Thomas Wold
Sigve André Evensen Skaugvoll

Ensemble Classifier Managing Uncertainty in Accelerometer Data within Human Activity Recognition Systems

Master's thesis in Master of Science in Informatics
Supervisor: Kerstin Bach
June 2019

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Thomas Wold
Sigve André Evensen Skaugvoll

Ensemble Classifier Managing Uncertainty in Accelerometer Data within Human Activity Recognition Systems

Master's thesis in Master of Science in Informatics
Supervisor: Kerstin Bach
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

The authors want to give a special thanks to Kerstin Bach for supervising this thesis and Atle Kongsvold for helping with generating new data for sensor no-wear time detection.

Abstract

Human activity recognition (HAR) is a field of study that aims to recognize activities from data acquired by video or wearable sensors. The biggest health study in Norway, HUNT, has recently ended it's fourth study where 38 756 participants have recorded activity data while wearing three-axis accelerometer on their thigh and back. HAR systems often require all sensors to be operative and attached to the participant at all times, and shows weaknesses when performing activity recognition, as a lot of misclassifications occur due to sensors lying still after being detached from the subject's body during activity recording. To make HAR systems more robust against this issue, this thesis researches on a new type of ensemble classifier where a meta classifier predicts sensor no-wear time, eliminates faulty sensor streams and dynamically adjust the LSTM-RNN sensor position specific classification models used, depending on the data available. The developed meta classifier is trained on a new "Sensor No-Wear Time" dataset that consists of real-world data, and is able to predict sensor no-wear time with 97.2% accuracy and shows promising results towards making more valid contributions towards public health research, as it eliminates up to several days of misclassifications where sensors have been detached. Research done in this thesis shows that individual models for thigh and back are struggling to classify certain static activities. A model for both sensors combined is therefore the best option for activity classification as it achieves an accuracy of 85.1% compared to the existing HAR system's 76.5%, and outperforms individual models when classifying static activities. Storing classification results for all participants in HUNT requires huge amounts of storage space, and $Feather$ is proving to be the file format that is best suited for storing activity classification results, as the result file size for each participant is reduced from 2.5 GB to 941 KB with a new compression algorithm. This results in a total reduction of 99.96%, as necessary storage space is reduced from 96.89 TB to 0.036469396 TB for all HUNT4 participants.

Sammendrag

Human activity recognition (HAR) er et forskningsområde med mål om å klassifisere aktiviteter utført av personer ved hjelp av data hentet inn av video eller sensorer festet på kroppen. HUNT er den største helseundersøkelsen i Norge, og har nylig avsluttet den fjerde undersøkelsen, hvor totalt 38 756 personer har deltatt og hatt to sensorer festet på kroppen som har registrert aktivitetsdata i tre akser. HAR systemer krever ofte at alle sensorer er operative og festet til kroppen til en hver tid. På dette grunnlaget viser systemene svakheter ved at det oppstår mye feilklassifiseringer på grunn av uforutsette hendelser, som at sensoren går tom for strøm, montert i feil retning, posisjon eller tatt av under aktivitest registreringen. For å gjøre HAR systemer mer robuste mot dette problemet, forsker denne masteren på bruken av en ny type ensemble classifier, der en meta classifier klassifiserer hvilke sensorer som registrerer gyldig data og eliminerer ugyldig data, før den dynamisk endrer hvilken LSTM-RNN aktivitets klassifiserings modell den gyldige dataen blir sendt til. Den utviklede meta classifieren er trent på ett nytt datasett, "Sensor no-wear time", som består av data som ikke er samlet inn under kontrollerte omgivelser, og oppnår en nøyaktighet på 97.2%. Den viser lovende resultater ved å komme med gyldige bidrag til forskning innen folkehelse siden flere dager med feilklassifisering kan bli unngått. Forskningen gjort i denne masteren viser at individuelle klassifiserings modeller for lår og rygg sliter med å klassifisere statiske aktiviteter som sitting og ligging. Modellen for begge sensorene kombinert oppnår en nøyaktighet på 85.1% sammenlignet opp mot det eksisterende systemet som oppnår 76.5%, er derfor det beste alternativet for aktivitets klassifisering, ettersom den klarer å differensiere mellom de statiske aktivitetene. Lagring av aktivitets klassifieserings resultetene med filformatet $Feather$ gir de minste filstørrelsene, og gir en reduksjon fra 2.5 GB til 941 KB per deltager med en enkel komprimerings algoritme. Dette resulterer i en total reduksjon på 99.96%, da nødvendig lagringsplass går fra 96.89 TB til 0.036469396 TB for alle deltagere i HUNT4.

Preface

The research project has been conducted under the supervision of Associate Professor Kerstin Bach, Department of Computer and Information Science at the Norwegian University of Science and Technology, during the academic year 2018-2019. Research conducted is part of a master thesis written by two graduate students specializing in Artificial Intelligence and Database and Search. Thus the theme of the thesis and research goals are within the scope of the students specialty, and the research with its results should be considered as equally important as other research conducted for the same theme. Motivation for research human activity recognition and detecting sensor no-wear time is based on the important contribution and impact such technology has on the day to day life. The intended audience are expected to have some knowledge about machine learning and the human activity recognition problem. The thesis is structured as eight parts, each with a dedicated theme. The purpose and content of each part is presented on separate parts pages and each chapter starts with a an introductory section outlining how the the chapter is structured.

Table of Contents

List of Tables

List of Figures

Abbreviations

Human Activity Recognition	=	HAR
Nord-Trøndelag Health Study	=	HUNT
Norges teknisk-naturvitenskapelige universitet	=	NTNU
Sensor no-wear time	=	SNT
Structured Literature Review	=	SLR
Research Question	=	RQ
Inclusion Criteria	=	IC
Exclusion Criteria	=	EC
Quality Criteria	=	QC
Artificial Intelligence	=	AI
Long Short-Term Memory	=	LSTM
Convolutional Neural Network	=	CNN
Artificial Neural Network	=	ANN
Decision Tree	=	DT
Random Forest Classifier	=	RFC
Activity Recognition Chain	=	ARC
Trondheim Free Living Dataset	=	TFL

Chapter 1

Introduction

Human activity recognition (HAR) is a field of study which aims to recognize activities based on data acquired from video or wearable sensors. The amount of research conducted on this topic has increased considerably in the past few years as the technology is improving and data are becoming more accessible. Wearable sensors like accelerometers are widely used in HAR research because they have become more powerful, affordable and accessible than ever before. Compared to video-based activity recognition, these sensors can be easily placed on the human body due to their relatively small size. Making data acquisition notably easier and less time consuming as no equipment setup is needed. Additionally, sensors does not cope with any privacy issues as the participants are not being filmed. HAR applications are used in numerous areas, ranging from medical research, rehabilitation, tracking personal health to surveillance for security.

Within medical research, HAR systems is especially useful because of the ability to record activity data over long time periods from numerous participants. Gathering data through objective measurements is really important towards the work of public health, as it is more accurate than subjective measurements since people does not have extensive knowledge about how much time they spend on different activities. The gathered data is also highly useful for health promotion and disease prevention as it addresses broad issues that can affect the health and well-being of populations.

Located in Nord-Trøndelag county in Norway, the HUNT Research Center (HUNT Research Center, 2019) has coordinated the Nord-Trøndelag Health Study (HUNT) since 1984. HUNT is the largest health study in Norway where data is collected from more than 120,000 people through four different population studies named HUNT1, HUNT2, HUNT3 and HUNT 4. In these four studies there has been gathered valuable biological material and public health data which is used in both national and international research.

The last HUNT study, HUNT4, was coordinated in cooperation with the Faculty of Medicine and Health Sciences at the Norwegian University of Science and Technology

(NTNU) and completed in March 2019. Citizens in Nord-Trøndelag who are older than 13 years old are invited to participate in the study by completing a questionnaire. Each participant that answers the questionnaire is invited to go through a further examination where different physical measurements are documented and biological materials are gathered. Furthermore they are given the opportunity to wear two accelerometers on the body for a period of seven days. The data that is gathered from the accelerometers, located on the participants thigh and lower back, will be the core of this thesis.

State of the art HAR systems have come along way and manages to achieve high accuracies. Reinsve (Reinsve, 2018) managed to achieve above 94% accuracy in his master thesis in 2018 with the use of a random forest classifier. Many state of the art HAR systems shows weaknesses when it comes to robustness, as they often require all sensors to be operative and attached to the participant at all times. This is a weakness because when humans wear the sensors, there is a possibility for one or more sensors to be displaced, stop working, run out of battery-power or is detached. Thus making many HAR systems not robust and agile enough to classify accurately when the required sensors data is faulty or missing. To address this issue, the thesis aims to extend the HAR system that is utilized for analyzing data from the HUNT study (presented in chapter 5.1), referred to as the existing HAR system throughout this thesis. The proposed HAR system will extend this by developing an ensemble classifier consisting of a meta classifier and position specific sensor machine learning models for back-, thigh-, and a combination of both. When sensor data is faulty or missing, the meta classifier will detect which sensors data stream has the malicious features and eliminate them, thereafter iteratively choose between the models that are trained on features from only valid sensor streams.

The concept of the meta-classifier is based on Reinsve's (Reinsve, 2018) master thesis, where he did a feasibility study to identify if a sensor had no-wear time, the study was called sensor no-wear time (SNT) and the classifier he used was Random Forest Classifier, and the study tried to detect if participants are wearing both accelerometers or not based on temperature. As this was only a feasibility study, this thesis aims to create a more comprehensive real-world dataset and a new model utilizing more than just temperature features.

When HUNT4 was completed, 38 756 people participated in the process of recording activity data with sensors attached to their body. It is a big motivational factor for this thesis to contribute work towards public health by developing a reliable HAR system which utilizes and benefits from sensor no-wear time detection and position specific models. As participant might take sensors of while doing data recording, it is important that classification can be performed with data from either both or one sensor. Additionally, this thesis aim to minimize the file size of the activity classification results. Currently the file size of the classification results is approximately 2.5 GB per subject, and it will require huge amount of storage space to store results from all participants

1.1 Goals and Research Questions

- **`Goal 1:`** Explore state of the art research on HAR systems regarding choices of Deep Learning Models and system infrastructure.
  - **`Research Question 1`**: Does state of the art HAR systems use position-specific models and iteratively choose between the best suited trained model?
  - **`Research Question 2`**: What is the state of the art HAR system architecture, with regards to system scalability and storage?
- **`Goal 2:`** Create a new SNT dataset with real world data and improve the sensor no-wear time accuracy from Reinsve's feasibility study by developing a meta classifier that utilizes more features.
  - **`Research Question 1`**: Will adding distance moved and temperature memory as features, improve the accuracy of the meta classifier?
- **`Goal 3:`** Improve the activity classification accuracy and performance of the existing HAR system by developing an ensemble of back, thigh and back and thigh models.
  - **`Research Question 1`**: How will an ensemble of models for activity classification affect the overall accuracy of HAR systems.
  - **`Research Question 2`**: How will models for thigh and back perform individually with regards to accuracy compared to both of them combined.
- **`Goal 4:`** Improve how the framework stores results of the classification
  - **`Research Question 1`**: What kind of storage methods do other HAR systems use to store their results?
  - **`Research Question 2`**: What is the most effective way to store the result with regards to the memory allocation?

1.2 Research Method

To conduct the research, the *research process* presented in chapter 3 in Oates (Oates, 2006) is applied. First *experience and motivation* and a *literature review* is conducted, which generates the previously presented research questions and puts the work presented in a *conceptual framework*. To find answers to the research questions, *design and creation* research strategy are executed. Design and Create research strategy is defined in definition 1. Data that is used for conducting experiments are generated as quantitative data through an *observation data generation method* and later analyzed in a *quantitative data analysis method*.

1.2.1 Conceptual Framework

This section describes the conceptual framework that illustrates the research to be conducted and what the expected findings are. After conducting the SLR (see chapter 2), the key factors that compile the research topic could be identified, and are presented in figure 1.1.

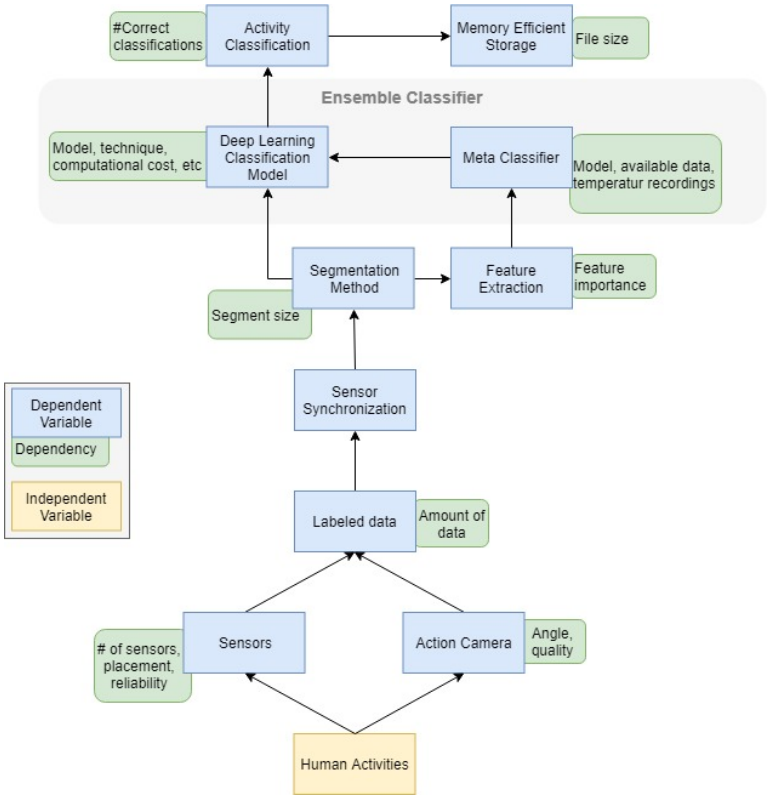


Figure 1.1: Conceptual Framework Variables

1.2.2 Observation Data Generation Method and Data Analysis

Observation data generation is the act of watching and paying attention to what people actually do, rather then what they allegedly state they do. To observe the participants and gather data for the activity recognition part of this thesis, two body-sensors, an action camera and spreadsheets are used. When the data is collected, the sensor data and video footage is synchronized and then then footage is used to label the sensor data with corresponding activities performed. The data analysis is done by machine learning, see section 3.1.

1.2.3 Design and Creation Strategy

The proposed ensemble classifier (see section 5.2), is an extension of the existing HAR system, and focuses on maintaining system reliability even though sensors data is faulty or missing. Additionally this work focuses on minimizing the size of the classification results. Thus, developing new classification models and a memory efficient storage method is the aim of the design and creation strategy.

Definition 1. Design and creation: focuses on developing new IT products, or artifacts. Often the new IT product is a computer-based system, but it can also be some element of the development process such as a new construct, model or method. - Design and creation definition - Briony J Oates, Researching Information Systems and Computing

The proposed system is evaluated on classification accuracy and reduction in memory allocation for classification results through comparison. Comparing sensor no-wear time and activity classification accuracy is done by performing leave-one-out training and comparing the results using quality metrics (defined in section 3.2.3) and comparing results with the existing HAR system and Reinsve's (Reinsve, 2018) feasibility study. For storing the classification results, different file types are tested and compared to find maximized reduction in memory allocation.

1.3 Thesis Structure

Each chapter starts with an introduction describing the content of the chapter.

- **Chapter 2: Structured Literature Review** presents the process for conducting structured literature review and current state of the art HAR systems, with regards to position specific models, system architecture and storing results.
- **Chapter 3: Background Theory** introduces the theory that is necessary to understand the content of this thesis.
- **Chapter 4: Data and Datasets** describes the Trondheim Free Living and Sensor No-Wear Time datasets used for conducting experiments for this thesis.
- **Chapter 5: Pipeline** elaborates the pipeline used in the existing HAR system and the proposed HAR system.
- **Chapter 6: Experiment** presents the experiments conducted with corresponding setup and results.
- **Chapter 7: Evaluation and Discussion** discusses the results from the conducted experiments and compares the proposed pipeline with existing HAR and Reinsve's feasibility study.
- **Chapter 8: Conclusion and Further Work** summarizes the work done throughout the thesis and discusses future work.

