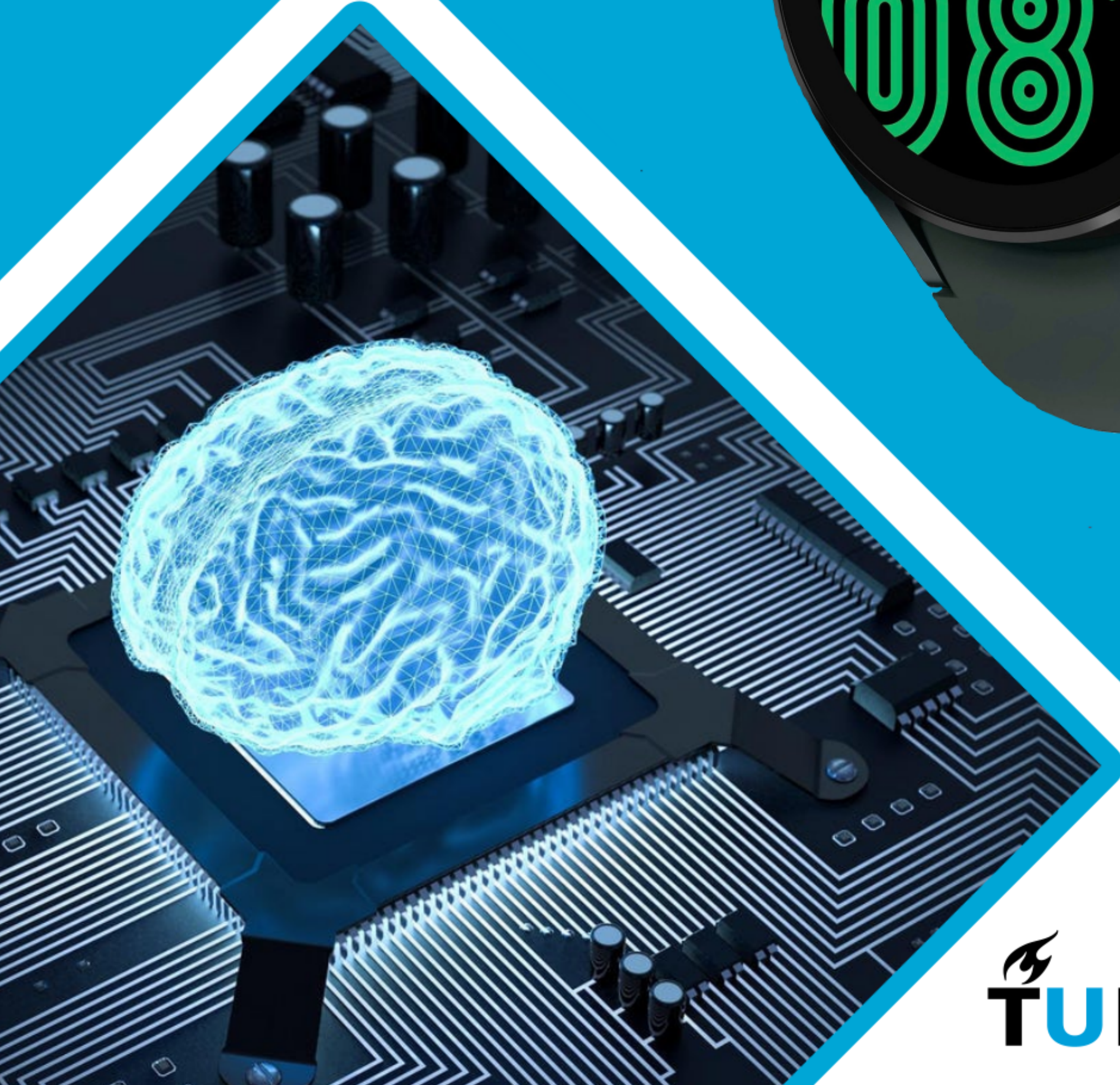
Facilitating healthcare using smartwatches

Bachelor Graduation Thesis

Ricardo Cavalini
Ibrahim Hassan
Thomas Pouwels

June, 2022



TU Delft

Thesis

Bachelor Graduation Project: Improving healthcare using smartwatches

by

Subgroup Data Classification

Ricardo Cavalini   5176484
Ibrahim Hassan   5144264
Thomas Pouwels   4705971

June 9, 2022
Delft

Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Electrical Engineering Programme

Abstract

This report entails one of two subsystems in a system to provide a web-based platform for smartwatch data acquisition. Human activity recognition is designed and implemented using various machine learning approaches and then tested on the data acquired using our own platform. Together with the web-based platform, this provides a solid base for more research using data gathered from smartwatches.

The human activity recognition is implemented first using a classical machine learning approach with feature extraction and a random forest classifier. Then, both a convolutional neural network and a recurrent neural network are implemented using Tensorflow [1]. Tests are performed to see how much data points are needed for sufficient classification accuracy, whether filtering of the data improves the accuracy of the models, and what model performs best.

Preface

This report is written as part of the Bachelor Graduation Project of the Electrical Engineering Programme at the TU Delft. The smartwatch app developed in the project is written for WearOS, and both the web-based platform and the classification methods are developed using Python. The GitLab repository can be found here: `https://gitlab.tudelft.nl/babdi/bapsmartwatch`

We would like to offer our thanks to our daily supervisor dr. B. Abdikivanani and our supervisor prof.dr.ir. A.J. van der Veen, who have helped and supported us during the past months. Next to that, we would like to thank our colleagues that designed the other subsystem, Shijie Liao, Robbin Poll and Marijn Sluijs. It was a pleasure working together.

Ricardo Cavalini, Ibrahim Hassan & Thomas Pouwels,
Delft, June 2022

Contents

1 Introduction

The presence of smartwatches is expending rapidly with 40 million smartwatches sold in Q4 of 2021 and a year-over-year shipments increase of 21% in the same year [2]. The newest smartwatches contain more types of sensors, which provide lots of new opportunities for engineers and researchers. Smartwatches become an accessible base for developing suitable applications without requiring extensive resources. Smartwatch sensors have become increasingly more reliable over time. This has improved their recognition and acknowledgement in healthcare services and research.

Healthcare
Smartwatches started with simple functions like heart rate detection and setting alerts for medication reminders. By now they have now progressed to more advanced applications like early detection of diseases such as arrhythmia, abnormalities in the heart rhythm. The sensors that make those advanced applications in healthcare possible are the PPG and ECG sensors.

Newer smartwatch models (such as the Apple Watch 7 [3]) use these sensors to detect signs of arrhythmia in their health app. Multiple studies have been conducted to test the detection accuracy of atrial fibrillation (a type of arrhythmia) using ECG data and conclude that smartwatches are already quite accurate [4–6]. The comparison in these studies is made with traditional 12-lead ECG's, which are more extensive and time-consuming for the patient and healthcare worker. Recently, Mishra et al. [7] even used smartwatches for pre-symptomatic detection of COVID-19.

Human activity recognition
Smartwatches also contain other sensors like the the accelerometer and the gyroscope which open the door to research in human activity recognition (HAR). HAR is an active field of research due to the many possible applications it has in domains such as healthcare monitoring [8], surveillance, and exercise tracking. For instance, HAR can recognise early mobility for Intensive Care Unit patients. Early mobility is essential for ICU patients who suffer from long-time immobilisation [9].

Improvements
More research is still needed to approve or improve the performance of diagnosis based on smartwatch sensor data. For example, some research questions that still need to be answered are: how good is the accuracy of the smartwatch sensors compared to traditional methods? How does their reliability hold up under different conditions such as swimming and jogging?

The first step to investigate these questions, is to collect data using these smartwatches and build the required database for them. A platform which enables easy and fast data collection from smartwatches is necessary and highly invaluable for researchers. Therefore the goal of this project is to provide a stable base for research on smartwatch data. To make this possible, a smartwatch app and a web based platform are designed and implemented.

Our app ”GetSmart” reads out the sensor data from the smartwatch and sends it to the web-based platform, where it can be downloaded. This allows for fast and easy collection and generation of data sets from smartwatches sensors.

In the second part of the project, which is the main focus of this thesis, we focus on Human Activity Recognition (HAR). Accelerometer and gyroscope data is collected using the platform and used to generate a data set. This data set is used to train and test machine learning algorithms for HAR. It is important to note that initially the algorithms are designed using an existing data set, as the first part of our project was still in progress.

Human Activity Recognition

As mentioned before, HAR has many applications in domains such as healthcare monitoring, surveillance, and exercise tracking. There are, however, some challenges that HAR faces such as data collection, privacy and power consumption.

Multiple measuring methods can be applied for HAR. Cameras are one of the possibilities to collect data for HAR. While cameras provide lots of useful data, the privacy invasion that comes with it causes big societal problems. Although there are many solutions to reduce collecting sensitive data [10] [11], they can't ensure privacy-preserving video's. Besides privacy, another main problem with collecting data using a camera is that the participants are bound to the view of the camera.

