Linda Kolb, BSc

Using Artificial Intelligence to Classify Activities Captured in Smart Homes

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing. Dr.techn. Roman Kern

Institute for Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Bad Aussee, October 2019

This document is set in Palatino, compiled with pdfLaTeX2e and Biber.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Contents

Abstract v

1. Introduction 1

2. Related Work 7
2.1. Background . . . 7
2.2. State of the Art . . . 7

3. Use Cases & Requirements 13

4. Data Preparation 17
4.1. Data Collection . . . 17
4.2. Exploratory Data Analysis . . . 20
4.3. Data Preprocessing . . . 25
4.4. Feature Selection . . . 28

5. Chosen Classification Algorithms 31
5.1. Standard Machine Learning Algorithms . . . 32
5.2. Deep Learning Algorithms . . . 32

6. Evaluation 35
6.1. Standard Machine Learning Algorithms . . . 36
6.2. LSTM network . . . 40
6.3. CNN . . . 40

7. Conclusions 47

A. Code Snippets 51

Bibliography 53

List of Figures

4.1. Layout of binary sensors 18
4.2. Layout of proximity sensors 19
4.3. Layout of intelligent floor modules 19
4.4. Bar Chart visualising the frequencies of each activity in the training data set 21
4.5. Bar Chart visualising the frequencies of each activity in the testing data set 22
4.6. GANTT Chart showing which activities the habitant performed 23
4.7. Durations per activity in the training data set 24
4.8. Acceleration data from wrist watch 27

5.1. Input Tensor for CNN and LSTM network 33
5.2. CNN architecture 34

6.1. Metrics of different machine learning algorithms 38
6.2. RMSE of different machine learning algorithms 39
6.3. Confusion matrix for the classification results of Logistic Regression 41
6.4. Confusion matrix for the classification results of the LSTM model 43
6.5. Different metrics of the deep learning models LSTM and CNN over the epochs. 44
6.6. Confusion matrix for the classification results of the CNN model 45
6.7. Different metrics of the 10 runs of the experiments using LSTM and CNN over the epochs. 46

1. Introduction

Due to the expansion of the internet and an increasing digitalisation of everyday appliances, new ways of using computers to improve the living quality of humans have evolved. Recent development in wide area networks and sensors becoming inexpensive and less power consuming facilitate the technological extension of ordinary objects to an ubiquitous net of interconnected devices. Those devices can communicate with each other based on standard communication protocols known as the Internet of Things (IoT).

One application of IoT is human activity recognition in environments that contain additional technological enhancements which make them"smart". An example for such an environment is a smart home where sensors detect when a person lies down in the bedroom to sleep. Once the smart home system picks up on the activity using sensors, it recognises the activity as "sleeping" using a classification algorithm. The system assists the person and switches off the heating of the room without any human interference.

The topic of real live activity recognition in assisted living is relevant to the world's society because of various reasons: Ann and Lau, 2014 lists application areas such as surveillance systems, healthcare, eldercare and human computer interaction. Since the population is growing older and older, new ways of dealing with eldercare have to be found. The goal is to enable elderly people to stay in their own home for as long as possible instead of sending them to nursing homes which might not be a long-term option due to financial reasons. A monitoring smart home collects sensor data, detects activities, reacts and assists the human when needed. Other promising use cases of human activity recognition are assistance with coordination and scheduling. It becomes easier for the home owner to manage tasks as for example sustainable and economic heating or energy management.

This thesis tries to answer the research question: How can artificial intelligence help to classify activities captured in smart environments? Artificial intelligence describes systems that act "intelligently" which means that the system simulates intelligent decisions - just like a human. For a human being it would be fairly easy to put a label on an observed human activity. However, a computer must learn how to distinguish between different activities and what action has to be taken accordingly. Using already labelled data to train a machine learning model to recognise certain activities by their features is categorised as a supervised learning task.

Massive amounts of data are required to ensure that a generalised model can actually differentiate between activities. The data set used to train the model of this project is the data set from the Ubiquituous Computing and Ambient Intelligence (UCAMI) Cup which is explained in detail in Espinilla, Medina, and Nugent, 2018. Llabelled data sets for activity recognition are seldom available due to the time-consuming effort and necessary infrastructure of creating them. The data has been captured inside a controlled environment that tried to simulate the real life. A 24 year old male student has simulated living in the smart home for ten days. The data set includes data from four different sources with labels for 24 different categories of activities of daily living. The captured sensor recordings are split into a labelled training set from seven days and three days worth of test data.

The list of activities and their definition according to Espinilla, Medina, and Nugent, 2018 are presented in Table 1.1.

Activity Name	**Description of Activity**
Take medication	This activity involved the inhabitant going to the kitchen, taking some water, removing medication from a box and swallowing the pills.
Prepare breakfast	This activity involved the inhabitant going to the kitchen, taking some products for lunch. This activity can involve (i) making a cup of tea with kettle or (ii) making a hot chocolate drink with milk in the microwave. This activity involves placing things to eat in the dining room, but not sitting down to eat.

Prepare lunch	This activity involved the inhabitant going to the kitchen, and taking some products from the refrigerator and pantry. This activity can involve (i) preparing a plate of hot food on the fire, for example pasta or (ii) heating a precooked dish in the microwave. This activity also involves placing things to eat in the dining room, but not sitting down to eat.
Prepare dinner	This activity involved the inhabitant going to the kitchen, and taking some products from the refrigerator and pantry. This activity can involve (i) preparing a plate of hot food on the fire, for example pasta or (ii) heating a precooked dish in the microwave. This activity also involves placing things to eat in the dining room, but not sitting down to eat.
Breakfast	This activity involved the inhabitant going to the dining room in the kitchen in the morning and sitting down to eat. When the inhabitant finishes eating, they place the utensils in the sink or in the dishwasher.
Lunch	This activity involved the inhabitant going to the dining room in the kitchen in the afternoon and sitting down to eat. When the inhabitant finishes eating, he places the utensils in the sink or in the dishwasher.
Dinner	This activity involved the inhabitant going to the dining room in the kitchen in the evening and sitting down to eat. When the inhabitant finishes eating, they place the utensils in the sink or in the dishwasher.
Eat a snack	This activity involved the inhabitant going to the kitchen to take fruit or a snack, and to eat it in the kitchen or in the living room. This activity can imply that the utensils are placed in the sink or in the dishwasher.
Watch TV	This activity involved the inhabitant going to the living room, taking the remote control, sitting down on the sofa and when he was finished, the remote control was left close to the TV.

Enter the SmartLab	This activity involved the inhabitant entering the SmartLab through the entrance at the main door and putting the keys into a small basket.
Play a video game	This activity involved the inhabitant going to the living room, taking the remote controls of the TV and XBOX, and sitting on the sofa. When the inhabitant finishes playing, he gets up from the sofa and places the controls near the TV.
Relax on the sofa	This activity involved the inhabitant going to the living room, sitting on the sofa and after several minutes, getting up off the sofa.
Leave the SmartLab	This activity involved the inhabitant going to the entrance, opening the main door and leaving the SmartLab, then closing the main door.
Visit in the SmartLab	This activity involved the inhabitant going to the entrance, opening the main door, chatting with someone at the main door, and then closing the door.
Put waste in the bin	This activity involved the inhabitant going to the kitchen, picking up the waste, then taking the keys from a small basket in the entrance and exiting the SmartLab. Usually, the inhabitant comes back after around 2 min, leaving the keys back in the small basket.
Wash hands	This activity involved the inhabitant going to the bathroom, opening/closing the tap, lathering his hands, and then rinsing and drying them.
Brush teeth	This activity involved the inhabitant going to the bathroom and brushing his teeth and opening/closing the tap.
Use the toilet	This activity involved the inhabitant going to the bathroom and using the toilet, opening/closing the toilet lid and pulling the cistern.
Wash dishes	This activity involved the inhabitant going to the kitchen and placing the dirty dishes in the dishwasher, and then placing the dishes back in the right place.

Put washing into the washing machine	This activity involved the inhabitant going to the bedroom, picking up the laundry basket, going to the kitchen, putting clothes in the washing machine, waiting around 20 min and then taking the clothes out of the washing machine and placing them in the bedroom closet.
Work at the table	This activity involved the inhabitant going to the workplace, sitting down, doing work, and finally, getting up.
Dressing	This activity involved the inhabitant going to the bedroom, putting dirty clothes in the laundry basket, opening the closet, putting on clean clothes and then closing the closet.
Go to the bed	This activity involved the inhabitant going to the bedroom, lying in bed and sleeping. This activity is terminated once the inhabitant stays 1 min in bed.
Wake up	This activity involved the inhabitant getting up and out of the bed.

Table 1.1.: The activities of daily living that are represented in the data set along with their description.

It is a non-trivial task to deal with a large amount of data which comes with using sensors in the domain of activity recognition. The contribution of this academic work to research in this field is an analysis of what kind of classification algorithms can be used and how they compare.

The theoretical way of searching for the solution of the research question is to study different methods other researchers have used to incorporate artificial intelligence to classify human activity recognition in the context of IoT. The main objective is to choose a fitting method for classification and to settle on optimal input parameters.

Regarding the practical part of this work, a system is implemented to test the conclusions from the literature review. The mentioned data set from

UCAMI Cup is processed and several machine learning and deep learning models are tested to classify the data. Since human activity recognition using sensors from a smart environment has only recently developed, not many solutions for machine learning algorithms have been evaluated. Deep learning has barely been used in the context of human activity recognition. It is more known in the domains of image classification and natural language processing. After the models have been trained, validated and tested, a confusion matrix as well as other metrics are obtained to compare the performance of the classification. The data set has already been explored by other researchers who performed different classification approaches. The results of this thesis are compared to the existing ideas. Experiments of different solutions applied to the same data set can help data scientists learn from each other and develop new algorithms based on what has worked and has not worked for others.

Chapter 2 discusses the theoretical background of the used algorithms and concepts and gives an overview of the state of the art in the field of human activity recognition. Chapter 3 analyses the use cases and requirements for the problem stated. Chapter 4 explores the input data set and explains how the data is prepared. Chapter 5 shows the classification algorithms in detail and how the experiments are performed. The results of the algorithms are evaluated in Chapter 6. Chapter 7 summarises the work of this thesis and gives some concluding remarks.

2. Related Work

2.1. Background

The chosen programming language is Python[1] including the powerful tools Pandas[2] (preprocessing) and scikit-learn[3] (machine learning algorithms) as well as Matplotlib[4] (plots for visualisation). In terms of deep learning, the neural network library Keras[5] are used. The hyperparameter optimisation for the Keras models are implemented using Talos[6].

2.2. State of the Art

Relevant literature is researched using keywords from the following list:

- Activity recognition in smart environments
- Activity recognition classification
- Sensor-based activity recognition
- Time series classification
- Activity recognition machine learning smart sensor
- Activity recognition for activities of daily living
- Multivariate time series classification
- CNN activity recognition
- CNN with LSTM

[1] *Python Programming Language* 2019.
[2] *Pandas - Python Data Analysis Library* 2019.
[3] *scikit-learn - Machine Learning in Python* 2019.
[4] *Matplotlib - Plotting in Python* 2019.
[5] *Keras - Library for neural networks* 2019.
[6] *Talos - Hyperparameter Optimization for Keras Models* 2019.