Chapter 2

Structured Literature Review

This chapter elaborates on the process for the structured literature review that is conducted for this thesis, before giving a summary of the most relevant research studies found and finally a summary of the chapter.

2.1 SLR Framework

Structured Literature Review (SLR) is a way of synthesizing the information available from relevant primary studies to a set of research questions (RQ). It is carried out using a strict framework and protocol. The advantages of conducting an SLR is that it helps identifying existing solutions, avoid bias in the work, help to avoid duplication of previous work and it makes it possible to identify knowledge gaps and highlight the areas where additional research is required.

SLR consists of three phases; planning, conducting and reporting. Phase one and phase two were further divided into sub-phases.

- Define the research question (phase 1)
- Identification of research (phase 2)
- Selection of primary studies (phase 2)
- Quality assessment (phase 2)
- Summary of related work (phase 3)

2.2 Planning

The planning phase consists of defining research questions, which is defined based on the main goals of this thesis in section 1.1. The purpose of this is to obtain a sufficient overview of the previous work that consists within the field of HAR systems regarding

the use of position-specific models and system architecture. Knowledge obtained from the SLR will give a clear understanding of any possible knowledge gaps and identify potential areas for this thesis to do further research on.

The following research questions were defined:

`Goal 1:` Explore state of the art research on HAR systems regarding choices of Deep Learning Models and system infrastructure.

- **`Research Question 1`**: Does any of the state of the art HAR systems use position- specific models and iterative choose between the best suited trained model?
- **`Research Question 2`**: What is the state of the art HAR system architecture, with regards to system scalability and storage?

2.3 Conducting

The conduction phase is utilized to find and evaluate studies that are relevant to the research questions from the planning phase, as well as extracting information from these studies. Utilizing this phase means to further divide it into three sub phases; Identification of research, Selection of primary studies and Quality assessment.

2.3.1 Identification of Research

In this phase, a strategy on *where* and *how* to retrieve literature that is relevant to the defined research questions from the planning phase is defined. First a list of search engines needed to be specified. IEEE was chosen as the main research database for finding and accessing relevant literature, as it has more than four million documents within fields like computer science. Google Scholar, a web search engine for academic publishing, was also used to get more span when searching for literature. Additionally, JMIR (Journal of Medial Internet Research) and PubMed Central (US National Library of Medicine National Institutes of Health) is also used as they include literature from a technical and medical point of view. Secondly a search query is needed to be able to search for literature at the specified search engines. The process of creating this query consists firstly of taking key terms which is highly related to the research questions and combining them into groups. Each group should contain other terms which is either synonyms, different form or have a similar meaning to the key term. The defined key terms and their respective groups are listed in table 2.1

The search query is generated by applying ”OR” withing the group between each term and ”AND” between each group. By applying this strategy, it is desirable to to get a set of relevant literature that covers all aspects of the research questions which thereafter can go through the selection phase. The generated search query is shown in figure 2.1 and is based on table 2.1.

	Group 1 Problem	Group 2 Tools	Group 3 Solution	Group 4 Data Input	Group 5 Data Output
Term 1	Human Activity Recognition	Deep Learning	Architecture	Wearable sensors	Storage
Term 2	Activity Recognition	Artificial Intelligence	Infrastructure	Body sensor	Memory allocation
Term 3	HAR	Recurrent Neural Network	-	Position specific sensors	Storing results
Term 4	-	Long Short-Term Memory	-	Accelerometer	-
Term 5	-	-	-	Time series	-

Table 2.1: Initial search terms and group

```
('Human Activity Recognition' OR 'Activity Recognition' OR 'HAR')
AND
('Deep learning' OR 'Artificial Intelligence'
    OR 'Recurrent neural network' OR 'Long short-term memory')
AND
('Architecture' OR 'Infrastructure')
AND
('Wearable sensors' OR 'Body sensors'
    OR 'Position specific sensors' OR 'Accelerometer'
    OR 'Time series')
AND
( 'Storage' OR 'Memory allocation' OR 'Storing results')
```

Figure 2.1: SLR query

To try and ensure that the initial search terms where relevant and good choices, multiple queries with different terms were tested. The terms that seemed to return the most accurate primary studies, based on title, were then selected for further use.

2.3.2 Selection of Primary Studies

While conducting the search strategy that is elaborated in section 2.3.1, a large number of literature is most likely to be returned. To reduce it to a more manageable size, some selection criteria is defined. The results given by the search query is filtered by removing duplicates, literature published before 2015 and literature that is not a journal, article or conference paper. The remaining literature after this process will proceed to the next sub-phase.

2.3.3 Quality Assessment

Studies that proceeds from the the previous sub-phase will go through three different screening stages. At each stage, the studies have to fulfill a set of inclusion criteria (IC), see table 2.2, to proceed to the next screening stage. The goal is to filter out studies that are not relevant to the research question this SLR is supposed to answer. Studies are also filtered out if they fulfill one of the exclusion criteria (EC) defined in table 2.2.

#	Description
IC 1	The study main concern is Human Activity Recognition
IC 2	The study focuses on deep learning
IC 3	The study describes the implemented machine learning model
IC 4	The study focuses on accelerometers
IC 5	The study should include figures explaining their solution
IC 6	The sensors should be positioned closely to the thigh and back
EC 1	The study is using video for activity recognition
EC 2	The title includes “smart home”, “video”, “vision”, ”daily living activities” etc

Table 2.2: Inclusion Criteria & Exclusion Criteria

Title and Abstraction

The first screening stage consists of reading the title and abstraction of the studies returned from the search query. At this stage IC1, IC2, IC3 and IC4 are the inclusion criteria the the studies needed to fulfill to advance to the next stage. After filtering and screening, a total of 71 studies proceed to the next screening stage. The reference list of every study that passed this stage is scanned to locate any additional studies that could be of relevance. Title and abstraction screening is performed to decide their relevance.

Introduction, Figures and Conclusion

Out of 71 studies that proceeded from the previous stage will thereafter go through introduction, figures and conclusion screening. At this stage all IC had to be fulfilled to be able to proceed to the next screening stage. Only 20 studies were marked as relevant for further screening. This stage made it possible to eliminate studies with misleading title and or abstraction, thus ensuring that the most relevant studies to proceeded to the next stage.

Full Text Skim

Studies that passed introduction, figures and conclusion screening, will go through a full text screening where each study is given a score based on the set of quality criterias (QC) in table 2.3. Each QC yields either zero, a half or one point, and each study can achieve a total score of seven. For a study to proceed to the data extraction stage it have to surpass a threshold of 2.5 points. A total of 10 studies achieved a higher scored than the defined threshold, and are listed in table 2.5.

#	Description	Points
QC1	The study is in context of using Deep Learning	[0, 0.5, 1]
QC2	Are system or algorithmic design decisions justified?	[0, 0.5, 1]
QC3	The study explicitly states the system architecture/infrastructure	[0, 0.5, 1]
QC4	The study describes how they store the result	[0, 0.5, 1]
QC5	The study uses multiple models to predict	[0, 0.5, 1]
QC6	The study uses position-specific sensor models	[0, 0.5, 1]
QC7	The study takes missing sensor data in to consideration	[0, 0.5, 1]

Table 2.3: Quality criteria

The full text screening revealed that the query had not generated as many relevant papers as initially thought, based on the low scores. Thus it was found necessary to re-factor the terms and groups with focus on getting results that focuses on storing classification results and maintaining accuracy with faulty sensors, to identify possible existing studies regarding the research questions. The new special queries can be seen in table 2.4. Further, after using the special queries, no additional studies was added to the SLR.

Special Query #	Description
SQ1	(“Human Activity Recognition” OR “Activity Recognition” OR “HAR”) AND (”Classification” OR ”Interference” OR ”Transactions” OR ”Sequential”) AND (“Storage” OR “Memory allocation” OR “Saving” OR ”Store”)
SQ2	(”Classification” OR ”Interference” OR ”Transactions” OR ”Sequential”) AND (“Storage” OR “Memory allocation” OR “Saving” OR ”Store”)

Table 2.4: Special RQ Query

Article	QC1	QC2	QC3	QC4	QC5	QC6	QC7
Multimodal Multi-Stream Deep Learning for Egocentric Activity Recognition - (Song et al., 2016)	1	1	0.5	0	1	1	0
Augmented Intelligence: Enhancing Human Capabilities - (Hebbar, 2017)	1	0.5	0.5	0	1	0.5	0
Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video - (Pigou et al., 2015)	1	0.5	1	0	0.5	0.5	0
Dealing with the Effects of Sensor Displacement in Wearable Activity Recognition - (Banos et al., 2014)	0	0.5	0.5	0	1	1	0.5
Deep Activity Recognition Models with Triaxial Accelerometers - (Alsheikh et al., 2015)	1	0.5	1	0	1	0	0
Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition - (Ordóñez et al., 2016)	1	0.5	0.5	0	1	0.5	0
Deep feature learning and selection for activity recognition - (Mohammad et al., 2018)	1	1	1	0	0.5	0	0
Sequential Deep Learning for Human Action Recognition - (Baccouche et al., 2011)	1	0.5	1	0	1	0	0
Deep learning fusion conceptual frameworks for complex human activity recognition using mobile and wearable sensors - (Friday et al., 2018)	1	0.5	0.5	0	1	0	0
LSTM-RNNs combined with scene information for human activity recognition - (Chen et al., 2017)	1	0.5	1	0	0	0	0.5

Table 2.5: Post-screening Primary Studies

2.4 Summary of Related Work

Summary of related work consists of both phase two and three, where phase two is to analyze and extract relevant information from studies that proceeded from the full text screening stage. Phase three, reporting, is using the extracted information to best answer research goal one and synthesis the information extracted for the research question presented in section 2.2.

RQ1 - Does any of the state of the art HAR systems use position- specific models and iterative choose between the best suited trained model?

To find answer to RQ1, information will be extracted from studies that has a score higher than zero on QC5 and QC7, as they focuses on position specific sensors and using multiple models for classification.

(Song et al., 2016) combines two different models, convolution neural networks (CNN) and Long Short-Term Memory (LSTM), to learn features from separate sensors. The convolution neural network is used to analyze spatial video captured from wearable devices and long short-term memory is used to analyze long-term temporal data from different low-dimensional sensors. The output of the two models are then fused using different pooling techniques to compute the prediction results. The idea behind this is that spatial and temporal information will help with the activity recognition, because the spatial information from video can help the classifier eliminate activities that are less likely to be performed at a certain location. Additionally, the paper states that LSTM is better at classifying long-term temporal information, which CNN is incapable off doing.

(Banos et al., 2014) presents the effects of sensor displacement on activity recognition. This study is relevant for this thesis as it aims to increase the robustness and maintaining high accuracy of on-body sensor-based human activity recognition by dealing with inconvenient events with sensors. The effects of sensor displacement are analyzed by three different methods, single sensor activity inference, feature fusion multi-sensor activity inference and decision fusion multi-sensor activity inference with an hierarchical weighted classifier. Results of this study state that the decision fusion multi-sensor method shows a notable tolerance to sensor displacement as decisions are considered independently and then errors from displaced sensors are collectively coped with.

(Pigou et al., 2015) state that using a simple temporal feature method is not sufficient for gesture recognition and therefore proposes a new end-to-end trainable neural network architecture incorporating temporal convolutions and bidirectional recurrence. The paper combines two deep learning classification models, CNN and RNN to identify if there are advantages to have unparalleled spatial feature extraction and recurrence to model the feature evolution. The paper focuses on improving the features final classification instead of optimizing the classification model and does not use sensor-specific models for classifications, but rather a combination. A combination of CNN and bidirectional-LSTM-RNN is analyzed to learn the temporal dynamics in sequential data, and a combination of two new proposed architectures, temporal convolutions and temporal convolutions with recurrent

neural network (RNN) are analyzed to reduce spatial and temporal dimension. The latter architecture proved to handle high-level temporal dependencies while resolving the need for a sliding window to implement frame-wise video classification. The paper states that adding temporal convolutions significantly increases accuracy but introduces jagged predictions This is eliminated by adding recurrence, which leads to think that using LSTM-RNN for activity recognition should be further investigated, as the paper identified that RNNs outperform non-recurrent networks, achieving more than 92% precision.

(Baccouche et al., 2011) also proposes a two-step combination of CNN and RNN classifiers for human activity recognition. The proposed two step architecture consists by first automatically learning spatio-temporal features and then classifying the sequence features from step one, using a LSTM-RNN. Based on results from experiments, it states that the proposed architecture outperforms existing deep models and competes with the best state of the art architectures. Experiments conducted proved that the proposed approach outperforms related deep models on multiple data sets with performance-accuracy above 92%, by learning spatial-temporal features and classifying entire sequences.

Integrating a Semantic, Context and Sentiment analyzer into a combined Deep Neural Network (DNN) for speech-recognition is the aim of (Hebbar, 2017). Speech recognition using environment sensor network for context and speech as features, can be related to the HAR problem as the prediction of text-to-speech is dependent on previous input in the same way prediction activities are on previous movement. The proposed DNN uses bidirectional LSTM-RNN and swaps each hidden layer sequence with forward sequence and backward sequence, ensuring that each hidden layer receives input from both forward and backward sequences. The proposed model takes from the multiple different classification models, and operates on the information developing n hidden layers to obtain the desired output. This yields better judgment and adaption regarding the input. The models used are trained on different input sensor data (features), but the output from the models are propagated into a single deep neural network, which expect the output from previous layer models as input. Results from experiments conducted states that the proposed system achieves higher accuracy than existing solutions, which support the hypothesis that using sensor specific models can improve the accuracy for classification systems.

(Alsheikh et al., 2015) stands out based on classifier choices, as the paper proposes to use Hidden Markov Model (HMM) to learn temporal patterns in human activities, trying to find the most probable sequences of hidden activities that produce observed sequences of inputs. The proposed solution combines Restricted Boltzmann Machines (RBM) and HMM and achieves results that outperforming existing methods for HAR. The sensor data is passed trough spectrogram to extract input feature-windows, which then is passed to the RBM for computing intrinsic features which then is passed to HMM for recognizing the underlying human activities by using posterior probability distribution. Thus the paper proposes a three-step classification system, combining multiple models (non-sensor specific) and a step manipulating the features for better learning. Experiments conducted resulted in achieving an impressive accuracy of 99.13%. The system performs well, but does not use sensor specific models and does not iterative chose between using the HMM

or RBM methods for classification.

(Friday et al., 2018) proposes a new conceptual deep learning framework that focuses on automatic feature extraction to improve system computation time and recognition accuracy, using seven convolutional layers, two gated recurrent units and Support Vector Machine (SVM). Thus, the proposed system utilizes multiple classification methods for activity recognition and focuses on sensor data while proposing two new system architectures, recurrent convolutional neural network and a hybrid SVM. A three-stage classification framework is proposed, first the input data goes trough a pre-processing stage, then the phase, which the paper focuses on, feature extraction consisting of a seven layer CNN (capturing the local regional features) before a RNN aggregates the features. Finally classification on the extracted features are done by a SVM. The paper states that the framework is still under implementation, and thus there are no performance evaluation available.

(Stewart et al., 2018) presents a dual-accelerometer system for classifying physical activity for children and adults, As the system uses two accelerometers attached to the thigh and lower back, it is very similar to the HAR system that HUNT is utilizing. This study aims the check the validity of a dual-accelerometer system, as well as examine the efficacy of using two sensors relative to sensors individually. When using both sensors for classifying physical activity, an accuracy higher than 98% was achieved. Using individual sensors for classification a drop in accuracy up to 26% is shown in one, but not on both sensors for some of the activities. The study uses Random Forrest for classifying activities, but states that RNN may be more suitable for time-series data collected from accelerometers. This can explain the high drop of accuracy when classifying with individual sensors.