To overcome these issues, smartphones [12] [13] and wearable sensors are valid solutions. Smartphones are less viable if the position change of the smartphone gets taken into account. Changing the phone's position, e.g. from pocket to purse, can influence the incoming data of the HAR. In doing so, the accuracy of the prediction model gets influenced. For wearable sensors, this is not the case. The privacy infringement in these scenarios is also limited to the user of the device. In contrast to cameras, an accelerometer does not record it's surroundings.

Wearable sensors can be divided into two categories: sensors on multiple parts of the body [14], or smartwatches [15] [16]. The first category is an accurate way to measure human activity, especially with suitable measuring points. However, depending on the position and the connections, users may find the sensors uncomfortable and can thereby influence the data by moving differently than normal. Smartwatches however are the most viable solution as they are relatively comfortable and do not interfere with movement.

Acquisition

Commercially available smartwatches from manufacturers such as Samsung, Apple, and Garmin already provide users insight on the recordings such as heart rate, ECG, and blood pressure. They are presented in a graphical user interface on an app but they do not provide or save continuous recordings of raw data that can be used for research and development. Several approaches have been proposed in literature to gather required data for researchers like the Real-time Online Activity and Mobility Monitor (ROAMM) framework [17], where several characteristics of such a framework are listed. The ROAMM framework consists of an application for data collection, a server for data storage, and an online monitoring tool.

Recognition

After acquiring the data from the smartwatch, the next step is to process and use it for activity recognition. This can be done using machine learning. Many classification models are proposed in literature that use sensor data to perform human activity classification. They range from traditional machine learning to deep learning algorithms. It is also possible to combine different classifiers in an ensemble. Hossain Shuvo et al. [18] propose an ensemble of a support vector machine classifier (SVM) and a random forest classifier with a Convolutional Neural Network (CNN). The first two models are machine learning models while the latter is a deep learning model. An ensemble of two deep learning models is also possible. Verma et al. [19] discuss a combination of a CNN and a Gated recurrent unit (GRU).

Problem description

As described in the previous sections, there are many applications for data recorded by smartwatches. However, to the best of our knowledge, there is no easy and user friendly platform to collect raw smartwatch sensor data. In this project we have developed an easy-to-use smartwatch application and a web-based platform for researchers, to record smartwatch data, send the data to the platform and retrieve the data from the platform for their applications. The recordings can be downloaded as a Comma Separated Value (CSV) file from the server. The recordings made during this project will be used to train a machine learning model for HAR, in order to demonstrate the usefulness of raw smartwatch data collected in this way.

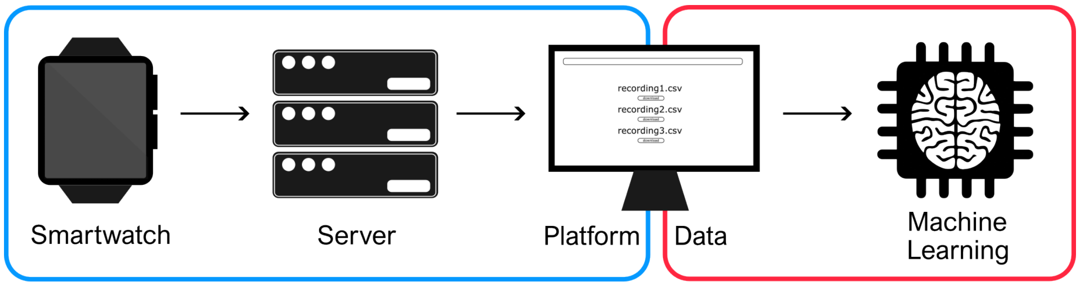


Figure 1.1: Project overview.

Subdivision
To address these problems, the six group members are divided into two subgroups. Figure 1.1 shows the proposed workflow. One of the subgroups (circled in blue) focuses on the data acquisition and storing the data on a web-based platform, while the other subgroup (in red) focuses on activity recognition. This thesis will focus on the human activity recognition subgroup, the thesis of the other subgroup can be found in [20].

Thesis outline
This thesis is structured as follows. Chapter 2 will describe the program of requirements of the complete system and the design criteria for the classification algorithms. Chapter 3 describes the recordings used for the classification models and different algorithms used to preprocess those recordings. In Chapter 4 the algorithms will be designed and implemented. These design choices will be determined by and referenced back to the requirements. Chapter 5 will describe the verification of the designed algorithms and present our results. The conclusion and discussion can then be found in Chapter 6.

2 Programme of Requirements

The programme of requirements is split into two parts. First, the general requirements of the complete system are given. The second part are the requirements for the design criteria of the HAR algorithm. The mandatory requirements must to be met for the project to be successful. The trade-off requirements will be fulfilled if time allows, they are designed to improve the system.

General Requirements

The goal of the project is to facilitate research on smartwatch applications. First, we allow for the easy recording of smartwatch data using an app and web-based platform. The app on the smartwatch collects and sends raw sensor data to a server. The server receives the data and makes recordings. These recordings can then be used in various settings. For this thesis, we will focus on a use case for this data, namely HAR.

1. The smartwatch application must acquire sensor data from a commercially available smartwatch.
2. The smartwatch application must transmit sensor data to the server.
3. The server must be able to record the sensor data.
4. The website of the server must present the active users and recordings.
5. The server must integrate a machine learning algorithm to recognise the wearers' current activity.

To integrate a successful machine learning algorithm to recognise the wearers' current activity, the criteria for the design of the HAR algorithm need to be clarified.

2.1. Design criteria for HAR

The design criteria are dependent both on the general requirements and the design choices of the other subsystem. Below they will both be briefly discussed and then summarised in a programme of requirements for the subsystem.

Design Limitations

The first design limitation is the limited time that can be spend on the project. Therefore, only a limited number of activities are chosen for the models to classify. These activities are sitting, standing, walking, running and walking stairs. Although limited, these 5 activities do make up a large amount of time spent wearing a smartwatch. They also pose some interesting design challenges, since there is a lot of ambiguity in the data for standing and sitting, and walking and walking stairs.

Another limitation is that the model has to be able to predict based on short segments. This limitation is based on the fact that a user can change activity often and in short intervals. For example, A 5-minute segment of data from the smartwatch could include the user sitting at their desk, walking to the stairs and up to get a coffee, and standing there for a chat.

Requirements for Classification

1. **Mandatory** The software must classify:
   - sitting
   - standing
   - walking
   - running
   - walking stairs
2. **Mandatory** The model must be able to continuously classify human activity based on data segments of no more than 5 seconds length.
3. **Trade-off** The models will be trained on GetSmart data sets.
4. **Trade-off** The software must be able to annotate the GetSmart data with the recognised activity on the web-based platform.
5. **Trade-off** The software will classify falling.

3 Data and Preprocessing

As has been discussed in the introduction and the programme of requirements, activity recognition will be done using smartwatch data. In this chapter, the sensors and data sets used and the preprocessing steps that have been taken will be explained before going into the machine learning models in the next chapter.

3.1. Sensors

Two of the smartwatch sensors will be used for recognition, namely the accelerometer and the gyroscope. The accelerometer in the smartwatch is an electromechanical device that measures acceleration force in the x, y and z direction in meters per second squared.

The gyroscope sensor is a device capable of measuring the angular velocity of an object, by means of the Corioles acceleration. Together they give a complete description of the way the watch moves over time.

3.2. Data set

Our goal is for machine learning models to be trained and tested on data sets acquired using the platform designed by the data acquisition group. However, since both groups are working in parallel, for the first iterations these models are trained on an already existing data set. This way hyperparameters of the models can be tweaked and some experiments can already be done before the GetSmart data is available. The data set we are going to use for this is the WISDM data set.

3.2.1. WISDM Data set

The WISDM Smartphone and Smartwatch Activity and Biometrics Data set was obtained from the UCI Machine Learning Repository [21] and recorded by Weiss et al. [22].

The data set consists of raw accelerometer and gyroscope data recorded by smartwatch and phone at a sampling rate $f_s = 20$Hz. The data has been collected from 51 subjects as they perform 18 different tasks for approximately 3 minutes each. The data is labelled with the performed activity and a subject ID. From this data set, only the raw data recorded by the smartwatch and only 5 out of 18 tasks were used, namely sitting, standing, walking, jogging and climbing stairs. 5 Seconds of accelerometer data from 3 activities is visualised in Figure 3.3. The class distribution across the data set is shown in Figure 3.1.

3.2.2. GetSmart Data set

The data recorded by the Data Acquisition subgroup using their GetSmart app consists of the same 5 activities (sitting, standing, walking, jogging and climbing stairs), this time recorded from 4 different subjects, and for approximately 5 minutes per task. The sample frequency of the data set is $f_s = 50$Hz. The data is visualised in Figure 3.4. Compared to Figure 3.3, a lot more detail can be seen in the graphs. The class distribution across the data set is shown in figure 3.2.