- Conditional Random Fields
- ResNet
- GoogLeNet
- VGG

Appropriate search engines for literature research in this domain are Semantic Scholar[7], the sensors section of the MDPI Open Acces Journal[8] and Google Scholar[9]. Additionally, the website of the Ulster Institutional Repository[10] at Ulster University has related papers to the problem statement of this thesis. The papers explaining and analysing the chosen dataset where taken from the proceedings of the UCAMI 2018 conference. The relevance of found papers is determined by the year it was published in and by the similarity to the problem domain.

The chosen data set for the project has already been analysed by different researchers to settle on a good classification solution. Some of them presented strong results based on the accuracy of how many classes were classified correctly. Karvonen and Kleyko, 2018 approached the problem with a domain knowledge-based solution. They chose to use only input data from the binary sensors out of all four data sources included in the data set. The paper states that conventional machine learning algorithms such as Support Vector Machines do not consider the temporal relationships between activities. They argue further that Recurrent Neural Networks which are able to learn sequences are more complex and would need a greater amount of training data than what is provided in the data set. This would lead to overfitting of the model. The presented solution is an expert system similar to a Finite State Machine with an accuracy of 81.3%. This type of system has the advantage that it is easy for humans to understand. Especially in the ehealth sector, the users of this system should be able to comprehend and reproduce how the classification was done. An issue with this type of solution is that once the inhabitant of the smart home changes the daily habits, the system struggles to classify the activities correctly. Another problem with sequences and this data set is that the data was captured in a controlled environment. The test person in the lab

[7] *Semantic Scholar* 2019.
[8] *Sensors - MDPI Open Access Journal* 2019.
[9] *Google Scholar* 2019.
[10] *Ulster Institutional Repository, Ulster University* 2019.

might behave differently because the person is aware that the activities are recorded. Furthermore, humans might switch their habits our of intrinsic or extrinsic motivation. The data set is also fairly small to provide a reliable input to train the model.

The details of the solution proposed by Lago and Inoue, 2018 describe a combination of a probabilistic and a descriptive model. They apply the binary sensor and proximity sensor data as inputs for the activity recognition process. The probabilistic model is based on a hidden markov chain and represents each activity as a state. Since the test person in the smart home lab switches from activity to activity, the result is a sequence of observations which fits for the application of hidden markov chains. The emission probabilities for the Markov Chain are calculated using a neural network and also considers the mean duration for every type of the 24 activities. The description-based activity recognition model utilises the verbal description of every activity given by the data set description. The logic of the model is implemented using Java classes, but due to the difficulty of transforming the activity descriptions into logic, the activities "wash hands" and "wash dishes" were skipped.

Salomón and Tîrnăucă, 2018 solved the time series classification problem with the given data set using a semi-supervised machine learning algorithm with a weighted finite automata and regular expressions. Additionally to using the binary and proximity senors, their contribution uses the floor sensor data to figure out the test person's position in the room through frequency distribution. The proposed system results in an accuracy of 90.65%. They argue that the errors in their prediction model are mainly caused by wrong predictions of starting and ending times of the activities. They believe this is due to the human transcribers making errors and errors in the data set.

Ceron, López, and Eskofier, 2018 submitted a solution using the CRoss Industry Standard Process for Data Mining (CRISP-DM) methodology for data mining projects. The six phases of this methodology according to Wirth and Hipp, 2000 are:

1. Business understanding
2. Data understanding
3. Data preparation

4. Modeling
5. Evaluation
6. Deployment

Jiménez and Seco, 2018 presents a multi-event naive Bayes classifier to estimate which activities were performed. This approach takes the input information from all four data sources. Using the training data set, the accuracy of the classification is 68%. Performing the classification with the test data set reaches an accuracy of 60.5%.

In Razzaq et al., 2018 the researchers achieve a 94% accuracy for training data and a 47% accuracy for the test data set using a FilteredClassifier from the Weka Tool *Weka 3: Data Mining Software in Java* 2019. This system also used all available sensor data to train the model.

The system by Ceron, López, and Eskofier, 2018 takes input data from the event streams of the binary senors, the proximity data, the acceleration data from the smart watch and the location data from the intelligent floor. The used model is based on J48 (the base classifier), Ib1, support vector machines, random forest, AdaBoostM1 and bagging. A problem that is mentioned concerning this solution is that the data set has a class imbalance: There are not enough samples of the activities "wash dishes" and "Playing video game" to predict those activities reliably. This issue could have been solved by simply balancing out the samples in the data set through under- or oversampling. Even though the 10-fold-cross-validation on the given training set resulted in a classification accuracy of 92.1%, the accuracy for classifying activities of the test data set was only 60.1%. The activities related to meals of the day were added into one single event e.g. "dinner", "lunch" and "breakfast" were summarised to the activity "eating". This step was also performed to the activities related to preparing those meals. Merging those activities increased the classification accuracy by 13%. The bodily movements of those activities are indistinguishable, only the time of the day is different.

According to Razzaq et al., n.d., the main issue with standard pattern recognition methods (support vector machines, hidden markov models, naive bayes) used for sensor-based activity recognition is that the features are always extracted with hand-crafted approaches, based on experience and domain knowledge of the data scientist programming the model. The

extracted features end up being rather shallow e.g. statistical features like mean or variance. This might not be enough to learn complex activities. Another problem with activity recognition is that usually the labeled data set is small due to the fact that it had to be labeled by someone and a large amount is needed to learn activities that might not always yield exactly the same sensor data in the same way. Additionally, most standard pattern recognition algorithms learn from static data, but sensor data for activity recognition comes in sequences of sensor data observation where each sample has different sensor events that only make sense when being considered sequentially. Alternatively, deep learning models can perform abstract feature extraction and model building simultaneously. One option of a deep learning model relevant to the problem domain of activity recognition are convolutional neural networks (CNNs). When using time series data, CNNs can leverage the abilities of detecting local patterns Razzaq et al., n.d.

Hammerla and Plötz, 2016 explored several deep learning algorithms for human activity recognition in their paper. Different benchmark data sets were used to compare deep feed-forward networks (DNN), convolutional networks (CNN) and recurrent networks based on LSTM (long short-term memory). Their results show that bi-directional LSTMs performed significantly better than other state-of-the art classification algorithms.

Another machine learning algorithm that has become popular for time series classification are conditional random fields (CRFs). Nazerfard et al., n.d. conclude that CRFs can outperform hidden markov models applied to sensor-based human activity recognition. CRFs are probabilistic models mainly used for natural language processing and can learn observation sequences.

3. Use Cases & Requirements

This section includes three different use case diagrams to demonstrate in detail how a user might interact with the system presented in this thesis. The proposed use cases are related to currently relevant topics like the growing percentage of elderly people in the world's society (ehealth, eldercare) as well as the disposal of waste (waste management). The general idea of the system is to classify activities of daily living with data from a smart home, analyse the data, track anomalies and take action accordingly. The table explains the use cases using a Sensing-Logic-Action approach. Sensing describes the data sources for receiving the sensor information, logic characterises the analysing and decision making part of the system and action defines what action the system will take in this specific use case. The tables 3.1, 3.2 and 3.2 describe the potential use cases in detail.

Title	Turning off the heating after the inhabitant goes to sleep
Problem description	The inhabitant of the smart home goes to sleep and wants the heating to be off to save energy.
Sensing	Location data from intelligent floor, binary and proxmity sensor data (for example proximity sensors of the bed), acceleration data from wrist watch.
Logic	The person lies down in the bed and stays there for moe than one minute. The logic part of the system recocgnises that the activity "go to the bed" has started.
Action	The smart home system turns off the heating in case it is on.

Table 3.1.: A use case for turning off the heating once the inhabitant goes to sleep.

Title	Tracking meal preparation for the elderly
Problem description	An elderly person might not be able to prepare a meal by themselves or forget to have a meal regularly.
Sensing	Location data from intelligent floor, binary and proxmity sensor data (for example magnetic contact and proximity sensors for the fridge), acceleration data from wrist watch.
Logic	A segment of the day (morning, afternoon, evening) ends without the activity of eating which means that tis activitiy has been skipped. The involved activities are Act05 "breakfast", Act06 "lunch" and Act07 "dinner" .
Action	A delivery service is notified to provide a meal for the inhabitant.

Table 3.2.: A use case for tracking meals using the proposed system.

Title	Efficient waste management in the home
Problem description	The activity "put waste in the bin" is monitored to ensure that waste is picked up once the waste bin outside of the smart home is full.
Sensing	Location data from intelligent floor, binary and proxmity sensor data (for example magnetic contact and proximity sensors for the trash), acceleration data from wrist watch.
Logic	The logic part of the system tracks how often this activity happens and when the waste collection was last called. The related activity is Act15 "put waste in the bin".
Action	The related action is to call waste collection after a pre-defined number of occurrences of this activity.

Table 3.3.: A use case for tracking waste disposal using the proposed system.

4. Data Preparation

This chapter describes how the data was capured followed by the preprocessing and analysis of the data set. The chosen programming language is Python[1] including the powerful tools Pandas[2] as well as matplotlib[3] for the plots.

4.1. Data Collection

The sensor data used for this project is gained from the data recorded inside a smart lab. The data set was published for the UCAMI Cup which was a competition for human activity recognition. A single habitant, a 24 year old student, carried out 24 different activities of daily living in a smart lab at the University of Jaén.

The smart lab tries to reproduce the environment of a real life apartment and to capture what the habitant is doing inside of the lab. The smart lab measures 25 square meter and has five different areas inside of it: entrance hall, kitchen, living room with a work desk and bedroom with an integrated bathroom (see Figure 4.3 for the interior of the lab). Several sensors have been placed inside of the lab in order to record the habitant's actions. The sensors collecting the data are deployed in close proximity or even directly attached to objects of interest. The four types of data sources included in the UCAMI data set are:

- 30 binary sensors transmitting a binary value for either magnetic contact, motion or pressure (see Figure 4.1 for their layout)

[1] *Python Programming Language* 2019.
[2] *Pandas - Python Data Analysis Library* 2019.
[3] *Matplotlib - Plotting in Python* 2019.

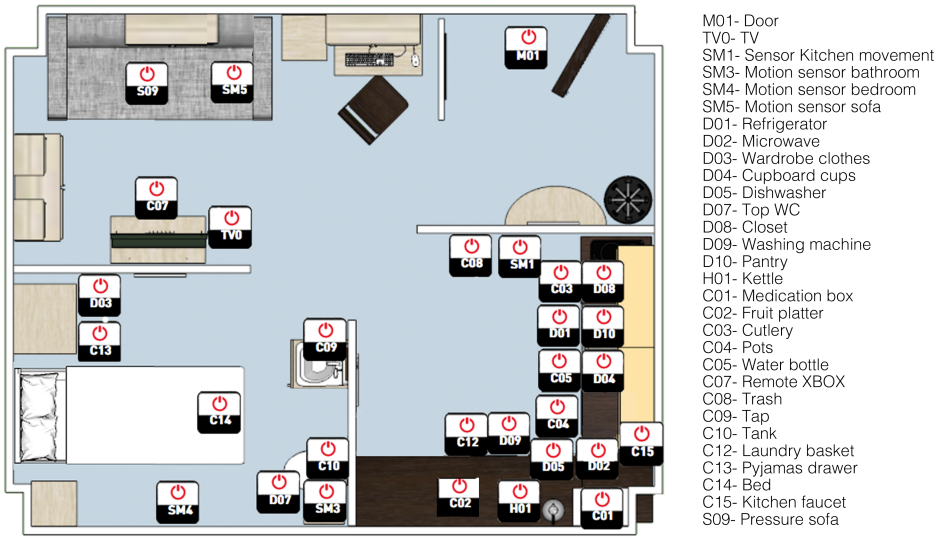


Figure 4.1.: Layout of binary sensors in smart home lab. Reprinted from Espinilla, Medina, and Nugent, 2018 (CC BY-NC 4.0).

