KU LEUVEN

Identification of inertial features for the optimal recognition of physical activities based on wearable devices

Simon Perneel

Thesis submitted for the degree of Master of Science in Artificial Intelligence, option Engineering and Computer Science

Thesis supervisors:
Prof. Dr. ir. E. Aertbeliën
Prof. Dr. ir. H. Bruyninckx

Assessor:
Dr. N. Tsiogkas
Prof. Dr. K. Luyten

Mentor:
Dr. ir. A. Ancillao

Academic year 2020 – 2021

Preface

Writing a master's thesis is a process that involves a lot. During this process I have gained experience and knowledge in various areas: programming machine learning algorithms in python, collecting data from experiments, biomechanics, working with inertial sensors, ... Therefore, I would like to thank the people who gave me this opportunity and who assisted me during my thesis.

First of all, I would like to thank my promotors, Prof. Aertbeliën and Prof. Bruyninckx for assigning and following up my thesis. I also would like to give acknowledgment to my daily advisor, Mr Ancillao. He was always ready to help or to give me advice.

Second, my parents for their unconditional support and believe in me. My brother for revising my thesis. Thank also to my friends and family, who provided the necessary leisure and distraction at regular times. Finally, thanks to all the volunteers that participated in the experiments for the data I needed.

Simon Perneel

Contents

Abstract

Wearable sensors for human activity recognition have become very popular in the past decade thanks to the advancements in sensor technology. A Magnetic Inertial Measurement Unit (MIMU) is a wearable device that combines an accelerometer, gyroscope and magnetometer in one unit. Activity recognition with MIMUs can be very useful for medical, sports and game applications. These wearable sensors produce a lot of data, enabling many possible features for classification. However, it is important to keep computing power, memory and energy requirements within bounds. Therefore, it is of interest to identify a minimal set of features to reliably identify the activities. In other studies, features from the magnetometer are hardly used and their added value is not clear. Most studies also presume a fixed orientation and/or placement of the sensors on the body, which is not always the case in real-life applications.

This thesis aims to implement an activity recognition system that recognizes a number of activities with high accuracy. An optimal set of features is sought, where it was ensured that the features are independent of the sensor orientation of the body. Furthermore, the aim was also to check the usefulness of magnetometer features and best positions to wear the sensor. Two approaches for time series classification are compared: classification on 'raw' data, where there is no explicit feature construction, and classification with manually extracted and selected features. The data was collected trough dedicated experiments. Nine subjects took part and performed seven basic activities with sensors placed on the wrist, ankle and upper thigh.

Both the sequential feature selector algorithm and a random forest classifier showed that features from the magnetometer do not add value to features from the accelerometer and gyroscope. An optimal set of 20 features was derived from the accelerometer and gyroscope measurements. Features from the ankle sensor were found to be the most discriminative. The sensor on the wrist is a less suitable position for recognizing the activities. The proposed recognition system achieves an overall accuracy and F1-score of 0.97 and 0.98 using all three sensors, respectively. Lastly, the analysis examines the tuning of the window size and different classifiers for the recognition system.

Keywords: human activity recognition, machine learning, signal processing

List of Figures and Tables

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

AC	Alternating Component
AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Net
DC	Direct Component
DFT	Discrete Fourier Transform
HAR	Human Activity Recognition
HCI	Human Computer Interaction
kNN	k-Nearest Neighbours
L1O	Leave-One-Subject-Out cross-validation
MET	Metabolic Equivalent of Task
MIMU	Magnetic Inertial Measurement Unit
MLP	Multi-Layer Perceptron
PCA	Principal Component Analysis
RAM	Random Access Memory
RF	Random Forest
SFFS	Sequential Forward Feature Selection
STFT	Short-Time Fourier Transform
SVM	Support Vector Machines

Symbols

m/s^2	Acceleration
deg/s	Angular frequency
G	Magnetic field
MET	Metabolic Equivalent of Task

Chapter 1

Introduction

1.1 Human Activity Recognition

Human activity recognition (HAR) is the ability of an intelligent system to detect physical activities. It aims to recognize the actions of a person automatically based on sensor readings and machine learning methods. It is a key research area within the Human-Computer Interaction (HCI) field of study that originated in the 1980s. The information that one can obtain from these activities, is useful in a variety of applications. For example, assisted living systems for smart homes, where elderly people are provided support to live independently [42]. In order to provide this support, a person's behaviour is observed with the help of HAR. Healthcare monitoring is another interesting field of application. Physiotherapist can for example remotely track a patient's execution of physical exercises or daily activities [13]. Other applications are performance enhancement and injury prevention in sports, interactive gaming and much more.

Traditionally, recognition was computer vision-based, and researchers used cameras to identify different activities [38] [16]. However, this method has some important drawbacks. It is confined to a fixed space, where environmental parameters (e.g. brightness) must be set correctly. With a single camera, the 3D space is captured onto a 2D image, which causes a loss of information. Moreover, the place of the activity on the images must be determined before the activity can be recognized. The use of a camera may also violate a person's privacy.

A better option for recognizing activity is to measure activity where it 'occurs', i.e. on the body. This is possible with inertial sensors [23]. With the rise of microelectromechanical systems (MEMS) during the past decade, high performance miniature versions could be made, and these inertial sensors also became cheaper and widely available. A magnetic inertial measurement unit (MIMU) is a device that combines several inertial sensors in one unit. Using accelerometers, gyroscopes and a magnetometer, the unit measures acceleration, angular rate and orientation of a body. The main advantage of MIMU sensors compared to vision-recognition is the

flexibility of use. They are not confined to a fixed space and can be used in different environments. Furthermore, they are able to record data for long periods. They have fewer requirements for processing power, memory, and energy. They also preserve better a user's privacy. Hence, inertial sensors are an effective way for human activity recognition.

However, there are still some challenges for activity recognition using inertial sensors. Some of these challenges will be addressed in this thesis.

1.2 Related work

Quite a lot of studies have been conducted in the past on activity recognition with inertial sensors. They differ in the number of sensors used, the placement of the sensors on the body, the used signals from the inertial data, etc.

Some studies only use measurements from the accelerometer, some use a combination of accelerometer and gyroscope data. Magnetometer measurements are rarely used. Most studies in activity recognition also assume that the sensors are worn in a fixed position and a fixed place. However, this is not feasible in practice. Table 1.1 gives some examples of studies in human activity recognition and their main differences.

Reference	recognized activities	sensor units used	used sensors	orientation invariant
[3]	19	5	accelerometer gyroscope magnetometer	✗
[4]	15	6	accelerometer	partial
[9]	19	5	accelerometer gyroscope magnetometer	✓
[25]	15	1	accelerometer	✗
[39]	7	1	accelerometer gyroscope	✗
[49]	6	5	accelerometer gyroscope	✗

Table 1.1: Examples of existing works in human activity recognition and their characteristics

1.3 Objectives and overview of the thesis

The objective of this thesis is in the first place to implement an AI classification algorithm that recognizes a number of physical activities with high performance. The classification is done based on measurements from inertial sensors. These measurements will be collected trough self-designed experiments. In the context of this thesis, an offline execution of the recognition system is chosen. An online recognition is more challenging to implement in software because the data collection, processing and classification is done in real time. Therefore, the data will be recorded first, and the recognition is performed afterwards.

Firstly, it will be investigated what the minimum set of features is to reliably identify the activities. In particular, features will be used that are independent of the orientation from the sensors on the body. Second, the usefulness of magnetometer features will be examined. Measurements from the magnetometer are rarely used in other studies and their added value is not clear. Lastly, sensors will be placed on different positions on the body to see which place is more suitable for recognizing the activities.

The thesis is divided in six chapters. Chapter 1 is the introduction to the work. Chapter 2 presents a literature review and the frequently used techniques for HAR, to provide context and background information for the rest of the thesis. Chapter 3 describes how the protocol of the self-designed experiments is defined, and how the data was collected. Chapter 4 and 5 presents the results obtained in this thesis. In particular, Chapter 4 explains the examined set of features and compares different approaches for feature selection. Chapter 5 assesses the performance of the used features and classifiers. It also checks the robustness of the recognition system against changes in orientation of the sensor. Finally, Chapter 6 discusses the insights gained in this thesis and some conclusions are drawn.

Chapter 2

State-of-the-art of Human Activity Recognition

Activity recognition systems are being developed for more than 20 years, and the research domain has matured. Research in activity recognition with inertial sensors has a lot in common with other domains like signal processing and general pattern recognition, but activity recognition systems also have their own specific constraints and there are still some key research challenges specific to human activity recognition.

The sequence of data acquisition, signal processing and machine learning techniques that perform the recognition, is called the activity recognition chain (ARC). This is a generic framework around which activity recognition systems are build. Figure 2.1 shows the different steps involved in the chain. Each part of the framework has its difficulties and there are several design choices that must be made carefully. This chapter provides a literature review of research towards activity recognition. It also describes in detail the different parts of the ARC and the considerations associated with it. In the end, some open research challenges that are specific to human activity recognition are outlined.

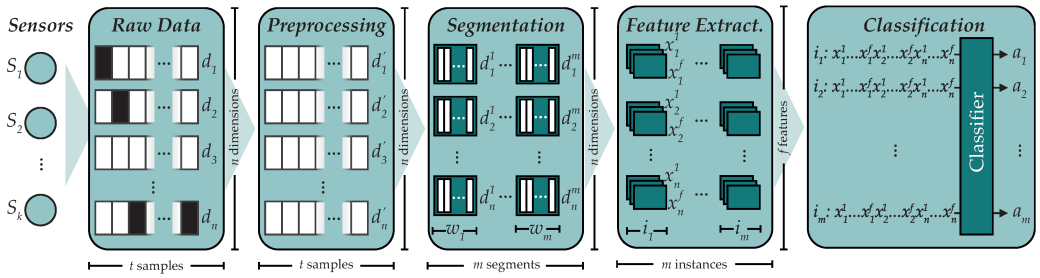


Figure 2.1: Different steps in the human activity recognition chain [40].

2.1 Data acquisition

The first step in the recognition process is to acquire data from (multiple) inertial sensors. There is no consensus on the number and placement of sensors. Nevertheless, this has an impact of the accuracy of the HAR system [5]. The number and placement of the sensors depends on the variety of activities one wants to recognize. Wearing multiple sensors can be burdensome and impractical. Consequently, researchers strive to use a minimum number of sensors while the developed model retains the highest possible accuracy. In some studies, one wants to look at body movements as a whole and places a sensor close to the center of mass of the body, such as the lower back [33], chest [26] or waist [17]. Other studies place multiple sensors on different parts of the body. Frequently used places include the wrist, upper arm, upper thigh and ankle [41] [6] [15].

2.2 Pre-processing

The data that the sensors record on the body during different activities is not in the right form to use right away. Hence, it is called *raw data*. The pre-processing stage converts the raw time series data into a pre-processed time series data format. It prepares the obtained data for feature extraction. This is a generic operation; it should only depend on the data itself and not on any specific activity or person. The raw sensor output at a point in time can be described with a vector:

$$s_i = (d^1, d^2, d^3, \ldots, d^t) \qquad \text{for } i = 1, \ldots, k \tag{2.1}$$

With k the number of sensors and d^i the multi-dimensional values of the MIMU sensor.

As a first pre-processing step, artifacts in the signals have to be filtered out (dealing with missing values, outliers, sensor malfunction, etc.). Then, the body acceleration and gravitational acceleration has to be separated. The high-frequency or AC component is related to the body acceleration and thus the dynamic motion of the subject. This can be exploited to identify dynamic activities (e.g. walking, running, jumping, etc.). On the other hand, the low frequency or DC component is related to the gravity. This component can be exploited to identify static postures (e.g., standing, sitting ...) [28]. A high-pass filter is often used to separate the gravitational component and the body component. [30] suggest using a Butterworth high pass filter with a cut-off frequency around 0.5 % of the sample rate.

The pre-processing stage also includes calibration, synchronization, conversion of units and resampling of the accelerometer and gyroscope signals if needed. Normalization is also part of the pre-processing. The aim of normalization is to bring the signals from different subjects to a common scale, without distorting differences in the ranges of values.

Lastly, sensor-fusion is sometimes done. Sensor-fusion brings data from different types of sensors together and extracts new information with algorithms (e.g. Kalman

Filter, Bayesian network, etc.). For example, with the combination of an accelerometer, gyroscope and compass, a Kalman filter sensor fusion algorithm can determine the orientation of an object. At the end of this stage, a pre-processed time series D' is obtained:

$$D' = \begin{pmatrix} d'^{1}_{1} & \dots & d'^{t}_{1} \\ \vdots & \dots & \vdots \\ d'^{i}_{n} & \dots & d'^{t}_{n} \end{pmatrix} \tag{2.2}$$

with d'_i one dimension of the pre-processed signal, n the dimensions and t the total number of samples

2.3 Data segmentation

The data segmentation step stage aims to identify the segments of the pre-processed signals that contains information about the activity. Segmentation means that data is divided into different segments. Each segment w_i is defined by its start time t_1 and its end time t_2. After the segmentation stage, a set of segments W containing a potential activity y is obtained:

$$W = \{w_1, \dots, w_n\} \tag{2.3}$$

The segmentation is an ambiguous process. Activities are not separated by pauses, and thus the start and end times are not exactly defined. Several activities can also take place at the same time. A 'drinking' activity can take place at the same time as a 'walking' activity, for example. There are different methods for data segmentation: sliding windows, event-defined windows and activity defined events [8].

- **Sliding windows**: with this method, a fixed-size window is moved over the time series signal to extract a segment. This is the simplest approach of the three. There are two parameters that have to be set: the window size and the step size. There is no clear agreement on these parameter settings. The window size influences the accuracy of recognition. A larger window size may yield a better performance, but the system has to 'wait' longer for a segment to become available. This delay is important when doing online classification. However, as stated in [8], larger window sizes do not necessarily translate into a better recognition performance. The step size is the number of time steps that are between two consecutive segments. It determines the amount of overlap of the segments. In previous studies, window sizes from 0.1 s ranging to 12.8 s and more have been used [8]. The influence of window and step size on the performance of the model is evaluated further in this thesis.

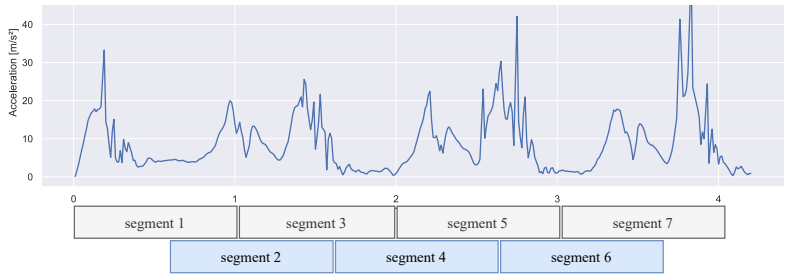


Figure 2.2: Segmentation with sliding windows. These windows have a size of 100 samples and a step size of 50, which results in 50% overlap between the segments.