Trying to combine features and then use a fusion-technique to increase the activity classification accuracy are common features in the studies presented above. The idea of combining multiple sensors, using a n-step classification method which fuses classifier outputs before a final classifier trained on the fused features, instead of choosing the best classification model for the current input features to classify which activity is being performed differs from what this thesis propose. The proposed solution aims at analyzing at the feature and then decide on a pre-trained sensor-specific model to use for the classification, based on the current input features. Another common research area in the state of the art research is automatic feature- learning and selection to improve the performance, instead of trying to identify and exclude corrupted features. Utilizing both temporal and spatial features to improve system accuracy is also much researched and proven to show accuracy higher than 90%.

RQ2 - What is the state of the art HAR system architecture, with regards to system scalability and storage?

For RQ2, will information be extracted from studies that has a score higher that 0 on QC3 and QC4, as they focuses on system architecture and storage methods.

(Chen et al., 2017) states that location information can improve the recognition accuracy of HAR systems. The study presents a component-based system architecture that

is created with usability and scalability taken in to consideration. It consists of 10 different software components and is presented with a figure. For storing raw sensor data from accelerometers and gyroscopes, a sensor processor component is used, but it is not elaborated what kind of storage method that is being used. With regards to system scalability, the study utilizes a slow intelligence system server witch manages communication between components. This allows the system to be extended with further component without any major system changes.

(Hebbar, 2017) states that ”existing systems are not fully aware of the context”, which indicates that classifiers might be unable to predict future state of the input provided appropriately, resulting in iterating trough the data in the wrong direction (using bidirectional methods), increasing the error gradient and decreasing the learning rate.

(Alsheikh et al., 2015) shows strong improvements on real world problems using only triaxial accelerometers as input for deep learning methods, competing with the best state of the art activity recognition systems. The proposed system focuses on both learning the classifiers weights used for recognition, but also manipulating and identifying activity membership probabilities for the input features for better utilization and learning for the activity classifier. Further the proposed system is split into three phases, first data gathering, then offline learning and finally online activity recognition and inference. The choice to keep the training offline, is because of the computational burden is heavy and not thus not suited for online-training on mobile devices. Conducting offline-training based on the computational cost and amount of data to train on makes a strong case, which is why this thesis proposes a system that does both offline- training and classification.

(Mohammad et al., 2018) focuses on activity recognition from short sequences of sensor readings with the aim of proposing a middle ground, between accuracy and performance speed using deep neural architecture for feature learning and feature selection. Data generation is gathered trough a mobile sensor-devices, such as mobile phones or smart watches. Further the paper states that when utilizing mobile devices it is paramount to reduce energy consumption and thus proposes a new system architecture solution. First the new proposed solution ”employs a large deep system followed by recursive aggressive feature selection applied to all neurons of the deep system treated as feature extractors.” The new architecture aims to first learn features, then select and extract features, which then are used to train the network. Feature learning is the main focus of the paper and proposes a multi-CNN architecture for slow feature extraction and three transformers, fast Fourier Transform, spectrogram and tensor manipulation on the features. To reduce features the feature selection can be interpreted as a network pruning method, reducing the computational cost, increasing the performance speed. Pruning is done by selecting weights and neurons that do not contribute to the calculation of selected features. The classification model in the paper is LSTM-RNN and uses an online-method for training (training is done as data comes in, and not on pre-gathered data), which spreads the classification time consumed. Thus improves classification performance speed, as it only classifies one-by-one input sequence and not multiple at once, which requires a longer time to finish. Final result proved by producing a large set of informative features and then keeping the most

informative for training, that could outperform state of the art on six out of seven publicity available data sets. Thus the paper proves that smaller but smarter networks in terms of feature selection and number of neurons can be equally accurate as deeper network, and smaller networks are easier to scale and multiply as the computational cost and resources become cheaper.

(Friday et al., 2018) states that ”sensor based method provide better advantages over other techniques as it is not limited to geographical location, economical to acquire, ease of deployment and does not pose any health hazard due to radiation.” and that ”Feature extraction is an important stage as it helps to reduce computation time and ensure enhanced recognition accuracy”. The latter statement can be seen as one of the biggest focus areas within HAR systems performance, especially automatic feature selection and extraction, as a huge amount state of the art studies focuses on reducing the system computation cost and speed, while maintaining high performance accuracy.

(Ordóñez et al., 2016) states that combining convolutional and LSTM recurrent layers shows advantages when performing activity recognition from wearable sensor data. In this architecture, the convolutional layers act as feature extractors and the LSTM layers model the temporal dynamics of the activation of the feature maps. The study presents a framework that can perform activity recognition on sensors individually or fuse multiple sensors (accelerometers, gyroscopes and magnetic sensors) to improve performance. When fusing all sensors, the framework improves by an average of 20% accuracy

From the papers presented above it is clear that there is not much focus in state of the art with regards to system scalability and classification storage. State of the art system architecture is experimenting with offline training based on the computational costs of processing the huge amount of data needed to train a good HAR system. Online classification based on the same idea that the computational costs are lower for classification than training because the classification input can be a stream of features and does not require a huge amount of computational power as input streams can be classified continuously and fast enough to avoid building up a queue. Thus utilizing mobile devices are applicable. A lot of the state of the art research is proposing architectures that focuses on feature learning and extraction, to be able to reduce the computational power needed for training and classification, by pruning features and reducing the number of neurons needed in a deep learning network. The most used models are a combination of CNN and LSTM-RNN and tries to utilize spatial features to improve accuracy, and thus presents component based framework.

2.5 Summary

The SLR discovered knowledge gaps and that there is limited to none primary studies that alone can answer the research questions. The primary studies that passed all screening phases are used as information and knowledge sources, to help the thesis answer the research question, by proposing a new system architecture, based on the ideas and research conducted in the primary studies. The primary studies focuses on feature learning and

extraction, trying to correct and or handle missing or faulty input data using feature transformations, as well as input from different sources by utilizing one or multiple classifiers in a n-step classification method, which fuses the result from the different classifiers and uses the fused features as input to a final classifier.

The proposed system will look at the sensor data to try and identify which sensor is faulty and then eliminate that input data and pass the remaining features trough a pre-trained position specific models. Based on the information gathered from trying to answer research question one, the proposed classifier will be utilizing LSTM-RNN for classification, as the studies show that HAR systems based on LSTM-RNN classifiers achieve accuracy's above 90%. As (Chen et al., 2017) is the only study that briefly presents storage, it gives a clear indication that storage in not something that researchers spends a lot of resources on and that there is done limited work on the two topics.

State of the art HAR systems does not use position-specific models and iteratively choose between the best suited trained model, and there is no common system architecture for scalability and storage. Minimizing the knowledge gap by conduction research and doing experiments is a motivational factor for this thesis as there are huge amounts of data that needs to be stored and accessed and a high probability for corruption of a sensor data stream.

Chapter 3

Background Theory

This chapter describes theory that is necessary to understand for the conducted experiments in this thesis. First the chapter describes different types of machine learning methods and models. Then an in-depth explanation of human activity recognition and the activity recognition chain that describes to process for activity classification. Finally performance measures used for evaluating and comparing the experiment results are presented.

3.1 Machine Learning

Machine learning as a computer science field, is the task of building algorithms for computer software that automatically improves their performance for a specific task over time by learning. Over the past years, the use of machine learning has increased remarkably and is being used in fields like marketing, medical diagnostics, robotics and search engines. It is desirable for computer software to learn, as it is not possible for humans to analyze huge amounts of data for complex problems without help of machines. How computer software is able to learn is defined in definition 2.

Definition 2. A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. (Mitchell, 1997)

3.1.1 Types of Machine Learning

Machine learning consists of three different types of learning methods; supervised, unsupervised and reinforcement learning. Distinguishing features between these methods depends on the type of feedback that is provided to the method to learn from. An additional hybrid method between supervised and unsupervised learning exists, the different methods are described in depth in Russel (Russell, 2016). This sections gives a brief introduction to the different types of machine learning.

Supervised Learning

A simple example of supervised learning is thinking about it as a student and a teacher. The student (model) is asked to learn to recognize different people from images (input). For each image the student makes an educated guess, the image is of person X (output), and the teacher gives the student feedback in the form of wrong, person X has blond hair or correct, it is person X.

In supervised learning the agent observes some example input-output pairs and learns a function that maps from input to output. Russel (Russell, 2016) describes that supervised learning requires "the prior identification of relevant inputs and correct outputs" and therefore is not able to operate autonomously without the help of a human. Experiments conducted in this thesis (see chapter 6) are of the type supervised learning, because the training data consists of input-output pairs, annotated by humans. Supervised learning is mostly used for classification and regression problems.

Unsupervised Learning

Unsupervised learning can be thought of as learning without a teacher. The student learns to recognize the different individuals based on images by trying to find structures in the data to distinguish the individuals.

In unsupervised learning the agent learns patterns in the input even though no explicit feedback is supplied. A common unsupervised learning task is clustering (discover the inherent groupings in the data), e.g. grouping together different fruits, based on input features like shape, color and weight.

Reinforcement Learning

In reinforcement learning the agent learns from a series of reinforcement–rewards or -punishments. It is up to the agent to decide which of the actions prior to the reinforcement were most responsible for it. An agent can learn a transition model for its inputs and to predict the next inputs. Thus, the task is to use observed rewards to learn an optimal (or nearly optimal) policy for the environment.

Not all problems fit to supervised- or unsupervised learning, and software playing chess is a good example where reinforcement learning is suitable. In chess there are tens of thousands of moves that can be played, and creating a knowledge base would be impossible and tedious work. The software has to learn by reinforcement when something good happens.

To distinguish between unsupervised- and reinforcement learning, if a person wants to buy a car, the unsupervised method would try and learn what underlying patterns the person likes about cars. The reinforcement method would suggest cars to the person, and getting feedback on what he or she likes, and then suggesting new cars based on the persons preferences.

Hybrid methods

Additionally to these three methods, there are also a hybrid methods between supervised- and unsupervised learning called semi-supervised learning, where different types of input

is used for the method to learn from. Input data consists of both labeled and unlabeled examples, where as the labeled data helps the method identify that there exist some groups in the data and what they might be, but then trained on the unlabeled data to define the underlying patterns that distinguishes the different groups in the data. Semi-supervised learning, can be divided into two different categories, inductive and transductive. Inductive learning is used to enhance performance by combining the usage of labeled and unlabeled input data from the whole problem. Transductive learning is more related to unsupervised learning and is used to predict specific examples given specific examples from a domain.

3.1.2 Architectures

Artificial Neural Networks

Artificial neural networks (ANN) is founded on the hypothesis that mental activity consists primarily of electrochemical activity in networks of brain cells called neurons. There are many types and structures of ANN, but in its simplest form, feed-forward ANN, can be modelled as a number of nodes connected through direct links in one direction (see figure 3.1a). The first layer is always the input layer, the middle layers are called hidden layers and the last layer is always the output layer. Each link between nodes has a numeric weight associated with it, which determines the strength and signal of the connections. Links are updated after an input has moved from the input layer to the output layer using backwards propagation and gradient decent. Backwards propagation is a method used to calculate a gradient that is needed in the calculation of the weights to be used in the network. Gradient Descent is an optimization algorithm, based on a convex function, that tweaks parameters iteratively to minimize a given function to its local minimum. When moving forward through the network, each neuron has a function called activation function. The function decides the strength of the output signal the node fires, and then the signal is sent forward through the network, reaching the output layer.

There exist a variety of activation functions, a simple linear activation function and its output is described in equation 3.1 where n is number of features, x_i is the input feature, w_i is the weight vector and b is the bias.

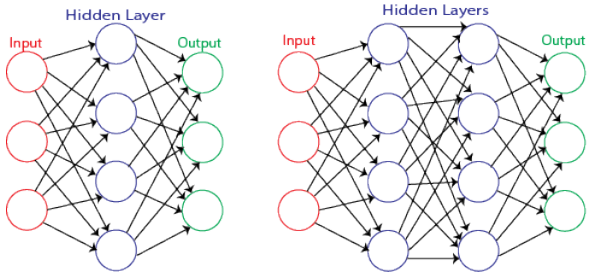
$$f \sum_{i=1}^{n} (x_i * w_i + b) \tag{3.1}$$

Output from the neural network is based on the strongest output signal from each neuron in the output layer.

Deep Learning

Deep learning refers most of the time to ANNs that has multiple hidden layers in the network (see figure 3.1b). More hidden layers help the ANN to generalize and learn more complex tasks. Each layer learns to transform its input data into a slightly more abstract and composite representation. It is not always the best solution to use too many hidden layers or neurons, because it could lead to overfitting in the network. Another concern about deep learning is its potential requirement for high computational power and long training

time, as ANNs can become computationally heavy by introducing too many neurons and weights for calculations.



(a) ANN with the input, output and one hidden layer

(b) Deep ANN with input layer, two hidden layers and output layer (blue)

Figure 3.1: ANN structure

Recurrent Neural Networks and Long Short-term Memory

RNN is an ANN architecture that employ recurrence. The recurrence passes information from a previous iteration forward through the network. The following equations describe how RNNs work;

$$h^{(t)} = tanh(Ux^{(t)} + Wh^{(t-1)} + b) \tag{3.2}$$

Equation 3.2 is computing the hidden state for an entity in the sequence, denoted as $h^{(t)}$. U is the weight associated with the corresponding input at entity $x^{(t)}$, and W is the weight associated with the previous hidden state. Bias is denoted as b.

$$\hat{y}^{(t)} = softmax(Vh^{(t)} + c) \tag{3.3}$$

Equation 3.3 is computing the output (classification) of the RNN, denoted as $\hat{y}^{(t)}$. V is the weight associated with the hidden state $h^{(t)}$ computed in equation 3.2, and bias is denoted as c.

RNNs has proved successful when applied to problems where the input data are temporal-bound, such as sequence data or when order is important. The output of the recurrent architecture can either be output for every entity in the input sequence or an output over the entire sequence. The essential ingredient which differentiate RNNs from feed-forward neural networks is the hidden state, that represents a summary of the entities seen in the past, for the same input sequence.

Vanishing gradients and gradient explosion are two problems with training RNNs. During training of the network, equations 3.2 and 3.3 can be derived for the expression 3.4

$$\frac{\partial L}{\partial W} \tag{3.4}$$

by using the chain rule, that calculate the derivatives of composite functions, U to show that the multiplication of W is the equivalent to multiplying a real number over and over

again. This might lead to the product shrinking to zero or exploding to infinity, introducing vanishing gradients. For in depth explanation about the cause of vanishing and gradient explosion, see Ketar (Ketkar, 2017). There are techniques to deal with the gradients, such as to re-scale the gradient once it exceeds a pre-defined threshold. This technique is simple and computational efficient, but has one drawback, as additional hyper parameter is introduced to the architecture.

Many variants of RNNs exist, Bidirectional RNN is one of them and works in the same way as standard RNNs but, with additional information propagating through the network. The key concept behind it is the use of future input information to make predictions for the current entity.

Another variant of RNN is LSTM which introduces a cell state to overcome the vanishing gradient problem and uses gradient clipping technique to overcome the exploding gradient. Ketar (Ketkar, 2017) describes the architecture of LSTM networks with the equations 3.5 through 3.9, where $\bigodot$ denotes point-wise multiplication of two vectors, the functions σ, g and h are non-linear activation functions, all the W and R terms are weight matrices and all the b terms are bias terms. x is the input features at time t.