- Proximity data between a smart watch worn by an inhabitant and a set of 15 Bluetooth Low Engergy (BLE) beacons positioned as can be seen in Figure 4.2
- Acceleration data from 3 axes generated by a smart watch
- Location information provided by an intelligent floor with 40 tiles (see Figure 4.3)

As the layouts of the deployed sensors show, the devices capturing the data are stationary except for the acceleration data coming from the wrist watch. The objects are placed close to areas inside the smart lab that are related to the activities defined in Table 1.1.

The basic idea of creating the proposed system is to develop techniques and tools to provide solutions for improving assistance in smart homes. Using a model for activity recognition can predict what kind of activity the habitant of the smart home is performing. Depending on the activity, actions can be taken during the activity or after it is over. For example, the smart home can assess how often the habitant has taken the trash out. After a specific

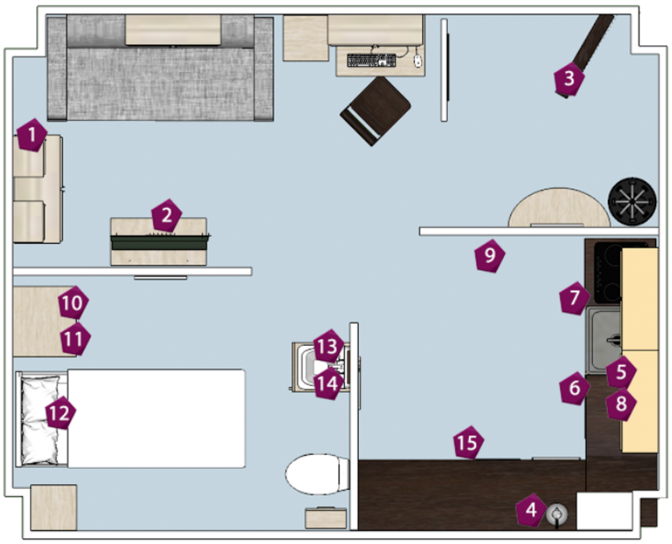


Figure 4.2.: Layout of proximity sensors in smart home lab. Reprinted from Espinilla, Medina, and Nugent, 2018 (CC BY-NC 4.0).

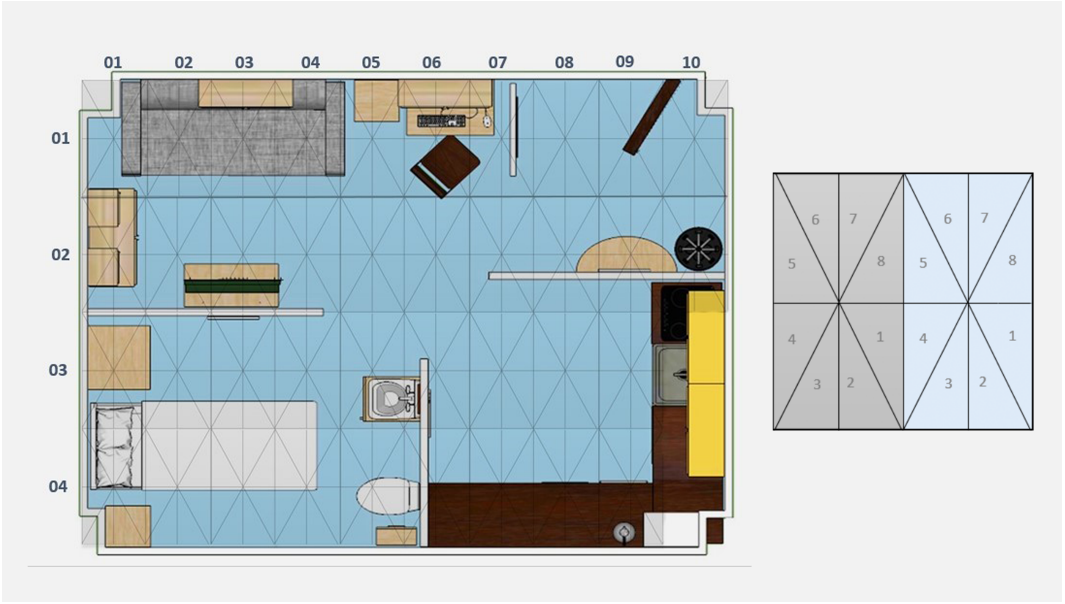


Figure 4.3.: Layout of intelligent floor modules in smart home lab. Reprinted from Espinilla, Medina, and Nugent, 2018 (CC BY-NC 4.0).

amount of time, the smart home system can alert the waste collection to collect all the trash bags. The description of the activity "put waste in the bin" is: "This activity involved the inhabitant going to the kitchen, picking up the waste, then taking the keys from a small basket in the entrance and exiting the Smart Lab. Usually, the inhabitant comes back after around 2 min, leaving the keys back in the small basket." The related sensors to this activity are the binary sensor for the magnetic contact on the trash bin and on the door, the intelligent floor tiles in the kitchen and in the entrance and the proximity sensors on the door and garbage can. There are eleven samples of this activity in the training set and four in the test set.

4.2. Exploratory Data Analysis

When comparing frequencies of each activity from the testing set in Figure 4.4, it is clear that the classes are unbalanced since there are high differences in the frequencies. The activity that is seen most often is "Brush teeth" with 21 counted occurrences. 169 activities were recorded in total. The activities "Relax on the sofa", "Visit in the SmartLab" and "Play a video game" only occurred once in the testing set.

In the testing data set of only three days worth of data, there are 77 activities (see Figure 4.5 for their frequencies). Two activities do not occur in the testing data set: "eat a snack" and "put washing into washing machine". The activity "relax on the sofa" which is only present once in the training set, was tracked eight times in the testing set.

Between the activities, there are times where sensor data is present, but no activity is assigned. When preprocessing, those samples are labelled with "no activity" in the data set. The "no activity" samples were omitted of the data set in the end because they barely occur in the testing set since the labelling process worked differently there. Figure 4.6 shows an example of the activities in the morning segment of the 31st of October, 2017. The activities are happening in sequential order, never in parallel. In between the recorded activities, there are gaps that result from sensor data with timestamps that are not assigned to any activity. The GANTT chart shows a typical morning: The habitant wakes up, uses the toilet, washes his hands,

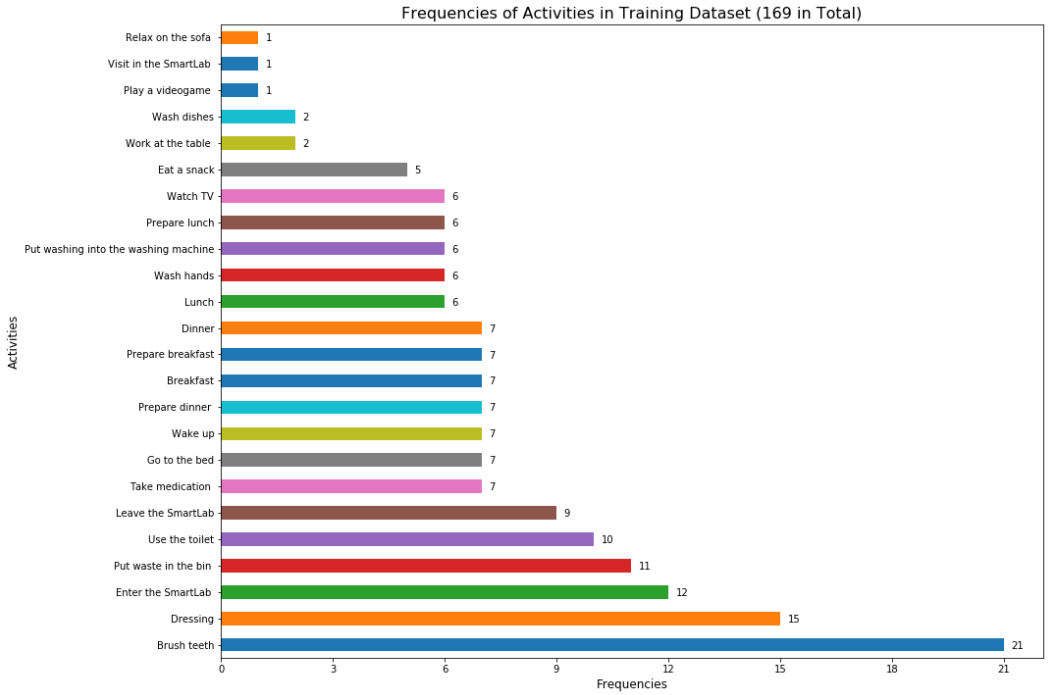


Figure 4.4.: Bar Chart visualising the frequencies of each activity in the training data set.

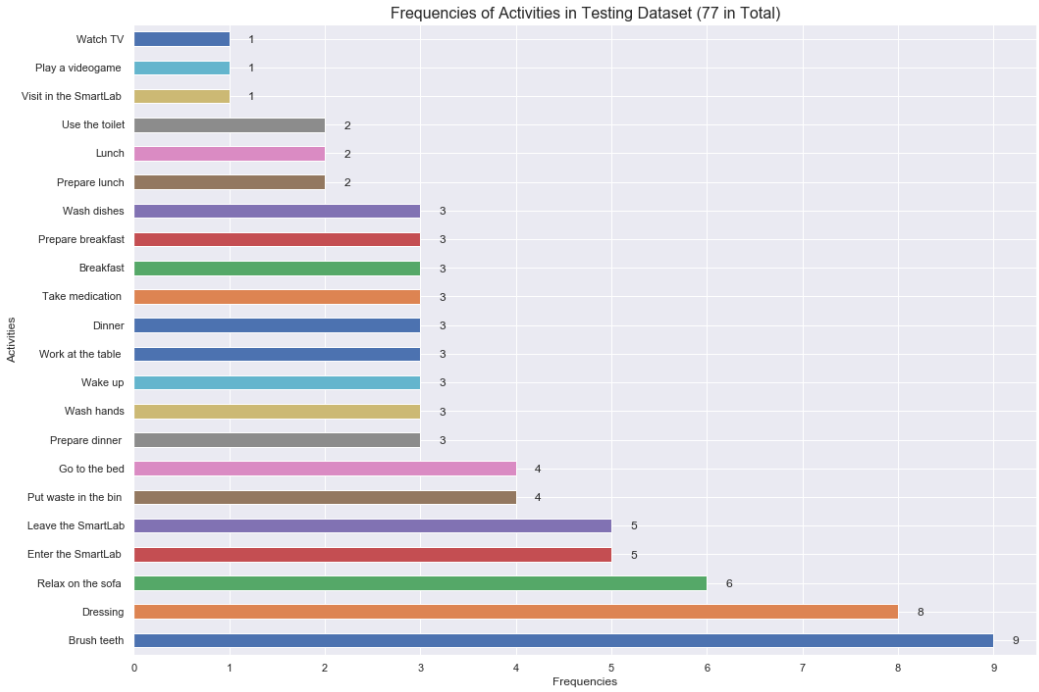


Figure 4.5.: Bar Chart visualising the frequencies of each activity in the testing data set.