- **Energy-defined windows:** this method is built on the fact that different activities have different intensities. This difference in intensities or energy levels is recorded by the inertial sensors. The energy of a signal $s(t)$ is defined as: $E = \int_{-\infty}^{+\infty} |s(t)|^2 \, dx$. By means of thresholding on the energy of the signals, different segments can be distinguished that are expected to be related to the same activity. Study [19] used this approach.
- **Event-defined windows:** the data is divided based on the detection of activity changes. Because events are not uniformly distributed in time, the size of the corresponding windows is not fixed. Sometimes, additional sensors are used to locate the events. GPS data on a phone can be used to divide a long stream of data from the accelerometer into different events. Or external sources such as a calendar that can contain information about the start and end of an activity or weekly repeated events. Study [7] used this approach.

The sliding windows or 'windowing' method is the most popular method of the three. It is quite simple; it does not require additional processing and thus has a low computational load. Because of this, it is well suited for real-time applications. This method will also be used further in this thesis. The drawback is that the windows are fixed in size and they are 'agnostic' about the type of activity of underlying time series data [12].

2.4 Feature extraction

Feature extraction and selection is a critical part of any kind of classification task. Feature extraction methods filter out relevant information and obtain quantitative measures of the signals. A feature is the numerical representation of data. The extracted features are discriminative for the activities and it allows to compare different time windows. Features can be computed based on knowledge about the activity ('handcrafted') or automatically with software packages [32]. The challenging task of feature selection identifying all highly relevant and discard weakly relevant attributes. This is especially time consuming and sometimes hard for time series classification. Also, the choice of the features is problem-specific and depends on the

activities that have to be recognized. The extracted features can be seen as feature vectors X_i of the set of segments W:

$$X_i = \mathcal{F}(D', w_i) \tag{2.4}$$

with $\mathcal{F}$ the function of the feature extraction and w_i the segment of the pre-processed time series D'. The resulting set of features is called the feature space. Features corresponding to the same activity should be at close distance in the feature space. While features of different type of activities should be at large distance. Moreover, optimal features should not depend on the person or the variability of a given activity. The more features in the feature space, the more training data will be required to make a meaningful model. A typical guideline is that there should be at least five training examples for each dimension in the representation [45]. When the number of features increases, the required computing power and memory of the classification algorithm will also strongly increase. Hence, it is important to identify a minimum set of features that achieves the wanted recognition performance. There are some methods that help reducing the features to those that are believed the most useful for a model to predict the target activity.

Usually, it is assumed that the orientation of the sensors is fixed, and the features are therefore orientation dependent. However, this is not the case in real-life applications. Sensor position and orientation can change over time (e.g., a MIMU sensor in a phone). Some recent studies focus therefore on finding orientation-independent features [47] [48]. These are discussed in Section 2.5.

2.4.1 Relevant features for HAR

Many different types of features have been used in previous studies. This section provides an overview of the most commonly used features for human activity recognition.

- **Statistical features**: these features are frequently used as they are simple and perform well on a lot of activity recognition problems.

Statistical features	References
mean	[12] [3] [28] [49] [25] [4] [41]
min	[3] [41]
max	[3][41]
range	[41] [25]
variance	[12] [3] [4] [41]
covariance	[20] [4]
standard deviation	[41] [25]
kurtosis	[3] [20] [4] [41]
skewness	[3] [4] [41]

Table 2.1: Overview of the most commonly used statistical features

- **Time-domain features**: these are features that are specific for signals that vary in time.

Time domain features	References
autocorrelation	[28] [49] [25]
RMS amplitude/ signal energy	[28] [49] [20] [4]
zero-crossing rate	[20]
median absolute deviation	[20]

Table 2.2: Overview of the most commonly used time domain features

- **Frequency-domain features**: These are features that are derived from time-frequency transforms, usually the Short-Time Fourier Transform (STFT).

frequency domain features	References
entropy	[28] [4] [41] [25]
spectral peak	[49] [4]
position of the peaks	[49] [41]
spectral energy	[20] [41]

Table 2.3: Overview of the most commonly used frequency domain features

- **Time-frequency domain features**: These are features intentionally crafted from time-series signals. A wavelet is an oscillation of limited duration with an average value of zero. They may be used to extract information from accelerometry data. Similar to Fourier analysis, a wavelet analysis decomposes the signal into its elements, which allows to identify the frequency content of a signal over time. A discrete wavelet transform of a sampled signal is a wavelet approximation of this signal. The coefficients of this transform correspond to the characteristics of the signal and can be used as a feature [34] [18].

2.5 Feature selection and dimensionality reduction techniques

Feature selection and dimensionality reduction methods are both used to reduce the number of features in a dataset. They are important and frequently used in activity recognition, and machine learning in general. These methods aim to remove irrelevant or redundant features. This provides a faster and more cost-effective classification, a better understanding of the underlying data and may improve the accuracy of the model. However, there is an important difference between the two. Feature selection only selects and omits given features without changing them, while dimensionality reduction transforms features into a lower dimension. Two frequently

used methods are discussed: SFFS and PCA. These will also be used further in this thesis when programming the HAR-system to find good features.

2.5.1 Sequential Forward Feature Selection (SFFS)

SFFS selects a subset of features in order to minimize the classification error. This algorithm is useful because it gives insight into which factors are most discriminative for recognizing the activities. It also reduces the number of features in the feature space which results in less data. The algorithm works as follows; first, the best feature according to some criterion function (e.g. prediction accuracy) is selected. Then, pairs of features are made using one of the remaining features and the selected feature, and the best pair is picked. Next, triplets are made with one of the remaining features and this best pair, and the best triplet is picked. This process continues until a predefined number of features are selected. The flowchart is shown in Figure 2.3.

A variation of the algorithm is the Sequential Backward Feature Selection algorithm (SBFS), which is similar to SFFS. Here they start with the full set of features, and the worst feature is discarded. SFFS works best when the optimal subset is small, SBFS when the optimal subset is large.

The main limitation of SFFS is that it is not able to remove features that become not useful after addition of other features. Study [28] stated that when a small number of features are used (< 10), the SFFS methods gives better results than PCA. When more features are used, classification rates are similar with SFFS or PCA.

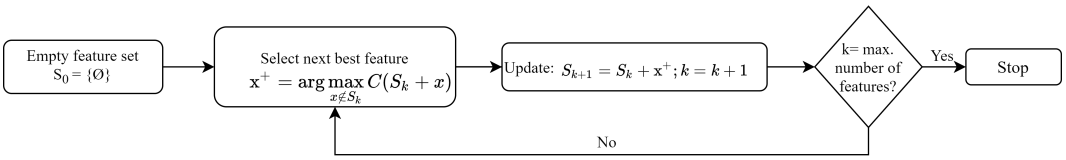


Figure 2.3: flowchart SFFS algorithm.

2.5.2 Principal Component Analysis (PCA)

PCA is by far the most popular feature reduction technique. This method transforms the original set of features into a new reduced set of features, called principal components. A principal component is a linear combination of the original variables. A variable system is fitted on the data such that the greatest variance of the data set comes to lie on the first axis, the second greatest variance on the second axis, etc. The features have to be scaled before applying PCA, otherwise the components would be biased towards features with high variance, which leads to false results. The resulting components are uncorrelated since they are orthogonal in the sample space. If this technique is used on a feature set with a large number of variables, the amount of explained variation can be compressed to a few components. This method is depicted in Figure 2.4.

A drawback of PCA is that the features are transformed and their units are thus lost. Also, the components itself are less interpretable.

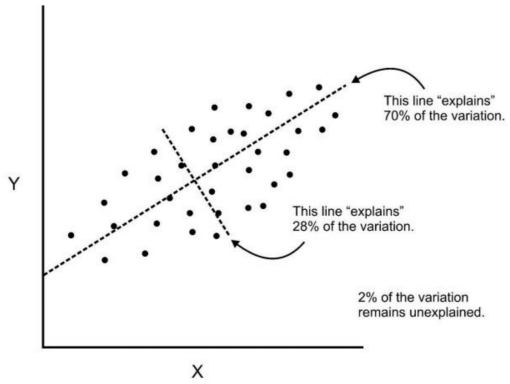


Figure 2.4: PCA - projection of the data on two principal components

2.6 Machine Learning Classification Techniques

In this stage, the selected/reduced features are used as an input to a classification algorithm. The task of the classifier is to label the feature vector extracted from each time window. It then makes a prediction on what activity is being performed during the time window. The aim is to make this prediction as accurate as possible. The next sections describe the main ideas of some state-of-the-art machine learning classifiers that also will be used further in Chapter 5.

2.6.1 k-Nearest Neighbours (k-NN)

This method is among the simplest of all classification algorithms. The idea is to memorize all feature vectors of the training set. New instances are classified by majority voting of the k closest instances in the training set. The distance between two feature vectors is usually defined as the Euclidean distance between the vector components. The optimal k value depends on the data set. Larger k values make the model less sensitive to noise but increase bias as the decision boundaries become smoother. The k-NN algorithm can be slow at prediction time because all instances must be passed trough to determine the closest [44]. Also, in a high dimensional space the distance to all neighbours becomes more or less the same, and the notion of near and far neighbours is less pronounced.

2.6.2 Random Forest Classifier (RF)

A Random Forest classifier consists of a large number of decision trees that operate as an ensemble. Each tree is trained by randomly sampled data and the information gain is used as splitting criterion for the nodes. The obtained trees are barely correlated. Then, each tree in the random forest makes a prediction of the class, and the final

prediction is the class with the most votes. The large number of barely correlated models operates as a committee that protect each other from individual errors [46]. Therefore, it is a simple, yet powerful method.

2.6.3 Neural networks

Many types of neural network architectures exist. Feedforward neural networks are often used for multi-class classification. A multi-layer perceptron (MLP) is a feedforward network composed of at least three layers with nodes or 'neurons'. There is an input layer, an output layer, and a hidden layer in between. The nodes of these layers are fully connected.

Each node in the network has a certain weight that is optimized with a backpropagation algorithm. An activation function maps the input values and the weights at each node to a value used by the next layer. Typically, a non-linear activation function is used in order to give the network the ability to learn non-linear patterns between the input (=feature vectors) and the output (=classes). The number of nodes at the input is equal to the number of features of the feature vector that has to be classified.

The number of nodes at the output is equal to the number of classes. The values at the output nodes range from 0 to 1 and represent the probability of the assigned class. The class with the highest probability is predicted by the net [11]. Figure 2.5 shows an example of a MLP architecture.

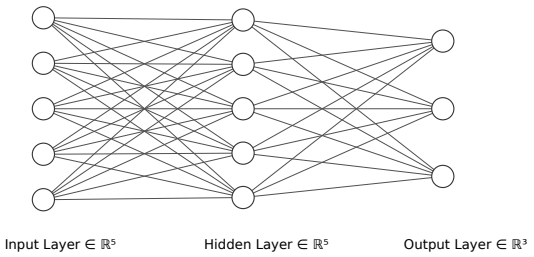


Figure 2.5: MLP architecture for classifying an instance with five features and three possible classes

2.6.4 Naive Bayes (NB)

This is a set of classifiers that works based on Bayes' Theorem of probability [43]:

$$P(c|x_1, \ldots, x_n) = \frac{P(c)P(x_1, \ldots, x_n|c)}{P(x_1, \ldots x_n)}$$

With $P(c|x)$ the probability of the class c given the values of the features x, $P(c)$ the probability of the class, $P(x|c)$ is the likelihood of the features given the class and $P(x)$ the prior probability of the features. An important assumption that is made is that the presence of a particular feature in a class is unrelated to the presence of any other feature. With this assumption, the formula can be reduced to:

$$P(c|x_1, \ldots, x_n) = \frac{P(c) \prod_{i=1}^{n} P(x_i|c)}{P(x_1, \ldots x_n)}$$

The denominator is constant given the input. The class with the highest probability gets predicted:

$$\hat{y} = \arg\max_y P(c) \prod_{i=1}^{n} P(x_i|c)$$

A Maximum A Posteriori (MAP) estimate is used to estimate $P(y)$ and $P(x_i|y)$. The NB classifiers differ mostly in the assumption of the distribution $P(x_i|c)$. Further in this thesis, the likelihood of the features is assumed to follow a Gaussian distribution.

A NB classifier is a quite simple model and it is very fast compared to the other classifiers. It typically does not need a lot of training data. The shortcoming of this method is that feature values almost never are independent of each other. Hence, these types of classifiers are called 'naive'. The probability outputs themselves are not very reliable. Nevertheless, the class with the highest probability that gets predicted often corresponds with the true class.

Naive Bayes classifiers are frequently used in applications that require fast classification on small datasets, such as spam filtering. It generally works well on multi-class classification problems [29].

2.6.5 Support Vector Machines (SVM)

Support Vector Machines gained popularity in the late 1990's. It is a widely applicable machine learning method that gives very good predictive models in a lot of cases [11]. Fundamentally, a support vector machine is a supervised binary classifier that finds the decision boundary that separates the classes and maximizes the margins. These margins are the distances between this decision boundary or 'hyperplane' and the instances (in this case: feature vectors) closest to the line. This is represented in Figure 2.6.

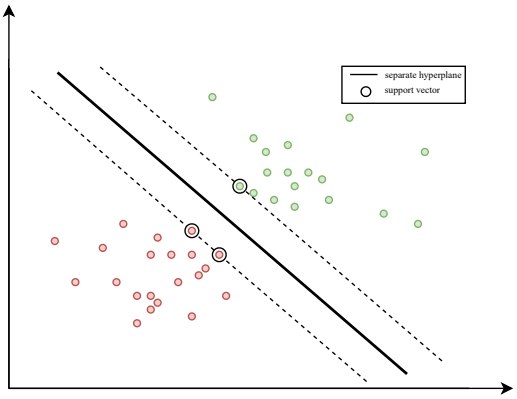


Figure 2.6: Support vectors and hyperplane for classification of a two-dimensional input space

It is assumed here that the two classes are linearly separable, i.e., there exists a hyperplane that separates them. The linear support vector machine finds the maximal margin hyperplane. However, in reality, datasets are usually not linearly separable. SVM also address the non-linearly separable cases by adding two concepts: *soft margins* and *kernel tricks*. With soft margins, the SVM tolerates a few instances to be misclassified. It makes a trade-off between finding a line that on the one hand maximizes the margin and on the other hand minimizes the misclassification.

The idea behind SVMs is to map the non-linear dataset into a higher dimensional space, which makes it possible to find a hyperplane to separate the instances. However, the computations within the higher-dimensional space are expensive. This is where the kernel trick comes in. Without going into the mathematical technicalities, it allows to operate in the original feature space without needing to compute the coordinates of the instances in the higher-dimensional space. There are different kernel functions that perform this mapping. The most frequently used are the radial basis function kernel (RBF) and the polynomial kernel.