The most important element of the LSTM is the cell state c, expressed in equation 3.8, where the state is updated based on the block input z, equation 3.5, and the previous cell state c^{t-1}. The input gate i, equation 3.6, determines what block input makes it into the cell state and the forget gate f, equation 3.7, determines how much of the previous cell state retain in the new updated cell state. The p-terms are called peephole connections, which allow for a fraction of the cell state to factor into the computation of the term in question. o, equation 3.9, is the output gate and $\hat{y}$, equation 3.10, is the output after point-wise multiplication of output gate and the hidden cell states. Figure 3.2a shows how all the discussed equations builds a network, that can be self-looped and, thus creating the RNN architecture.

$$z^{(t)} = g(W_z x^{(t)} + R_z \hat{y}^{t-1} + b_z) \tag{3.5}$$

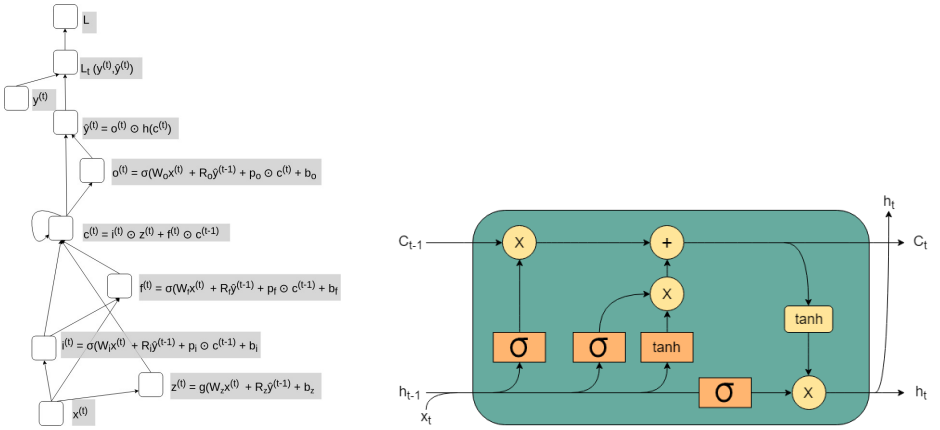
$$i^{(t)} = \sigma(W_i x^{(t)} + R_i \hat{y}^{t-1} + p_i \bigodot c^{(t-1)} + b_i) \tag{3.6}$$

$$f^{(t)} = \sigma(W_f x^{(t)} + R_f \hat{y}^{t-1} + p_f \bigodot c^{(t-1)} + b_f) \tag{3.7}$$

$$c^{(t)} = i^{(t)} \bigodot z^{(t)} + f^{(t)} \bigodot c^{(t-1)} \tag{3.8}$$

$$o^{(t)} = \sigma(W_o x^{(t)} + R_o \hat{y}^{t-1} + p_o \bigodot c^{(t)} + b_o) \tag{3.9}$$

$$\hat{y}^{(t)} = o^{(t)} \bigodot h(c^{(t)}) \tag{3.10}$$



(a) LSTM Network Architecture **(b)** LSTM Cell

Figure 3.2: (a) Detailed illustration of how the LSTM network is connected. **(b)** Detailed illustration of an internal LSTM cell

Figure 3.2b shows the LSTM cell architecture, and how the different input gates and states are maintained. The gates can be thought of as standard neurons in a ANN, as they compute an activation output using a activation function. The different gates are depicted as $sigmoid$ functions in the figure. The left most gate is the forget gate, next going towards the right are the input gate. Then there is the calculation of the cell state, depicted as $tanh$ function. The right most $sigmoid$ function is the output gate.

LSTM cell flow is to first calculate what to forget by running the input and previous hidden state trough an activation function, then multiplying it with the previous cell state C_t. The new signal C_t after first operation is then added to the output of calculating what to retain in the cell by running the raw input signal and previous hidden state trough an activation function and multiplying it with the output of running the input and previous hidden state trough $tanh$ function. This new signal is the new cell state C_t.

The new cell state is then sent out of the cell, but also a copy is sent to a $tanh$ function to calculate the new hidden state. After the new cell state is passed trough the function, it is multiplied with result of the output gate which is running the input and previous hidden state trough a activation function. The result signal is the new hidden cell state.

Decision Trees

A Decision tree (DT) is a tree structured representation (see figure 3.3) of different decisions and situations that are used for classification purposes. Classification of instances works by sorting them down the tree from the root, answering each node's question and follows the corresponding path down the tree to a leaf node. Each node in the tree specifies a test of some attribute of the instance. Branches descending from a node represents a possible value for an attribute. Leaf nodes are the answer or class to give the input. Building a DT is five steps on repeat; calculate entropy of every attribute, then split the data into subsets, using the attribute with lowest entropy (or highest information gain). Then create

a decision node in the tree with the attribute used to split the dataset. Repeat the five steps on the remaining data subsets and using only the attributes that are not a decision node.

Entropy is a measure of the amount of uncertainty in the input data set, and is calculated for each attribute, to decide which attributes gives us the most information about the data, e.i finds the attribute that creates the highest amount of different subsets by splitting the dataset, so that the tree can be as little as possible and few attributes can be used to decide the output classification.

Information gain is a measure of the difference in entropy from before to after the dataset is split on an attribute (e.i in other words, how much uncertainty in the dataset was reduced after splitting.)

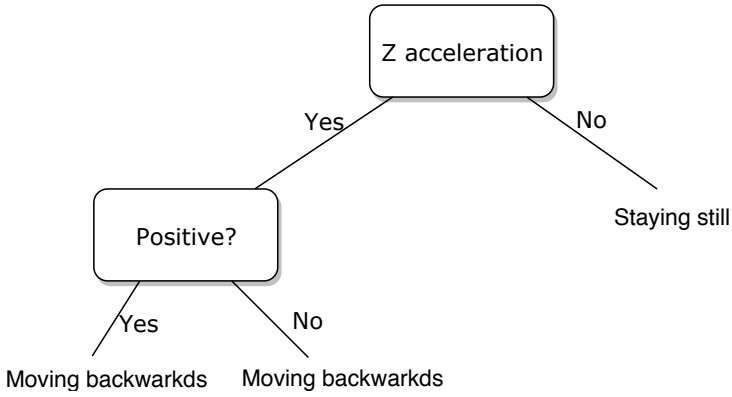


Figure 3.3: Decision Tree Architecture

Random Forest Classifier

Random forest classifier (RFC) is an ensemble machine learning method that are used for classification, meaning it combines models into one big model, where all the individual models classify the input and then a voting is done by the RFC. A simple voting method can be to choose the class that got most classifications by the individual models. Each individual model tries to learn different types of input data. One can think of ensemble methods as a team, where each member (individual model) has a specialty field, then when the team is gonna vote on a decision, the option with most votes are chosen or the option with the most certainty of success (member confidence). Hopefully the individual team member that has the needed specialty is the most confident model, gets chosen and is correct. The method aggregates multiple decision trees to produce a more generalized model because single decision trees tend to over fit as they grow very deep.

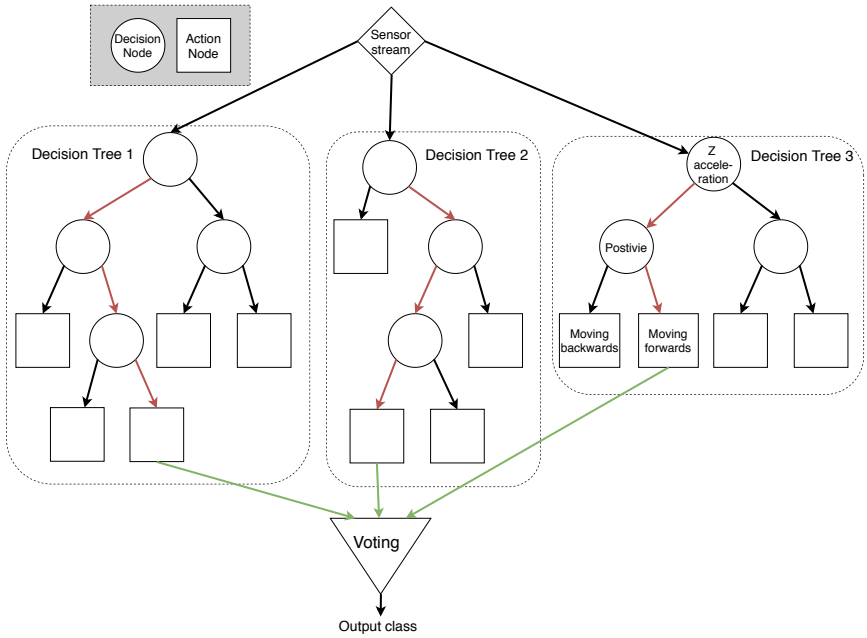


Figure 3.4: Random Forest Architecture

3.2 Human Activity Recognition

Human activity recognition is a supervised learning problem, where the task T is to recognize and classify activities performed by an individual. Experience E, which consists of labelled data, is given to the software as movements recorded by sensors placed on the individuals body. The system is evaluated based on a performance measure P that measures how many unseen motions from different individuals the system correctly classifies.

A more comprehensive and detailed definition of HAR systems is stated by Lara and Labrador (Lara and Labrador, 2013) in definition 3.

Definition 3. Given a set $S = S_0, ..., S_{k-1}$ of k time series, each one from a particular measured attribute, and all defined within time interval I = $[t_\alpha, t_\omega]$, the goal is to find a temporal partition $(I_0, ..., I_{r-1})$ of I, based on the data in S, and a set of labels representing the activity performed during each interval I_j (e.g., sitting, walking, etc.). This implies that time intervals I_j are consecutive, non-empty, non-overlapping, and such that $\bigcup_{j=0}^{r-1} I_j = I$, (Lara and Labrador, 2013).

3.2.1 Activity Recognition Chain

Bulling et. al (Bulling et al., 2014) states that HAR systems can be presented and as a sequence of operations, called Activity Recognition Chain (ARC), to recognize activi-

ties. Hessen and Tessem (Hessen and Tessem, 2016) describes the operations in detail and presents the ARC visually as in figure in their thesis as figure 2.1 in Chapter 2.

This thesis focuses primarily on classifying sensors no-wear time with a meta classifier before doing the activity classification with position specific models. Therefore the ARC presented by Hessen and Tessem (Hessen and Tessem, 2016) has been extended with an additional step, meta classifier. The activity classification step has also been extended with two additional position-specific LSTM classification models. These two steps functions as an ensemble classifier, see section 5.2 for an in-depth explanation on how the proposed system works as an ensemble classifier. Section 3.2.2 will describe these steps in detail and figure 3.5 show the extended ARC.

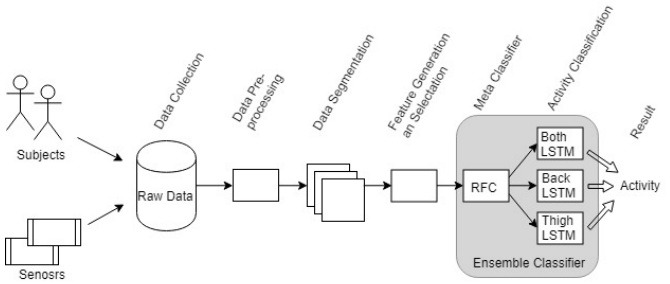


Figure 3.5: Activity Recognition Chain

3.2.2 Classification

Classification is a supervised learning technique where the task is to determine to what group a certain observation belongs too. Using HAR as an example, the goal is to determine what kind of activity an individual does at a given time utilizing acceleration in x, y and z directions recorded from accelerometers attached on an individuals body.

Hypothesis 1. To improve a HAR system's classification accuracy and overcome faulty sensor data, it is better to have position specific activity classification models rather than try to manipulate the faulty data.

As most state of the art HAR classifiers aim to overcome faulty sensor data stream by rotation and transformation of data (see section 2.4), this thesis aims to improve HAR classification based on hypothesis 1. The improved classification aims to identify which sensors streams are faulty by using a meta-classifier and then eliminate features from faulty sensors streams in the input features. After the identification and elimination of of faulty data in a stream, one of the individual models in the proposed ensemble classifier is used to perform the final activity classification.

The proposed system has a meta classifier that detects sensor no-wear time when data is faulty or missing, and three position specific classifiers that performs activity classification. The classifiers are listed below and figure 3.6 shows the classification for one extracted feature window, where the window is propagated through the figure from bottom up.

- **Meta** classifier that labels sensor recordings with '1' (both), '2' (thigh), '3' (back) and '4' (none) based on temperature and distance moved.

- **Back, Thigh and Both** position specific sensor models takes sensor features from the meta classifier which is labeled respectively with '1', '2', and '3' as input, and performs activity classification on them. Output is an activity performed at a specific timestamp.

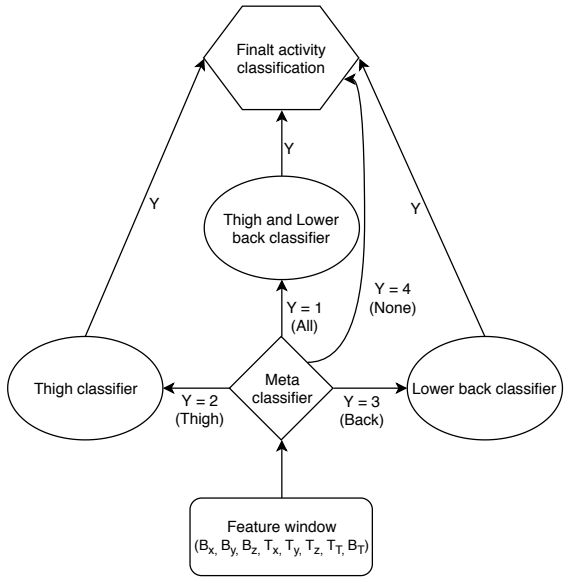


Figure 3.6: Ensemble classifier architecture

Meta Classifier

The idea behind the proposed meta classifier is based on Reinsve's (Reinsve, 2018) master thesis where he did a feasibility study on using RFC to identify sensor no-wear time, which achieved an accuracy of 95.6%. The proposed meta classifier aims to achieve better overall performance measures by adding features such ass temperature memory and distance moved trough feature extraction. Thus hypothesis 2 was defined.

Hypothesis 2. Adding features about previous windows and the current windows movement from the accelerations are going to improve the overall Accuracy, Precision, Specificity, Recall and F_1 score of the meta classifier.

Both the meta classifier and RFC from the feasibility study utilizes the sensor's on-board temperature sensor to record temperature while the sensor is turned on. Collected data is treated as input to the meta classifier which classifies different sensor configurations. Throughout the thesis sensor configuration will be used to represent the non-faulty sensor data streams, that can be identified. There are four possible sensor configurations, only

back- , only thigh-, both- or none of the sensors are attached and records valid features. Reinsve's (Reinsve, 2018) feasibility study showed that differentiating between both sensors attached and no sensors attached was to difficult to learn and therefore decided to drop the class, N - No Sensors Attached. A more in-depth-discussion about this can be found in Reinsve (Reinsve, 2018) in section 7.2.3. The proposed meta classifier aims to be able to differentiate between these two labels by adding additional features, temperature memory and distance moved. By adding temperature memory, it is desirable that the meta classifier will be able to remember if the temperature is falling , indicating that it might be unattached to the subject. Distance moved aims to detect changes in sensor acceleration. Very small changes in acceleration should give an indication that the sensor is lying still. Based on this, hypothesis 3 was created.

Hypothesis 3. Adding features about previous windows temperature and distance moved to the current window are going to help the meta classifier differentiate between the two classes 1(A - Both Sensors Attached) and 4(N - No Sensors Attached).

Input to the meta classifier in this thesis is the temperature readings from both sensors as well three new features, acceleration distance and two different temperature memory features. For more comprehensive explanation of RFC features, see section 5.2.3.

These extra features are added to achieve better overall Accuracy, Precision, Recall and F_1 score with the new meta classier as opposed to the one used in the feasibility study. Appendix A section A.1 describes both the input data to the RFC, segmented into windows and windows reduced to extracted features. Chapter 5 explains the entire data pre-processing sequence and table 5.1 summarizes the different features.