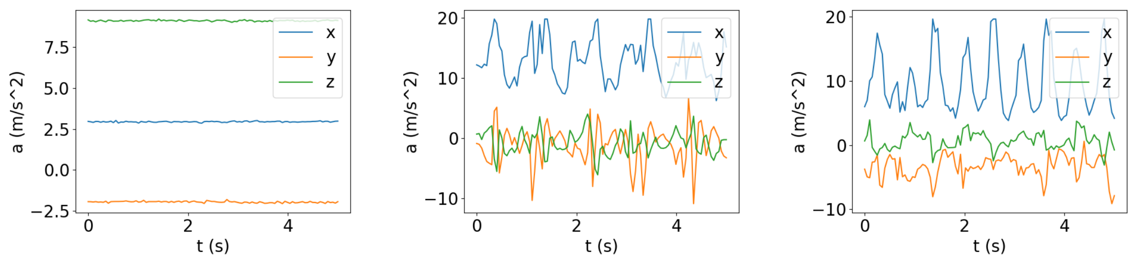


Figure 3.3: 5 seconds of accelerometer data of different activities, from left to right: sitting, walking, jogging (WISDM data set)

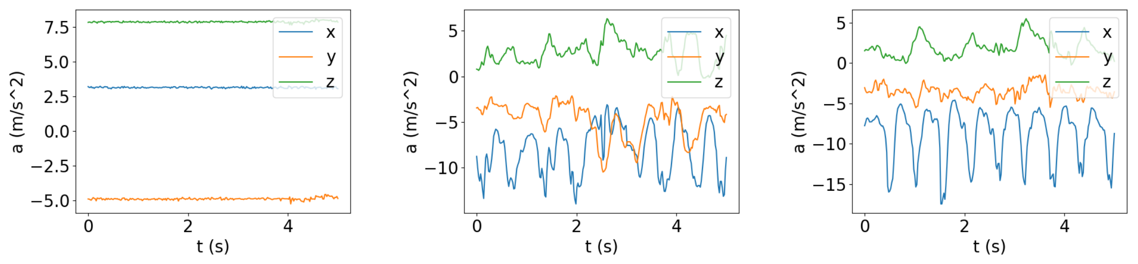


Figure 3.4: 5 seconds of accelerometer data of different activities, from left to right: sitting, walking, jogging (GetSmart data set)

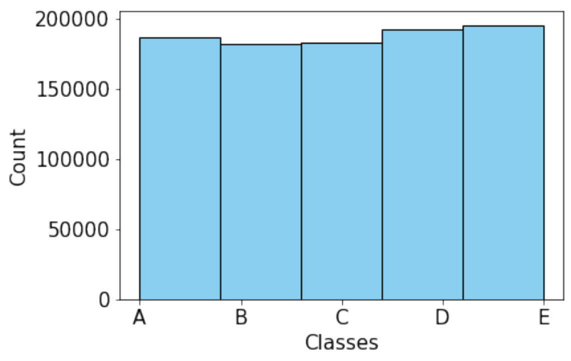


Figure 3.1: Class distribution WISDM data set.

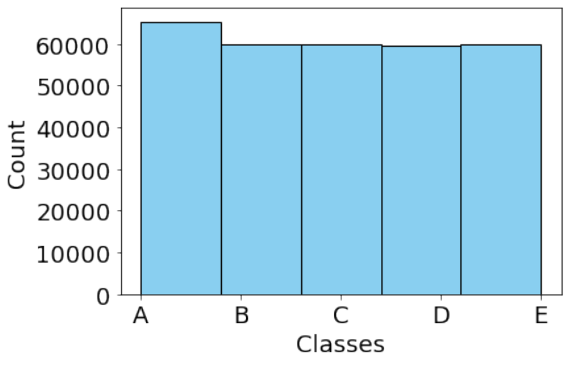


Figure 3.2: Class distribution GetSmart data set.

3.3. Processing

Before using the data set as input for the classification algorithms, the data set might have to be filtered to remove noise and artefacts. Besides filtering, the recordings have to be cut into smaller segments to account for the fast change of activities of the user and improve recognition results.

3.3.1. Filtering

An example of unfiltered accelerometer data of a subject sitting is shown on the left in figure 3.5. All three signals have a clear DC component, with magnitude $|a| = \sqrt{x^2 + y^2 + z^2} \approx 9.78\, ms^{-2}$, approximately the gravity of Earth g.

An argument could thus be made to filter the data. This filtering is done using a Butterworth high-pass filter with a cut-off frequency at 0.3Hz [23]. After filtering (see second subfigure in 3.5), $|a| \approx 1.6 \times 10^{-4}$ and the DC component is indeed removed from the data. This does, however, also remove information from the data, mostly the orientation of the watch. Therefore, we will later investigate if the filtering really improves the accuracy of activity detection or not.

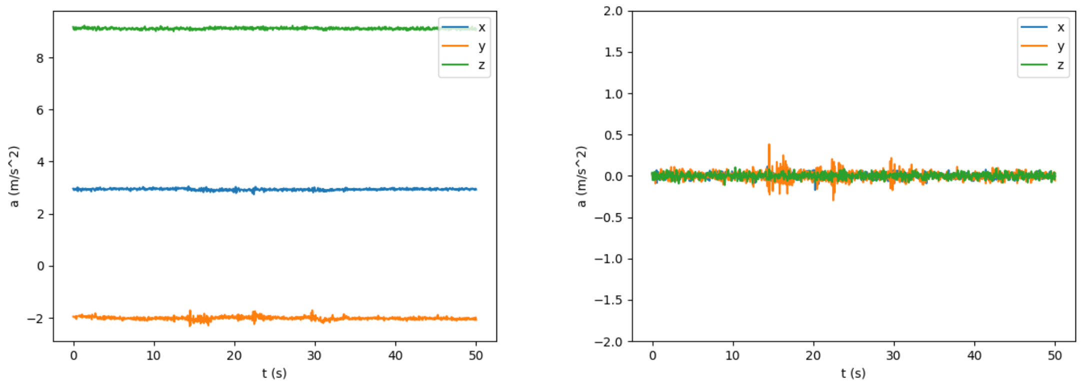


Figure 3.5: left: 50 seconds of unfiltered accelerometer data; right: the same data after filtering

3.3.2. Sliding Window

As described in Section 2.1, a user can change activity often and in short intervals. Therefore, their activity should be recognised based on only a short segment. To do so, a sliding window is implemented. This way only the last n samples of the data will be considered for the classification, as illustrated in Figure 3.6.

The amount of samples n to be considered is dependent on the sample rate of the data and the minimal duration in seconds needed by the model to differentiate between different activities. Recent papers [23, 24] suggest segment sizes of around 3 seconds, which would mean $n = f_s \times t = 60$ samples for the WISDM set.

The other variable to be considered is the amount of samples the window shifts, k samples. Using the sample frequency $f_s = 20Hz$ of the WISDM data set described in Section 3.2 and $k = 1$, the model classifies every 50 milliseconds. This can be computationally expensive, especially at higher sampling frequencies. A trade-off between accuracy and used resources should therefore be made.

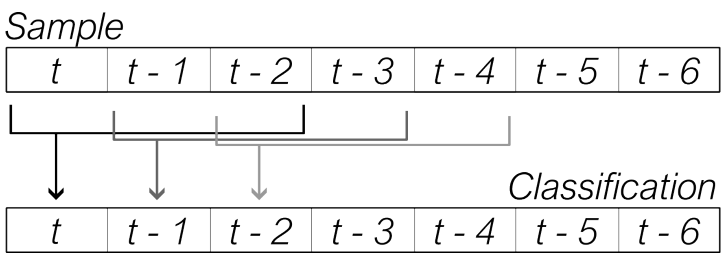


Figure 3.6: An example of a sliding window

3.3.3. Train and test sets

After slicing the data set into segments, it is split into a train and a test set. The train set will be used to train the different machine learning models, while the test set will be used to independently test the model's accuracy: it will not be used when training the models. The data set is split into 70% train and 30% test set. The same 'random state' is used in Python every run, to make sure the test and train set are the same between different runs.

4 Design and Implementation

To recognise the human activities mentioned in chapter 2.1, different recognition models are implemented. This chapter starts with a description of the proposed workflow. Next to that, the implementation and design choices of the machine-and-deep learning models are addressed.

4.1. Workflow

Figure 4.1 shows the proposed pipeline for human activity recognition. First, the recorded sensor data is split into a train and a test set. These sets are used to train the model and validate its performance. Once training is completed, the recognition algorithm is deployed on the web-based platform described in chapter 1 to perform HAR on real-time sensor data. For this, the sliding window is used to select a segment upon which the recognition is based. Together with the recognition algorithm, the platform can gather sensor data from a smartwatch and annotate the sensor data simultaneously.

In the next sections, first a classical machine learning approach and then two deep learning approaches for the recognition are discussed and implemented.