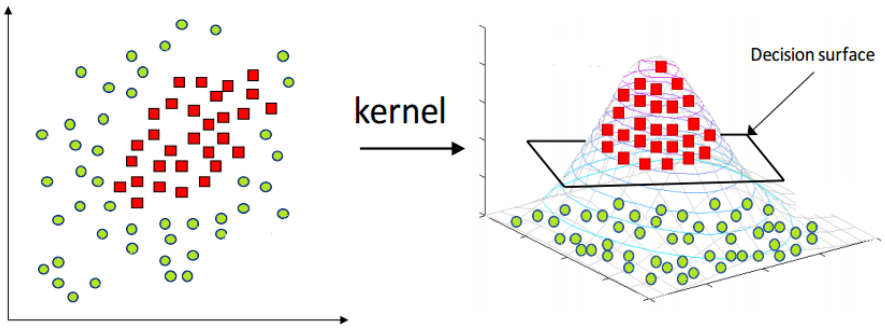


Figure 2.7: Mapping of the data from 2-dimensional space to 3-dimensional space [50]

SVM support binary classification and it separates instances in two classes. In

order to be able to use SVM on multiclass classification, the problem is broken down into multiple binary classification problems. The *One-vs-One* approach breaks the multiclass problem down in multiple binary classification problems. An optimal hyperplane is sought to separate between every two classes. The points of other classes are neglected. For each pair of classes, a hyperplane is constructed. As a result, the classifier uses $\frac{m(m-1)}{2}$ SVMs. With the *One-vs-Rest* approach, the separation takes all points into account, and divides them into two groups: one group for the class points and one group for all other points from the other classes. This is repeatedly done until all classes are separated. The *One-vs-Rest* approach uses m classifiers. This approach is less work than the *One-to-one* approach and will train faster. But it also leads to less optimal separation of the classes.

2.6.6 Classification on raw data

An alternative approach to the HAR chain is to classify activities based on raw data from the inertial sensors. In this method, there is little pre-processing and there is no manual feature engineering and selection needed to be done. Due to the great advances in deep learning and convolutional neural nets (CNNs), this approach has been used in recent years with good results [21] [10] [22].

A discriminative deep learning model is a classifier that directly learns the transformation between the raw input of the time series (e.g., x, y, z-acceleration) and the classes. The model outputs a probability distribution over the possible classes. Hence, this approach to time-series classification is called an *end-to-end* approach.

Because neural nets have a very parallel architecture, they can predict the classes in an efficient way. A drawback of neural nets is that the weights directly depend on the dimension and length of the inputted time series [21]. As a result, the neural networks are not directly transferable.

In Chapter 4, the performance of a multi-layer perceptron model will be tested out as a point of comparison for a classifier with handcrafted features.

2.7 Evaluation of the HAR

There are several techniques to assess the performance of the classifiers. Cross-validation is used to estimate the skill of a classifier model on unseen data. Two common cross-validation techniques that are used further in this thesis are discussed.

2.7.1 k-Fold cross-validation

The k-fold cross-validation is a popular evaluation method in machine learning. With this method, the data set is divided at random in k equal groups. For example, if k is taken ten, we have a 10-Fold cross-validation. The procedure is as follows. One group is taken out and used on the test set. The remaining groups form the training data set. Then the classifier model is fitted on the feature vectors in the training set

and evaluated with the test set. The evaluation score of the model is retained, and the model is discarded. Then this process is repeated 10 times for each group. The performance of the model is now determined by taking the average of the individual evaluation scores.

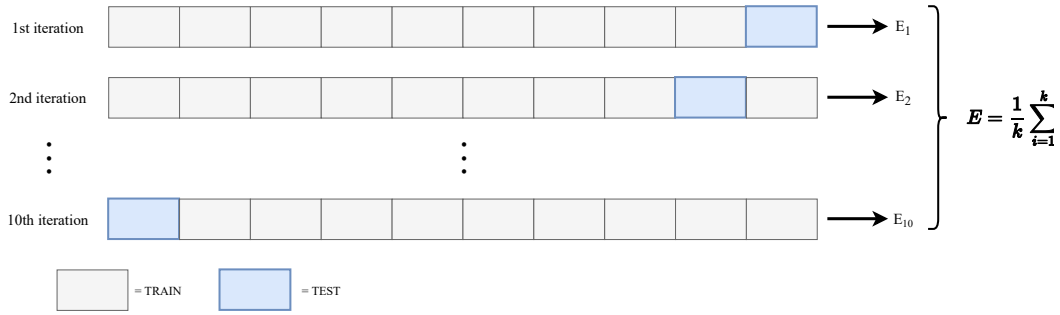


Figure 2.8: k-fold cross-validation procedure

2.7.2 Leave-One-Subject-Out cross-validation

The Leave-One-Subject-Out (L1O) cross-validation technique works similarly. The main difference is in the division of the data set. With L1O, the data is divided subject-wise. Each group contains only the data of one subject. The test set contains the feature vectors of one subject and the classifier is trained with the feature vectors from the remaining subjects. An evaluation score is made based on the classification of the left-out test set. Then, this process is then repeated for each subject. L1O cross-validation is more challenging than k-Fold as it is highly dependent on the variation in execution of the activities by the subjects. The training and test set are related to different persons. Nevertheless, this validation method is preferred for activity recognition because it assesses the generalizability of the system to an unseen subject [9].

2.7.3 Evaluation metrics

There are various metrics which allow to measure the accuracy of a classifier's prediction. These metrics also allow to compare the performance of different classification models. The metrics that are frequently used are listed below.

- **Accuracy**: the intuitive and most straightforward metric.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

  This metric has a clear shortcoming. When the data set is imbalanced (which is common in real-life examples), accuracy of the model can be misleading. Suppose we have a binary classification problem and 95 out of 100 points are positive. A dumb model that only predicts positive, will get an accuracy of 95%.

- **Precision**: this is the number of true positives divided by all predicted positives (true positives and false positives). It means the percentage of the predicted class which truly belongs to the class.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall**: this metric refers to the percentage of total relevant results which are correctly classified. It means the percentage of correctly predicted cases out of all cases that belong that a certain class.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- $F1$**-score**: For problems where both precision and recall are important, a trade-off is needed. Because it is not possible to maximize both precision and recall, this metric is a combination of both. It is particularly useful because it is robust to class imbalance. It is calculated as the harmonic mean of precision and recall:

$$F1 = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$$

  An $F1$-score of 1 means perfect precision and recall, while a score of 0 indicates either the recall or precision is zero.

2.8 Challenges

A number of challenges that are specific to human activity recognition are listed below.

- **Generalizable model**: Activities may be executed differently by different persons. An activity can also be executed differently by the same person. This is called intraclass variability. If the HAR-system is trained for multiple persons, the model has to be robust against this variability. A sufficient number of training data and person-independent features are important for a model that generalizes well.

- **Trade-offs**: When designing a HAR, there is always a trade-off to be made between accuracy, latency of the recognition and the available processing power and energy [51]. When a recognition system is implemented in a mobile device, it has to meet the energy and processing requirements of the device, while providing a reasonable accuracy and trade-off.

- **Sensor orientation and placement**. Most studies assume that the placement of the sensors is fixed. For example, a smartwatch can be worn on both arms, but the exact position can change over time. The orientation of a smartphone in someone's pocket can also differ. To make the recognition

system robust against change in orientation, orientation-invariant features are needed.

- **Privacy concerns**: A number of studies have shown that sensitive personal data can be derived from accelerometer readings. Inertial data can reveal information about the activities of a person, its identity, health condition, ... [27]. A good security mechanism for the inertial data is important for the privacy of the user.

2.9 Conclusion

This chapter presented the theoretical background of human activity recognition. HAR recognition follow a generic framework which is called the activity recognition chain. It has 5 main stages. Acquisition of the raw data, pre-processing, segmentation, feature extraction and classification. Using deep learning and neural nets, the feature extraction can possibly be omitted.

The evaluation of the recognition system is also an important part. The different methods and considerations for each of these parts are explained in this chapter and the methods used further in this thesis were justified.

Chapter 3

Experiments and data processing

The data used in this thesis to set up a HAR system was collected from self-designed experiments. In order to have reliable data, a protocol for the experiments was defined. Several people participated and performed different type of activities. This chapter describes more in detail how the experiment is set up and how the data was obtained and processed afterwards.

3.1 Setup of the experiment

3.1.1 Hardware

The activity recognition is done based on inertial data from MIMUs. The used units in the experiment are the *MTw Awinda* wireless motion trackers from *XSens* [36]. These units combine a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer. Its specifications are listed in Table 3.1 and the dimensions in Figure 3.1. A useful aspect of this system is that multiple units can be synchronized while taking measurements. Combinations of signal can therefore also be used (e.g., the difference in acceleration between two sensors). Three units are used in combination with one receiver station.

Feature	Value
accelerometer range	0± 160 m/s^2
gyroscope range	0± 2000 deg/s
magnetometer range	0± 1.9 G
dynamic accuracy gyroscope	0.75 deg RMS (roll/pitch) 1.5 deg RMS (jaw)
static accuracy orientation	0.5 deg RMS (roll/pitch) 1.5 deg RMS (jaw)
max. sample frequency	120 Hz
max. range of transmission (indoor)	20 m
weight	16 g

Table 3.1: Main specifications of the XSens MTW Awinda MIMU sensor

3.1.2 Subjects

Nine subjects (volunteers) participated in the study, ranging from 16 to 54 years old. All subjects are healthy and there are both men and women among the subjects. More physiological information on the subjects is shown in Table 3.2.

ID	**Gender**	**Age**	**Height (cm)**	**Weight (kg)**	**BMI (kg/m^2)**	**Dominant side**
tp 1	Male	23	183	68	20.3	Left
tp 2	Male	22	193	74	19.9	Left
tp 3	Male	22	190	91	25.2	Right
tp 4	Male	16	198	75	19.1	Right
tp 5	Female	54	174	66	21.8	Right
tp 6	Male	54	175	76	24.8	Right
tp 7	Female	24	175	64	21.9	Right
tp 8	Male	25	194	96	25.6	Left
tp 9	Male	17	189	77	21.6	Right
avg	-	28.6	186	76.4	22.1	-
std	-	13.9	8.7	10.3	2.3	-

Table 3.2: Physiological information of the subjects

3.1.3 The protocol

The experiments followed a fixed protocol. Seven basic activities had to be performed by the subjects. These activities include walking, running/jogging, going upstairs, going downstairs, a sit-to-stance, a stance-to-sit and a standing jump. These are chosen in order to have activities of different activity groups. These are listed in Table 3.3. Some activities are very similar, such as sit-to-stance/ stance-to-sit and

walking/ going upstairs/ going downstairs. Optimal feature selection will be needed to properly separate these activities.

The activities were carried out as follows. Running and walking activities were performed in a corridor of the department (20 m long). The subject runs/walks down the corridor once, turns without interruption and returns to the start. So a total of 40 m of walking/running is recorded. The stair climbing and descending is done on a staircase of 30 steps, briefly interrupted by two intermediate floors. The sitting down/ standing up was done with a regular stationary chair. The jumping activity is a standing jump. The subjects were asked to jump twice, with one second in between.

The subject received a signal every time to start the activity. Recording only started when the activity had started. Recording stopped as soon the activity was finished. In this way, the recording only contains the requested activity, and does not have to be cut afterwards.

It is also important for the HAR-system to be person-independent. In order to make the model robust to inter-person variability, the participants were asked to perform the activities in their own natural way. The intra-class variability is further ensured by collecting enough training data. That is why the subjects were requested to repeat each activity five times.

Activity	Activity group	MET
Sit-to-stance	Transitional activity	1.8
Stance-to-sit	Transitional activity	1.5
Downstairs	Medium-level activity	3.5
Walking	Medium-level activity	3.5
Upstairs	High-level activity	4-8
Running (jogging)	High-level activity	7.0
Jumping	High-level activity	8.0

Table 3.3: List of the activities, their division into activity groups and their MET (= ratio of the work metabolic rate to the resting metabolic rate) as given in the compendium of physical activities [2].

The units were placed on the wrist, ankle and on the upper left thigh. These three positions were chosen because these constitute different body parts, which allows to better distinguish the different activities. The wrist and the pocket are also user-friendly placements. If the system would be used in practice, it is more likely that a sensor would be put on the wrist or in a pocket at the level of the thigh, than for example on a person's back or knee.

Afterwards, during the analysis, it was examined which of the three places on the body is the best to distinguish the activities. The MIMU sensors were always put in the same place. The orientation was kept the same. However, arm and leg lengths vary from person to person, so accelerations and speeds may be different. The placement and orientation of the sensors is shown in Figure 3.1.

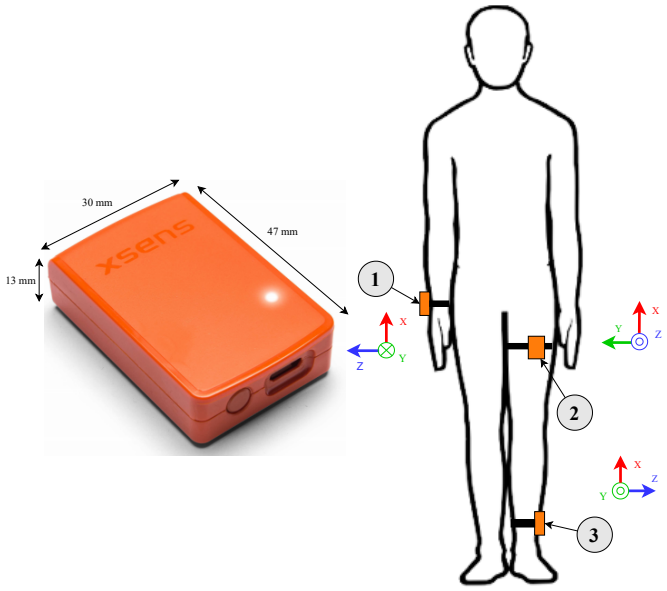


Figure 3.1: Placement of the MIMU sensors. On the left, the dimensions of the sensor is shown. On the right, the placement of the sensors on the right wrist (1), the middle of the upper left thigh (2), and the just above the left ankle (3). The 3-axis symbol is added as a reference for the orientation.

One additional experiment was carried out. Subject 1 performed the protocol a second time, but now with the orientation of the sensors changed. The sensor units were rotated 180° around its y-axis and 180° along the x-axis. Ideally, this would have been done with more subjects. However, because the execution of all activities takes up quite some time, only one subject performed the activities also with a changed orientation of the sensors.

The additional experiment was necessary to afterwards test whether the features are indeed orientation invariant and therefore the recognition is not dependent on the orientation of the sensors on the body.