When the data pre-processing is done, the resulting input data with a shape of $extracted\ feature\ windows = [\frac{\text{number or samples in input data}}{\text{sequence length}}, \#\ extracted\ features]$ Then the $extracted\ feature\ windows$ are fed to the meta classifier, which returns an array with the "guessed" classification. $guessed\ classification = [\frac{\text{number or samples in input data}}{\text{sequence length}}]$ where each element is one of the seen during training.

3.2.3 Performance Measures

Performance measures are quality metrics used to measure how the system improves over time and the final expected overall performance for a system. Standard performance measures for classification machine learning tasks are measured in terms of *positive* and *negative* classifications (P = correct, N = incorrect), *true positive* and *true negative* (TP, TN respectively) which denotes correctly classifying an entity as positive or negative and finally *false positive* and *false negative* (FP and FN respectively), which refers to a wrong classification.

Accuracy is a quality metric used for evaluating classification tasks, and is defined in definition 4

Definition 4. Accuracy refers to a measure of the degree to which the predictions of a model match the reality being modeled. $accuracy = P(\lambda(X) = Y)$, where XY is a joint distribution and the classification model λ is a function $X \rightarrow Y$. Sometimes, this quantity is expressed as a percentage rather than a value between 0.0 and 1.0.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.11}$$

(Sammut and Webb, 2010a)

Precision is a metric that uses *true positives* and is defined in definition 5

Definition 5. Precision is defined as a ratio of true positives (TP) and the total number of positives predicted by a model.

$$Precision = \frac{TP}{TP + FP} \tag{3.12}$$

(Ting, 2010b)

Recall is also used in evaluating classification tasks and is defined in definition 6

Definition 6. Recall is measuring the ratio between *true positives* and total number of positives in the data, also known as sensitivity.

$$Recall = \frac{TP}{TP + FN} \tag{3.13}$$

(Sammut and Webb, 2010c)

Specificity and Sensitivity, defined in definition 7, is used together as a quality measure representing the predictive performance of a classification model.

Definition 7. Specificity is the fraction of negative examples predicted correctly by a model, while Sensitivity is the fraction of positive examples predicted correctly by a model.

$$Specificity = \frac{TN}{TN + FP} \tag{3.14}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.15}$$

(Ting, 2010c)

F_1-measure is an evaluation metric, defined in definition 8, that relies on both *precision* and *recall*. Note that $F_1 - measure$ does not take into consideration true negatives. When working with imbalanced data sets such as HUNT, F_1-measure is a better metric than accuracy, because it gives the harmonic means and thus is applicable when the average of rates is desired.

Definition 8. $F_\beta = (1 + \beta^2) * \frac{precision * recall}{(\beta^2 * precision) + recall}$, substituting β with 1 gives

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{3.16}$$

(Sammut and Webb, 2010b)

Confusion Matrix is used to visualize the classification performance. A comprehensive definition is found in definition 9

Definition 9. A confusion matrix summarizes the classification performance of a classifier with respect to some test data. It is a two-dimensional matrix, indexed in one dimension by the true class of an object and in the other by the class that the classifier assigns. Table 3.1 presents an example confusion matrix for a binary classification task.

(Ting, 2010a)

		Assigned Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

Table 3.1: Confusion matrix example

A confusion matrix helps to visually find which classes the classifier misclassifies, through a two-dimensional matrix, visualized in table 3.2 with a three-class classification task with the classes Walking, Sitting and Standing. The confusion matrix shows that the classifier is not confused about classifying walking, as it classifies 10 out of 10 walking-examples correctly. Sitting should be improved as the confusion matrix shows that the classifier is struggling to identify and assign the correct class to examples that are labeled with sitting in the test set. Furthermore the confusion matrix shows that the classifier is actually quite accurate at classifying the standing examples and is only classifying one example as walking. The miss-classification could mean that there are some anomalies in the data that are worth investigating.

		Assigned Class		
		Sitting	Standing	Walking
Actual Class	Sitting	10	2	0
	Standing	0	7	1
	Walking	0	3	8

Table 3.2: Three-classification confusion matrix example

Chapter 4

Data and Datasets

The chapter describes the data and the datasets that are used in this thesis. First the Trondheim Free Living dataset is described and then the Sensor No-wear Time dataset.

4.1 Data Collection

As a part of the HUNT study, collection of participants movement data is an essential part. To record movement data, two Axivity AX3 3-Axis Accelerometer sensors (Axivity, 2019) are used. These sensors are able to record oriental changes in x, y and z axis, ambient light and temperature continuously for 14 days at 100 Hz (100 recordings per second). For the HUNT study, the sensors are placed on the participants lower back and right thigh and records data at 50 Hz over a time period of seven days.

Figure 4.1: Axivity AX3 3-Axis Accelerometer

(a) Placement of thigh sensor

(b) Placement of back sensor

Figure 4.2: Sensor placement on participants body

4.2 Datasets

Two different datasets are used for this thesis, the Trondheim Free Living (TFL) dataset and the Sensor No-wear Time (SNT) dataset. These two datasets will be elaborated in detail below.

4.2.1 Trondheim Free Living dataset

The TFL dataset was introduced in the master thesis by Vågeskar (Vågeskar, 2017) and consists of labeled movement data from 22 different participants, recorded in an out-of-lab environment. The existing HAR system is trained on the this dataset, therefore the position specific models developed in this thesis will also use the TFL dataset for training in order to compare results as accurate as possible. Chapter 3, section 3.1.8 in Vågenskar (Vågeskar, 2017), describes the creation process of the TFL dataset.

Subjects

The 22 adult subjects in the TFL dataset consists of 15 males and 7 females, each with an unique id ranging from 001 to 022. Subject 001 through 005 and 007 was excluded in this thesis because of differences in sensor placement.

4.2.2 Sensor No-wear Time dataset

Reinsve (Reinsve, 2018) created a new dataset for his thesis, called ”Sensor no-wear time”. With this dataset he aimed to detect if sensors are attached to the participants body or not during data recording by utilizing the temperature readings from the on board temperature sensor on the Axivity AX3. The temperature readings is not a direct measurements of

skin temperature, although it is expected that the readings would correlate to the skin temperature of the person wearing the senors.

In Reinsve (Reinsve, 2018) the data collection resulted in four different recordings, each where a sensor was place on the right thigh and lower back of each participant. Recordings were done by two male adults following two different protocols. These two protocols are described step-by-step below.

Protocol 1 (4 steps)

1. Put two sensors on and be active for about an hour.
2. Take back sensor off and let it lie on a table for an hour.
3. Put on the back sensor and then take thigh sensor off and let it lie on a table for an hour.
4. Take off thigh sensor (both sensors are off).

Protocol 2 (6 steps)

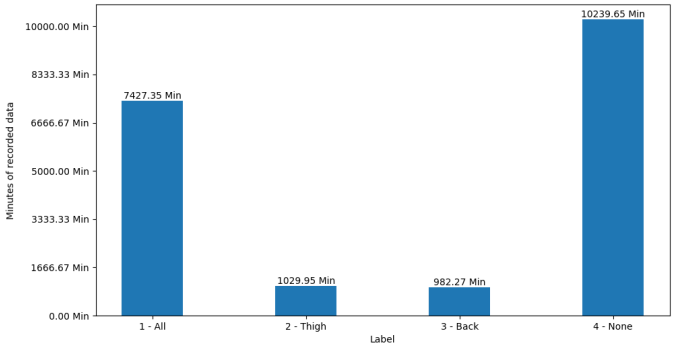
1. Put two sensors on and be active for about an hour.
2. Take back sensor off and let it lie on a table for an hour.
3. Put on the back sensor with opposite device orientation.
4. Take off thigh sensor and let it lie on a table for an hour.
5. Put on the thigh sensor with opposite device orientation.
6. Take off both sensors.

Data collection

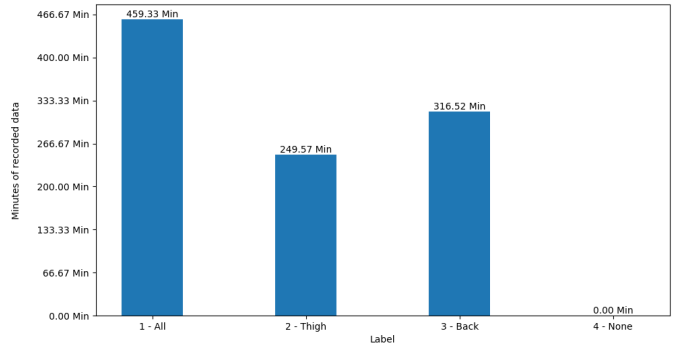
Initially, data generated for the SNT dataset was not comprehensive enough for the meta classifier to maintain high accuracy, as it only contains four three-hour recordings with small amounts of sensor configurations. Hence for the purpose of this thesis, a new SNT dataset with higher rates of variation had to be created. The data generation process for the new SNT dataset consists of recordings from two different subject groups.
In the first group, the authors of this thesis wears two sensors and alternates between the four sensor configurations at different times throughout the day. This is done to generate more variation in the data and each recording has a duration between two and four days.
The second group, collects data from different HUNT4 participants and was supervised by Atle Kongsvold. Each participant followed the protocol described below, but did not have to follow the steps in any specific order, and the time between each step is decided by each individual. Every time a sensor were attached or detached, the timestamp of the respective action were written down in a spreadsheet, and corrected trough analyzing each sensor stream with the OMGUI (Open Lab, 2018) software.

Protocol for the second part of data generation

- Take both sensors off for 5 minutes (let sensors lie on a table). Put sensors back on.
- Take both sensors off for 10 minutes (let sensors lie on a table). Put sensors back on.
- Take both sensors off for 15 minutes (let sensors lie on a table). Put sensors back on.
- Take both sensors off and take a shower (let sensors lie in the bathroom). Put sensors back on.



(a) Data Distribution for the SNT dataset



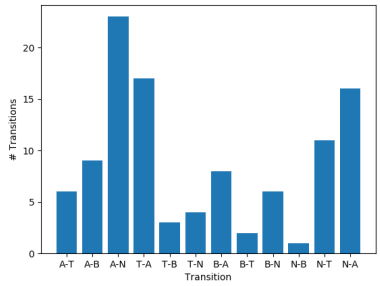
(b) Data Distribution for Reinsve SNT dataset

Figure 4.3: Label distribution

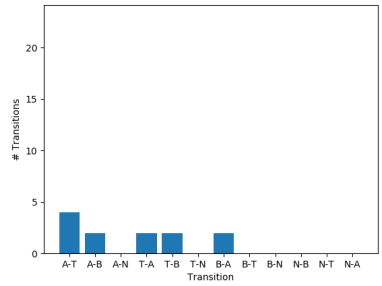
When analyzing figure 4.3a, it becomes clear that the dataset is very unbalanced. The figure shows that there are 7 427,35 minutes of collected data where both sensors are attached, that is 7,21 times more data than data with only the thigh sensor attached. The

small amount of data with only one sensors attached can mean that during leave-one-out training, if a test dataset is chosen with most of its valid features are only from one sensor, it would probably result in bad performance as it encounters data it has never seen before or learned to classify.

Further analysis of the SNT datasets reveals that the sensor transitions are more balanced than the actual data distribution, see figure 4.3a and 4.4. This happens because the time spent with the sensor configuration after a transition can vary. As the dataset is more balanced with regard to transitions, there is a high possibility that the training data contains the transitions found in the test data. It should therefore be able to learn the transitions and sensor configurations even though it has few hours of some of the sensor configurations. If the model can learn to recognize the different transitions, it will help with learning when one sensor is detached or attached. The new SNT dataset has a total of 106 transitions compared to the 12 transitions in the SNT dataset from Reinsve's (Reinsve, 2018) feasibility study. Figure 4.4a and 4.4b show the different transitions for the new SNT dataset and Reinsve's (Reinsve, 2018) SNT dataset respectively. Table 4.1 describes the abbreviations for the different sensor transitions used in figure 4.4.



(a) Sensor transitions for the new SNT dataset



(b) Sensor transitions for the SNT dataset from Reinsve

Figure 4.4: Sensor transitions

Abbreviation	Transitioning from	Transitioning to
A-T	All	Thigh
A-B	All	Back
A-N	All	None
T-A	Thigh	All
T-B	Thigh	Back
T-N	Thigh	None
B-A	Back	All
B-T	Back	Thigh
B-N	Back	None
N-B	None	Back
N-T	None	Thigh
N-A	None	All

Table 4.1: Abbreviations used in figure 4.4

Subjects

A total of six adult males participated in the data collection process for the new SNT dataset. There are no female subject participants because at the time of recording, no female subjects was available for data generation. Even though there are no female subjects it should not have any effect, as the data collected or the models ability to learn is dependent on the temperature recordings, and females and males has the same body temperature. Each participant was given an unique id ranging from 001 to 006. If a subject has multiple recording, a integer suffix is added to the id to distinguish between the recordings, e.g 001.1. Table 4.2 shows a full overview of the subject in the new SNT dataset.

Subject id	Age	Gender	# Recordings
001	24	M	2
002	26	M	3
003	49	M	1
004	28	M	1
005	34	M	1
006	28	M	1

Table 4.2: Age, gender and number of recording for each participant in the SNT dataset

Structure and annotation

Enabling the use of new data recordings to train the meta classifier, recordings needed to be labeled. The labeling process consisted of looking at the timestamps written down during the recording process in the spreadsheet (see table 4.3), and comparing them to temperature and acceleration changes visualized in OMGUI (Open Lab, 2018) to get exact timestamp when a sensor is taken attached or detached. The spreadsheet is then converted into a JSON-file (Crockford, 2019) manually by identifying the transitions and then writing the sensor configurations interval into JSON format. JSON files are created for each recording and later used to label the data in the pre-processing for training the meta classifier. Figure 4.2.2 shows how a annotation file for one recording looks like. All collected data is labeled with four different labels, 1 (for $both$ sensors), 2 (for $thigh$ sensor only), 3 (for $back$ sensor only) and 4 (for $none$ sensors). The keys are the $label\ value$, and each key has an array as value. Each element in the array, is representing a sensor configuration with date, start- and stop times when the different sensors are being used or not.

```
{
    "1": [
      [
        "2019-03-19",
        "19:11:00",
        "19:58:00"
      ],
      [
        "2019-03-19",
        "20:06:00",
        "20:37:00"
      ],
      [
        "2019-03-19",
        "20:58:00",
        "21:27:00"
      ],
      [
        "2019-03-19",
        "21:34:00",
        "21:47:00"
      ],
      [
        "2019-03-19",
        "21:55:00",
        "22:00:00"
      ]
    ],
    "2": [
      []
    ],
    "3": [
      []
    ],
    "4": [
      [
        "2019-03-19",
        "19:58:00",
        "20:06:00"
      ],
      [
        "2019-03-19",
        "20:37:00",
        "20:58:00"
      ],
      [
        "2019-03-19",
        "21:27:00",
        "21:34:00"
      ],
      [
        "2019-03-19",
        "21:47:00",
        "21:55:00"
      ]
    ]
}
```

Listing 4.1: JSON annotation for subject 4 recording 1

Time	Action	Label Name	Label Value
14:15	Both on	All	1
19:59	Thigh off	Back	3
22:21	Back off	None	4
23:19	Back on	Back	3
23:20	Thigh on	Both	1
11:40	Back off	Thigh	2
19:35	Back on	Both	1
21:58	Back off	Thigh	2
22:20	Back on	Both	1
22:21	Thigh off	Back	3
00:53	Thigh on	Both	1
13:23	Back off	Thigh	2
13:26	Thigh off	None	4

Table 4.3: Recording 001.2's annotation spreadsheet

Chapter 5

Pipeline

This chapter starts by shortly describing and introduce the existing HAR system the thesis is based on. Next is the description of the implementation and architecture of the proposed pipeline developed for this thesis and research experiments. Then each step in the pipeline architecture is elaborated.