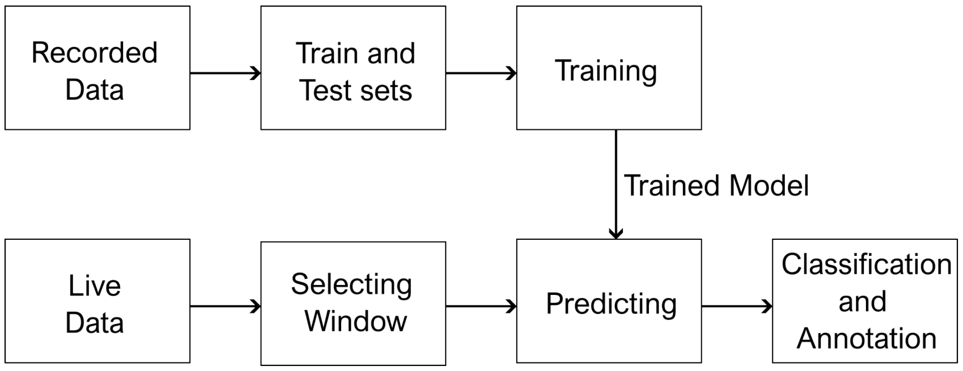


Figure 4.1: Pipeline for human activity recognition.

4.2. Classical Approach

Using the workflow in Figure 4.1 for the classical machine learning approach, the recorded data (raw data) for the train and test sets will be used to extract features. These features go through feature selection to reduce the number of input variables. The selected features are used to train the classical machine learning algorithm, random forest.

4.2.1. Features Extraction

Raw data from the accelerometer and gyroscope in chapter 3 give little to no information to the classical machine learning algorithms to build their model on. This is where feature extraction plays a big role.

Extracting features from raw data gives statistical insight. With statistical insight, classical machine learning algorithms can build a better model. The features extracted from the raw data (Table 4.1) are popular for HAR tasks, due to their simplicity and high performance [25–27]. These features will be extracted in the time domain, and time-frequency domain using the discrete wavelet transform (DWT).

Features:	**Description:**
Mean	The mean is the average value of the window segment.
Variance	The variance is used to measure the average deviation from the mean.
Median	The median is the value in the middle when the window segment is ordered from the least to the greatest.
Standard Deviation	The standard deviation is used to measure how far a group of values deviates from the mean.
Maximum	The maximum value of the window segment.
Minimum	The minimum value of the window segment.
Peak to Peak	The peak to peak is a difference between the maximum and the minimum.
Root Mean Square	The root mean square is the root taken from the average value of the window segment squared.
Skewness	The skewness is used to measure the asymmetry of a distribution.
Kurtosis	The kurtosis is used to measure how heavy the distribution tails differ from a normal distribution.
Impuls Factor	The impuls factor is used to compare the maximum to the mean of the window segment.
Crest Factor	The crest factor is used to compare the maximum to the root mean square of the window segment.
Interquartile Range	The interquartile is used to measure the middle 50 % of the window segment when the window segment is ordered from small to large.
Correlation Coefficient	The correlation coefficient is used to describe the strength between two relative movements of two window segments; the accelerometer and the gyroscope.

Table 4.1: 14 extracted features and description for classical machine learning models.

Time Domain Features

The time domain is a term used to describe the analysis of physical signals or mathematical functions. Where time domain features extraction is the technique arising from time domain analysis. In related studies, time domain feature extraction is the most popular. The popularity comes from not having to perform prepossessing steps on the raw data. This results in having less computational time in the overall classification algorithm.

Time-frequency Domain Features

Time-frequency domain is a combination of time and frequency domain analysis. It approximates the sinusoidal frequency and phase content of a local segment while the signal changes over time. Multiple techniques are available to analyse such a domain.

Short Term Frequency Transform
This method is introduced by Dennis Gabor [28]. His method decomposes a given signal into a number of small fixed windows and performs the Fourier transform on those windows.

Each time window of the Short Term Frequency Transform(STFT), has a certain frequency and time resolution (Figure 4.3). The higher the time resolution, the lower the frequency resolution and vice versa. The downside to this fixed window is the loss of information in time and frequency.

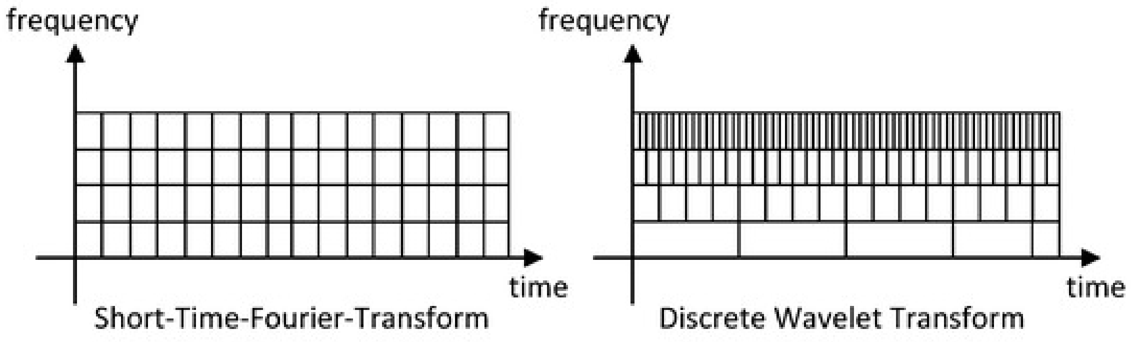


Figure 4.2: Comparison of a STFT and a Wavelet Transform for time-frequency analysis of a signal [29].

Wavelet Transform
The second technique is the Wavelet Transform, which is built upon the method of Dennis Gabor. The wavelet transform provides a multi-resolution representation using wavelets(”little waves”). The wavelet transform can analyse continuous-time signals or discrete-time signals. As for Human Activity, only discrete signals are dealt with.

The DWT starts with a spectral transformation of the whole signal. This gives high-frequency resolution, but low time resolution. Where after you go up a ’level’ and slice the time window. This gives a higher time resolution at the expense of frequency resolution. The windows gradually get smaller with every level, which ends up with a high time resolution, but a low-frequency resolution (Figure 4.3).

Within the DWT there are two types of decomposition, single-level decomposition and multilevel decomposition. Single level decomposition on the window segment results in two coefficients, the approximation coefficients (cA) and the detail coefficients (cD). cA is based on lowpass decompositions and cD is based on a highpass decomposition. multilevel decomposition is executed on cA and results again in a cA and dA, which gives more detailed information. However, multi-level decomposition will also negatively influence computational power and can decrease the stability of the model [30]. As a result, single-level decomposition is used for feature extraction.

4.2.2. Feature Selection

The features extracted from each window segment in the time domain and time-frequency domain are combined into a ”feature space”. With the feature space, the classical machine learning algorithm can use it as input for training and testing the model. As mentioned in the last section features give more statistical insight into the raw data, which can result in building a better model, which is not always the case. Features within the feature space can also have a negative effect on the model. Some features become redundant based or are irrelevant from the start. If the model bases the prediction on those features, it yields a lower accuracy.

This is where feature selection comes into play. Random forest uses a build-in technique to calculate the score for all features for a certain model. This score represents the ”importance” of each feature based on the model. The technique iterates over multiple decision trees and takes the average of all models and indicates the range of importance for every feature. The results can be seen in Figure A.1,A.3 and A.2.

From there, Recursive Feature Elimination (RFE) is used [31]. RFE is a feature selection method that is based on the feature importance defined by the build-in technique of the random forest model. It does this by repeatedly training on the feature space by removing the least significant feature in every iteration until all the features are cut. Figure **??** shows the number of features plotted against the cross-validation accuracy.

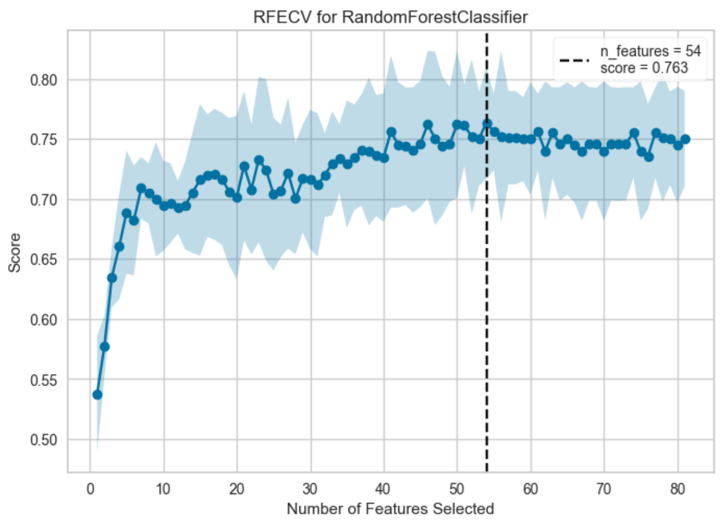


Figure 4.3: Number of selected features plotted against the f1-score.The blue margins represent the range of variety which the model iterates over different feature subsets. The blue dots represent the average of all iterations for a specific number of features used. The black line represents the number of features which yield the highest f1-score.