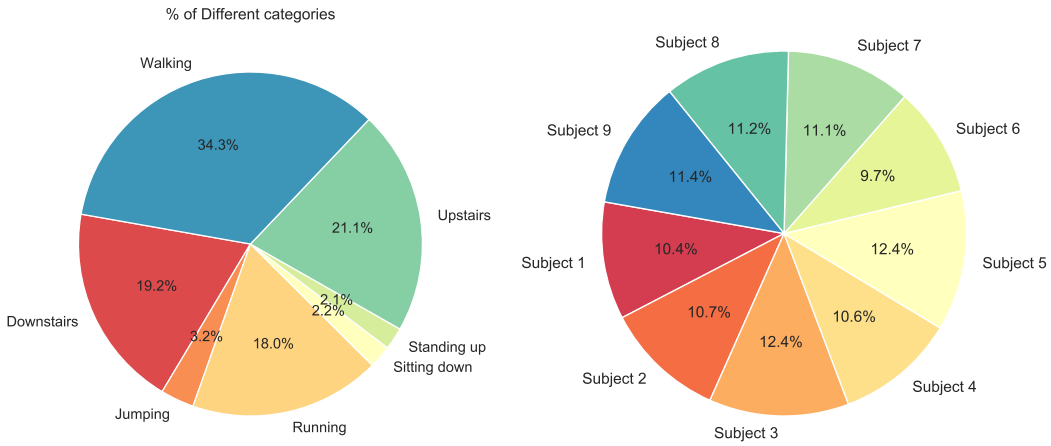
3.2 Data acquisition

The following parameters have been set for the experiments:

- Sample rate f_s: 100 Hz: due to the sampling theorem, the Nyquist frequency is $\frac{f_s}{2} = 50$ Hz. This is the maximum frequency in the signals that can be sampled without loss of information. According to [24], human activity frequencies are between 0 Hz and 30 Hz. As a result, the sample rate is definitely set high enough to ensure that signal variations between samples are not lost. Sampling frequencies that are too high are also not desirable because (too) many samples must be processed and the system will consume more energy. Consequently,

the sample rate could have been set lower to 60 or 80 Hz, which would have been more optimal.

All activities were recorded inside. Finally, each activity trial was exported from the XSens environment as csv files. To facilitate further processing, a template is used for naming the files: "`tp{x}-{y}-{activity}-{sensor}.csv`". With x the number of the subject, y the activity trial number, the activity name and the ID of the sensor respectively. The distribution of the obtained data set is shown in Figure 3.2. As can be seen, the data set is imbalanced. This is because for example the walking and running activity took more time to perform than the 'standing up' or 'sitting down' activities. Hence, to compare models, metrics were used that are independent of class imbalance, such as precision and $F1$-score.



(a) activity-wise

(b) subject-wise

Figure 3.2: Distribution of the collected data

Some example data from the recorded activities is shown in Figure 3.3 on the following page. The running and walking activities have a clear repetitive pattern. The acceleration and angular velocity patterns of the stance-to-sit and sit-to-stance activity are very similar.

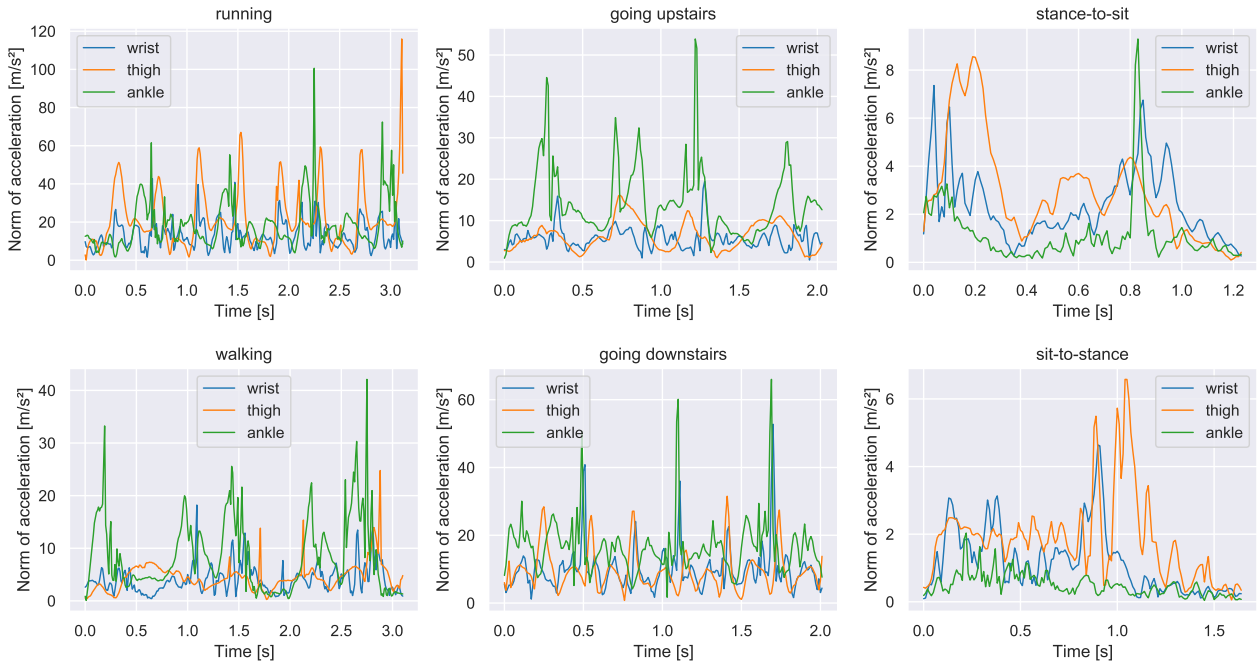


Figure 3.3: Some example accelerometry data from the different activities

3.3 Data processing

The HAR-system was implemented in Python. The *Pandas* library was used for handling the data, and *Scikit-learn* for the classification algorithms [35] [37]. The code implementation has been made public and can be viewed on GitHub [1]. All computing tasks are done on an Acer laptop with an Intel Core i7-8550U CPU @1.80 GHz, 4 cores, 8 logical processors and 8GB RAM.

3.3.1 Pre-processing

Data packets that were not received during the execution of an activity are retransmitted after a recording. These packets were put back in the right place in the time series. The gravitational component also was separated from the body acceleration. This is done automatically by the XSens system. Afterwards, the data from the 3 sensors are merged into one table. This was done with the `preprocess(...)` function.

3.3.2 Segmentation

Each activity trial was divided into time-window segments. This was done with the `segmentation(...)` function. It takes the pre-processed data, the windows length and the amount of overlap as an input. It returns a list containing the segments and the activity label of each segment.

[1] link: `https://github.com/simonperneel/MAI-Thesis-HAR`

3.4 Conclusion

This chapter showed how the experiment was set up, and the data that was acquired. It describes how the data was further processed, up to the segmentation of the time series. The next stage in the activity recognition chain is the feature extraction and the classification. These steps are discussed in more detail in the following chapters, since these are crucial steps.

Chapter 4

Identification of optimal feature vectors

This chapter aims to identify an optimal set of features that enable good activity recognition. An optimal feature vector is a set of features that enables a high classification accuracy, while the computational load is kept low. There are two main approaches for time series classification. These are shown in Figure 4.1. The *feature engineering* approach aims to construct and select good features from the time series. This is the traditional method. These features are manually chosen or 'handcrafted'. The extracted features are fed in a discriminative classifier. There also exist software packages that automatically extract and select features from time series.

The *end-to-end* classifier directly maps the raw input data to a probability of each class at the output. The feature learning and classification is incorporated in one model. Neural network architectures like a multi layer perceptron (MLP) or a convolutional neural net (CNN) are used for this. These two approaches are discussed in the following sections.

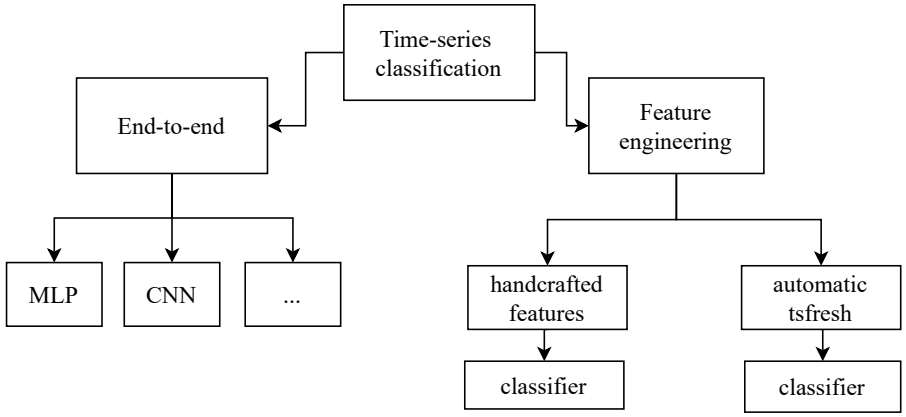


Figure 4.1: Different approaches for TSC

4.1 Classification on raw data

As explained in Section 2.6.6, time series classification can also be done on raw data. The main advantage of this end-to-end approach is that no feature engineering has to be done. The feature learning and classification is combined in one model. This approach is implemented to see how the performance compares to the classification based on handcrafted features.

The *Keras* API in Python is used to program the model. It is a deep learning API running on top of the machine learning platform *TensorFlow*. A multi-layer perceptron model is used. It is the most simple deep learning architecture and has been used in other time series classification tasks [21] [1]. The architecture is shown in Figure 4.2. There are three hidden layers with each 100 neurons. The layers are fully connected. The final layer is a discriminative layer with one node for each class. Each node outputs a probability that the input segment belongs to the class. This probability distribution is ensured by the previous layer. This is a fully connected layer with a softmax activation function and as many neurons as the number of classes.

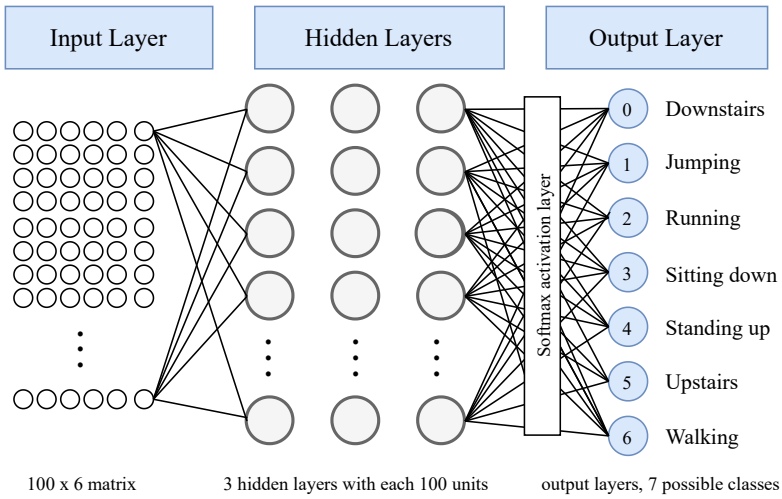


Figure 4.2: used multi-layer perceptron architecture for classification.

With classification on raw data, there is little pre-processing needed. The data just needs to be in the right shape to feed into the network. The magnitude of the acceleration and angular velocity is used as input to obtain orientation-invariance. These two data streams are divided into segments of 100 samples. There are 3 sensor units, with each 2 streams (accelerometer & gyroscope, magnetometer is not used). Therefore, each segment is a two-dimensional matrix of the shape 100x6. These segments are fed into the neural net.

A limitation of the MLP is that each sample is threated independently and has its own weight. Therefore, the time information is lost. The weights in the network are learned during training with an optimization algorithm. This algorithm minimizes the error that comes with a certain value of the weights. The error is calculated using a loss function. The loss function that is used here is categorical cross entropy:

$$\text{Loss}(X) = -\sum_{i=1}^{C} y_i \log \hat{y}_i$$

With X denoting the time series, $\hat{y}_i$ the probability of X having the class y equal to class i out of C classes in the data set.

The model trains up to 20 epochs, and training was stopped when the accuracy of the validation set stopped improving over the last two epochs. This regularization method is called 'early stopping' and is used to avoid that the neural net overfits on the training data.

An overview of the performance of this MLP is displayed in Table 4.1 and the confusion matrix is shown in Figure 4.3 on the next page. The model has an accuracy of 80% with a standard deviation of 5% over all subjects. This is when all three sensors are used. These scores are obtained with the L1O cross-validation method as described in Section 2.7.2. The recognition performance varies strongly from person to person and is not optimal. As can be seen in the confusion matrix, the model has problems with distinguishing similar activities, like sitting down and standing up, and going downstairs versus walking.

In addition, the performance was also examined with only one sensor. The accuracy of recognition decreases 7% with only the sensor on the ankle. It drops further to 54% and 51% with only the measurements from the wrist and thigh respectively. Therefore, one can conclude that, according to the MLP, the sensor on the ankle is the most discriminative to classify these seven activities. a single sensor on the wrist or thigh is not optimal, as the accuracy and F1-score are low.

	All sensors - magn.		Only ankle - magn.		Only wrist - magn.		Only thigh - magn.	
Activity	Precision	F1-score	Precision	F1-score	Precision	F1-score	Precision	F1-score
Downstairs	0.62	0.63	0.46	0.47	0.38	0.20	0.32	0.28
Jumping	0.96	0.74	0.80	0.45	0.67	0.51	0.88	0.41
Running	0.91	0.96	0.93	0.90	0.90	0.93	0.83	0.84
Sitting down	0.70	0.71	0.56	0.48	0.58	0.54	0.46	0.34
Standing up	0.66	0.69	0.50	0.58	0.30	0.32	0.41	0.49
Upstairs	0.84	0.83	0.76	0.78	0.28	0.17	0.38	0.34
Walking	0.81	0.81	0.77	0.77	0.48	0.60	0.48	0.54
Average	**0.79**	**0.77**	**0.68**	**0.63**	**0.51**	**0.47**	**0.54**	**0.46**
Accuracy	**80% ± 5%**		73% ± 8%		54% ± 6%		51% ± 7%	

Table 4.1: Evaluation of the MLP model with L1O cross-validation. The magnitude of acceleration and angular velocity is used as input data.

The performance of an MLP with the acceleration and angular velocity along the x,y,z-axis as input was also evaluated. The disadvantage of this method is that the sensor must always be oriented in the same way on the body, which will not always be the case in real-life. However, the recognition performance is clearly better. Especially the sitting down and standing up activities are now well recognized. The results are shown in Table 4.2. An accuracy of 88% over all subjects is obtained. With a single sensor on the ankle, the accuracy is 85%. The sensor on the ankle is also the most discriminative in this case.

	All sensors - XYZ		Only ankle - XYZ		Only wrist - XYZ		Only thigh XYZ	
Activity	Precision	F1-score	Precision	F1-score	Precision	F1-score	Precision	F1-score
Downstairs	0.74	0.79	0.79	0.75	0.58	0.64	0.62	0.65
Jumping	0.96	0.91	0.66	0.63	0.77	0.65	0.82	0.60
Running	0.91	0.93	0.86	0.90	0.91	0.92	0.85	0.86
Sitting down	0.93	0.93	0.82	0.87	0.76	0.77	0.89	0.89
Standing up	0.90	0.89	0.84	0.77	0.80	0.65	0.89	0.85
Upstairs	0.92	0.94	0.91	0.91	0.49	0.51	0.80	0.75
Walking	0.93	0.87	0.86	0.86	0.78	0.71	0.77	0.78
Average	**0.90**	**0.89**	**0.82**	**0.81**	**0.73**	**0.69**	**0.81**	**0.77**
Accuracy	**88% ± 9%**		85% ± 7%		70% ± 11%		77% ± 10%	

Table 4.2: Evaluation of the MLP with L1O cross-validation. The acceleration and angular velocity along the x, y and z-axis is taken as input data.