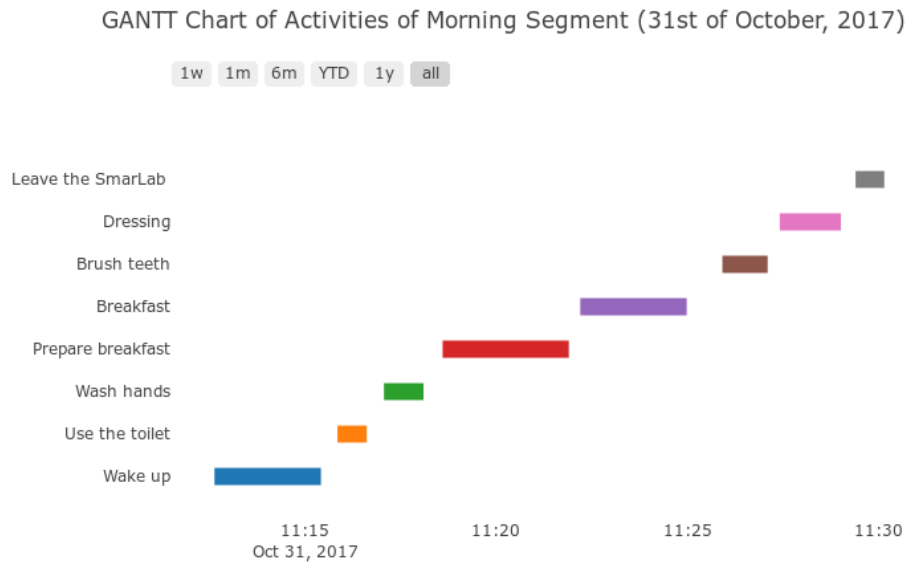


Figure 4.6.: This GANTT Chart shows which sequence of activities the habitant performed in the morning segment of the 31st of October, 2017.

prepares breakfast, eats breakfast, brushes his teeth, dresses himself and then leaves the smart lab. The activities "wake up", "prepare breakfast" and "breakfast" itself take the longest time. The shortest activities from that morning are "leave the smart lab" and "use the toilet".

In Figure 4.7 the durations of the activities taken from the training set are visualised. A few outliers are visible in the box plot. The activities that tend to result in the longest durations are "put washing into the washing machine", "work at the table" and "watch TV" Nevertheless, the activity "watch TV" has a large interquartile range which indicates that there are various samples with highly fluctuating durations for this activity in particular.

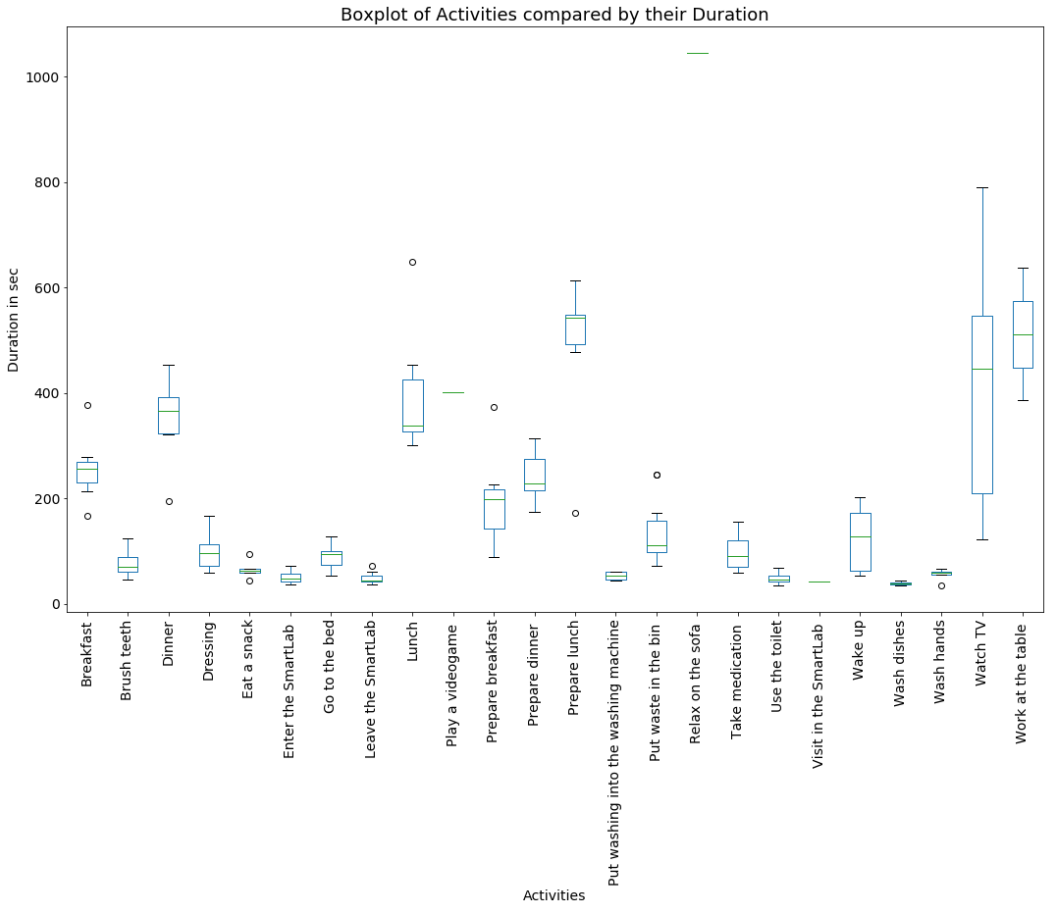


Figure 4.7.: The durations per activity in the training data set are shown as a box plot.

4.3. Data Preprocessing

The data is split into testing and training according to the days the sensor data was captured inside the smart lab: Seven days were used for the training, three days were used for the testing. The data set is divided into different times of the day - morning, afternoon and evening. The testing data is preprocessed in the same way as the training data.

The sensors have different frequencies at which data is being sent. The highest amount of sensor information comes from the acceleration data source, collected with a sample frequency of 50 Hz. First, each of the data sources is merged into samples of a duration of five seconds. This step is carried out because it is easier for a model to recognise an activity with a broader time slot that has more information stored to hint at what activity could be performed currently. Sometimes it happens that multiple sensor data is present for the same five second slot, therefore the sensor data needs to be aggregated into one value for each sensor per five seconds. The floor data is transformed into one feature per module and is set to 1 if there was sensor data collected in that time slot, otherwise the value is 0 meaning there is no sensor data coming from a module in that specific slot. The binary sensors and proximity sensors are added to the features as one variable per device. The binary sensors already come in a binary format, 1 signaling that there is motion, pressure or no magnetic contact present. The value is 1 for all samples until a 0 is received from the sensor meaning for example that the pressure is not there anymore. If there are multiple entries for the same object in the same time slot, the value is summarised to 1 in case there was a 1 in the merged samples coming from the sensor during that slot. Otherwise the value for that object is 0. The proximity sensor data is measured using a Received Signal Strength Indicator (RSSI) which translates to how close an object with a proximity sensor attached is to the wrist watch worn by the habitant. Thus, the proximity data is added with 0 meaning no proximity at all. Due to the quality of the signal, the RSSI is preprocessed in the way that any measurement greater than -97 means proximity - the higher the value, the closer the distance to the object placed inside the apartment. If the RSSI measure gives evidence for proximity, the value for the respective device is 1, otherwise it is 0. When merging the proximity data into the slots, 1 is selected as a the value is used. The acceleration data is included with the

three axes X, Y and Z - each as their own feature. The mean of the sensors is used when merging the data into the five seconds slot.

After resampling the existing sensor data into five seconds slots, all data from all four data sources is merged using the continuous timestamp from the captured sensor data for alignment. If there is sensor data missing for any of the floor, binary sensors or proximity features, 0 is assigned to this value since no data being present is a high sign that there is nothing of interest happening around those sensors. For the acceleration data, the last valid observation is propagated forward to fill missing data. Taking the mean would have smoothed down spikes in the acceleration data as can be seen in Figure 4.8. This step influences the performance of the classification, but only in a small lowering of the accuracy percentage. In the plot of the acceleration data it is visible that the mean smoothes the data but keeps the trend which could be an interesting aspect when looking at time series data and sequences of activities. An additional step could be to add a rolling mean (aggregating multiple data points to one mean value) of a certain amount of seconds as another feature as it is visualised in the plot.

Since the activities related to the different meals of the day depend strongly on the segment during which the data was captured, the segment is added as its own feature. This way the model should be able to differentiate between breakfast, lunch and dinner. The activity label, which is the class for the prediction algorithms, is added according to the timestamp. When there is sensor data present, but no activity assigned to it, the sample is labelled with “no activity” which can be seen as its own, additional class added to the 24 activities of daily living presented in in Table 1.1. Since the samples of this “no activity” class mostly occur in the training sample and not that much in the testing sample due to different ways of labelling the data, this class is eliminated.

The features from the binary sensors, proximity and from the intelligent floor are binary making it irrelevant to perform standardisation or normalisation. The only ordinal data in the data set is the acceleration data and the segment of the day. This data sources are normalised using the MinMaxScaler from the scikit-learn preprocessing class which scales the values in a range between 0 and 1. Using no normalisation does not change the outcome

Figure 4.8.: Acceleration data from a wrist watch. All 3 axes (X, Y and Z) are shown with the mean and a rolling mean of 30 seconds.

of the algorithms significantly. The idea of normalisation is to bring the features into a similar range which can lead to faster convergence.

The preprocessing steps resulted in 5.344 samples for training and 2.920 samples for the testing.

4.4. Feature Selection

After merging the sensor data into one second slots with inputs from all four data sources, it becomes apparent that the original data set includes a great amount of features and an extremely sparse input matrix. Therefore feature selection is performed. A Pearson correlation is calculated to check if certain features in the training set have a linear relation. When looking at the result of the Pearson correlation, it is noticeable that the binary sensors "kitchen faucet (C015)", "cutlery (C03)" and "laundry basket (C012)" are not present in the training set and neither in the testing set. The sensor "Remote XBOX (C07)" is only present in the testing set, yielding 1 only once. Due to the fact that the model will not be able to learn anything from those sensors, the four features are removed. When analysing the Pearson correlation even further, it is visible that the adjacent floor tiles have a minor association with each other. The sofa pressure sensor (binary sensor) in the living room and the proximity sensor of the kettle in the kitchen have a very high association of 92.6%. This is interesting since they are in completely different spots in the apartment. Some features from sensor data collecting devices that are placed close to each other in the smart lab show an association of medium strength, for example the proximity sensors from the bed and the pyjamas drawer.