While 54 features result in the highest f1-score of 0.763. 41 features results in approximately the same score. Taking computational power into consideration, only the 41 most important features will be used in training and testing the classical machine learning algorithm.

4.2.3. Random Forest

Random forest is a supervised learning algorithm and was first created by Tin Kam Ho [32]. Random Forest is an ensemble of multiple Decision trees, where every classification of every decision tree has the same weight [33]. This results in a classification system based on the majority of the votes (Figure 4.4).

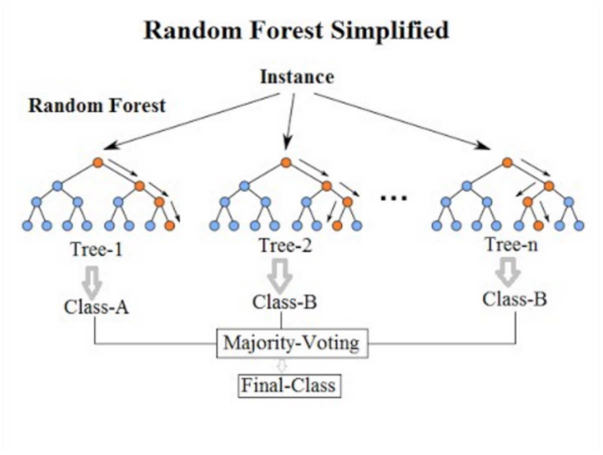


Figure 4.4: Diagram of a Random Forest Tree Classification Algorithm[34].

A single decision tree works as follows. It starts at the root node and decides on a feature boundary whether it should branch to the left or right to another node. This is continued until it branches into a leaf and classifies the data. A single decision tree defines its classification based on the best features when splitting on nodes.

Random forest introduces extra randomness when growing the decision trees and thereby defines a decision tree on random combinations of features. Random forest is also less influenced by outliers than other classification algorithms, because of recursive partitioning and local model fitting.

Before the implementation of the random forest model, a closer look at the parameters has to be taken. By performing hyper parameter tuning, the most favourable parameters can be chosen to yield the highest accuracy. The parameters, their definition and the variables over which the tuning iterates are shown in the following table:

Parameters:	**Description:**	**Tuning range:**
n_estimators:	Number of trees in the forest.	5, 20, 50, 100, 200, 500, 750, 1000
min_samples_split:	Minimum samples required to split an internal node	2, 6 or 10
min_samples_leaf:	Minimum samples required to be at a leaf node	1, 3 or 4
max_depth:	Maximum depth of a tree.	10, 20, 30, ..., 110, 120
bootstrap:	Whether bootstrap samples are used when building trees. If False, the whole data set is used to build each tree.	True or False

Table 4.2: Parameters with description and the variables over which hyper parameter tuning occurs of the random forest model.

Hyper parameter tuning resulted in the following variables:

- n_estimators = 750
- min_samples_split = 2
- min_samples_leaf = 1
- max_depth = 20
- bootstrap = False

The implementation of the random forest model is built with the resulted variables mentioned above. The input to build the model is based on feature importance and selection, where the 41 most important features are used.

4.3. Deep Learning

The advantage of deep learning over classical machine learning is that no handcrafted features have to be inserted into the classification algorithms. Deep learning models learn patterns from the data themselves, thus making it possible to learn very complex relations in the data. Based on the complex time-series nature of the sensor data, two types of deep learning algorithms are considered for classification, namely convolutional neural networks (CNN) and recurrent neural networks (RNN).

4.3.1. Convolutional Neural Network

CNN is a class of neural networks (NN) that can be used for pattern recognition within data, e.g. images and time series. A traditional CNN consists of a convolutional layer, pooling layer and a fully connected layer [35] as shown in figure **??**. In contrast to traditional NN, CNN is great at extracting features from data with its convolutional layers to find complex relationships within the data. The CNN architecture used in this thesis is the Fully Convolutional Network (FCN). It was proposed by Wang et al. [36], who tested a multilayer perceptron (MLP), the FCN, and Residual network (ResNet) models on 44 UCR time-series data sets [37] and compared their performance against other state-of-the-art models. The authors concluded that the FCN has a superior performance compared to the other models.

The architecture of an FCN is shown in figure 4.5. It mainly consists of three blocks containing a 1-dimensional convolutional layer, a batch normalization layer, and an activation function. Batch normalization normalizes the output of the convolutional layer before passing it to the ReLu activation function. This makes the model more stable by tackling the vanishing/exploding gradient problem and it reduces the training time. Global average-pooling is used in this architecture as pooling layer. The resulting vector flows into the softmax layer for the classification.

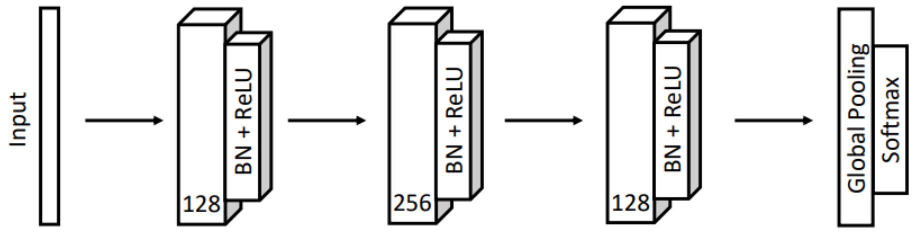


Figure 4.5: Fully convolutional Network [36].

4.3.2. Recurrent Neural Network

One of the drawbacks of using feedforward or convolutional neural networks on sequential data is that the order of the samples is not taken into account. A solution to this problem is the Recurrent Neural Network (RNN). In the Recurrent Neural Network, the input is fed into the network sequentially, where it updates a hidden state variable. The state then carries over the information from this step to the next one.

The way this state gets updated can be as simple as a $tanh$ activation, or as complex as the LSTM [38] or GRU [39]. The drawback of using the simple RNN is that it only has relatively short memory (in the order of 10 discrete-time steps) due to the vanishing gradient problem in back propagation through time [40].

LSTM and GRU

Comparison on the performance of the GRU, LSTM and RNN by Chung et al. [41] shows that both LSTM and GRU outperform the simple RNN, as expected. Results on comparing the LSTM and GRU were inconclusive, suggesting ”the choice of the type of gated recurrent unit may depend heavily on the data set and corresponding task” [41, p 7]. One notable difference between the LSTM and GRU is the amount of hidden layers: where the LSTM has 4, the GRU only has 3. This should make computations slightly faster.

Preliminary results showed both a shorter training time and a slightly better accuracy using the GRU, and it has therefore been chosen over the LSTM.

4.3.3. Final designs

Layers and Nodes

Next to the type of deep learning model, some hyperparameters have to be chosen before implementing the final model. The first one is the amount of layers, and the nodes per layer. For the FCN, the topology as described by Wang et al. [36] is used. For the GRU, a first model with 1 layer and 32 nodes was used, after which the model was expanded until a balance between complexity and accuracy was found. Next to that, somethings that improved the accuracy a lot are separate normalisation for the accelerometer and gyroscope inputs, as well as adding a small (16 nodes) fully connected layer before going into the GRU layers. Presumably this is because the input range of the accelerometer and the gyroscope differ a lot. Squashing both together would make all gyroscope data points close to zero.

Regularisation

Regularisation is a way to help models avoid overfitting. There exist different regularisation methods including dropout and batch normalisation. As mentioned earlier, batch normalisation stabilizes the model and reduces the training time. Dropout regularisation, on the other hand, works by ”dropping” neurons and their corresponding connections during training. That prevents neurons to learn the characteristics of the training data, and therefore it prevents overfitting.

When training on the GetSmart data set, considering its relatively small size, dropout layers are added to the FCN architecture besides batch normalisation to avoid overfitting. The GRU uses batch normalisation as well as dropout layers in its architecture.

Layer	Nodes	# of Parameters
1D Convolution Layer	128	6272
Batch Normalisation	128	512
Dropout Regularisation	-	-
1D Convolution Layer	256	164096
Batch Normalisation	256	1024
Dropout Regularisation	-	-
1D Convolution Layer	128	98432
Batch Normalisation	128	512
Max Pooling Layer	128	-
Fully Connected Output Layer	5	645

Table 4.3: The implemented FCN Architecture, total number of parameters: 271,493

Layer	Nodes	# of Parameters
Gyroscope Batch Normalization	3	12
Accelerometer Batch Normalization	3	12
Fully Connected Layer Gyroscope	16	64
Fully Connected Layer Accelerometer	16	64
Dropout Regularization	-	-
GRU Layer	128	62208
GRU Layer	64	37248
Fully Connected Output Layer	5	325

Table 4.4: The implemented GRU architecture, total number of parameters: 99,933

5 Testing and Verification

In this chapter the models implemented in Chapter 4 will be tested on both the WISDM data set [22] as mentioned in Section 3.2, as well as on the data recorded by the data acquisition subgroup. Their performances will be compared and discussed.