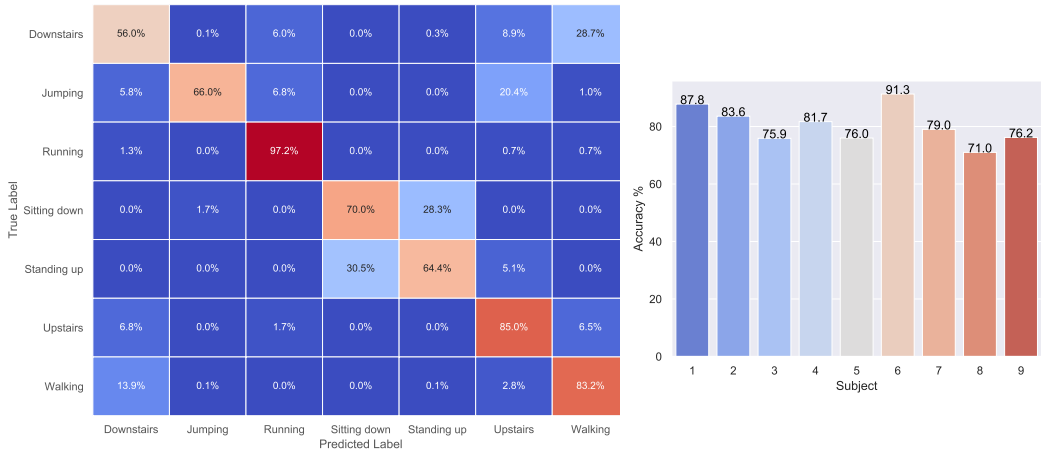


Figure 4.3: Left: Normalized confusion matrix of the MLP model with the magnitude of acceleration and angular velocity of all three sensors as input. The elements on the diagonal are the recall values for each class. Right: Accuracy on each subject with L1O cross-validation

In conclusion, classification on time series data using a MLP gives a reasonable, but not optimal performance. The model recognized well activities like jumping, running and going upstairs, but had problems with activities that are similar like standing and sitting down. This could be due to a lack of training data, which is often a problem with deep learning. Another reason is that the input data (i.e. magnitude of acceleration and angular velocity) do not suffice to distinguish between standing up and sitting down. This will probably require specific features to make this distinction. This is discussed in the next section. With the MLP model, the performance also is very variable from person to person, with an accuracy of 91% for the best, and 71% for the worst recognition. Therefore, it is not very generalizable.

The accuracy increased by 8% on average when using the x,y,z-components of acceleration and angular velocity, but this method is not robust against changes in sensor orientation. The ankle is the best placement on the body for recognizing the activities when using a single sensor for recognition.

4.2 Handcrafted features

4.2.1 Orientation invariance

In order to make the orientation independent of sensor orientation, the L_2-norm or vector magnitude vm is calculated from the tri-axial signals. It is defined as the square root of the sum of the squares of the values along each axis:

$$vm(t) = \sqrt{x(t)^2 + y(t)^2 + z(t)^2}$$

This is calculated for the accelerometer and gyroscope signals. All further extracted features are calculated with the vector magnitude.

The magnetometer measures the magnetic field of the earth, along with any additional magnetic fields or disturbances. Because the magnitude of this vector does not change significantly, it is meaningless to calculate the norm as a feature. Therefore, the magnitude of the magnetic field along each axis is used for feature extraction.

4.2.2 Extracted features

As a starting point, a large number of features are extracted that are commonly used in other HAR-studies. The computed features from the **accelerometer** and **gyroscope** are listed below. These are computed for each of the three units.

- **statistical**: mean, max., standard deviation, kurtosis and skewness
- **time features**: rms, autocorrelation, variance
- **frequency features**: five largest coefficients of the Discrete Fourier Transform (DFT) and the corresponding frequencies of the sensor at the **thigh**.

This results in 68 features (3 units x 2 sensors x 8 features + 1 unit x 2 sensors x 10 features). Features were also derived from the **magnetometer** to examine if they are useful for activity recognition.

- **statistical**: mean, max., standard deviation, kurtosis and skewness
- **time features**: rms, autocorrelation, variance

These are 72 features (3 units x 3 axis x 8 features). A total of 140 features are extracted. This is far too much for a lightweight classifier. It takes a long time to calculate all these features. Some features are highly correlated and therefore redundant. Other features may not be relevant because they provide no useful information of the activity. The number of features therefore can be strongly reduced. The next section aims to identify the optimal set of features for classifying the seven activities.

4.2.3 Feature reduction

Using all the extracted features will not result in the most predictive model. Therefore, one wants to reduce the number of features to the most important ones or the 'optimal' set.

A first way to measure the importance of the features is with a random forest classifier, which consists of several decision trees. The importance score of a feature can be determined based on the reduction in entropy of a node in a decision tree. In each node of the tree, a feature value is evaluated. If a node causes a large reduction of entropy, the

evaluated feature in this node is important. Hence, this feature will get a high important score.

A random forest classifier is fitted on the extracted features to evaluate the importance of all extracted features. All features were then ranked according to their importance. Among the 30 least important features, 20 are from the magnetometer. For each sensor, the average scores of their features were determined. The result is shown in Table 4.3. As can be seen, the average importance score of the magnetometer features is about four times lower than the score of the features from the accelerometer and gyroscope.

	accelerometer features	gyroscope features	magnetometer features
mean relative importance score	0.114	0.104	0.025

Table 4.3: mean importance scores for features from accelerometer, gyroscope and magnetometer. The scores are based on the reduction of entropy when evaluating a feature in a node of a random forest classifier.

A second way to select the most discriminative features for classifying the activities is with SFFS. The working principle of this algorithm is described in Section 2.5.1. A SFFS algorithm has been implemented with the *scikit-learn* library, with a random forest classifier as estimator. The algorithm was set to select the 30 most important features.

The three most important features that were selected are from the ankle and wrist sensor. These are the autocorrelation of the acceleration at the ankle, the RMS value of the angular velocity at the ankle and the standard deviation of the acceleration at the wrist. The values of these top three features are plotted for each activity in Figure 4.4. From the figure, it is clear why these features are important. The running, walking and jumping activity are already fairly distinguishable from the other activities with only these three features.

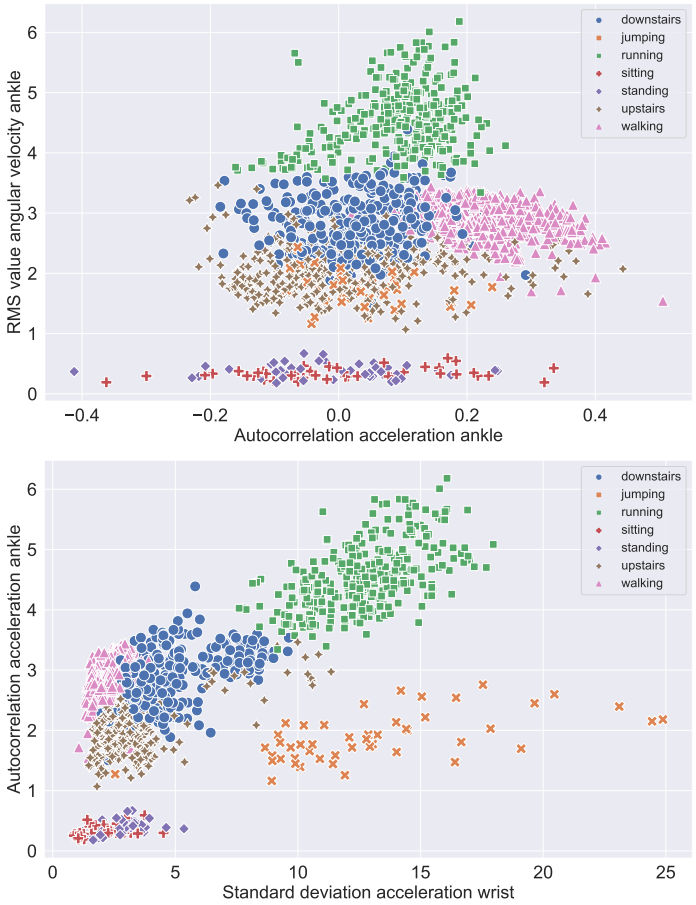


Figure 4.4: Scatterplot of the first three selected features by SFFS plotted for each activity segment

The full list of selected features by SFFS is attached in Appendix A. Of these 30 features, there is only one from the magnetometer. This confirms that these magnetometer-based features are not very useful for recognizing the activities. Therefore, features from the magnetometer are omitted further.

Now, the aim is to reduce the initial set of 140 features to a minimal set of features that characterize the activities well. Therefore, the best features selected by SFFS were added one by one, after which the F1-score is evaluated each time. The F1-score is used because it is a better metric to evaluate the performance of a classifier on imbalanced data sets. A random forest classifier is used as classification algorithm.

The result is shown in Figure 4.5 on the following page. As can be seen, the performance increases strongly with the first three features. After this, it continues to increase gradually with the addition of more features. With the top 17 features selected by SFFS used, a maximum F1-score is reached. The F1-score does not improve much when adding more features after this point. Therefore, we will continue with these 17 features, referenced as the 'minimal set'. This set of features is listed in Appendix A.1. This reduced set of features allows for a more efficient recognition. The classification is faster and less processing needs to

be done. Table 4.4 shows a comparison between the comprehensive set and the minimal set.

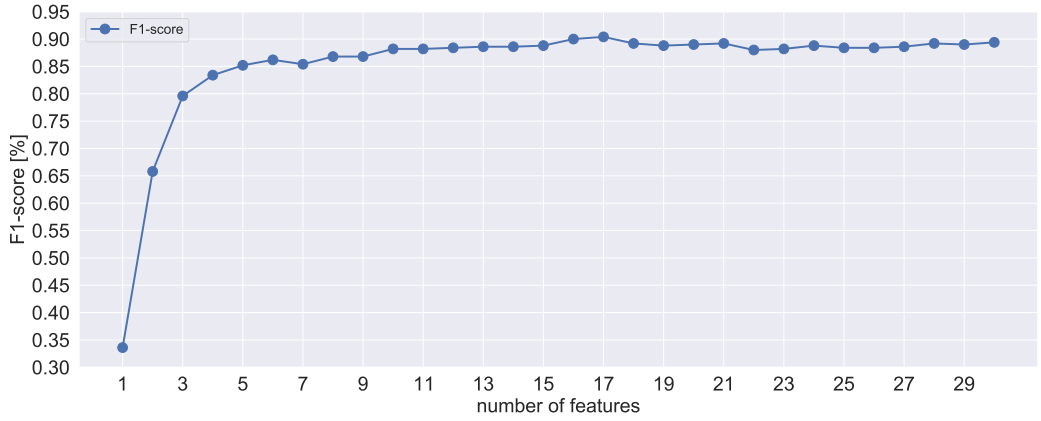


Figure 4.5: Progression of the F1-score as a function of the number of features used in classification. The score is averaged over 5 runs and determined with L1O cross-validation and a random forest classifier.

	initial set extracted features	optimal reduced set with SFFS
number of features	140	17
computing time [s]	72.2	12.1
avg. precision	0.92	0.90
avg. recall	0.92	0.90
avg. $F1$-score	0.92	0.90

Table 4.4: Reduction of the initial set of features to an optimal set of features. Precision and $F1$ score are determined with L1O cross-validation. The computation time is the time needed to calculate the features from all segments in the data set.

With these 17 features, the recognition performance is just slightly less than with all extracted features (average $F1$-score of 0.90 vs. 0.92). The evaluation scores per class and the confusion matrix is displayed in Figure 4.6. As can be seen, all classes are recognized properly with the reduced set of features, except for the 'standing up' and 'sitting down' activity. These are very similar activities that cannot be distinguished with these features. The next section describes an appropriate feature to disentangle these patterns.

Activity	Precision	Recall	F1-score
Downstairs	0.96	0.95	0.96
Jumping	0.98	0.90	0.94
Running	1.00	1.00	1.00
Sitting down	0.70	0.64	0.67
Standing up	0.67	0.76	0.71
Upstairs	0.97	0.96	0.97
Walking	0.97	0.99	0.98
Average	0.90	0.90	0.90
Accuracy		0.96	

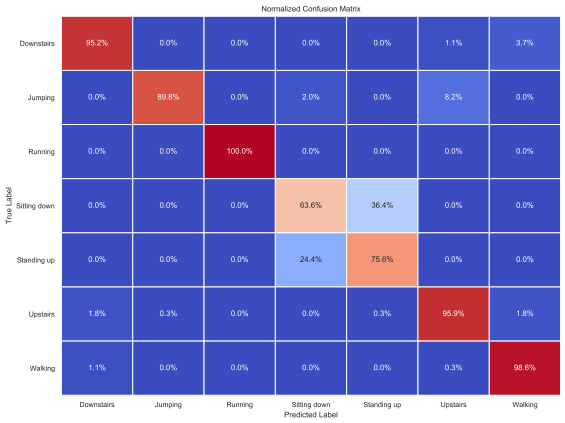


Figure 4.6: Left: Recognition performance with the 17 selected features and a random forest classifier. Accuracy is the average accuracy determined with L1O cross-validation. Right: normalized confusion matrix

A third way to reduce the number of features is with PCA, as explained in Section 2.5.2. For a practical application, it is not very useful to use this method. PCA would need to calculate all features, and then reduce them to the principal components at classification time. This is more time-consuming and inefficient. SFFS is better in this regard. It can select the most relevant features in advance. When a system is deployed, only the reduced number of features need to be calculated. This is why the PCA feature reduction method was not implemented in this thesis.

4.2.4 Additional feature

The features from the accelerometer and gyroscope turned out not to be sufficient to distinguish between the sit-to-stand and stand-to-sit activity. After all, the acceleration and rotational speed of these activities are the same. The only difference is that the action is inverse in time. Thus, there is still an additional feature needed to distinguish between the two and recognize them properly.

The rotation of the sensor at the thigh is appropriate for this. The rotation direction of the thigh is different for standing up and sitting down. The XSens MTW sensors provide the orientation between the sensor-fixed coordinate system, and the earth-fixed reference coordinate system (= X positive pointing to magnetic North, Y according to right-handed coordinates (West) and Z positive pointing up) [36]. In this way, the orientation of the sensor with respect to the earth is known. This allows to determine the rotation of the sensor at any time.

As a feature, the orientation of the sensor at the beginning of a segment is compared with the orientation at the end of the segment. It can be deduced now how the sensor rotates with respect to the earth reference frame. Figure 4.7 shows an example. When a person is going to sit, the orientation of the sensor at the thigh changes. In this case, the sensor rotates clockwise around the y-axis. When the person is standing up, the sensor will rotate counter clockwise around the y-axis. This can be used to distinguish between the two activities.