Since there are 40 modules for the intelligent floor which is quite a large number of features, it was tried to collapse those features into rooms (entrance, bedroom, kitchen, living room) according to the drawings of the tile arrangement. This feature reduction results in a lower accuracy and is therefore not used in the final run of the machine learning algorithm.

The feature selection process results in the following 83 features being used for the algorithms described:

- Binary senors: only 26 sensors where selected (binary)
- Intelligent floor: 40 modules (transformed to binary input data) (binary)
- Acceleration from a wrist watch: X, Z and Y axis (scaled between 0 and 1)
- Proximity data from 15 different objects (scaled between 0 and 1, 1 meaning the highest proximity)
- The segment of the day: morning (0), afternoon (1), evening (2): This feature is particularly important to distinguish between breakfast, lunch and dinner due to the time of the day.

The samples are related to each other. To represent the sequential dependence of the samples, lagged features from the sample before are used. This procedure adds the features of the samples of eighteen seconds before the current data point resulting in 1.577 features per sample, not including the class labels for those previous samples. Including the class label ends in an accuracy of over 90% for a majority of the algorithms, but it can be speculated that the model only learned to guess the activity class that occurred before, not being able to recognise when an actual new activity starts.

To reduce features and improve the accuracy, principal component analysis is performed on the data. Nevertheless, the feature reduction only worsens the accuracy and is therefore not chosen as a preprocessing step for the machine learning pipeline of this project.

5. Chosen Classification Algorithms

The algorithms used to classify the activities are presented. See Chapter 6 for their comparison. The chosen programming language is Python[1] including the powerful tools Pandas[2] and scikit-learn[3] as well as matplotlib[4] for the plots.The deep learning models were created using Keras[5].

The problem presented in this academic work is a classification of activities based on multivariate time series data. Time series data is continuous data that measures changes in Internet of Things solutions, for example a smart environment like the smart lab mentioned in this thesis. The same features are tracked over time and stored away together with their timestamp for further analysis. Multivariate means that there are more than a single feature present in the samples to predict the class.

In the domain of activity recognition, it proves difficult to classify a five seconds slot as part of an activity of a human. The classification of each sample in the activity recognition is related to the samples that occurred before the currently examined sample since the samples are divided into slots of five seconds. Thus, a model that can learn sequences to include in the rules for the classification has to be found.

[1] *Python Programming Language* 2019.
[2] *Pandas - Python Data Analysis Library* 2019.
[3] *scikit-learn - Machine Learning in Python* 2019.
[4] *Matplotlib - Plotting in Python* 2019.
[5] *Keras - Library for neural networks* 2019.

5.1. Standard Machine Learning Algorithms

A list of ten basic classification algorithms from the scikit-learn library in Python[6] is chosen to test their predictive accuracy on the human activity recognition data set:

- Logistic Regression
- Naive Bayes
- Decision Tree
- Support Vector Machine
- K-Nearest Neighbors
- Random Forest
- Bagging
- Extra Tree
- Gradient Boosting
- Neural Network

The algorithms are executed with the parameters as shown in Listing A.1 in the appendix. The hyperparameters of the best performing algorithms according to the result metrics were optimised using grid search yielding the parameters shown in the code sample.

5.2. Deep Learning Algorithms

Recent development in artificial intelligence explore deep learning and show how neural networks can prove as more powerful solutions than basic algorithms. A deep learning algorithm that is becoming popular in time series prediction and classification is long short-term memory (LSTM) which is an artificial recurrent neural network. This algorithm is used in addition to the standard machine learning models listed above. An LSTM is able to learn the sequence dependence of features. The hyperparameters are chosen as shown in Listing A.2 in the appendix after tweaking them with various trial and error runs.

[6] *scikit-learn - Machine Learning in Python* 2019.

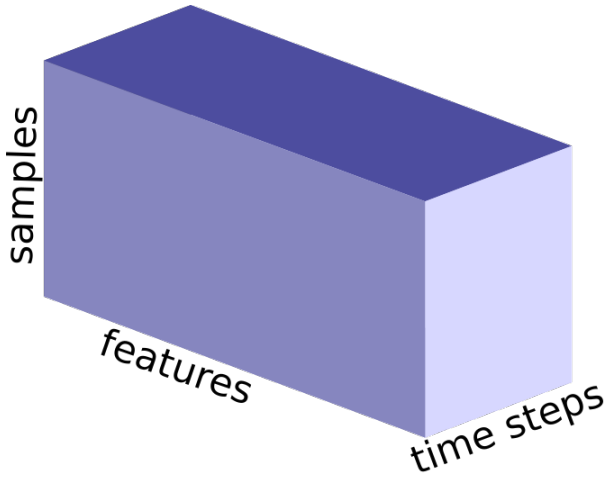


Figure 5.1.: The input for the CNN and LSTM network is a 3D tensor with the shape samples x time steps x features per step

Another deep neural network is convolutional neural network (CNN) which is primarily used in image classification. Using a hyperparameter optimisation for Keras called Talos[7], the hyperparameters for the CNN are found (see in Listing A.2 in the appendix). As with the LSTM, dropout is used. L1 regularisation is tested as well, but results in a worse accuracy.

The network architecture is shown in Figure Figure 5.2. The ntwork consists of the following layer setup: convolution (64 filters) → max pooling → flatten → fully connected layer → dropout → softmax output layer. The input for the CNN and LSTM network is a tensor with the shape samples x time steps x features per step (see Figure 5.1). Using 40 lagged time steps results in a better accuracy instead of 18 which were used for the standard algorithms. The output is a one-hot encoded matrix with the classes as columns. Using a one-hot encoded matrix is beneficial in case a new type of activity is added to the model. Then the model does not have to be re-trained. If a vector with the size of the classes would be used, the model would have to be trained again which takes time.

To combat overfitting of the deep learning models, early stopping, regularisation and dropout were used. The learning rate was set to a small value because it

[7] *Talos - Hyperparameter Optimization for Keras Models* 2019.

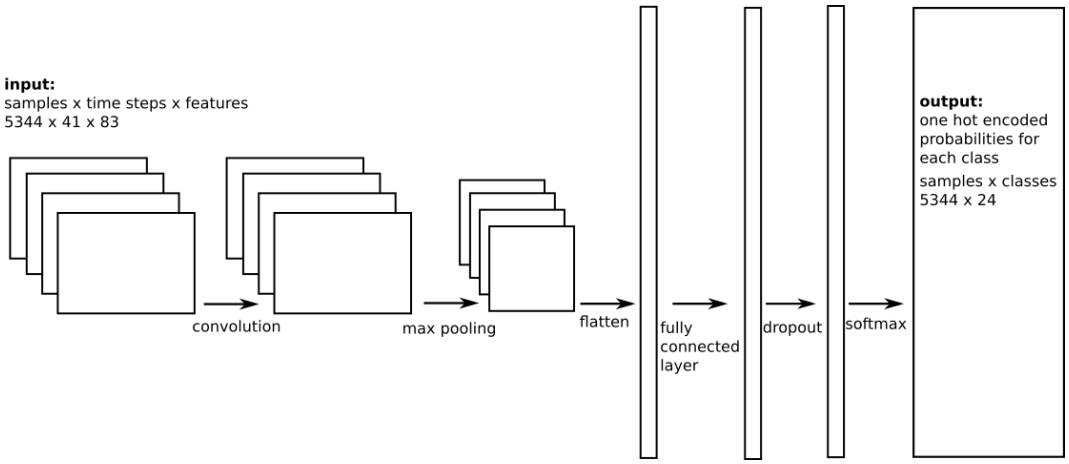


Figure 5.2.: The network architecture of the CNN consists of: convolution → max pooling → flatten → fully connected layer → dropout → softmax

showed a higher accuracy. Batch normalisation speeds up the training, but when it was used with the deep learning networks in this thesis, the model did indeed learn faster but the accuracy dropped significantly. Therefore batch normalisation was not used. As an optimizer, Nadam (Nesterov Adam optimizer) was chosen.

6. Evaluation

In this chapter, the results of the machine learning algorithms used for the classification of different activities of daily living are compared. The chosen metrics were accuracy, recall (weighted), precision (weighted), F1 score (weighted), Matthews correlation coefficient (MCC) and root mean square error (RMSE), the standard deviation of the prediction errors. Recall, precision and F1 score were weighted to deal with the label imbalance of the data set. The formulas for accuracy, precision, recall and F1 score are as follows:

$$Accuracy = \frac{TruePositives}{ActualResults}$$

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

6.1. Standard Machine Learning Algorithms

The metrics of the results of the standard machine learning algorithms are displayed in Figure 6.1 and in Table 6.1. The RMSE is compared in Figure 6.2 . From analysing the metrics, it becomes apparent that the precision metric has a higher spike in the bar chart than the rest of the metric. With the formula in mind, the precision measurement illustrates that the models did not classify a high amount of false positives in comparison to the true positives. Most of the positive values were simply never predicted. For the support vector machine and random forest algorithms, the recall was higher than precision. The algorithm that resulted in the overall best performance was logistic regression: Accuracy: 63.53% , recall: 64.86%, precision: 70.991%, F1 score: 63.39%, MCC: 61.91%, RMSE: 0.18. This means that using the model obtained from logistic regression, 63.53% of the samples can be classified correctly. The worst result came from the naive bayes algorithm. Naive Bayes works with the assumption that all input features are conditionally independent. Since some features have a strong strength of association according to the Pearson correlation, this might be the reason that the algorithm performs poorly. Logistic regression for example still works well when features are correlated. The training and testing accuracy percentage were nearly the same for each individual model.

The RMSE was the lowest for logistic regression. The highest RMSE came from the k-nearest neighbours algorithm which also showed weak performance. The RMSE of naive bayes was actually not that high compared to the other algorithms even though its accuracy was the lowest.