5.1. Metrics

The classification models are assessed against one metric, namely accuracy. Accuracy is defined as the fraction of the total True Positive(TP) plus the total True Negative(TN) instances over the total number of predictions:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} * 100\% \ [\%] \tag{5.1}$$

The main problem with using accuracy is that it is ineffective when used on unbalanced data sets. In a binary classification problem where class A occurs 95% of the time and class B only 5% of the time, always guessing A gives 95% accuracy.

As the class distribution plots in chapter 3 indicate, the classes in WISDM and the recorded data set are only slightly unbalanced. The class ”Walking” occurs at approximately 21.4% in the data set, while the other classes occur at around 19.6%. Using accuracy as the performance metric should thus suffice.

5.2. Segment size and filtering

To find the right segment size, sample frequency, and the usefulness of filtering, different tests are conducted. The classification models are analysed on three different segment sizes, two different sample frequencies, and on (un)filtered data. The segment size is determined to find the minimum duration of an activity needed to obtain the correct classification. The sampling frequency of the sensor data is 50 Hz. The data is down-sampled to 25 Hz to investigate what influence the sample density has on the model's prediction. A low-pass filter is then needed to filter out frequency components above 12.5 Hz to avoid aliasing. The influence of filtering out earth's gravity from the accelerometer data by applying a high pass filter, as mentioned in section 3.3.1, on the classification is also analysed.

5.3. WISDM tests

The WISDM data set is tested according to the above set metrics. The resulted accuracy can be seen in Table 5.1. As the WISDM data set is recorded on a sampling frequency of 20 Hz, there was no need to downsample and thereby no need to add a low-pass filter to avoid aliasing. The tested models, discussed in chapter 4, were Random Forest, FCN and GRU. Every model was tested with the following possibilities:

- no filter or high-pass filter;
- 2, 4 or 6 second segments.

Model	Filtering	Segment size (s)	Accuracy
Random Forest	None	2	0.786629
		4	0.832372
		6	0.865020
	$f_c = 0.3Hz$	2	0.789016
		4	0.828200
		6	0.865130
FCN	None	2	0.906825
		4	0.919808
		6	0.945457
	$f_c = 0.3Hz$	2	0.894176
		4	0.923115
		6	0.944484
GRU	None	2	0.941113
		4	0.988501
		6	**0.998606**
	$f_c = 0.3Hz$	2	0.930186
		4	0.987140
		6	0.996113

Table 5.1: Accuracy of 3 different models, depending on segment size and how these were filtered (WISDM Data set).

Table 5.1 shows the results of these tests. The two important things to note are the increase in accuracy when the segment size increases, and the negligible difference in accuracy with or without filtering. The segment size will be investigated further later on, but for now these results do indicate there is no reason to apply a high pass filter to the data set.

5.3.1. Segment size

In Table 5.2 the test results, based on a sampling frequency of 20 Hz and no filter, can be seen. The accuracy of the models depending on window size has been plotted in 5.1. This test is conducted to further explore the influence of the sample size.

The random forest classifier seems to improve it's accuracy almost linearly with window size. The FCN and GRU however do increase performance initially, but when the window size becomes larger than 4 seconds, there is almost no more added benefit. Since predicting on a smaller window has the benefits that a new activity is recognised sooner and training time is shorter, a window size of 4 seconds is chosen.

5.4. Tests on GetSmart data

The recorded data by the data acquisition group, will be tested using the the test plan in Figure 5.2. The high-pass filter is not included, considering the tests on WISDM data set concluded to lower accuracy's compared to not filtering.

5.4.1. Over-fitting

Table 5.3 shows the classification results of the random forest classifier, FCN, and GRU on the GetSmart data set. These results are based on the train-test split with ratio 70:30 as described in Section 3.3.3. At a first glance, the high accuracy's already seems to good to be true. The most obvious explanation would be that the model has over fitted to the data set, which is not unlikely at all since the data set is relatively small. To test this hypothesis, leave-one-out cross-validation can be used.

Model	f_s (Hz)	Segment size (s)	Accuracy
Random Forest	20	2	0.782287
	20	3	0.815384
	20	4	0.835347
	20	5	0.856403
	20	6	0.870503
FCN	20	2	0.977321
	20	3	0.98997
	20	4	0.994087
	20	5	0.990006
	20	6	0.99791
GRU	20	2	0.966735
	20	3	0.981852
	20	4	0.985599
	20	5	0.988948
	20	6	0.993949

Table 5.2: Accuracy of 3 different models, depending on segment size (WISDM data set).

Model	f_s (Hz)	Segment size (s)	Accuracy
Random Forest	25	3	0.916530
		4	0.922158
		5	0.927474
	50	3	0.912258
		4	0.924241
		5	0.935045
FCN	25	3	0.998795
		4	0.998575
		5	0.996928
	50	3	0.997919
		4	0.998794
		5	0.999013
GRU	25	3	0.998686
		4	0.998355
		5	0.998683
	50	3	0.998028
		4	0.998246
		5	0.999122

Table 5.3: Accuracy of 3 different models, depending on segment size and sample frequency (GetSmart data set).

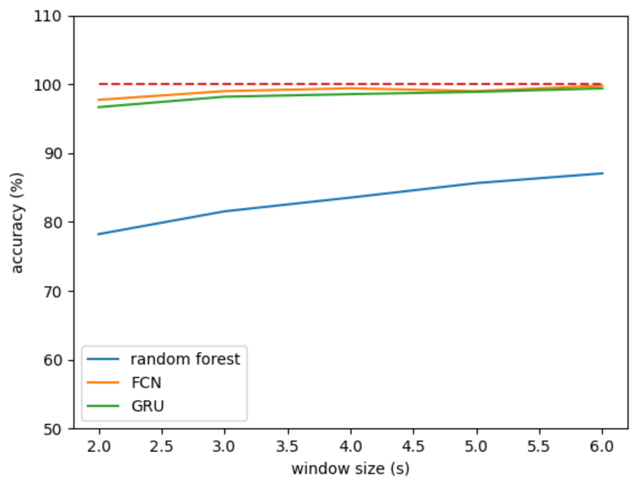


Figure 5.1: Accuracy vs window size for Random Forest, FCN and GRU

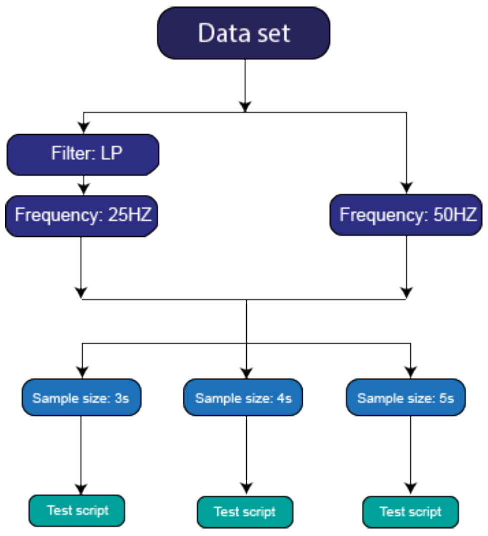


Figure 5.2: Test plan for GetSmart data set
,

5.4.2. Leave one out cross-validation

Leave-One-Out Cross-Validation (LOOCV) is a method to validate the performance of a model. The data set of size s, on which the model is tested, is split into a train and test set with sizes $s-1$ and 1, respectively. The model is trained s times by iterating over every possible combination for the trainset and tested on every instance.

As mentioned in chapter 3, the data set from the data acquisition subgroup is recorded by four persons performing five different activities. LOOCV is, for this case, applied by training the model on three test persons and validating it on the remaining subject. This means that each model has to be trained and tested 4 times. To save on computation time, the test is only ran using $f_s = 25$Hz and a segment size of 4 seconds. The results can be found in 5.4 and do indeed show that the averaged accuracy drops to approximately 81% for the random forest classifier, 87% for the FCN and 86% for the GRU.

Model	f_s	Segment size(s)	Train accuracy	LOOCV Accuracy	Fold
Random	25	4	-	0.671537	0
Forest	25	4	-	0.843829	1
	25	4	-	0.826739	2
	25	4	-	0.910641	3
FCN	25	4	0.999486	0.877749	0
	25	4	0.998617	0.907563	1
	25	4	0.999156	0.842761	2
	25	4	0.998617	0.867692	3
GRU	25	4	0.999684	0.932337	0
	25	4	0.998740	0.816639	1
	25	4	0.999015	0.764196	2
	25	4	0.998705	0.918128	3

Table 5.4: Train and cross-validation accuracy of 3 different models (GetSmart data set).

6
Conclusion and Discussion

6.1. Conclusion

In this study, we presented different machine-and deep learning models for human activity recognition. These are the Random Forest classifier, the Fully Convolutional Network (FCN), and the Gated Recurrent Unit (GRU). The first one being a classical machine learning model while the latter are deep learning models. The main difference between machine-and deep learning is that in deep learning no handcrafted features have to be inserted into the classification models. For Random Forest, many features were extracted from the smartwatch sensor data. The most relevant features are listed in table 4.2. These features were selected with Random Forest feature importance using Recursive Feature Elimination (RFE).