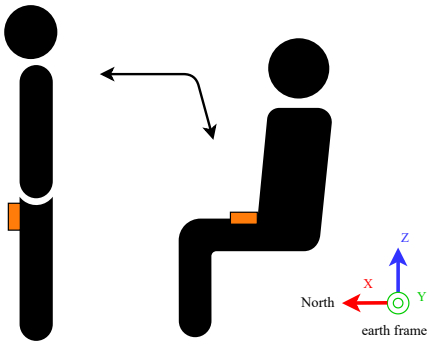


Figure 4.7: Stance-to-sit and sit-to-stance activity - rotation of the thigh sensor

The feature is implemented in the code as follows. The orientation at the beginning of the activity is given by the quaternion q_1. The orientation at the end is given by the quaternion q_2. The rotation of the sensor is represented by the difference quaternion q_{diff}. It represents the rotation of the sensor. If this difference quaternion is multiplied with the first orientation quaternion, we get the quaternion of the end orientation.

Mathematically:

$$
\begin{aligned}
& q_{\text{diff}} \cdot q_1 = q_2 \\
\Rightarrow\ & q_{\text{diff}} = q_2 \cdot \text{inverse}(q1) \\
& \text{The inverse is equal to the conjugate for unit quaternions:} \\
\Rightarrow\ & q_{\text{diff}} = q_2 \cdot \text{conjugate}(q_1) \\
& q_{\text{diff}} = q_2 \cdot \text{conjugate}(\text{a} + \text{b}i + \text{c}j + \text{d}k) \\
& q_{\text{diff}} = q_2 \cdot (\text{a} - \text{b}i - \text{-c}j + \text{d}k)
\end{aligned}
$$

Finally, this difference quaternion q_{diff} is converted into Euler angles (roll, pitch, yaw) and used as a feature. It tells how the sensor rotates during the activity in terms of roll, pitch and yaw. For example, in Figure 4.7, when the person is sitting down, the sensor at the thigh will rotate clockwise around the y-axis (= negative pitch). When the person is standing up, it will rotate counter clockwise around the y-axis (= positive pitch). This feature is independent of the orientation of the sensor unit because it represents the rotation with respect to the fixed earth frame, and not with respect to the sensor coordinate system.

With this additional orientation feature, it is possible to distinguish between the 'standing up' and 'sitting down' activities. The precision of recognition improves from 70% and 67% to 96% and 94% for these activities. The obtained optimal set thus consists of 20 features; 17 selected by SFFS and these three additional features that reflect the orientation difference. A random forest classifier fitted on this set of features gives a good recognition performance. An accuracy and average F1-score of 97% and is achieved.

Figure 4.8 shows the classification metrics and confusion matrix of the system. The metrics are calculated with L1O cross-validation, where one subject's data is repeatedly left out to use as test set, and the other subject's data is used to train the model.

Activity	Precision	Recall	F1-score
Downstairs	0.95	0.94	0.95
Jumping	1.00	0.96	0.98
Running	1.00	1.00	1.00
Sitting down	0.96	0.98	0.97
Standing up	0.94	0.98	0.96
Upstairs	0.98	0.96	0.97
Walking	0.97	0.99	0.98
Average	0.97	0.96	0.97
Accuracy		0.97	

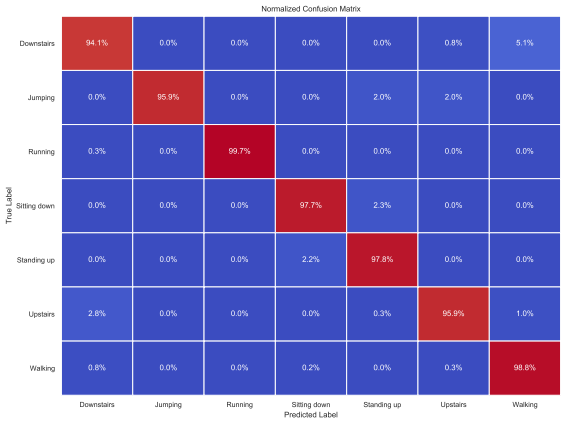


Figure 4.8: Left: Recognition performance with the optimal set of features and a random forest classifier. The scores are determined with L1O cross-validation. Right: normalized confusion matrix.

4.3 Automatic feature extraction

Human activity recognition can be challenging as the construction and extraction of the features has to be done manually and requires domain knowledge of the classification problem. There exist packages that automate the extraction of features from time-series. `Tsfresh` is such a package for python [32]. It calculates automatically features from time series for classification. This package was tested to see if it is usable, and what the advantages and disadvantages are.

First, the package provides functions to calculate a comprehensive set of features from the time series [31]. There are different settings possible that define whether all features, an 'efficient set', or a minimal set of features are calculated. Custom defined features can also be added. The efficient set of parameters are chosen here, which are all the features that are provided by tsfresh, without the computational expensive ones. They range from simple features like mean and maximum, to more complex one like the entropy or Benford correlation.

The second step is to select the relevant features from the comprehensive set of extracted features. The package provides functions to identify the strongly and weakly relevant attributes. It does this by using the fresh algorithm [14]. Fresh stands for FeatuRe Extraction based on Scalable Hypothesis tests. This algorithm selects good features based on hypothesis tests. They individually test each extracted feature for its significance in predicting the class.

4981 features were calculated from the accelerometer, gyroscope and magnetometer, which is computationally very expensive. Then, the number of relevant features to select is set to 20, the same number as previously with the handcrafted features. The set of features that were identified by tsfresh to be optimal are included in Appendix A.2. Also with the automatic feature selection, no features of the magnetometer were chosen.

The performance of a random forest classifier with these 20 selected features was evaluated. The classification metrics and confusion matrix are shown in Figure 4.9. The recognition is slightly less for all classes when compared with the handcrafted features (see Figure 4.8). In

particular, the automatic feature extraction falls short for recognizing the 'standing up' and 'sitting down' activities well. These activities are very similar and need a specific feature as described in previous section.

Activity	Precision	Recall	F1-score
Downstairs	0.67	0.75	0.70
Jumping	0.90	0.96	0.93
Running	0.99	1.00	0.99
Sitting down	0.61	0.70	0.65
Standing up	0.64	0.56	0.60
Upstairs	0.86	0.86	0.86
Walking	0.87	0.81	0.84
Accuracy		0.84	

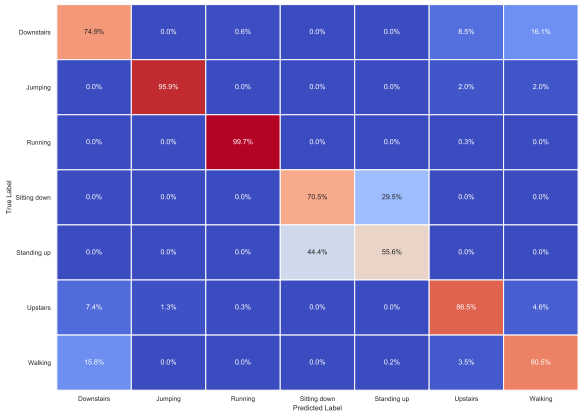


Figure 4.9: Left: Recognition performance with selected features by tsfresh. The scores are determined with L1O cross-validation. Right: normalized confusion matrix.

In conclusion, feature extraction with automatic packages like *tsfresh* are useful in the sense that no domain knowledge about the classification problem is needed. The feature construction and selection are done automatically. Because of this, faster development of the recognition system is possible compared to when it is done manually. However, the features that were selected by the fresh algorithm are not as optimal as the optimal handcrafted features in previous section. The package also does not provide features that allow to distinguish between standing up and sitting down.

4.4 Conclusion

In this chapter, two approaches have been considered for time series classification. With the end-to-end approach, no feature construction is done. The feature learning and classification is done within the same model. The more complex steps of feature construction and selection are omitted, which allows for rapid deployment of the system.

A multi-layer perceptron architecture was implemented, which achieved a reasonable performance with an accuracy and F1-score of 80% and 0.77%. A better performance may be achieved with more advanced architectures. Study [22] achieved an accuracy of 90% with a CNN on a similar data set with the same types of activities to recognize. The drawback of the end-to-end approach and neural networks in general is that you have no interpretability towards which movements or patterns are characteristic of the activity.

A better performance was obtained with the feature engineering approach. An optimal set of 20 features was found for classifying the activities properly. These features are statistical, time signal and frequency-based, plus an orientation feature to distinguish the 'standing up' and 'sitting down' activities.

Both the SFFS method and the importance score-based selection method showed that features of the magnetometer are less useful than features extracted from the accelerometer

and gyroscope measurements. Adding magnetometer features also did not improve the performance of the model. However, measurements from the magnetic field are used to calculate the orientation of the sensor. The XSens units calculates its orientation internally with sensor fusion from the accelerometer, gyroscope and magnetometer. The orientation is used in this work to distinguish between sitting and standing.

Chapter 5

Performance evaluation

At the beginning of the thesis, the goal was set to implement an activity recognition system with a high performance. In this chapter, the different parameters of the activity recognition chain are investigated in order to achieve this optimal recognition.

First, the influence of the time window size is identified. Next, five commonly used classifiers are assessed; the performance of these classifiers is compared using cross-validation.

A second point that was made, was that the recognition system should be invariant of sensor orientation. For this reason, we examined if the used features are indeed orientation-invariant. In the last part of the chapter, we check which of the three sensor placements is best for classifying the activities.

5.1 Influence of the window length

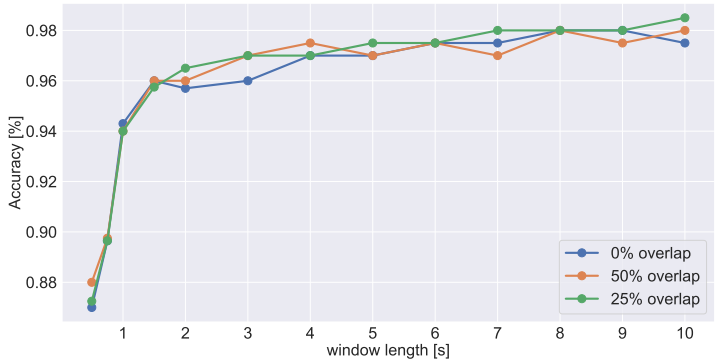
The segmentation of the signals is an important stage in activity recognition. Fixed-length time windows are commonly used, but there is no agreement on the optimal size of the windows. Smaller windows enable a faster recognition of the activity and require less processing. Contrastingly, larger window sizes can enclose better the typical pattern of an activity and may perform better.

In other studies, window sizes from 0.1 s to 12 s have been used. Therefore, we compared 13 values for the window length ranging from 0.5 s to 10 s. The movements were sampled with a frequency of 100 Hz, so a window size of 1 s corresponds to 100 data points. Furthermore, 3 values for overlap were tested: 0, 25 and 50% overlap. The scores are obtained with a random forest classifier and the 20 selected features from the previous chapter.

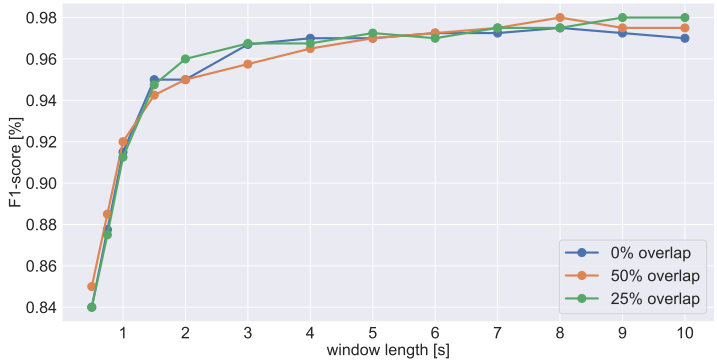
The effect on the performance of recognition for different window sizes is presented in Figure 5.1. The same trend can be observed with the accuracy and F1-score. The worst performance is obtained with the smallest window of 0.5 s. Also, the performance increases as the length of the window increases.

However, from a window length of 4 s, the performance improvements are negligible. The biggest leap in performance is from 0.75 s to 1 s, with an improvement of 4% in accuracy and F1-score.

The amount of overlap between the segments has no major effect on performance. Nevertheless, an overlap of 25% seems the best for most sizes.



(a) accuracy scores



(b) F1-scores

Figure 5.1: Effect on the performance of recognition for different window sizes and amount of overlap

A window size of 3-4 s with an overlap of 25% seems to be optimal. These parameter settings present a good trade-off between performance (high accuracy) and speed of the recognition (short window). These parameter settings are similar to what was found to be optimal in other studies. Study [26] used a window size of 3.2 s with no overlap between the segments, on a similar dataset. Study [**?**] uses a window size of 4 s with 50% of overlap. Study [8] indicated that the optimal size for 'energetic' activities (i.e. walking, jogging, jumping, etc.) is between 1 and 3.25 s. Therefore, a window size of 3 s and 25% overlap are used in further models.

5.2 Classification and computational performance

The selection of the optimal features was discussed in the previous chapter. After this reduction to 20 optimal features, the next step in the activity recognition chain is the classification. There is no 'silver bullet' classifier that works best on all problems.

For this reason, six classifiers are implemented using the supervised learning tools from the *scikit-learn* library. These include a Random Forest (RF), Support Vector Machines (SVM), *k*-Nearest Neighbour (kNN), Decision Tree (DT), Multi-Layer Perceptron (MLP) and Naive Bayes classifier (NB). Their working principles are explained in Chapter 2.

Chosen hyperparameters

The hyperparameters for the different classifiers are tuned with a grid search. This is a tuning technique that tries to find the optimal values for the hyperparameters. It does an exhaustive search on different combinations of possible parameter values of the model. The obtained and used parameters are listed below.

- **RF**: 100 estimators, Gini split criterion for the nodes, max tree depth of 20.
- **SVM**: linear kernel, loss function: squared hinge, regularization parameter C=1, one-vs-rest multiclass strategy.
- **kNN**: number of neighbours: 5, distance metric: Euclidean distance.
- **DT**: entropy split criterion for the nodes, no maximum tree depth.
- **MLP**: 1 hidden layer with 100 units, *Relu* activation function, *adam* optimizer algorithm for training, regularization parameter α=0.0001.
- **NB**: assumed Gaussian distribution for the likelihood of the features.

Figure 5.2 presents the results. The $F1$-score is used to compare the classifiers. As explained in Section 2.7.3, this is a better metric for evaluating models on an imbalanced dataset. The scores are determined with two cross-validation techniques to see if the same trend holds. The black error bars indicate two standard deviations around the mean score for each iteration in the cross-validations. The scores are an average over 5 runs. As expected, the scores with L1O cross-validation are a bit lower than with 10-fold cross-validation. This is because with L1O, the test data comes from an 'unseen' subject, which is more challenging to recognize properly. While with 10-fold cross-validation, the training and test data possibly contain data from the same subject.

