In *4th International Conference on Ambulatory Monitoring of Physical Activity and Movement (ICAMPAM), Limerick, Ireland, June 10-12, 2015*, page 11, 2015.

Camilla Munch Nielsen, Nidhi Gupta, Lisbeth E Knudsen, and Andreas Holtermann. Association of objectively measured occupational walking and standing still with low back pain: a cross-sectional study. *Ergonomics*, pages 1–9, 2016.

Salvatore T March and Gerald F Smith. Design and natural science research on information technology. *Decision support systems*, 15(4):251–266, 1995.

Herbert A Simon. *The sciences of the artificial*. MIT press, 1996.

Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):33, 2014.

Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L Littman. Activity recognition from accelerometer data. In *AAAI*, volume 5, pages 1541–1546, 2005.

Heike Leutheuser, Dominik Schuldhaus, and Bjoern M Eskofier. Hierarchical, multi-sensor based classification of daily life activities: comparison with state-of-the-art algorithms using a benchmark dataset. *PloS one*, 8(10):e75196, 2013.

Ming Li, Viktor Rozgić, Gautam Thatte, Sangwon Lee, Adar Emken, Murali Annavaram, Urbashi Mitra, Donna Spruijt-Metz, and Shrikanth Narayanan. Multimodal physical activity recognition by fusing temporal and cepstral information. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 18(4): 369–380, 2010.

Louis Atallah, Benny Lo, Rachel King, and Guang-Zhong Yang. Sensor positioning for activity recognition using wearable accelerometers. *Biomedical Circuits and Systems, IEEE Transactions on*, 5(4):320–329, 2011.

Ian Cleland, Basel Kikhia, Chris Nugent, Andrey Boytsov, Josef Hallberg, Kåre Synnes, Sally McClean, and Dewar Finlay. Optimal placement of accelerometers for the detection of everyday activities. *Sensors*, 13(7):9183–9200, 2013.

Dean M Karantonis, Michael R Narayanan, Merryn Mathie, Nigel H Lovell, and Branko G Celler. Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. *Information Technology in Biomedicine, IEEE Transactions on*, 10(1):156–167, 2006.

Hans D Lüke. The origins of the sampling theorem. *IEEE Communications Magazine*, 37(4):106–108, 1999.

Roger Bartlett. *Introduction to sports biomechanics: Analysing human movement patterns*. Routledge, 2007.

Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *Pervasive computing*, pages 1–17. Springer, 2004.

Narayanan C Krishnan and Diane J Cook. Activity recognition on streaming sensor data. *Pervasive and mobile computing*, 10:138–154, 2014.

Simon Kozina, Mitja Lustrek, and Matjaz Gams. Dynamic signal segmentation for activity recognition. In *Proceedings of International Joint Conference on Artificial Intelligence, Barcelona, Spain*, volume 1622, page 1522. Citeseer, 2011.

Thomas Plötz, Nils Y Hammerla, Agata Rozga, Andrea Reavis, Nathan Call, and Gregory D Abowd. Automatic assessment of problem behavior in individuals with developmental disabilities. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 391–400. ACM, 2012.

Eric Guenterberg, Sarah Ostadabbas, Hassan Ghasemzadeh, and Roozbeh Jafari. An automatic segmentation technique in body sensor networks based on signal energy. In *Proceedings of the Fourth International Conference on Body Area Networks*, page 21. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

Tâm Huynh and Bernt Schiele. Analyzing features for activity recognition. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, pages 159–163. ACM, 2005.

Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.

Vincent T Van Hees, Lukas Gorzelniak, Emmanuel Carlos Dean Leon, Martin Eder, Marcelo Pias, Salman Taherian, Ulf Ekelund, Frida Renström, Paul W Franks, Alexander Horsch, et al. Separating movement and gravity components in an acceleration signal and implications for the assessment of human daily physical activity. *PloS one*, 8(4):e61691, 2013.

Erwin Kreyszig. Advanced engineering mathematics, 9th edition, chapter 11.1. 2006.

Waltenegus Dargie. Analysis of time and frequency domain features of accelerometer measurements. In *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*, pages 1–6. IEEE, 2009.

Syed Agha Muhammad, Bernd Niklas Klein, Kristof Van Laerhoven, and Klaus David. A feature set evaluation for activity recognition with body-worn inertial sensors. In *Constructing Ambient Intelligence*, pages 101–109. Springer, 2011.

Vipin Kumar Pang-Ning Tan, Micheal Steinbach. Introduction to data mining, international edition. 2006.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29 (6):82–97, 2012.

Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1915–1929, 2013.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), Buenos Aires, Argentina*, pages 25–31, 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Lawrence R Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93. ACM, 2000.

Yan Zhou and Sally Goldman. Democratic co-learning. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 594–602. IEEE, 2004.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

Tin Kam Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832–844, 1998.

Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

Oresti Banos, Juan-Manuel Galvez, Miguel Damas, Alberto Guillen, Luis-Javier Herrera, Hector Pomares, Ignacio Rojas, Claudia Villalonga, Choong Seon Hong, and Sungyoung Lee. Multiwindow fusion for wearable activity recognition. In *Advances in Computational Intelligence*, pages 290–297. Springer, 2015.

Donghai Guan, Weiwei Yuan, Young-Koo Lee, Andrey Gavrilov, and Sungyoung Lee. Activity recognition based on semi-supervised learning. In *Embedded and Real-Time Computing Systems and Applications, 2007. RTCSA 2007. 13th IEEE International Conference on*, pages 469–475. IEEE, 2007.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, page 3, 2010.

Hongqing Fang, Hao Si, and Long Chen. Recurrent neural network for human activity recognition in smart home. In *Proceedings of 2013 Chinese Intelligent Automation Conference*, pages 341–348. Springer, 2013.

Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *Human Behavior Understanding*, pages 29–39. Springer, 2011.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015.

Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

Bruce M Altevogt, Harvey R Colten, et al. *Sleep Disorders and Sleep Deprivation:: An Unmet Public Health Problem*. National Academies Press, 2006.