WISDM tests

The activities that are considered in this project are sitting, standing, walking, running, and walking stairs. The models are evaluated on the WISDM data set that contains smartwatch sensor data sampled at 20 Hz. This data set is used to find the optimal segment size and to investigate the usefulness of filtering. The models were tested for segment sizes ranging from 2 to 6 seconds and on filtered and unfiltered data. The test results in Table 5.1 show that filtering with a high-pass filter does not affect the performance of the models much, therefore unfiltered data is used during further tests on the GetSmart data set. Table 5.2 indicates that a segment size of 4 seconds is preferred since it is small enough to recognise new activities and big enough for decent classification accuracy.

GetSmart tests

The models are also evaluated on the data set recorded by the data acquisition subgroup. These tests showed that our models overfitted drastically on the data set as the results in Table 5.3 and Table 5.4 indicated. However, even when overfitting all models had an average accuracy above 80% using leave-one-out cross-validation. This does imply there is still a lot of room for improvement when training the models after gathering data from a larger pool of subjects, such as in the WISDM data set.

Programme of Requirements

To satisfy the mandatory requirements, the models had to be able to recognise 5 different types of activities: sitting, standing, walking, walking stairs and jogging, and do this based on segments of no more than 5 seconds length. The trade-off requirements included training the models on the GetSmart data set, annotating these data sets and training the models to detect falling.

Mandatory Requirements

For the first requirement, the Random Forest, FCN and GRU models are able to classify the WISDM data set with a respective accuracy of 84%, 99.4% and 98.6%, using 4 seconds segments. For the GetSmart data set, the models were overfitted, but using leave-one-out cross-validation an average accuracy of 81%, 87% and 86% respectively was still reached. Therefore, we consider this requirement reached.

The second requirement dealt with the segment size. A maximum of 5 seconds was set, and since the segments used for our final results were indeed shorter, this requirement was also reached.

Trade off requirements
For the trade off requirements the models needed to be trained and tested on the recorded GetSmart dataset. This has been done, but not necessarily as extensive as we would have liked, due to the relatively small size of the data set. We are therefore not able to draw conclusions about the accuracy of the model depending on sample frequency for example.

The second trade off requirement had us integrate the machine learning models on the web-based platform by the Data Acquisition subgroup. This has been implemented, but no tests have been done to check the accuracy of the system. As of now, we therefore consider this requirement not met.

For the final requirement the models had to be able to predict falling. Due to a lack of data sets and time, this has not been implemented.

Requirements met
To conclude, both mandatory requirements have been met, and a good deal of the first trade off requirement was also implemented. The implementation of the models on the GetSmart platform needs more time for testing and validation, but is certainly within reach.

6.2. Discussion

There are several limitations to our research. First of all, there are some issues with the GetSmart data set. The GetSmart data set (approximately 100 minutes of recording) is quite small compared to the WISDM data set (approximately 765 minutes of recording). Next to that, the GetSmart data set has little variety in its recordings, since the data is collected from only 4 different subjects. The WISDM data set on the other hand consists of data recorded from 51 different subjects. Carrying out such extensive data collection was also simply not within the scope of this project.

The other limitation of this research is the time we were able to spend on it, which resulted in not being able to fine tune and explore the data sets and models. For example, Table 5.1 took approximately 30 hours to compute. As a consequence we have done very little hyperparameter tuning of the models, although this can also be attributed to the fact that exploring the impact of segment sizes and filtering was of higher priority.

Recommendations
The developed models for HAR suffered from overfitting. As mentioned, the GetSmart data set is not extensive enough. In order to obtain more reliable and accurate models more recordings need to be added from different subjects. This will subsequently improve the accuracy and reliability of the machine and deep learning models.

In addition to that, more activities can be considered in the data set so that the models can be deployed on a larger number of tasks. Furthermore, more hyperparameter tuning can be performed to increase the accuracy of the models independent of the data sets used.

Bibliography

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

[2] B. Mohan, "Global smartwatch shipments hit a record high in q4 2021." `https://www.androidcentral.com/wearables/global-smartwatch-shipments-hit-a-record-high-in-q4-2021`. Accessed: 20-04-2022.

[3] Apple, "Specifications of the apple watch models." `https://www.apple.com/uk/watch/compare/`. Accessed: 21-04-2022.

[4] S. Abu-Alrub, M. Strik, F. D. Ramirez, N. Moussaoui, H. P. Racine, H. Marchand, S. Buliard, M. Haïssaguerre, S. Ploux, and P. Bordachar, "Smartwatch Electrocardiograms for Automated and Manual Diagnosis of Atrial Fibrillation: A Comparative Analysis of Three Models," *Frontiers in Cardiovascular Medicine*, vol. 9, 2022.

[5] T. Caillol, M. Strik, F. D. Ramirez, S. Abu-Alrub, H. Marchand, S. Buliard, N. Welte, S. Ploux, M. Haïssaguerre, and P. Bordachar, "Accuracy of a Smartwatch-Derived ECG for Diagnosing Bradyarrhythmias, Tachyarrhythmias, and Cardiac Ischemia," *Circulation: Arrhythmia and Electrophysiology*, vol. 14, no. 1, 2021.

[6] M. Nasarre, M. Strik, F. D. Ramirez, S. Buliard, H. Marchand, S. Abu-Alrub, S. Ploux, M. Haïssaguerre, and P. Bordachar, "Using a smartwatch electrocardiogram to detect abnormalities associated with sudden cardiac arrest in young adults," *EP Europace*, vol. 24, no. 3, pp. 406–412, 2021.

[7] T. Mishra, M. Wang, and A. e. a. Metwally, "Pre-symptomatic detection of covid-19 from smartwatch data," *Nature Biomedical Engineering*, vol. 4, pp. 1208–1220, Nov 2020.

[8] R. Liu, A. A. Ramli, H. Zhang, E. Henricson, and X. Liu, "An overview of human activity recognition using wearable sensors: Healthcare and artificial intelligence," *arXiv preprint arXiv:2103.15990*, 2021.

[9] R. Liu, S. A. Fazio, H. Zhang, A. A. Ramli, X. Liu, and J. Y. Adams, "Early mobility recognition for intensive care unit patients using accelerometers," *arXiv preprint arXiv:2106.15017*, 2021.

[10] M. S. Ryoo, B. Rothrock, C. Fleming, and H. J. Yang, "Privacy-Preserving Human Activity Recognition from Extreme Low Resolution," *Privacy-Preserving Human Activity Recognition from Extreme Low Resolution*, vol. 8, no. 1, pp. 1–8, 2017.

[11] I. Y. Jung, "A review of privacy-preserving human and human activity recognition," *International Journal on Smart Sensing and Intelligent Systems*, vol. 13, no. 1, pp. 1–13, 2020.

[12] M. Straczkiewicz, P. James, and J.-P. Onnela, "A systematic review of smartphone-based human activity recognition methods for health research," *npj Digital Medicine*, vol. 4, no. 1, 2021.

[13] R.-A. Voicu, "Human Physical Activity Recognition Using Smartphone Sensors," 01 2019.

[14] M. Z. Uddin, "Human activity recognition using wearable sensors, discriminant analysis, and long short-term memory-based neural structured learning," 08 2021.

[15] S. Mekruksavanich and A. Jitpattanakul, “Smartwatch-based human activity recognition using hybrid lstm network,” in *2020 IEEE SENSORS*, pp. 1–4, 2020.

[16] S. Mekruksavanich and A. Jitpattanakul, “Sensor-based complex human activity recognition from smartwatch data using hybrid deep learning network,” in *2021 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, pp. 1–4, 2021.

[17] S. Nair, M. Kheirkhahan, A. Davoudi, P. Rashidi, A. A. Wanigatunga, D. B. Corbett, T. M. Manini, and S. Ranka, “Roamm: A software infrastructure for real-time monitoring of personal health,” in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1–6, 2016.

[18] M. M. Hossain Shuvo, N. Ahmed, K. Nouduri, and K. Palaniappan, “A hybrid approach for human activity recognition with support vector machine and 1d convolutional neural network,” in *2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pp. 1–5, 2020.

[19] U. Verma, P. Tyagi, and M. Kaur, “Smartphone-based CNN-GRU Framework for Human Activity Recognition,” *2021 IEEE 2nd International Conference On Electrical Power and Energy Systems (ICEPES)*, pp. 1–4, 2021.

[20] R. Poll, M. Sluijs, and S. Liao, “Improving healthcare using smartwatches,” June 2022.

[21] D. Dua and C. Graff, “UCI Machine Learning Repository.” `https://archive.ics.uci.edu/ml`, 2017.