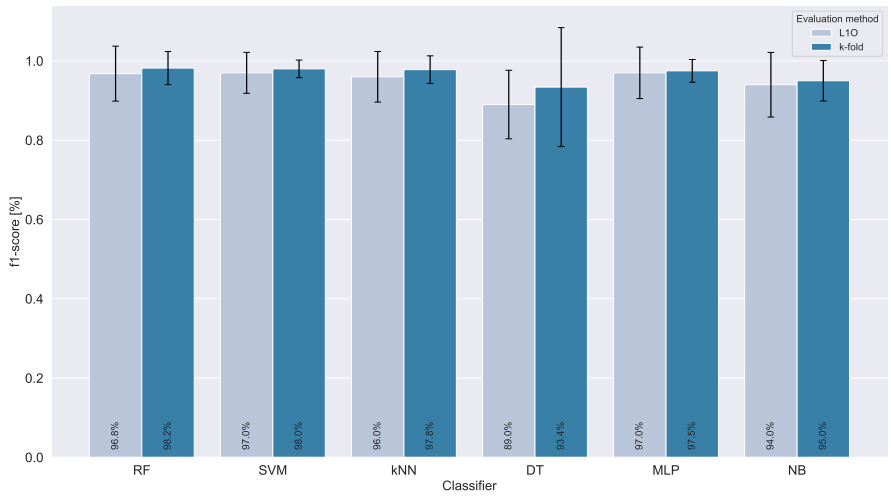


Figure 5.2: $F1$-scores for each classifier. Scores are determined with L1O and 10-Fold cross-validation. The error bars show ± two standard deviations around the mean of the different iterations of the cross-validation.

In addition to the accuracy of the model, the computational complexity is also important for practical applications. The classification speed and training time were determined for each classifier. Especially the prediction speed is important for an efficient model since

training of the classifier can be done in advance. Table 5.1 presents the average times over 5 runs. The unit of the times is in milliseconds.

	L1O						10-Fold					
	RF	SVM	kNN	DT	MLP	NB	RF	SVM	kNN	DT	MLP	NB
training time [ms]	426.9	128.2	3.1	65.2	2929.4	3.8	454.7	125.2	5.04	54.3	2522.1	5.1
classification time [ms]	13.81	2.26	18.6	2.7	3.5	1.8	14.5	2.1	18.4	2.1	3.0	2.0

Table 5.1: The training time is the time needed to fit the model for a single iteration in the cross-validation. The classification time is the time needed to predict the class for the test feature vectors in one iteration of the cross-validation. The times are averaged over 5 runs.

Discussion
All classifiers have a high $F1$-score. Only the decision tree has a distinct lower score. The SVM and MLP classifier obtained the largest score of 97% with L1O cross-validation. The RF classifier has the largest score of 98.2% with 10-fold cross-validation. However, the scores of the SVM, MLP, RF and kNN classifier are very close. The differences between these four are all within 1%. This indicates that the selected features characterize the activities well.

the kNN classifier is effective and has almost no 'training' time because the feature vectors just have to be stored in memory. However, it takes significantly more time to predict the class because the closest neighbour feature vectors have to be calculated during classification. Storing the training instances also requires a lot of memory. Because of this, the kNN classifier is not suitable for use in practical applications.

The MLP classifier has a good F1-score and a fast classification time, but its training time is significantly longer in comparison with the others. This is because the optimization of the weights is computationally expensive. The training of the random forest classifier is less expensive. This classifier also achieves a high score of 96,8%. The Naive Bayes classifier trains and classifies fast, but its $F1$-score is 3% less than the best classifier. Finally, the decision tree obtained the lowest score, and the $F1$-score varies widely from subject to subject. It may be a too simple model for this classification problem.

Overall, the best classifier is the SVM classifier. It has the highest accuracy and the variation of the score for each subject is smaller than for the other classifiers. Also, it is the second fastest classification algorithm. The SVM classifier has the best combination of performance and recognition speed. If one of the classifiers would be deployed in an embedded system with online classification, an SVM classifier is the best option here.

5.3 Orientation invariance test

By using the vector magnitude of the acceleration and angular velocity, the recognition should be robust against orientation changes. To test this, the seven activities were also performed with a rotation applied to the original orientation of the sensors. Otherwise, the protocol was the same.

The recognition of the activities performed with the rotated sensors is validated equal to the earlier models. The model is trained on the data of the subjects whose sensors were similarly oriented. But now the data of the person with rotated sensors is used as test data. This is repeated nine times, where each time one subject is left out from the train data to

obtain cross-validation. A random forest classifier is used.

The metrics are compared with the recognition performance achieved before, whereby the orientation of the sensors remained the same between the subjects. The results are presented in Table 5.2. As can be seen, the scores of the activities performed with rotated sensors are just slightly less. The difference is negligible, it is probably due to the variation in execution of the activities by the subject. This confirms that the used features with the vector magnitude are indeed orientation-invariant. More different orientations could be tried, but the recognition performance will most likely be the same.

	Reference orientation	Rotated sensors
accuracy	0.982	0.98
precision	0.988	0.974
recall	0.992	0.962
F1-score	0.982	0.964

$F1$**-scores**	Reference orientation	Rotated sensors
downstairs	0.958	0.958
jumping	1	0.966
running	1	1
sitting	0.996	0.936
standing	0.996	0.936
upstairs	0.998	0.998
walking	0.976	0.97

Table 5.2: Performance of recognition compared for the activities done by the subject with rotated sensors vs. the activities done by the subjects who had the same orientation of their sensors. Left: overall scores, Right: $F1$-scores per activity type.

5.4 Recognition with a single sensor

So far, features have been used from all three sensors on the body. With more sensors it is easier to achieve a good recognition performance because more measurements are available at different places on the body. In this section we look at how good the recognition is with one of the three sensors. From this, it can be deduced what the better placement is in order to recognize the activities.

Wrist sensor
Only the measurements from the wrist sensor are now used for recognizing the activities. Statistical, time and frequency features are extracted again from the accelerometer and gyroscope (same as in Section 4.2.2). Features from the magnetometer are left out, since it turned out in the previous chapter that they do not provide additional useful information. A total of 45 features are computed. The extracted features are further reduced to the 20 best features with the SFFS algorithm.

Thigh sensor
The same process is repeated for the thigh sensor. Now, only the measurements of the sensor at the upper thigh are considered. The same types of features are extracted (45 in total). SFFS selects the best 20 features from this. The selected features do differ from the selected features from the wrist or ankle sensor. This is because the movements differ on the different body parts, and thus also the most important features for each sensor.

Ankle sensor
The same process is also done for the ankle sensor. Features are extracted only from the measurements of the ankle sensor. Afterwards, 20 features are selected by SFFS.

Discussion

The performance of recognition for each sensor individually is compared with a random forest classifier and L1O cross-validation. Figure 5.3 compares the precision of the prediction for each class. Figure 5.4 compares the average scores for the recognition with a single sensor vs. recognition with all sensors.

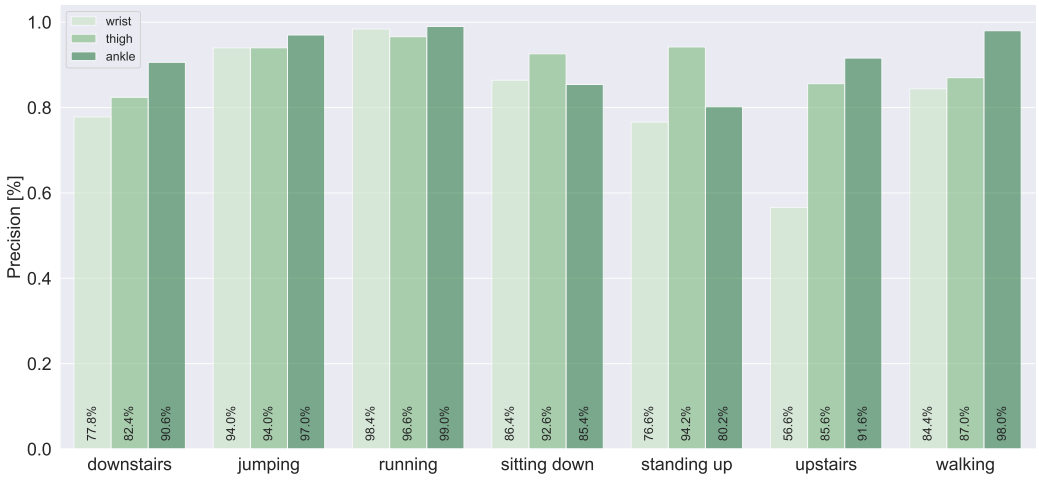


Figure 5.3: Precision compared for each activity when only one of the three sensor is used for classification. A random forest classifier is used, and the scores are determined with L1O cross-validation

The sensor on the ankle has the highest precision for classifying most classes. Only for the 'standing up' and 'sitting down' classes, the sensor at the thigh gets a higher score. This is because the thigh moves when a person stands up or sits down, and this movement has been defined in a feature. While the ankle does not move much, which makes it more difficult to distinguish between standing up and sitting down. When these two activities would be defined as a single type of activity, it would be easily recognized. The ankle sensor especially has a higher precision in recognizing going downstairs and walking.

The sensor at the thigh comes in second best for recognizing the activities. Its overall precision and $F1$-score are similar to these of the ankle sensors, because it recognizes better standing up and sitting down.

The sensor at the wrist performs the worst. As can be seen in Figure 5.4, the average precision and $F1$-score is about 10% lower than recognition with a sensor on the thigh or ankle. The wrist sensor does recognize high-intensity activities well (e.g., running and jumping). But it has difficulty distinguishing between walking, going upstairs and going downstairs. The wrist movements for these activities are indeed very similar, which makes it difficult to recognize them properly.

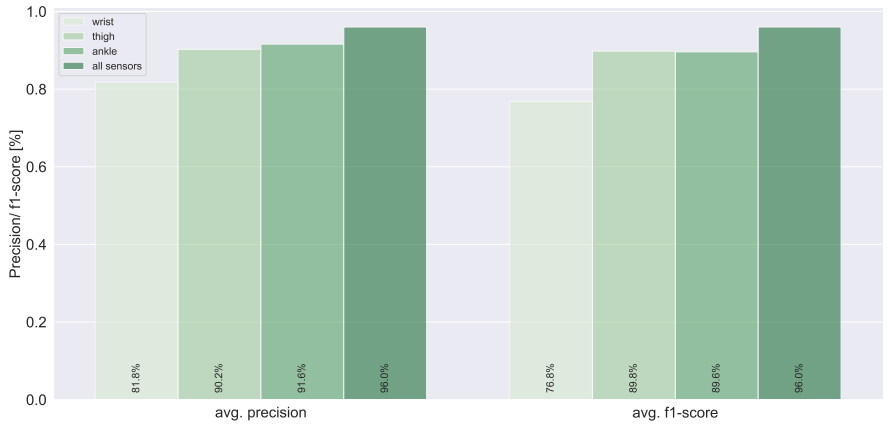


Figure 5.4: average precision and F1-score on classification when one of the three sensors is used vs. when all sensors are used.

In summary, sensors that are placed on the leg are more discriminative for recognizing these seven activities. This corresponds with what was seen with the selection of the optimal features in previous chapter; more than 2/3 of the features were derived from the leg sensors. The placement of the sensor on the ankle is more optimal than on the thigh. The thigh sensor is only favored for recognizing standing up/sitting down, due to the specific motion of these activities. The wrist sensor has the lowest performance of the three.

The recognition is the best with all three sensors. The average precision and F1-score are about 5% higher than the recognition with a single sensor on the ankle.

5.5 Conclusion

The optimal parameter for window length has been determined. The recognition was tried out with different window sizes and amount of overlap. Evaluation using cross-validation showed that the optimal window size is around 3-4 s. This is a good trade-off between performance and recognition speed. The minimal, 'cut-off' window size for a good performance is 1 s. Below this window size, the accuracy and F1-score drops strongly.

Different classifiers are implemented and tested. The RF, SVM, kNN and MLP classifiers have a similar, high F1-score. Still, the SVM classifier is the most optimal. It achieves the largest $F1$-score, while its training and classification time are among the best. In addition, the deviation from the average F1-score is also the smallest at SVM. That is, the recognition system is more generalizable across subjects with an SVM classifier than with others.

The recognition with one of the three sensors was also considered. The sensor placed on the ankle was the best in recognizing most type of activities. The sensor on the thigh has a lower performance in recognizing the activities, except for the 'standing up' and 'sitting down' activities. The sensor at the wrist achieves a significantly lower performance. Some of the recorded type of activities have very similar wrist movements, which makes the recognition more difficult. By using the vector magnitude, information on the directions of angular velocity and acceleration is lost. This is less of a problem for recognition with the ankle or thigh sensor because it still recognizes the activities well. But the wrist sensor may lack this information to be able to distinguish better between walking up/ going upstairs/ going downstairs.

The recognition with features from all three sensors is the best. When the optimal parameters and the optimal features from previous chapter are used, the recognition is almost flawless. The performance of this model is depicted in 5.5.

Activity	Precision	Recall	F1-score
Downstairs	0.99	0.94	0.97
Jumping	1.00	0.98	0.99
Running	1.00	1.00	1.00
Sitting down	0.98	0.98	0.98
Standing up	0.96	0.98	0.97
Upstairs	0.99	0.98	0.98
Walking	0.98	1	0.99
Average	0.99	0.98	0.98
Accuracy		0.97	

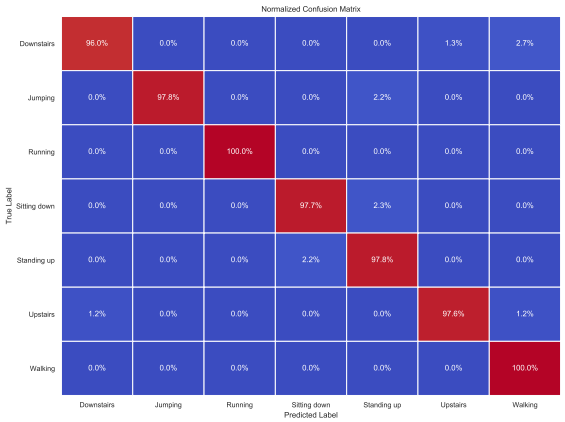


Figure 5.5: Left: metrics of the classification with the optimal parameters (window size of 4 s, 25% overlap, SVM classifier). The scores are obtained with L1O cross-validation. Right: normalized confusion matrix.

Chapter 6

Conclusion

Human activity recognition with inertial sensors is an emerging field of study. Activity recognition systems can be useful in medical, health and security applications, among others.

A lot of features can be derived from MIMU sensors to be used for AI methods. However, it is important that the recognition system is lightweight to be usable in embedded devices. Hence, it is of interest to identify a minimal number of features to reliably recognize the activities. Fewer features reduce the required processing, memory and energy requirements of the system.

The data was collected trough dedicated experiments. Nine subjects participated and performed a number of common activities following a fixed protocol. Three MIMU sensors were placed on the wrist, ankle and upper thigh. The performed activities include running, walking, going upstairs, going downstairs, jumping, sitting down and standing up.

This thesis aimed to implement an activity recognition system with a high performance, while using a minimal set of features. Two approaches for recognition of the time series data have been considered.

The classification on raw data or the end-to-end approach incorporates the feature learning and classification in one model. The main advantage of this method is that it eliminates the more complicated steps of feature extraction of traditional HAR system. When the model has finished training, the recognition can also be done faster, as no features have to be calculated.