The confusion matrix for the logistic regression algorithm is available in Figure 6.3. Most of the activities had a high chance of being predicted correctly. The confusion matrix illustrates that the biggest problem for the model was to spot the activity “relax on the sofa”. This activity was often misclassified as “lunch”, “dinner” and “watch TV”. This is probably because the correlation of the kettle sensor and the the sofa pressure sensor that was seen in the Pearson correlation already. Those sensors were mainly involved in the four activities. The confusion matrices for the other models show as well that the models had difficulties classifying the activity “relax on the sofa” correctly. Interestingly, “play a video game” was always wrongly

Algorithm	Accuracy	Recall	Precision	F1 score	MCC	RMSE
Logistic Regression	63.53%	64.86%	70.99%	63.39%	61.91%	4.28
Naive Bayes	34.45%	34.52%	63.41%	33.97%	33.06%	6.00
Decision Tree	40.92%	40.92%	49.24%	38.38%	38.62%	6.76
Support Vector Machine	54.18%	60.02%	74.78%	50.23%	53.62%	5.71
K-Nearest Neighbors	38.25%	47.33%	50.93%	38.45%	32.88%	7.34
Random Forest	49.11%	67.20%	53.34%	56.41%	47.58%	6.16
Bagging	49.62%	50.66%	50.96%	44.94%	47.99%	5.98
Extra Tree	53.53%	54.65%	60.91%	49.24%	52.21%	5.12
Gradient Boosting	55.41%	56.57%	56.03%	50.96%	54.45%	4.99
Neural Network	55.31%	56.47%	59.37%	50.85%	54.36%	4.95
CNN	69.04%	70.8%	73.1 %	68.9%	67.2%	3.8
LSTM	46.52%	52.6%	50.7%	47.5%	43.2%	5.67

Table 6.1.: Metrics of different machine learning algorithms

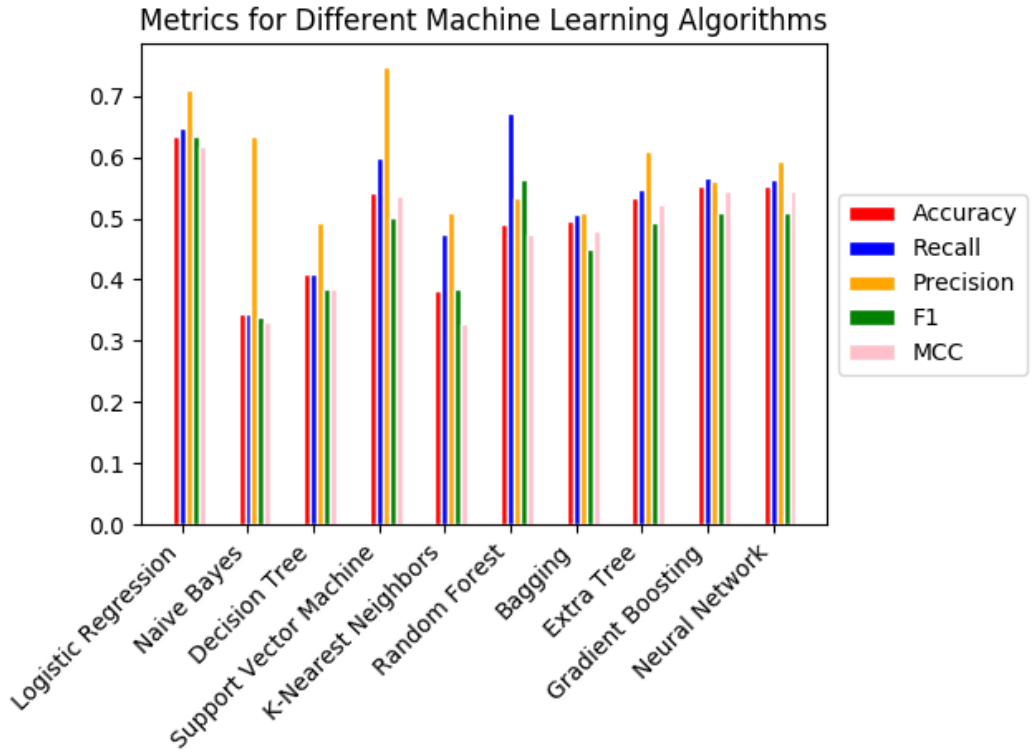


Figure 6.1.: Bar chart of metrics of different machine learning algorithms performed on the data set.

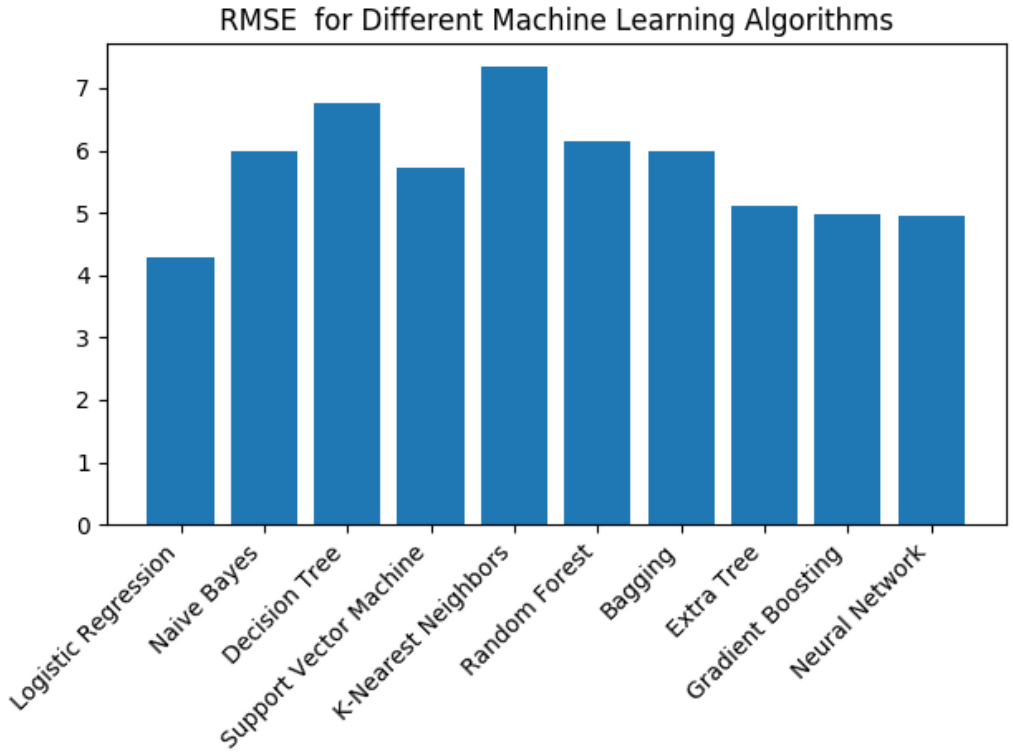


Figure 6.2.: Bar chart of RMSE of different machine learning algorithms performed on the data set.

classified as "watch TV". There was only one occurrence of this activity in the test and in the training set which might be the reason why the model cannot learn the patterns of this activity. "Use the toilet" was recognised as the activity "dinner" for all samples. The activity "wash dishes" was mostly confused with "put waste in the bin" and "take medication".

The logistic regression classification algorithm calculates the probability of each activity and then takes the label which results in the maximum of those probabilities. In Table 6.2 shows the probabilities of one sample. The highest value result was returned for activity "wash hands", but it was actually activity "brush teeth".

6.2. LSTM network

Surprisingly, the results from the LSTM model were worse than the ones from the standard algorithms described above with an average accuracy of 46.52% (+/-8.27) over 10 runs. The averaged RMSE is 5.67. The cause for the weak performance might be that the training data set is fairly small for a deep learning problem (5.344 samples for training and 2.920 samples for the testing). The samples were fed into the algorithm using 40 time steps, for the standard algorithms 18 were optimal. Figure 6.4 displays the confusion matrix of the LSTM model. The training time was notably longer than other algorithms proposed in this work. The shortest run out of the 10 experiments for the LSTM was 10 minutes, the shortest for the CNN was 50 seconds.

Figure 6.5 displays the validation and training loss of the LSTM network as well as the training accuracy. The accuracy does improve over the epochs, but the validation loss doesn't seem to become significantly smaller.

6.3. CNN

The performance of the CNN was similar to the best results from the standard algorithm with an average of 69.04% (+/-2.63) of accuracy in

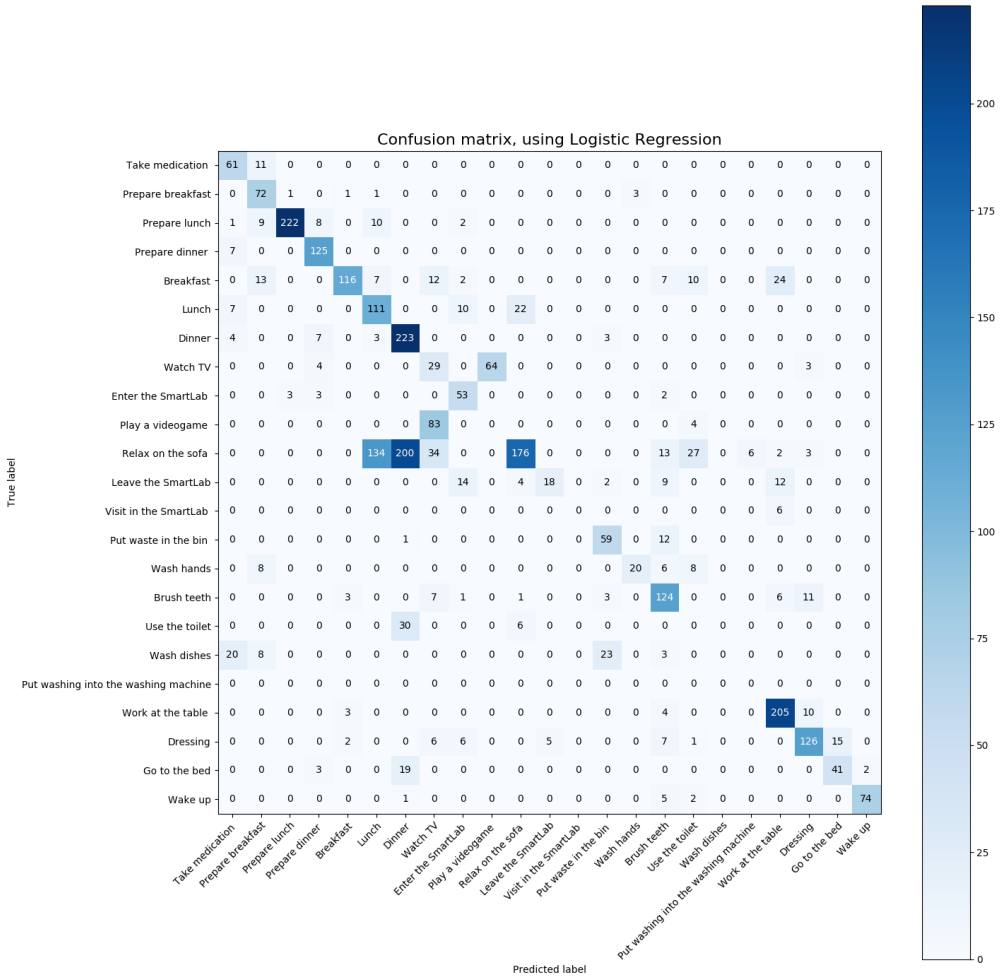


Figure 6.3.: Confusion matrix for the classification results of Logistic Regression performed on the data set using scikit-learn.