5.1 Existing HAR System

The pipeline that is created for this thesis is built on the code base that is utilized for the HUNT study, made by Håkon Måløy and Sverre Herland at NTNU. Initially, Måløy made the both sensors LSTM model that is used for activity classification, while Herland extended the model with a pipeline.

The code exists of two parts, a graphical web interface for data processing and the activity classification. The interface is meant to be a user friendly interface for end-users to start and monitor classification jobs for individual subjects.

Furthermore the code base has a pipeline, which can unzip a directory, and creates a time-synchronized $.csv$ file containing the sensor data streams from both sensors. The synchronized file is passed on to the activity classification model, trained on both sensors. The solution has only one classification model that assumes data from the two sensors has valid data.

The system architecture and infrastructure is very complex and uses a lot of custom classes and parser objects to instanciate and use the pipeline. It requires a modern Nvidia-GPU (Nvidia, 2019a) that is compatible with Tensorflow-GPU (Tensorflow, 2019) and Keras (Keras, 2019) Python packages. The entire project is containerized with Docker (Docker, 2019), and since the project is GPU dependent and Docker is running on a Linux distribution, Docker is wrapped in NVIDIA-docker (Nvidia, 2019b) software to enable docker access to the GPU.

5.1.1 The Original Pipeline Architecture

Training a model

There is no code in the extended pipeline to train a model, but there are implemented methods for processing data in the pipeline allowing for easy implementation of the LSTM model's training function. If trying to use the function before over writing the abstract HAR-LSTM class training function, an error (Not implemented) is raised. The model used by the exiting HAR system was developed by Håkon and trained in a separate system to an accuracy of 76.5%, and then the model configuration and weights was stored. Herland reproduced the model and reused the stored weights and then extended the model with a pipeline for starting activity classification on HUNT4 subjects and not training.

Inference

Before any activity classification can be performed, the user has to submit a new job through the graphical user interface. The submitted job must be a $.7z$ file with a sub directory with the same name as the root folder, without the $.7z$ suffix. Inside the sub directory, two cwa files must be present, one for the back sensor, where the file names must include "$\_B$", and one for the thigh sensor, where the file name must contain "$\_T$". This submission creates a database entry in an SQLite database, with a job id, and a status. A daemon thread regularly checks the database for new entries and if a new job is found, the daemon spawns a new child process to actually do the inference job. The new process, unzips the $.7z$ file and synchronizes the two cwa files based on timestamps and creates a new time synchronized file csv file with acceleration information from both sensors. When the data is done processing, an LSTM model is instantiated based on a configuration file and weights selected trough the user interface when submitting the job. The model then runs classification on the input data (the synchronized file), and saves the activity classification results in a $.csv$. For each recorded sample, it stores a the timestamp and the classified activity. Finally when the daemon thread knows that its child process is done, it marks the job as "completed" if the activity classification was successful. The job is marked as "error" if something went wrong during the child process execution. At last, the unzipped directory is deleted to save storage space.

5.2 The Proposed Pipeline Architecture

As mentioned previously, the pipeline that is developed for this thesis is built on the existing code base described in section 5.1. All of the interface and actual pipeline code was excluded. Thus the code used as a code base, is the AI model and helper-functions.

The new pipeline is designed in such a way that users only have to use pipeline functions, as the pipeline handles everything else in the background. Data processing steps like unzipping and time synchronization are done in the pipeline and can be changed through function arguments. With regards to sensor synchronization, an additional step has been implemented where temperature readings are added to the time synchronized file. Temperature readings are also added as features since they are suspected to help detect sensor no-wear time. Training different models and running classification can also be done

trough the pipeline. The pipeline does actually work as an ensemble classifier, section 3.1.2 describes how an ensemble classifier works, as RFC is one of many different ensemble classifiers. The developed solution is a new type of ensemble classifier where the voting is not on what the result should be, but rather what position specific models to use for classification.

Figure 3.4 on page 26 shows how a normal ensemble classifier conducts the voting, and figure 5.1 show how the pipeline works as an ensemble classifier. In the figure the red arrows illustrate an example traversal through classification for a segmented feature window. Greyed out objects are not used for that window's activity classification as output is only one activity from the voted position specific model.

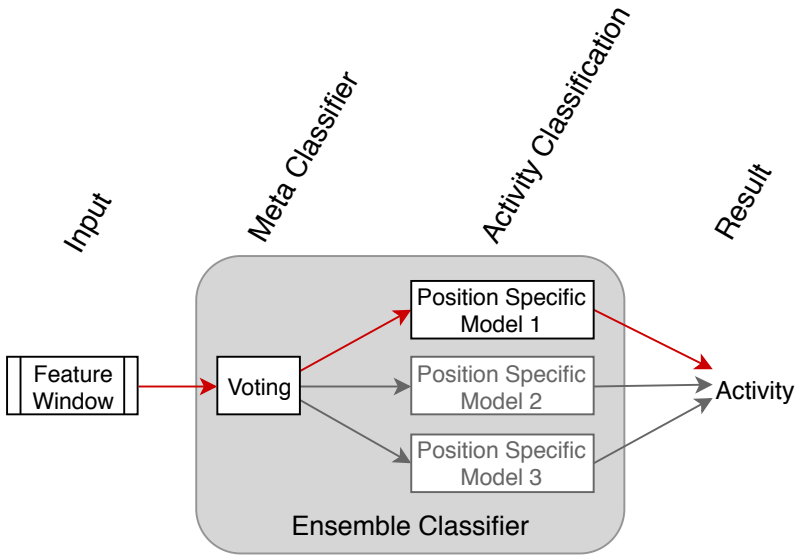


Figure 5.1: Illustration of how the pipeline works as an Ensemble classifier

Thus the voting is changed from "the majority of specialists with different specialty fields are saying x" to "whom among our specialists are the specialist on the current input, thus we should listen to that specialist". The ensemble classifier are consisting of a meta classifier that does the voting on specialists for the actual activity classification. Sensor no-wear time is classified by the meta classifier in parallel over five second windows, that are passed on to the corresponding position specific models, as described in section 3.2.2. The new pipeline utilizes $.yml$ configuration files to configure the activity classification models. Supporting multiple data-formats is important, so the pipeline supports the use of both $.7z$ files with the same hierarchy as the previous solutions and regular directories containing $.csv$ files.

5.2.1 Pipeline Flow

The following steps are how to use the pipeline for training and activity classification. Pipeline function methods are marked with (P) and script statements are marked with (S)

1. (S) : Initialize a Pipeline object.
2. (P) : Unzip .7z directories using the pipeline function ”unzip_multiple_directories”, by specifying a list with file paths. After unzipping, a time synchronized file is also created and saved in the same unzipped folder, if there already exists a unzipped folder with the same name at the specified unzip location the previous folder is overwritten.
3. (P) : The next step is to create training data from the data in the previous process using the pipeline function ”create_large_dataframe_from_multiple_input_directories”. In this function, the data processing described in section 5.2.2 is performed and labels are added. The function can either return a list of dataframes or one single dataframe. When training the meta classifier the function returns one large dataframe containing data from all the unzipped folders. For training the different LSTM models, the function returns a list of dataframes, one for each unzipped folder.
4. (S) : The script must now define some variables for the meta classifier features. If the given model is a LSTM, path for the configuration file must be defined.
5. (P) : Next step is to extract the different features from the data so it can be passed to the wanted model function. Extract the different sensor features, back- and thigh accelerations, back- and thigh temperature and labels into different variables, using the pipeline function ”get_features_and_labels_as_np_array”.
6. (P / S) : After defining configurations one can use the pipelines method to instantiate the wanted model, or one can instantiate them by importing the models module. The authors recommend using the pipeline functions, as it is the easiest way to instantiate a model by calling the pipeline function ”train_*_model” (* rfc or lstm). This instantiate and trains a model.
7. (P) When the model are done training, the pipeline has a built-in function for evaluation, classifying and even saving the model.
8. (P) : Optional, use the built-in function ”remove_files_or_dirs_from” to clean up the working environment.

5.2.2 Data Processing

All data recordings that goes through the pipeline for activity classification is located in $.7z$ folders as cwa files, one for back and one for thigh. This data has to go through different data processing steps before any activity recognition can be performed. Figure 5.2 shows each step in the data processing and what files that are in use. Step one is unzipping the $.7z$ file to make the cwa files available. A time synchronized csv file containing time

and acceleration information from both back and thigh is also made by running a time synchronization script made by Dan Jackson (Jackson, 2016) on the two cwa files. As this script does not support temperature extraction, each cwa file is also separately run through Axivity's cwa converter to make new csv files containing temperature. To have all information in one place, step three merges the temperature recordings from each sensor csv file with the time synchronized csv file, based on timestamps and creates a new pandas dataframe that is used for further processing. A dataframe is a tabular pandas data structure with labeled axes that can be thought of as a dictionary-like container, allowing indexing and selection using character- and string types. The synchronized dataframe does now include all the information that is needed for the meta classifier to determine if sensors data streams are faulty or not, and activity classification for the LSTM models. In the merging process, a lot of temperature reading are lost, as a result of timestamps in the sensor csv file does not match with the timestamps in the synchronized file. A matching timestamp occur approximately every third second. Hence in the next step, missing temperatures are generated by assuming that they are equal to the next valid temperature in the synchronized dataframe. This assumption is valid as there is only a few seconds apart between each valid temperature recording and temperature does not change that rapidly. After this process, one $.txt$ file for each senors is generated, containing temperatures equivalent to number of recordings in the synchronized dataframe. In the last step, the two $.txt$ files and the synchronized dataframe is concatenated together resulting in a synchronized dataframe with no missing data.

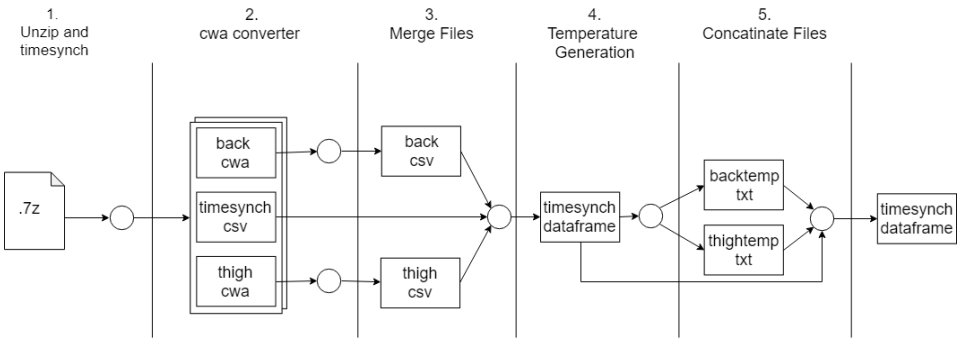


Figure 5.2: Data processing steps

Data Resampling

For the LSTM models, the training data are recorded at $100hz$. Before any training can be done, this data has to be downsampled to $50hz$ since the sensor data that is used for classification are recorded at $50hz$. The downsampling method is created by Måløy and Herland who have tested different downsampling techniques to find the best suited for the HUNT4 dataset. The core function of the downsampling is "scipy.signal.resample" from the Python package "SciPy" (SciPy.org, 2019). The re-sampling is utilizing Fourier transformations which assumes that the signal is periodic and re-samples the signal by applying a spacing; $len(x)/num * (spacingofx)$ on the data signal data

A problem with the re-sampling is that the data is being manipulated. As show in figure 5.3 with black circles, acceleration values from the raw data has either been increased or reduced in the re-sampled data. Grey lines indicate differences in peaks. For example if acceleration values increase at low intensity activities like walking, they become more similar to activities with higher intensity like running. Thus it will be harder for the LSTM models to differentiate between them.

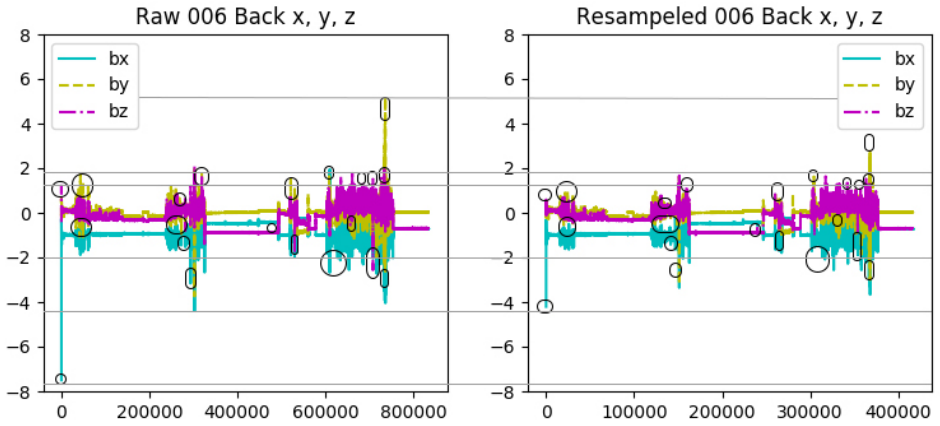


Figure 5.3: Differences between raw and resampled data stream for back sensor

5.2.3 Feature Engineering

Window Segmentation

Segmenting the sensor data stream for the meta classifier means to change the data input data from a continuous stream with $(nsamples, nfeatures)$. The data needs to be re-shaped into $(batch\ size, sequence\ length, num\ features)$, but as RFC does not support that data input format, instead we need to convert the data into $(num\ samples, 1, summary\ of\ records)$ which is the same as $(num\ samples, extracted\ features)$. To replace the $sequence\ length$ and $num\ features$ we need to extract features that represents the $sequence\ length$ of samples for a window, into summary features. The extracted features are described below in section 5.2.3, and for a more in depth explanation of the data format during window segmentation, see Appendix A.1.

Further when segmenting the sensor data stream for the the different LSTM models, the data stream can be reshaped from the input format $(nsamples, nfeatures)$ to the format $(batch\ size, sequence\ length, num\ features)$. The code for segmenting one sensor stream can be found in listing 5.2.3, and here one can notice that there is no feature extraction. In the LSTM windows, the features are x, y ,z. For a more in-depth explanation of the LSTM data segmentation, see Appendix A.2.

```
import numpy as np
import pandas.DataFrame as df
'''
```

```
Example df is
[timestamp (as index), bx, by, bz, tx, ty, tz, btemp, ttemp, label]
'''
    dataframes = [df(Dataset1),.., df(DatasetN)]
sequence_length = 250
batch_size = 512


X = np.concatenate([
        df[columns].values[ :
            (len(df) - len(df) % sequence_length)
        ]
    for df in dataframes])
        .reshape( -1, sequence_length, len(columns))

if stateful:
  X = X[ : (len(X) - len(X) % batch_size) ]
```

Listing 5.1: Pseudocode for LSTM sensor stream window segmentation

Feature Extraction

Since the meta classifier is a RFC, it can not process sequential data segmented into batches referred to as windows, with a given number of records/samples representing the window. The RFC must have one record/sample representing the entire feature window. For the temperature features wanted, this is not a problem since each window contains only five seconds with data. As the temperature does not change much over a short period of time, it is enough to extract few features to give a good summary of what happens with the temperature within a window. Additionally to temperature features, a distance feature is also extracted. Features that are extracted for the RFC are listed table 5.1.

Name	Definition	Description
Max	$Max(x)$	Highest value in the window
Min	$Min(x)$	Lowest value in the window
Max-Min-Delta	$Max(x) - Min(x)$	Difference between the highest and lowest value in the window
First-Last-Delta	$Last(x) - First(x)$	Difference between the first and last value in the window
Distance Moved	$\frac{1}{2} * avgAcceleration * (\delta t)^2$ avgAcceleration for the given axis δt is the window size	Acceleration distance moved, one feature for each axis
Temperature Memory	-	Memory of the last 10 minutes of Max-Min and First-Last calculations

Table 5.1: Features extracted for meta classifier

Max and Min temperature features help differentiate if the temperature readings in a window is stable or not. Figure 5.4 shows temperature readings from when a sensor is lying on a table with very small differences of temperature. When both max and min are almost equal we can argue that the sensor are either attached or detached during the entire window.