A deep neural network has been implemented according to this approach. This neural net has a reasonable performance, with an accuracy of 80% and average F1-score of 77% using all three sensors. However, the model has difficulties for clearly identifying standing up and sitting down, as well as going down vs. walking. The recognition with this model was not optimal yet. A possible reason is that there is too little training data. There is also still potential in improving the model, e.g. by using more advanced architectures like a CNN.

A better performance was achieved with the traditional approach of feature construction. As a starting point, a large number of statistical, time-domain and frequency domain features were extracted from the time series data. With the SFFS feature selection algorithm, this was reduced to 20 optimal features. As in other studies, statistical features (i.e. mean, max, kurtosis) and time-domain features (i.e. autocorrelation, RMS) also proved to be effective here. A minimal feature set was thus found that allows for a reliable recognition of the seven activities.

Another objective of this work was to verify the usefulness of features from the magnetometer. When the importance score is calculated for every extracted feature by means of a random forest classifier, the average score of the magnetometer features is about four times lower than features from the accelerometer and gyroscope. This corresponds with the result from the SFFS algorithm. Of the 30 features picked by the SFFS algorithm as the best features, only one is from the magnetometer. Therefore, if accelerometer and gyroscope measurements are available, features from magnetometer have no added value.

The best position to wear the sensor was also studied. Therefore, the recognition performance with a one of the three sensors was considered. The sensor placed on the ankle achieved the best performance overall, both for the end-to-end classification as well as the classification with feature construction. This is the optimal placement for recognizing most activities. However, only the sensor at the thigh makes it possible to distinguish between standing up and sitting down. The thigh sensor is the second-best placement. Recognition with a single sensor on the wrist achieves a significantly lower performance. However, it should be noted that the optimal placement of the sensors is highly dependent on the types of activities that have to be recognized.

In the last part of the thesis, the recognition system was optimized. The optimal window size has been determined at 3-4 s. It is a good trade-off between recognition speed and recognition performance. The performance drops significantly with window sizes less than 1 s. The SVM classifier was found to be the best classifier because of its high prediction accuracy, while the prediction time is still short.

The proposed recognition system achieves an overall accuracy and F1-score of 0.97 and 0.98 with all three sensors, respectively. These scores are determined with Leave-One-Out cross-validation. The validation method ensures that high scores reflect an user-independent recognition. The recognition performance of the system is orientation independent.

In this work, the data of the sensors on the different body parts was used independently. An interesting approach would be to combine the different time series into additional signals. From these signals, one could extract new features such as the difference in acceleration between two sensors. With these types of handcrafted features, it may be possible to achieve an optimal recognition with even less features.

Future work could further exploit the benefits of end-to-end classification methods. Good results were achieved in other studies last years with convolutional neural network architectures [52] [22]. However, the models of these studies are not robust to orientation changes of the sensor. Future work could focus on end-to-end classifiers whose performance is independent of sensor orientation.

Appendices

Appendix A

Identified optimal features

A.1 Selected optimal features by SFFS

Selected feature	name		name
1	autocorr. acceleration ankle (lag 10)	16	freq. largest peak of the DFT of the angular velocity thigh
2	RMS angular velocity ankle	17	mean angular velocity ankle
3	std. acceleration wrist	18	mean angular velocity thigh
4	autocorr. angular velocity wrist (lag 10)	19	kurtosis acceleration ankle
5	skewness angular velocity ankle	20	std. angular velocity ankle
6	RMS acceleration thigh	21	mean acceleration thigh
7	std. angular velocity wrist	22	variance acceleration ankle
8	autocorr. angular velocity thigh (lag 10)	23	max. magnitude magnetic field X-axis thigh
9	RMS angular velocity thigh	24	max. acceleration wrist
10	std. acceleration thigh	25	RMS acceleration ankle
11	skewness acceleration thigh	26	freq. second largest peak of the DFT of the acceleration thigh
12	RMS acceleration wrist	27	autocorr. angular velocity ankle (lag 10)
13	mean acceleration wrist	28	autocorr. acceleration wrist (lag 10)
14	variance acceleration thigh	29	kurtosis angular velocity wrist
15	max. angular velocity ankle	30	kurtosis angular velocity thigh

Table A.1: top 30 selected features by Sequential Forward Feature Selector algorithm. The first 17 features are used in the optimal set.

A.2 Selected optimal features by tsfresh

These are the top 20 features that were selected by the fresh algorithm with the tsfresh automatic feature construction package.

Selected feature	name		name
1	autocorr. acceleration ankle (lag 6)	11	acceleration thigh quantile 0.7
2	autocorr. acceleration ankle (lag 7)	12	acceleration wrist FFT coefficient 0
3	autocorr. acceleration ankle (lag 5)	13	angular velocity ankle quantile 0.2
4	c3 statistic angular velocity ankle (lag 1)	14	acceleration thigh AR coefficient 0
5	mean angular velocity ankle	15	acceleration wrist quantile 0.2
6	angular velocity ankle quantile 0.9	16	median angular velocity ankle
7	mean acceleration wrist	17	c3 statistic angular velocity ankle (lag 3)
8	abs. energy acceleration wrist	18	c3 statistic angular velocity ankle (lag 2)
9	sum values acceleration wrist	19	acceleration thigh quantile 0.6
10	median acceleration thigh	20	acceleration thigh change quantile 0.8

Table A.2: top 20 relevant features selected by fresh algorithm

Appendix B

Software

The written software for this thesis can be downloaded from the following repository on GitHub: `https://github.com/simonperneel/MAI-Thesis-HAR`. The repository also includes documentation and installation instructions.

The used libraries and their versions are displayed below:

type	library	version
deep learning	keras	2.4.3
	keras-applications	1.0.8
	keras-base	2.4.3
	keras-preprocessing	1.1.2
	tensorflow	2.3.0
machine learning	scikit-learn	0.24.1
feature extraction	tsfresh	0.17.0
math	numpy	1.19.2
	scipy	1.6.0
visualization	matplotlib	3.3.4
	seaborn	0.11.1
data analysis	pandas	1.2.1

Table B.1: Used python libraries and their versions

Bibliography

[1] N. Ackermann. Human activity recognition tutorial with keras and core ml. https://towardsdatascience.com/human-activity-recognition-har-tutorial-with-keras-and-core-ml-part-1-8c05e365dfa0, 2018. Accessed on 2021-04-25.

[2] B. E. Ainsworth, W. L. Haskell, S. D. Herrmann, N. Meckes, D. R. Bassett Jr, C. Tudor-Locke, J. L. Greer, J. Vezina, M. C. Whitt-Glover, and A. S. Leon. 2011 compendium of physical activities: A second update of codes and met values. Available at https://sites.google.com/site/compendiumofphysicalactivities/compendia, 2011.

[3] K. Altun and B. Barshan. Human activity recognition using inertial/magnetic sensor units. In *Human Behavior Understanding*, volume 6219 of *Lecture Notes in Computer Science*, pages 38–51. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[4] L. Atallah, B. Lo, R. King, and G.-Z. Yang. Sensor placement for activity detection using wearable accelerometers. In *2010 International Conference on Body Sensor Networks*, pages 24–29, 2010.

[5] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou, and Y. Amirat. Physical human activity recognition using wearable sensors. *Sensors*, 15(12):31314–31338, 2015.

[6] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou, and Y. Amirat. Physical human activity recognition using wearable sensors. *Sensors (Basel, Switzerland)*, 15(12):31314–31338, 2015.

[7] M. S. H. Aung, S. B. Thies, L. P. J. Kenney, D. Howard, R. W. Selles, A. H. Findlow, and J. Y. Goulermas. Automated detection of instantaneous gait events using time frequency analysis and manifold embedding. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 21(6):908–916, 2013.

[8] O. Banos, J. Galvez, M. Damas, H. Pomares, and I. Rojas. Window size impact in human activity recognition. *Sensors (Basel, Switzerland)*, 14:6474–99, 04 2014.

[9] B. Barshan and A. Yurtman. Classifying daily and sports activities invariantly to the positioning of wearable motion sensor units. *IEEE internet of things journal*, 7(6):4801–4815, 2020.

[10] A. Bevilacqua, K. MacDonald, A. Rangarej, V. Widjaya, B. Caulfield, and T. Kechadi. Human activity recognition with convolutional neural networks. 09 2018.

[11] H. Blockeel. *Machine Learning and Inductive Inference*. Acco, 2018-2019 edition, 2018.

[12] A. Bulling, U. Blanke, and B. Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46, 01 2013.

[13] D. M. Burns, N. Leung, M. Hardisty, C. M. Whyne, P. Henry, and S. McLachlin. Shoulder physiotherapy exercise recognition: machine learning the inertial signals from a smartwatch. *Physiological measurement*, 39(7):075007–075007, 2018.

[14] M. Christ, A. W. Kempa-Liehr, and M. Feindt. Distributed and parallel time series feature extraction for industrial big data applications. oct 2016.

[15] W. F. Fadel, J. K. Urbanek, S. R. Albertson, X. Li, A. K. Chomistek, and J. Harezlak. Differentiating between walking and stair climbing using raw accelerometry data. *Statistics in biosciences*, 11(2):334–354, 2019.

[16] D. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[17] A. Godfrey, R. Conway, D. Meagher, and G. OLaighin. Direct measurement of human movement by accelerometry. *Medical engineering & physics*, 30(10):1364–1386, 2008.

[18] A. Godfrey, R. Conway, D. Meagher, and G. OLaighin. Direct measurement of human movement by accelerometry. *Medical engineering & physics*, 30(10):1364–1386, 2008.

[19] E. Guenterberg, S. Ostadabbas, H. Ghasemzadeh, and R. Jafari. An automatic segmentation technique in body sensor networks based on signal energy. 04 2009.

[20] M. Guo, Z. Wang, N. Yang, Z. Li, and T. An. A multisensor multiclassifier hierarchical fusion model based on entropy weight for human activity recognition using wearable inertial sensors. 49(1):105–111, 2019.

[21] I. F. Hassan, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 07 2019. Copyright - Data Mining and Knowledge Discovery is a copyright of Springer, (2019). All Rights Reserved; Last updated - 2019-06-13.

[22] A. Ignatov. Real-time human activity recognition from accelerometer data using convolutional neural networks. *Applied Soft Computing*, 62:915–922, 2018.

[23] N. Kern, B. Schiele, and A. Schmidt. Multi-sensor activity context detection for wearable computing. In *Lecture notes in computer science*, pages 220–232, Berlin, 2003. Springer.

[24] A. Khan, N. Hammerla, S. Mellor, and T. Plötz. Optimising sampling rates for accelerometer-based human activity recognition. *Pattern Recognition Letters*, 73:33–40, apr 2016.

[25] A. M. Khan, Y.-K. Lee, S. Y. Lee, and T.-S. Kim. A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. 14(5):1166–1172, 2010.

[26] A. M. Khan, Y.-K. Lee, S. Y. Lee, and T.-S. Kim. A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. 14(5):1166–1172, 2010.

[27] J. Kroger, P. Raschke, and T. Bhuiyan. Privacy implications of accelerometer data: A review of possible inferences. pages 81–87, 01 2019.

[28] A. Mannini and A. M. Sabatini. Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors (Basel, Switzerland)*, 10(2):1154–1175, 2010.

[29] Marius Borcan. Naive Bayes Classifier Explained. Available at: https://towardsdatascience.com/naive-bayes-classifier-explained-54593abe6e18, mar 2020.

[30] M. Mathie. Monitoring and Interpreting Human Movement Patterns Using a Triaxial Accelerometer. Technical report, 2003.

[31] Maximilian Christ et al. Overview on extracted features Tsfresh documentation. Available at: https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html.

[32] Maximilian Christ et al. Tsfresh documentation. https://tsfresh.readthedocs.io/en/latest/index.html.

[33] H. Myklebust, N. Nunes, J. Hallen, and H. Gamboa. Morphological analysis of acceleration signals in cross-country skiing - information extraction and technique transitions detection. pages 510–517, 01 2011.

[34] B. Najafi, K. Aminian, A. Paraschiv-Ionescu, F. Loew, C. Bula, and P. Robert. Ambulatory system for human motion analysis using a kinematic sensor: monitoring of daily physical activity in the elderly. *IEEE transactions on biomedical engineering*, 50(6):711–723, 2003.

[35] T. pandas development team. pandas-dev/pandas: Pandas, Feb. 2020.

[36] M. Paulich, M. Schepers, N. Rudigkeit, and G. Bellusci. Xsens MTw Awinda: Miniature Wireless Inertial-Magnetic Motion Tracker for Highly Accurate 3D Kinematic Applications.

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

[38] A. Pentland. Looking at people: Sensing for ubiquitous and wearable computing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22:107–119, 2000.

[39] S. Rosati, G. Balestra, and M. Knaflitz. Comparison of Different Sets of Features for Human Activity Recognition by Wearable Sensors. 2018.

[40] Y. Saez, A. Baldominos, and P. Isasi. A comparison study of classifier algorithms for cross-person physical activity recognition. *Sensors (Basel, Switzerland)*, 17(1):66, 2017.

[41] V. N. T. Sang, S. Yano, and T. Kondo. On-body sensor positions hierarchical classification. *Sensors (Basel, Switzerland)*, 18(11):3612, 2018.

[42] S. Sangavi and B. M. Hashim. Human activity recognition for ambient assisted living. In *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, pages 1–4. IEEE, 2019.

[43] Sklearn developers. Naive Bayes. Available at: https://scikit-learn.org/stable/modules/naive_bayes.html.

[44] D. Subramanian. A simple introduction to k-nearest neighbors algorithm. https://towardsdatascience.com/a-simple-introduction-to-k-nearest-neighbors-algorithm-b3519ed98e, 2019. Accessed on 2021-04-25.

[45] K. Theodoridis. *Pattern recognition*. Elsevier/Academic Press, Amsterdam ; London, 4th ed. edition, 2009.

[46] T. Yiu. Understanding Random Forest. `https://towardsdatascience.com/understanding-random-forest-58381e0602d2`, 2019.

[47] A. Yurtman and B. Barshan. Activity recognition invariant to sensor orientation with wearable motion sensors. *Sensors (Basel, Switzerland)*, 17(8):1838, 2017.

[48] A. Yurtman, B. Barshan, and B. Fidan. Activity recognition invariant towearable sensor unit orientation using differential rotational transformations represented by quaternions. *Sensors (Basel, Switzerland)*, 18(8), 2018.

[49] T. Zebin, P. J. Scully, and K. B. Ozanyan. Evaluation of supervised classification algorithms for human activity recognition with inertial sensors. In *2017 IEEE SENSORS*, pages 1–3. IEEE, 2017.

[50] G. Zhang. What is the kernel trick? why is it important? Available at: `https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d`, 2018.

[51] M. Zhang and A. Sawchuk. Motion primitive-based human activity recognition using a bag-of-features approach. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, IHI '12, pages 631–640. ACM, 2012.

[52] Y. Zhang, Y. Zhang, Z. Zhang, J. Bao, and Y. Song. Human activity recognition based on time series analysis using U-Net. Technical report.