Activity	Probability
Take medication	0.00
Prepare breakfast	0.00
Prepare lunch	0.00
Prepare dinner	0.00
Breakfast	0.00
Lunch	0.00
Dinner	0.00
Eat a snack	0.00
Watch TV	0.00
Enter the SmartLab	0.00
Play a video game	0.00
Relax on the sofa	0.00
Leave the SmartLab	0.00
Visit in the SmartLab	0.00
Put waste in the bin	0.00
Wash hands	0.03
Brush teeth	0.93
Use the toilet	0.03
Wash dishes	0.00
Put washing into the washing machine	0.00
Work at the table	0.00
Dressing	0.00
Go to bed	0.00
Wake Up	0.00

Table 6.2.: Class probabilities for one sample using logistic regression

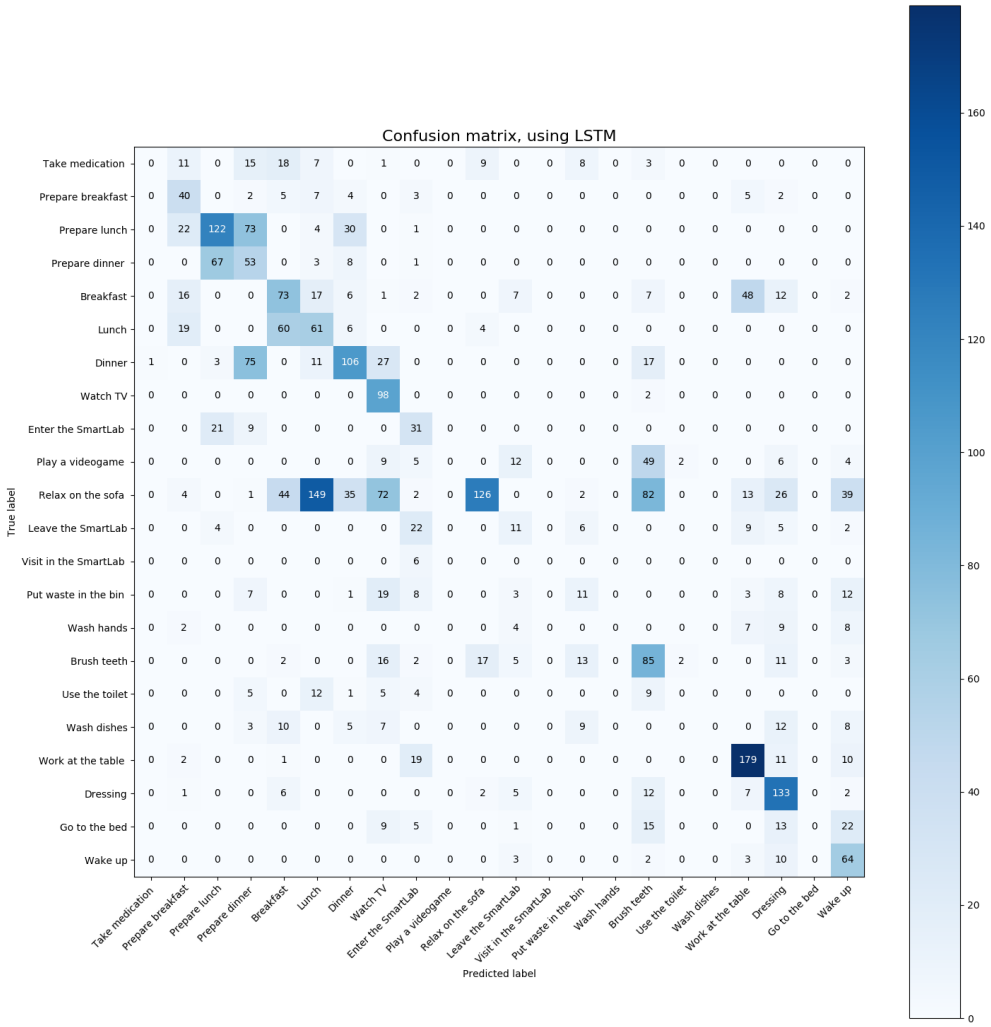
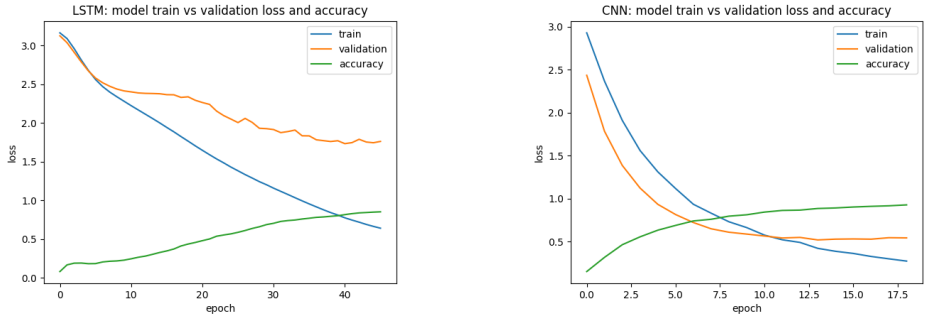


Figure 6.4.: Confusion matrix for the classification results of the deep learning model LSTM performed on the data set using Keras.



(a) Training loss, validation loss and training accuracy curves of LSTM model

(b) Training loss, validation loss and training accuracy curves of CNN model

Figure 6.5.: Different metrics of the deep learning models LSTM and CNN over the epochs.

10 runs. The averaged RMSE is 3.8 which is the lowest in all of the used algorithms. Notably, the training and validation accuracy were much higher than the testing accuracy (around 95% and 80%) . This can be a sign of overfitting of the model. Figure 6.6 displays the confusion matrix of the CNN model. The plot shows that the model had the same problem as the standard algorithms - it could hardly distinguish between "dinner" and "relax on the sofa". Same as in the confusion matrix of the logistic regression, "watch TV" was often misclassified as "play a video game".

Figure 6.5 illustrates how training and validation loss behave over the epochs of the CNN model. The training loss for the first couple of epochs is higher than the validation loss. This means that the model is better at predicting unseen data than on the training data which could be caused by a validation set with very different data. The training accuracy shows a well-formed curve close to 100%.

The deep learning algorithms were run 10 times to get an averaged result since the outcome can be different each run. The runs are compared in Figure 6.7. These graphs were created usign the Tensorboard visualisation[1]. There are two clusters visible - one is the CNN and one is from the LSTM. For the CNN, the different functions for each run are similar and nearly

[1] *TensorBoard - Visualisation for Tensorflow and Keras* 2019.

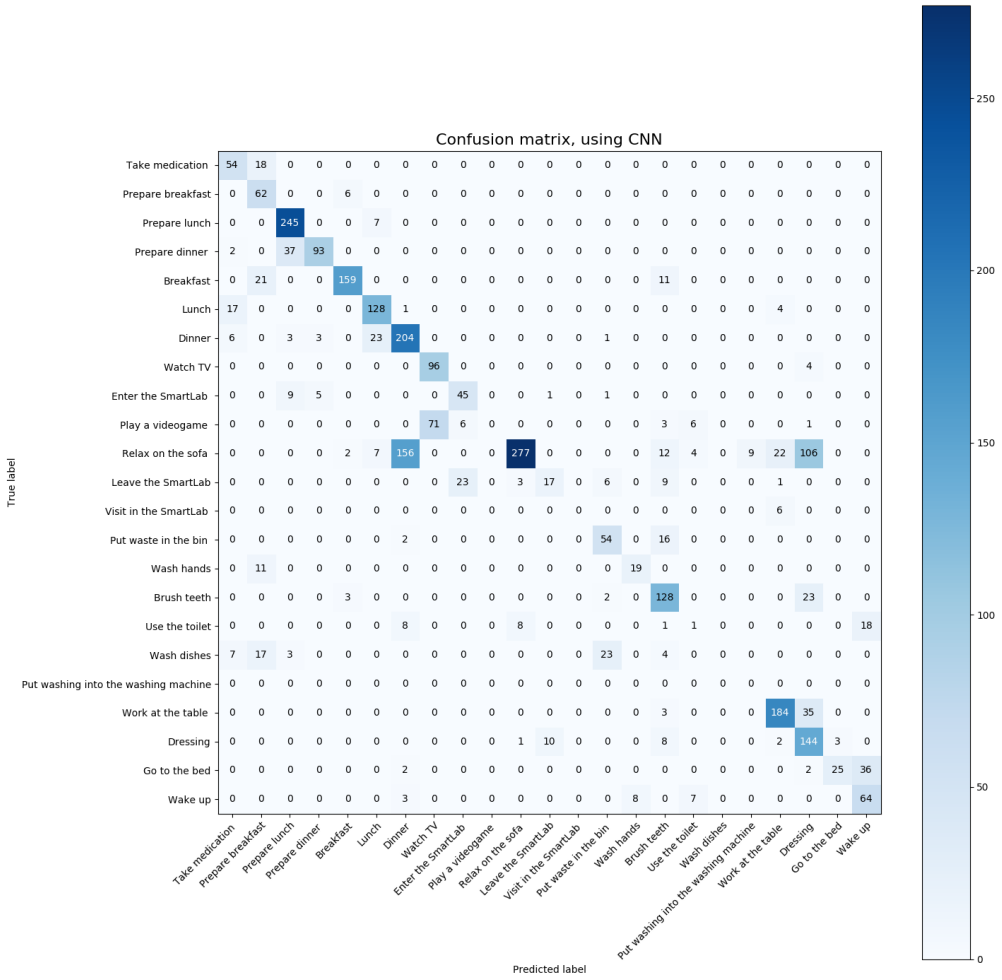
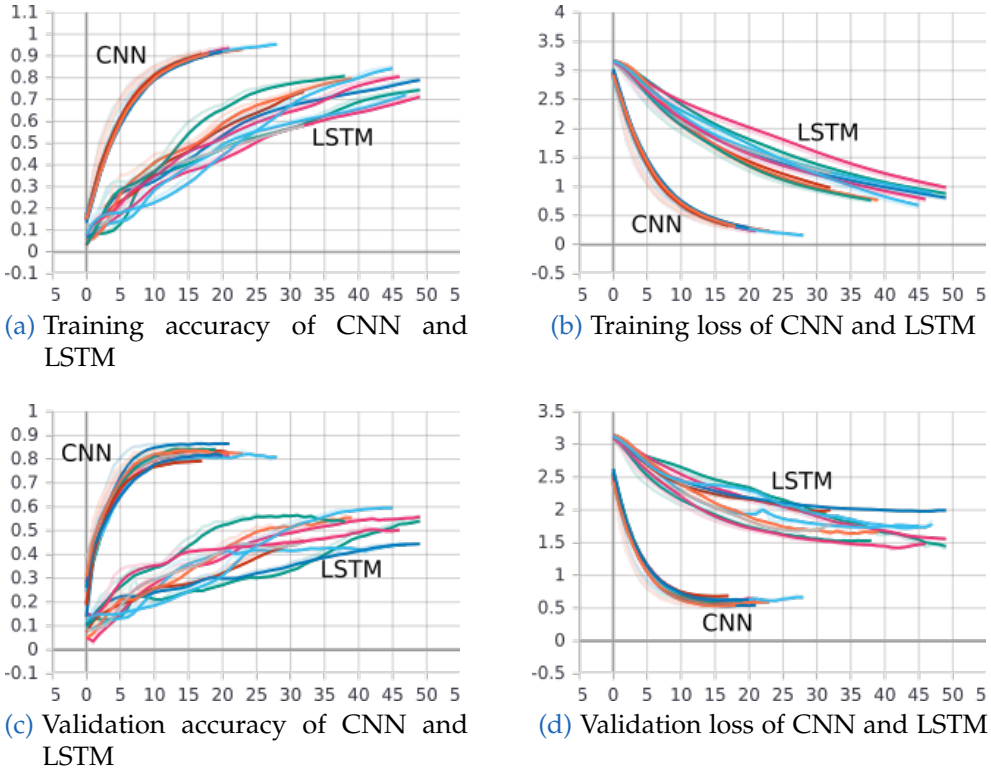


Figure 6.6.: Confusion matrix for the classification results of the deep learning model CNN performed on the data set using Keras.