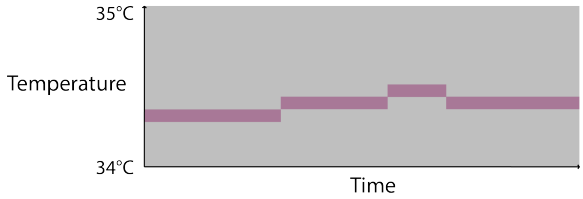


Figure 5.4: Window of temperature readings from a sensor that lies on a table

Max-Min-Delta and First-Last-Delta temperature feature calculates the difference between the max and min temperature value and the first and the last temperature value in the window. These two features will also help to differentiate if sensors are attached or detached during a given window. If the difference in both features are low, the sensor is either on or off the subjects body, but when the difference is high we can argue that the sensor is taken off or on during the given window as the temperature will raise or fall an the values will either be higher or lower than the other (as shown in figure 5.5)

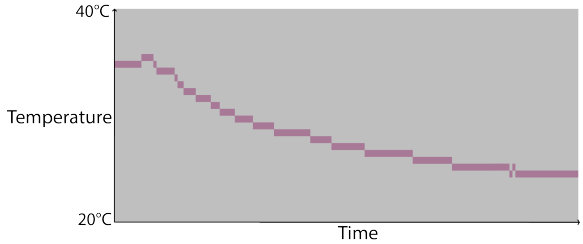


Figure 5.5: Window of temperature readings from a sensor at the point when it is taken off

Even though temperature features can give a good indication on sensors configurations, it is still not enough for the classifier to operate with good accuracy. The problem with only using temperature is that temperature changes in different climates. For example if a person goes outside during the winter sensors would record lower temperature readings as the sensor gets cold and the body has not managed to heat it up yet. During winter, the back sensor would most likely record higher temperature than the thigh sensors, as it is isolated from the cold by a winter coat. Sensor readings in cold temperatures can be as low as when the sensor are taken off and it would therefore be hard to classify if the sensor is on the body. The same issue occur if the sensor is taken off but lies on a table in a very hot room, as temperature readings can be close to body temperature.

Distance moved, where the acceleration for a window is converted into distance moved and added as a feature to the meta classifier to cope with this issue. The sensors are most

likely to lie totally still if the calculated distance is approximately zero. If this is combined with low and stable temperature readings, the classifier will identify that the sensor is detached. If the sensor is on the body, the calculated distance will be higher as people usually never stands or sit completely still. Combining this with high and stable temperature readings, the classifier will identify that the sensor is on a persons body.

A drawback with using distance moved as a feature could be if the sensor is placed in a backpack or other places where it can move around while not being attached to a persons body. If a person is walking with a backpack, the sensor would be moving in all directions and giving wrongfully accelerations. If the sensor movement ends where it started in the window, distance moved is equal to zero. Hopefully the distance moved in correlation with temperature recordings, makes the meta classifier smart enough to classify the different sensor configuration.

Temperature memory is added as a feature to help the meta classifier get context information about the window and the idea is also to make the meta classifier able to remember if the temperature is rising or falling over a given time. Giving more context information to meta classifier by introducing knowledge about previous windows, it is desirable that it can make better decisions about the state of the sensor. If the temperature drops over a longer period without any acceleration, its easier to state that the sensor is taken off. For the meta classifier to be able to know the temperature behaviour of previous windows, a temperature memory object is integrated. For each window, based on the temperature memory, a memory_max_min_delta- and memory_first_last_delta is calculated based on the last 10 minutes of temperatures and added as a feature.

A drawback of introducing memory to the meta classifier, is that if the training data contains wrongfully manipulated or malicious features, it will at some point be remembered and negatively influence the meta classifier. Since the memory can be quite big, remembering 10 minutes worth of windows, means that the samples can affect the meta classifier over time.

5.2.4 Classification

After data processing the classification process is started by running a pipeline function called ”parallel_pipeline_classification_run”. To run the function some function arguments are needed to be defined in advance; path to saved meta classifier, a dictionary object where the key is a string representation of the both sensors (1), thigh sensor (2) and back sensor (3). Each key contains another dictionary with the keys ”config”, ”saved_model” and ”weights”, where the values are paths to the corresponding configuration files, saved model or weights file. An example is show in listing 5.2.4. The function runs the entire classification pipeline from extracting the correct features for the meta classifier as well as the activity classifiers.

```
lstm_models_path = {
    "1": {
        "config": "../params/config.yml",
        "saved_model": "trained_models/both_model.h5",
        "weights": "trained_models/both_sensor_weights.h5"
    },
```

```
    "2": {
        "config": "../params/thigh_sensor_config.yml",
        "saved_model": "trained_models/thigh_model.h5",
        "weights": "trained_models/thigh_sensor_weights.h5"
    },
    "3": {
        "config": "../params/back_sensor_config.yml",
        "saved_model": "trained_models/back_model.h5",
        "weights": "trained_models/back_sensors_weights.h5"
    }
}
```

Listing 5.2: Dictionary containing meta information for LSTM models

The following are the steps in the pipeline function flow;

1. Segment the data into the specified window size. Then extract the temperature and distance features necessary for the meta classifier
2. For each window, create a new task for the meta classifier and add the task to a queue.
3. Create the number of specified meta classifiers to work in parallel to classify the jobs in the meta classifier queue.
4. Meta classifiers classify the windows using the features, then adds the window to a position specific classifier queue. If the window was classified as no sensor, it is directly added to the result without any activity classification.
5. When all the meta classifiers are done and there are no window-tasks left to classify, the next step is activity classification. Now the functions extracts the features needed for activity classification, segmenting them into the same windows as the meta classifier was classifying.
6. Then in order, all the windows classified as ”both” sensors are passed to a LSTM model, trained on data where both sensors where attached to the subject. Then all the ”thigh” windows are classified by a LSTM model trained with only thigh sensor data. Finally all ”back” windows are classified by a LSTM model trained with only back sensor data.
7. All the activity classification results and windows detected as faulty are gathered into one dataframe, and sorted on the starting timestamp for each window.
8. The final step is to minimize the result to avoid allocating unnecessary storage memory. To archive this, a simple compression solution is implemented. For all sequential windows with the same target class, instead of saving each timestamp in the window with corresponding target or saving the entire window. Only store the start timestamp of the first record in the first window as the start time and the end timestamp of the last record in the last window as end time. The target class, the position specific model used for classification and average confidence is also stored along with the start and end time. Then save the result as preferred file format. To decompress the result, generate new rows from the start- to the end time of a compressed window with the frequencies the original data was recorded in. Add the generated start and end time, confidence and the target for each row.

Chapter 6

Experiment

Chapter six presents the experiment environment and each experiment with setup and results. The experiments are executed in the order of the pipeline execution, starting with sensor no-wear-time detection, improving the overall accuracy for activity classification, comparing position specific models and finally memory efficiency.

6.1 Runtime Environment

All the experiments are executed on a server named Samuel01, hosted by the faculty of information technology and electrical engineering, Department of Computer Science (NTNU, 2019) at NTNU. Docker is used to create an isolated and reproducible working environment and NVIDIA-DOCKER enables the Docker container to use the host's GPU. And the 7z program was also installed in the container.

Experiments are executed on Linux, but can be executed on Windows and Mac as well Hardware and software information about Samuel is listed section 6.1.1 and 6.1.2. This information is given as a way to make our experiment and research reproducible. All the python-packages used are listed in a file called "requirements.txt", located in the GitHub (GitHub, Inc, 2019) repository [1] for this thesis, along with the code and datasets used.

For the experiments to run on another type of operating system, there are a few changes that need to be made. The timesynch script, omconverter and cwa_converter need to change the building commands from $make$ to $build$-scripts. For more information about detailed information about this, see Jackson (Jackson, 2016). Further in order to make the unzipping functionality work, a program that allows unzipping and interaction trough the terminal is needed. To enable the functionality change the line 32 in "src/utills/zip_utils.py" to work with the program's cli.

[1] https://github.com/skaugvoll/master_project

6.1.1 Hardware

Server hardware information for Samuel01 is listed below.

- CPU: Intel(R) Core(TM) i7-5930K @ 3.50GHz, 64 bit.
- GPU: Nvidia GeForce GTX TITAN X, @ 33MHz, 12GB RAM.
- Hardware architecture : x86_64
- Processor: x86_64
- Hardware platform: x86_64
- Memory: 62GiB System memory
- Storage: C610/X99 series chipset 6-port SATA Controller [AHCI mode]

Model	Size	Name
WDC WD4000FYYZ-0	3,7T	sda
WDC WD4000FYYZ-0	3,7T	sdb
INTEL SSDSC2BA40	372,6G	sdc
WDC WD101KFBX-68	9,1T	sdd

Table 6.1: Secondary storage information

6.1.2 Software

Software information about the host server Samuel01 is listed below.

- OS: GNU/Linux
- Kernal: #164-Ubuntu SMP Tue Oct 2 17:16:02 UTC 2018
- NVIDIA-DOCKER
- Docker

The docker container software is listed below

- Docker image: tensorflow:tensorflow:latest-gpu
- 7-Zip v9.20 developed by Igor Pavlov (Pavlov, 2010)

6.2 Improve Accuracy by Extending the Features of the Meta Classifier

Reinsve (Reinsve, 2018) did a feasibility study on the use of a RFC for identifying faulty sensor data-streams caused by sensors detached from a participant's body during activity recording. In his study, the RFC struggled to differentiate if both sensors are attached or

not, as it only considers temperature data. This experiment aims to improve the accuracy by introducing temperature memory and distance moved as features to the proposed meta classifier to help it differentiate between class 1 (Both sensors attached) or 4 (None sensors attached)

6.2.1 Setup

The proposed meta classifier is using the same Random Forest Classifier from the sklearn python package (scikit learn, 2019) as Reinsve (Reinsve, 2018) used in his feasibility study and is now classifying class 4 (None sensors attached). For this section, distance moved and temperature memory will be referred to as movement and memory respectively. First, leave-one-out training will be performed on the same data as Reinsve (Reinsve, 2018) used in his feasibility study. The data is pre-processed through the proposed pipeline and segmented into windows. Afterwards, leave-one-out training will also be performed on the entire new SNT dataset and a small subset of it. Each training will be performed four times with different feature configurations; without (wo) movement and memory, with (w) movement and without temperature memory, without movement and with memory and with both movement and memory. Additionally, leave-one-out classification will be performed again, with temperature memory and distance moved, on each participant to calculate Precision, Recall and F_1 score for each label individually. All results will be presented in tables in the next section, and the highest achieved performance measures are highlighted in bold. Configuration for the meta classifier are presented below.

- $Numberoftrees$: 100
- $Classes$:
  - 1 or A = Both Sensors attached
  - 2 or T = Thigh Sensors attached
  - 3 or B = Back Sensors attached
  - 4 or N = None Sensors attached
- $Features$: (see table 5.1)
  - temperature-max
  - temperature-min
  - difference in temperature max and min
  - difference in temperature first and last reading
  - distance moved calculated
  - max_min_delta and first_last_delta temperature for the temperatures held in memory
- $Class\ weight$: "Balanced"
- $train\ overlap$: 0.8
- $Batch\ size$: 512
- $Sequence\ length$: 250
- $Input\ data$: [001.1, 001.2, 002.1, 002.2, 002.3, 003, 004, 005, 006]

6.2.2 Results

Table 6.2 shows the average accuracy from two rounds of leave-one-out training, the first does not utilize movement and memory used as features, but the second does. The dataset used for training is the SNT dataset from Reinsve's (Reinsve, 2018) feasibility study. Precision, Specificity, Recall and F_1 score for the leave-one-out training with movement and memory is presented in table 6.3

Training datasets	Testing dataset	wo/movement, wo/memory	w/movement, w/memory
P1_V, P2_A, P2_V	P1_A	**0.917**	0.836
P1_A, P2_A, P2_V	P1_V	0.801	0.654
P1_A, P1_V, P2_V	P2_A	0.873	**0.988**
P1_A, P1_V, P2_A	P2_V	0.915	0.965
Average		0.876	0.861

Table 6.2: Meta classifier accuracy with and without the new features with the same data as Reinsve

Labels	Precision	Specificity	Recall	F_1 score
1 (All)	0.769	0.812	0.981	0.841
2 (Thigh)	0.988	0.997	0.990	0.989
3 (Back)	0.720	0.660	0.660	0.682
4 (None)	0.0	0.0	0.0	0.0

Table 6.3: Meta classifier average Precision, Specificity, Recall and F_1 score with the same data as Reinsve

Table 6.4 and 6.5 shows average accuracy from four different runs of leave-one-out training with different feature configurations for movement and memory. This is performed on a subset of the new SNT dataset in table 6.4 (approximately 18 778 052 rows of training data), and on the entire new SNT dataset in table 6.5 (approximately 52 477 924 rows of training data).

Training datasets	Testing dataset	wo/movement, wo/memory	w/movement, wo/memory	wo/movement, w/memory	w/movement, w/memory
002.1, 002.2, 003, 004	001.1	0.339	0.771	0.832	0.785
001.1, 002.2, 003, 004	002.1	**0.903**	0.938	0.774	0.956
001.1, 002.1, 003, 004	002.2	0.900	0.465	**0.949**	0.783
001.1, 002.1, 002.2, 004	003	0.296	0.956	0.365	0.948
001.1, 002.1 002.2, 003	004	0.842	**0.985**	0.910	**0.990**
Average		0.656	0.823	0.766	0.892

Table 6.4: Meta classifier accuracy for different feature configuration, with a subset of the SNT dataset

Trainingsets	Testingset	wo/movement, wo/memory	w/movement, wo/memory	wo/movement, w/memory	w/movement, w/memory
001.2, 002.1, 002.2, 002.3, 003, 004, 005, 006	001.1	0.942	**0.990**	0.947	**0.991**
001.1, 002.1, 002.2, 002.3, 003, 004, 005, 006	001.2	0.959	0.973	**0.991**	0.981
001.1, 001.2, 002.2, 002.3, 003, 004, 005, 006	002.1	0.938	0.947	0.939	0.947
001.1, 001.2, 002.1, 002.3, 003, 004, 005, 006	002.2	0.971	0.934	0.958	0.934
001.1, 001.2, 002.1, 002.2, 003, 004, 005, 006	002.3	**0.989**	**0.990**	0.989	0.988
001.1, 001.2, 002.1, 002.2, 002.3, 004, 005, 006	003	0.492	0.893	0.561	0.928
001.1, 001.2, 002.1, 002.2, 002.3, 003, 005, 006	004	0.919	**0.990**	0.952	0.979
001.1, 001.2, 002.1, 002.2, 002.3, 003, 004, 006	005	0.907	0.988	0.953	0.987
001.1, 001.2, 002.1, 002.2, 002.3, 003, 004, 005	006	0.949	0.977	0.968	0.990
Average		0.896	0.965	0.918	0.969

Table 6.5: Meta classifier accuracy for different feature configurations, with data from the entire SNT dataset

Table 6.6 shows Precision, Recall, and F_1 score for label 1 (Both), 2 (Thigh), 3 (Back) and 4 (None) together with accuracy when performing leave-one-out training on the new SNT dataset with movement and memory as features. Table 6.7 show the average Precision, Recall and F_1 score for each label.