[22] G. M. Weiss, K. Yoneda, and T. Hayajneh, “Smartphone and smartwatch-based biometrics using activities of daily living,” *IEEE Access*, vol. 7, pp. 133190–133202, 2019.

[23] D. Thakur and S. Biswas, “An Integration of feature extraction and Guided Regularized Random Forest feature selection for Smartphone based Human Activity Recognition,” *Journal of Network and Computer Applications*, p. 103417, 2022.

[24] Z. Chen, C. Jiang, and L. Xie, “A novel ensemble elm for human activity recognition using smartphone sensors,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2691–2699, 2019.

[25] C. Fan and F. Gao, “Enhanced Human Activity Recognition Using Wearable Sensors via a Hybrid Feature Selection Method,” *Sensors*, vol. 21, no. 19, p. 6434, 2021.

[26] A. Shaafi, O. Salem, and A. Mehaoua, “Improving human activity recognition algorithms using wireless body sensors and svm,” pp. 607–612, 2020.

[27] N. C. Minh, T. H. Dao, D. N. Tran, N. Q. Huy, N. T. Thu, and D. T. Tran, “Evaluation of smartphone and smartwatch accelerometer data in activity classification,” pp. 33–38, 2021.

[28] D. Gabor, “Theory of communication. Part 1: The analysis of information,” *Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[29] S. Fortes, “Transform-Based Multiresolution Decomposition for Degradation Detection in Cellular Networks,” 10 2020.

[30] M. Yang, Y.-F. Sang, C. Liu, and Z. Wang, “Discussion on the Choice of Decomposition Level for Wavelet Based Hydrological Time Series Modeling,” *Water*, vol. 8, no. 5, p. 197, 2016.

[31] “Recursive Feature Elimination — Yellowbrick v1.4 documentation,” 2019.

[32] T. K. Ho, “Random decision forests,” vol. 1, pp. 278–282 vol.1, 1995.

[33] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O’Reilly Media, 3 ed., 2022.

[34] Wikimedia Commons, “,” 05 2021.

[35] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” pp. 1–6, 2017.

[36] Z. Wang, W. Yan, and T. Oates, “Time series classification from scratch with deep neural networks: A strong baseline,” *CoRR*, vol. abs/1611.06455, 2016.

[37] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, “The ucr time series classification archive,” July 2015. `www.cs.ucr.edu/~eamonn/time_series_data/`.

[38] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder–decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1724–1734, Association for Computational Linguistics, Oct. 2014.

[40] P. Werbos, “Backpropagation through time: what it does and how to do it,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[41] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” 2014.

A
Features

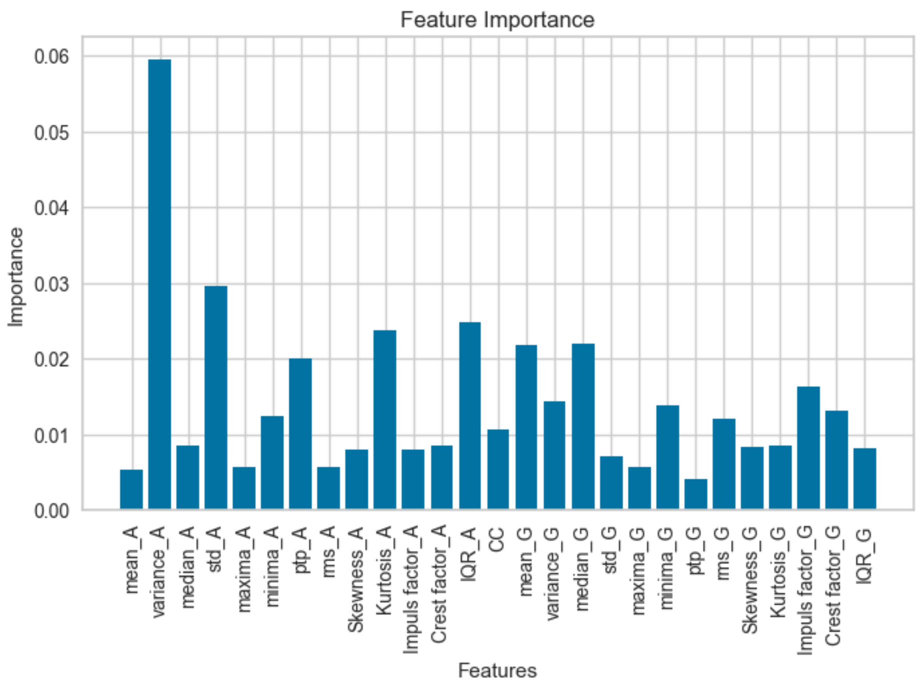


Figure A.1: Feature importance of the accelerometer and gyroscope in time domain.

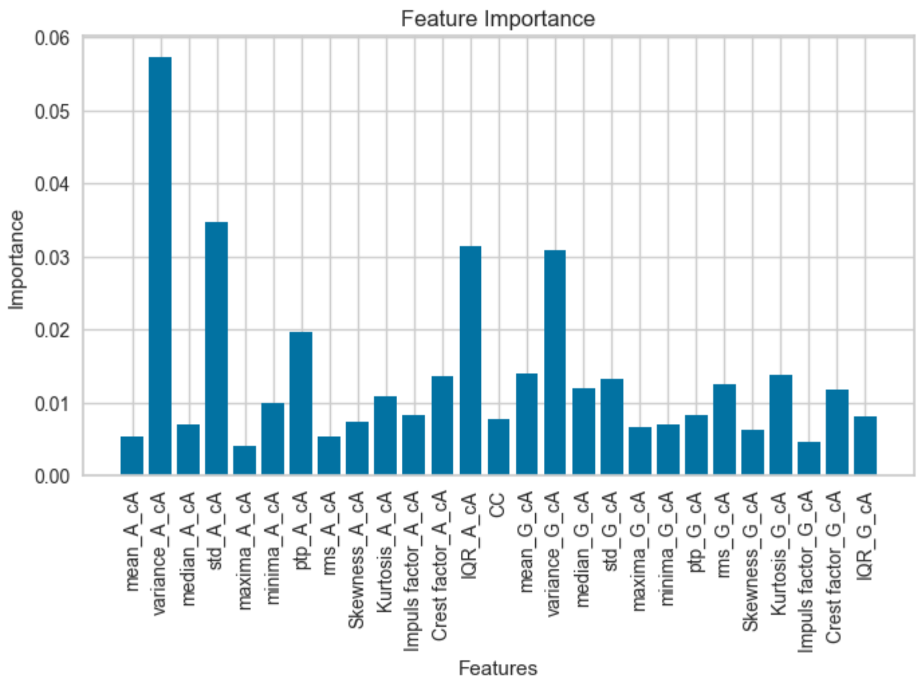


Figure A.2: Feature importance of the accelerometer and gyroscope in time-frequency domain of the approximation coefficient.

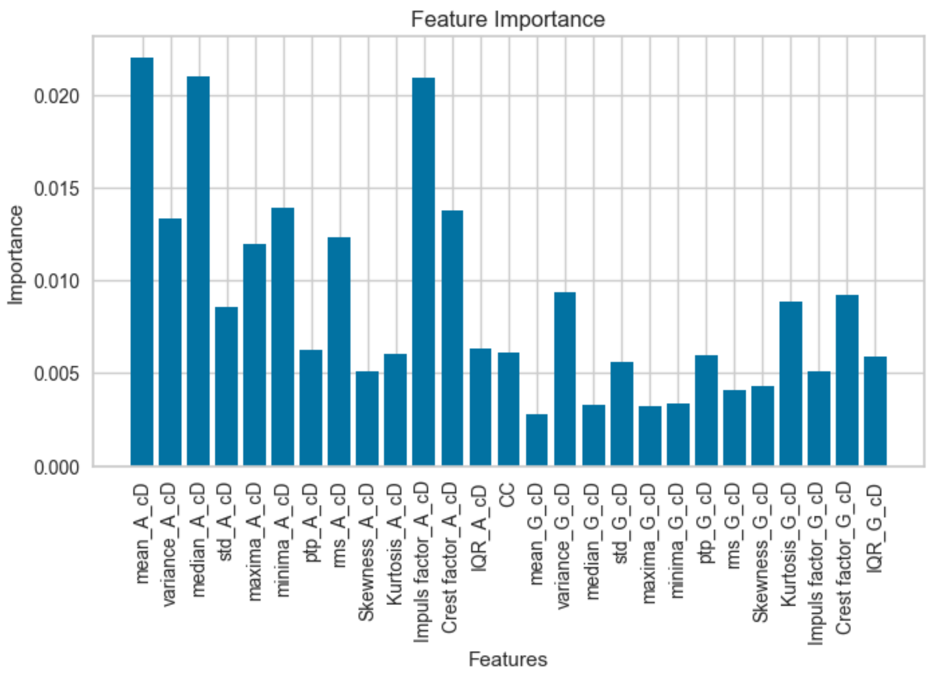


Figure A.3: Feature importance of the accelerometer and gyroscope in time-frequency domain of the detail coefficient.