(a) Training accuracy of CNN and LSTM

(b) Training loss of CNN and LSTM

(c) Validation accuracy of CNN and LSTM

(d) Validation loss of CNN and LSTM

Figure 6.7.: Different metrics of the 10 runs of the experiments using LSTM and CNN over the epochs.

logarithmic. The LSTM runs show a higher variance in their performance. The LSTM takes to take more time than the CNN and the function has a more linear growth. From this analysis, it is clear that the CNN converges faster than the LSTM since early stopping is used. Early stopping forces the deep learning algorithm to stop learning before the model overfits.

7. Conclusions

Ten different standard machine learning algorithms and two deep learning models (LSTM, CNN) were tested to classify activities of daily living using time series data. The data set contained sensor data from four different sources: intelligent floor, proximity, binary sensors and acceleration data from a smart watch. The sensor data was aligned by its timestamp and resampled into five seconds slots. Feature selection was performed to reduce the number of features. The parameters for the standard algorithms were found with grid search. For the LSTM, several different options for layers and hidden neurons were explored. The CNN hyperparameters were optimised using hyperparameter optimisation.

The best performing standard algorithm in terms of accuracy was logistic regression with 63.53%. Compared to Ceron, López, and Eskofier, 2018 who worked on a similar solution ending up in 60.1% accuracy, the results of the static machine learning methods presented were better. It must be noted that Ceron, López, and Eskofier, 2018 merged the meal activities into a single activity. The LSTM model resulted in 46.52% (+/-8.27), the CNN in 69.04% (+/-2.63) on average. Surprisingly, the deep learning methods were not able to classify more accurately than the standard algorithms used in this project.

The results can either be a sign that there is only a weak pattern to be learned from the preprocessed data or that the machine learning algorithms used are not a good choice. Human activities of daily living can barely be judged from such a short period of time since there features identifying this feature might not all be present in that single sample, but might have been included in a previous sample or are part of future samples. Static machine learning algorithms like logistic regression have difficulties with detecting patterns in sequences of sensor data.

A use case scenario for this model could be detecting how often the habitant of the smart home has performed the activity “put waste in the bin” and subsequently - depending on a threshold number - calling the garbage collection to retrieve the trash bags. It is fair to say that the LSTM could not classify this activity correctly. The logistic regression and CNN models both showed more reliable results in detecting this activity, but confused it sometimes with “brush teeth”.

Further improvements on the models could be done by performing data augmentation on the data set to get more data samples or simply more samples could be recorded. The class imbalance could also be fixed. For the LSTM in particular, the hyperparameters could be explored further. Another idea would be to use nine days of training and only one day for testing the accuracy of this algorithm. A better approach in terms of which algorithm to use could be a hybrid model using CNN layers paired with LSTM layers.

Appendix

Appendix A.

Code Snippets

Listing A.1: The machine learning algorithms from the scikit-learn library in Python are executed using mostly default parameters.

```
regressor = LogisticRegression(random_state=0, solver='lbfgs', multi_class='
    multinomial', max_iter=3000) # logistic regression
gnb = GaussianNB() # gaussian naive bayes
dt = tree.DecisionTreeClassifier() # decision tree
svc = SVC(C=1.0, decision_function_shape='ovr', degree=3, gamma='
    auto_deprecated', kernel='rbf') # support vector machine
knn = KNeighborsClassifier(n_neighbors=5) # k-nearest neighbors
rf = RandomForestClassifier(n_estimators=1000, max_depth=10, random_state=0) #
     random forest
b = BaggingClassifier() # bagging
et = ExtraTreesClassifier()  # extra tree
gb = GradientBoostingClassifier() # gradient boosting
nn = MLPClassifier(activation='relu', early_stopping=True, hidden_layer_sizes
    =(5,5), max_iter=500, shuffle=False, solver='sgd', validation_fraction
    =0.2, batch_size=5, learning_rate='adaptive', learning_rate_init=0.0001)
      # using neural network
```

Listing A.2: The hyperparameters for the long short-term memory recurrent neural network using the Keras library are executed with the parameters shown in this code sample.

```
n_outputs = 24  # number of classes

model = Sequential()
model.add(LSTM(8, input_shape=(train_X.shape[1], train_X.shape[2]),
    kernel_regularizer=regularizers.l2(0.0001)))
model.add(Dense(8, activation='relu', kernel_regularizer=regularizers.l2
    (0.0001)))
model.add(Dense(n_outputs, activation='softmax'))
optimizer = optimizers.Nadam(lr=0.0001, beta_1=0.9, beta_2=0.999, epsilon=None
    , schedule_decay=0.004)
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['
    acc'])
# early stopping
```

```
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=5)
# fit network
history = model.fit(train_X, train_y, epochs=50, batch_size=4, verbose=2,
    shuffle=False, validation_split=0.2, callbacks=[es])
```

Listing A.3: The hyperparameters for the convolutional neural network using the Keras library are executed with the parameters shown in this code sample.

```
n_outputs = 24  # number of classes
epochs = 50

model = Sequential()
model.add(Conv1D(filters=64, kernel_size=2, activation='relu', input_shape=(
    train_X.shape[1], train_X.shape[2])))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(n_outputs, activation='softmax'))
optimizer = optimizers.Nadam(lr=0.0001, beta_1=0.9, beta_2=0.999, epsilon=None
    , schedule_decay=0.004)
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['
    acc'])
# early stopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=5)
# fit network
history = model.fit(train_X, train_y, epochs=epochs, batch_size=5, verbose=2,
    shuffle=False, validation_split=0.25, callbacks=[es])
```

Bibliography

Ann, Ong Chin and Bee Theng Lau (Nov. 2014). "Human activity recognition: A review." In: *2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014)*. Penang, Malaysia: IEEE, pp. 389–393. DOI: 10.1109/ICCSCE.2014.7072750 (cit. on p. 1).

Ceron, Jesus, Diego López, and Bjoern Eskofier (Dec. 2018). "Human Activity Recognition Using Binary Sensors, BLE Beacons, an Intelligent Floor and Acceleration Data: A Machine Learning Approach." In: *The 12th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI 2018)*. Vol. 2. 19 1265. Punta Cana, Dominican Republic: MDPI. DOI: 10.3390/proceedings2191265 (cit. on pp. 9, 10, 47).

Espinilla, Macarena, Javier Medina, and Chris Nugent (Dec. 2018). "UCAmI Cup. Analyzing the UJA Human Activity Recognition Dataset of Activities of Daily Living." In: *The 12th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI 2018)*. Vol. 2. 19 1267. Punta Cana, Dominican Republic: MDPI. DOI: 10.3390/proceedings2191267 (cit. on pp. 2, 18, 19).

Google Scholar (2019). URL: scholar.google.co.uk/ (visited on 04/09/2019) (cit. on p. 8).

Hammerla, Nils Y. and Shane Halloran Thomas Plötz (2016). "Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables." In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pp. 1533–1540. URL: ijcai.org/Proceedings/16/Papers/220.pdf (cit. on p. 11).

Jiménez, Antonio R. and Fernando Seco (Dec. 2018). "Multi-Event Naive Bayes Classifier for Activity Recognition in the UCAmI Cup." In: *The 12th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI 2018)*. Vol. 2. 19 1264. Punta Cana, Dominican Republic: MDPI. DOI: 10.3390/proceedings2191264 (cit. on p. 10).

Karvonen, Niklas and Denis Kleyko (Dec. 2018). "A Domain Knowledge-Based Solution for Human Activity Recognition: The UJA Dataset Analysis." In: *The 12th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI 2018)*. Vol. 2. 19 1261. Punta Cana, Dominican Republic: MDPI. DOI: 10.3390/proceedings2191261 (cit. on p. 8).

Keras - Library for neural networks (2019). URL: keras.io (visited on 06/05/2019) (cit. on pp. 7, 31).

Lago, Paula and Sozu Inoue (Dec. 2018). "A Hybrid Model Using Hidden Markov Chain and Logic Model for Daily Living Activity Recognition." In: *The 12th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI 2018)*. Vol. 2. 19 1266. Punta Cana, Dominican Republic: MDPI. DOI: 10.3390/proceedings2191266 (cit. on p. 9).

Matplotlib - Plotting in Python (2019). URL: matplotlib.org (visited on 05/23/2019) (cit. on pp. 7, 17, 31).

Nazerfard, Ehsan et al. (n.d.). "Conditional Random Fields for Activity Recognition in Smart Environments." In: DOI: 10.1016/j.patrec.2018.02.010 (cit. on p. 11).

Pandas - Python Data Analysis Library (2019). URL: pandas.pydata.org (visited on 05/23/2019) (cit. on pp. 7, 17, 31).

Python Programming Language (2019). URL: python.org (visited on 05/23/2019) (cit. on pp. 7, 17, 31).

Razzaq, Muhammad Asif et al. (Dec. 2018). "Multimodal Sensor Data Fusion for Activity Recognition Using Filtered Classifier." In: *The 12th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI 2018)*. Vol. 2. 19 1262. Punta Cana, Dominican Republic: MDPI. DOI: 10.3390/proceedings2191262 (cit. on p. 10).

Razzaq, Muhammad Asif et al. (n.d.). "Multimodal Sensor Data Fusion for Activity Recognition Using Filtered Classifier." In: DOI: 10.3390/proceedings2191262 (cit. on pp. 10, 11).

Salomón, Sergio and Cristina Tîrnăucă (Dec. 2018). "Human Activity Recognition through Weighted Finite Automata." In: *The 12th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI 2018)*. Vol. 2. 19 1263. Punta Cana, Dominican Republic: MDPI. DOI: 10.3390/proceedings2191263 (cit. on p. 9).

scikit-learn - Machine Learning in Python (2019). URL: scikit-learn.org (visited on 05/23/2019) (cit. on pp. 7, 31, 32).

Semantic Scholar (2019). URL: semanticscholar.org (visited on 04/09/2019) (cit. on p. 8).

Sensors - MDPI Open Access Journal (2019). URL: mdpi.com/journal/sensors (visited on 04/09/2019) (cit. on p. 8).

Talos - Hyperparameter Optimization for Keras Models (2019). URL: github.com/autonomio/talos (visited on 06/14/2019) (cit. on pp. 7, 33).

TensorBoard - Visualisation for Tensorflow and Keras (2019). URL: www.tensorflow.org/guide/summaries_and_tensorboard (visited on 06/27/2019) (cit. on p. 44).

The 12th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI 2018) (Dec. 2018). Vol. 2. 19. Punta Cana, Dominican Republic: MDPI.

Ulster Institutional Repository, Ulster University (2019). URL: uir.ulster.ac.uk/ (visited on 04/09/2019) (cit. on p. 8).

Weka 3: Data Mining Software in Java (2019). URL: cs.waikato.ac.nz/ml/weka/ (visited on 04/09/2019) (cit. on p. 10).

Wirth, Rüdiger and Jochen Hipp (2000). "CRISP-DM: Towards a Standard Process Model for DataMining." In: *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, pp. 29–39. URL: pdfs.semanticscholar.org/48b9/293cfd4297f855867ca278f7069abc6a9pdf (cit. on p. 9).