Training datasets	Testing datasets	Accuracy	Label	Precision	Recall	F_1 score
001.2, 002.1, 002.2, 002.3, 003, 004, 005, 006	001.1	0.989	1 2 3 4	0.989 0.940 **1.0** 0.993	0.995 0.958 0.867 0.999	0.992 0.948 0.929 0.996
001.1, 002.1, 002.2, 002.3, 003, 004, 005, 006	001.2	0.984	1 2 3 4	0.962 **0.999** 0.999 0.999	0.999 0.845 **0.997** **1.0**	0.982 0.920 **0.998** **0.999**
001.1, 001.2, 002.2, 002.3, 003, 004, 005, 006	002.1	0.983	1 2 3 4	0.922 **0.999** **1.0** 0.999	**1.0** 0.897 0.738 0.999	0.960 0.895 0.850 **0.999**

001.1, 001.2, 002.1, 002.3, 003, 004, 005, 006	002.2	0.934	1 2 3 4	0.822 0.995 0.667 **1.0**	0.999 **0.994** 0.002 0.999	0.902 **0.994** 0.004 **0.999**
001.1, 001.2, 002.1, 002.2, 003, 004, 005, 006	002.3	**0.996**	1 2 3 4	0.990 0.981 **1.0** 0.998	0.997 0.974 0.972 0.999	0.993 0.977 0.986 **0.999**
001.1, 001.2, 002.1, 002.2, 002.3, 004, 005, 006	003	0.896	1 2 3 4	0.966 0 0 0.750	0.893 0 0 0.905	0.928 0 0 0.821
001.1, 001.2, 002.1, 002.2, 002.3, 003, 005, 006	004	0.989	1 2 3 4	0.991 0 0 0.997	0.999 0 0 0.852	**0.995** 0 0 0.919
001.1, 001.2, 002.1, 002.2, 002.3, 003, 004, 006	005	0.988	1 2 3 4	0.990 0 0 0.993	0.988 0 0 0.988	0.989 0 0 0.990
001.1, 001.2, 002.1, 002.2, 002.3, 003, 004, 005	006	0.988	1 2 3 4	**0.993** 0 0 0.958	0.997 0 0 0.816	**0.995** 0 0 0.881
Average		0.972				

Table 6.6: Meta classifier results from leave one out training, avg. accuracy: 97.2%

Labels	Precision	Recall	F_1 score
1	0.950	0.985	0.965
2	0.549	0.528	0.538
3	0.499	0.368	0.394
4	0.966	0.953	0.958

Table 6.7: Meta classifier average Precision, Recall and F_1 score for table 6.6

6.3 Utilizing Ensemble Classifier to Improve Accuracy of Activity Classification

This experiment aims to investigate if utilizing an ensemble classifier will increase accuracy of activity classification. In order to be able to measure accuracy, an annotated dataset is needed with information about which sensors are attached or not and which activities are being performed. A dataset like this does not exists for the HUNT study, as the concept of utilizing a meta classifier for detecting no sensor wear time, then using different models for the actual activity classification has not been tested before. This was also discovered in the SLR that there has not been conducted any research on this topic. Creating such a

dataset is also very time consuming and expensive, thus there has not been created such a datatset for this thesis.

As the necessary annotated dataset does not exist, this thesis will therefore prove that the pipeline described in chapter 5 works as a proof of concept and is able to classify when sensors are attached or not and then perform activity classification when sensors are attached. It is expected that many of the HUNT4 activity recordings contain data where one or more sensors are detached at some point during recording.

Sensors are expected to be detached from the body during activity recording, as participants take them off before a shower, they fall off or they just chooses to detach the sensor them self. If sensor no-wear time is not recognized, the classification will most like classify the data as lying or sitting, which will cause a lot of misclassified data

Figure 6.1 shows the activity classification results with the existing HAR system for subject 4000181 where each row represents one day. Almost half way through day four, the classification is predicting that the subject is lying, represented as the color blue. Since all of the other days are colored blue, it is suspected the the sensors are taken off and is lying still on a table, thus the classification is wrong.

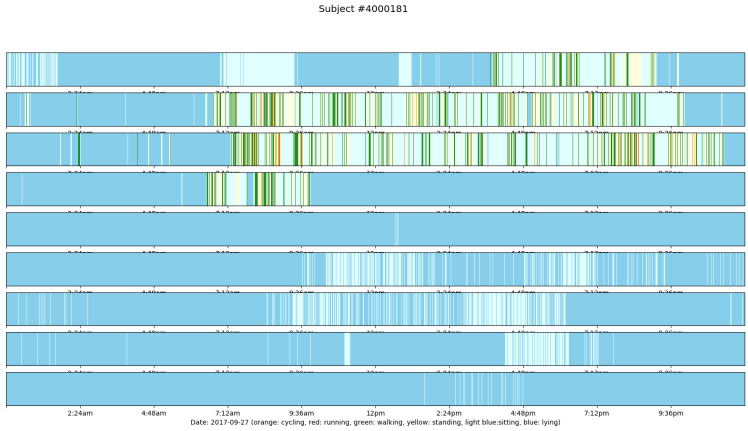


Figure 6.1: Subject suspected to have sensor no-wear time

6.3.1 Setup

In order to show the the pipeline works as expected, different HUNT4 subjects where it is expected that sensors are detached during activity recording, are executed through the pipeline. The results of these experiments will be presented in the same way as figure 6.1, where different colors represents different activities, but with additional colors, purple and white, that represents no-wear time and no data respectively. Two of the subjects will be presented in the results, while Appendix D, shows the the results for all the subjects.

6.3.2 Results

Figure 6.2 and 6.3 show activity classification results after running subject 4000181 and 4003601 through the pipeline. Both classifications are presented in forms of a daily overview chart, where each row starts at midnight, and spans through an entire day consisting of 24 hours.

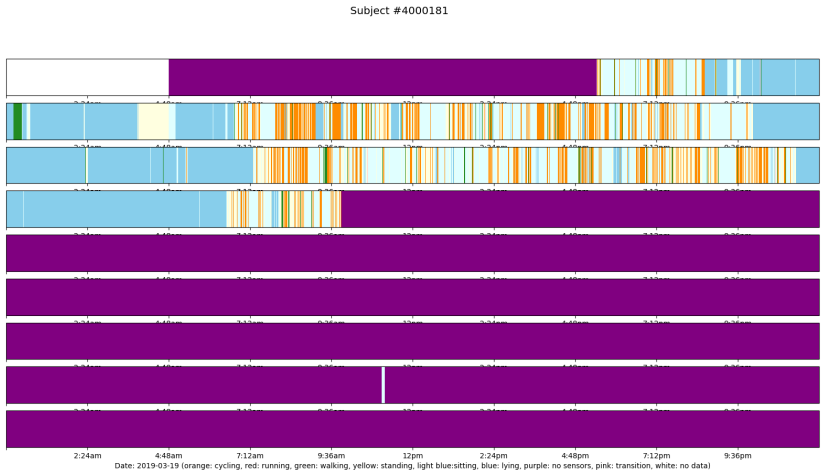


Figure 6.2: Activity classification for subject 4000181 that is suspected to have sensor no-wear time

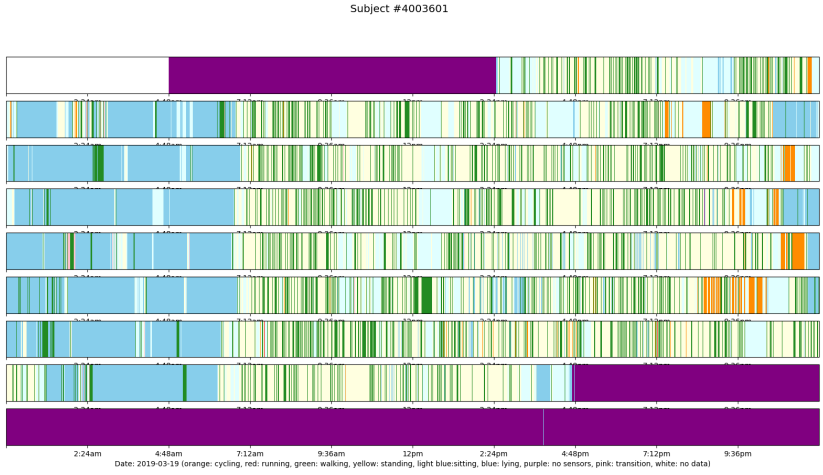


Figure 6.3: Activity classification for subject 4003601 that is suspected to have sensor no-wear time

6.4 Comparison of Individual Sensor Models Against a Combined Sensor Model

After the meta classifier is finished classifying sensor configurations, the data is sent to different position specific LSTM models for activity classification. Using two sensors for activity recognition is more difficult to handle than one sensor, as synchronization is needed when using two. Additionally it is more stressful for the participants to attach two sensors rather than one and it can be uncomfortable to wear the sensors over time, especially the back sensor. Expenses in terms of money, computational power and memory allocation, are also doubled when using two sensors instead of one. Since using two sensors has some drawbacks, this experiment will do testing on how individual models for thigh and back perform with regards to accuracy compared to a combined model of both to determine which sensors are really necessary.

6.4.1 Setup

To be able to compare the different model that are used for activity classification, leave-one-out training will be performed on the subjects from the TFL dataset. As performing leave-one-out training is quite time consuming, different optimizers, with 32 and 19 layers with dropout of true and false respectively, will be tested to get an indication on how they perform. 32 units in the first hidden layer were chosen because the existing HAR system is using 32 units for activity classification. As there are 19 unique activities to classify, 19 units where chosen for the second hidden layer. Each leave-one-out training is executed with 40 epochs, as empirical testing did not show any noticeable increase in accuracy by increasing number of epochs, compared to the increase in execution runtime. The results, average- , best- and worst accuracy for each configuration is presented in the section below.

6.4.2 Results

Table 6.8, 6.9 and 6.10 shows results from leave-one-out training with different optimizers for both-, thigh- and back model respectively. The highest average accuracy achieved in each table is highlighted in bold.

Training data	Back and Thigh layers [units, dropout]	Optimizer	AVG Accuracy	Best Accuracy (recording)	Worst Accuracy (recording)
(006-022)	32T 19F	SGD	0.682	0.921 (012)	0.03 (009)
(006-022)	32T 19F	RMSprop	0.747	0.951 (012)	0.516 (010)
(006-022)	32T 19F	Adagrad	**0.851**	0.956 (012)	0.604 (019)
(006-022)	32T 19F	Adadelta	0.828	0.939 (012)	0.574 (010)
(006-022)	32T 19F	Adam	0.837	0.962 (012)	0.650 (019)

(006-022)	32T 19F	Adamax	0.829	0.936 (016)	0.601 (019)
(006-022)	32T 19F	Nadam	0.843	0.963 (012)	0.561 (019)

Table 6.8: Leave One Out training for both sensors LSTM model

Training data	Layers [units, dropout]	Optimizer	AVG Accuracy	Best Accuracy (recording)	Worst Accuracy (recording)
(006-022)	32T 19F	SGD	0.68	0.879 (012)	0.140 (013)
(006-022)	32T 19F	RMSprop	0.842	0.944 (016)	0.592 (019)
(006-022)	32T 19F	Adagrad	0.831	0.938 (016)	0.584 (019)
(006-022)	32T 19F	Adadelta	**0.845**	0.938 (016)	0.589 (019)
(006-022)	32T 19F	Adam	0.773	0.925 (016)	0.201(011)
(006-022)	32T 19F	Adamax	0.825	0.936 (016)	0.585 (019)
(006-022)	32T 19F	Nadam	0.822	0.938 (016)	0.581 (019)

Table 6.9: Leave One Out training for thigh sensor LSTM model

Training data	Layers [units, dropout]	Optimizer	AVG Accuracy	Best Accuracy (recording)	Worst Accuracy (recording)
(006-022)	32T 19F	SGD	0.532	0.852 (012)	0.037 (010)
(006-022)	32T 19F	RMSprop	0.640	0.850 (012)	0.234 (009)
(006-022)	32T 19F	Adagrad	0.585	0.829 (012)	0.040 (009)
(006-022)	32T 19F	Adadelta	**0.665**	0.875 (012)	0.404 (010)
(006-022)	32T 19F	Adam	0.640	0.825 (012)	0.390 (009)
(006-022)	32T 19F	Adamax	0.630	0.835 (012)	0.038 (010)
(006-022)	32T 19F	Nadam	0.680	0.868 (012)	0.439 (021)

Table 6.10: Leave One Out training for back sensor LSTM model

6.5 Minimizing File Size of Results

As mentioned in chapter 1, there have been recorded activity data from 38 756 people in the HUNT4 study that was completed in March 2019. When activity classification is performed on a recording, it is desirable to store the results of the classification for further analysis and research. Currently, the results of the classification is saved as $.csv$ file with an average size of approximately 2,5GB per file if each sensor reading is saved with the corresponding classified activity and the record timestamp for that reading. If classification is performed on all participant, it will require an huge amount of storage as shown in equation 6.1

$$\begin{aligned} \text{\# participants} &= 38\,756 \\ \text{Result file size} &= 2{,}5GB \\ Memoryneeded &= 38\,756 * 2{,}5GB \quad = 96\,890GB = 96{,}890TB \end{aligned} \tag{6.1}$$

The experiment builds on trying to find the file format that is best suited for minimizing the file size of the classification results. The file formats are csv, $Pickle$, $H5$ and $Feather$, as they are commonly used in the machine learning community and are often compared against each other in terms of size and speed. csv is intended for long term storage and stores results in plain text with commas separating each column. $Pickle$ is serializing object structures by converting (flattening) the object hierarchy into a byte stream, but is not secure against erroneous or maliciously constructed data $H5$ simplifies the file structure to only include "datasets", which are multidimensional arrays, and Groups, which are container structures that can hold datasets and other groups. $Feather$ does not use compression, but stores dataframes as binary text with the main goal of lowering writing and reading times, but is usually bound by the local disk performance. $Feather$ is also not recommended for long term storage, as the format is not promised to be stable between versions. Additionally a decompression method (described in step 8 in section 5.2.4) will also be experimented with.

6.5.1 Setup

This experiment is divided into three parts where four different file types will be tested. In the first part, a 2.6 GB file will be converted to the different file formats to see how they perform on large files. Then activity classification will be performed on two different datasets in the second part. The two datasets used will differ in size to show the different speeds and sizes for classification on a small and large dataset. Recording 003 is used as the small dataset, as it contains only one day of data Subject 4003601 from the HUNT4 study, is used as the large dataset, and contains activity data recorded over seven consecutive days. The datasets will be tested with and without the decompression method. When performing activity classification, the uncompressed result file is writing out the classification for each window (containing 250 recording), and not every recorded sample. Thus making it much smaller than if each recorded sample was stored with the classification as in the result csv files saved by the existing HAR system.

Results will be presented in a table with file size, storing time and reading time for each file format.

6.5.2 Results

	CSV	Pickle	H5	Feather
Size (KB)	2990889	**919789**	1200466	2964001
Writing Time (s)	175.119	**5.806**	7.728	7.894
Reading Time (s)	26.317	3.429	51.459	**2.242**

Table 6.11: File size, Write- and Read times when converting a 2.6 GB file

Small Dataset

The uncompressed result for subject 003 have 4 319 rows and the compressed have 166 rows. Compressing the result yields a reduction of 96.16% rows.

Uncompressed	CSV	Pickle	H5	Feather
Size (KB)	254	**170**	178	**170**
Writing Time (s)	0.024	**0.001**	0.062	0.208
Reading Time (s)	0.014	**0.002**	0.092	0.003

Table 6.12: File size, Write- and Read times after activity classification on subject 003 without compression

Compressed	CSV	Pickle	H5	Feather
Size (KB)	12	**8**	15	**8**
Writing Time (s)	0.002	**0.0009**	0.03	0.025
Reading Time (s)	0.004	**0.001**	0.091	0.002

Table 6.13: File size, Write- and Read times after activity classification on subject 003 with compression

Large Dataset

The uncompressed result for subject 4003601 have 149 040 rows and the compressed have 24 063 rows. Compressing the result yields a reduction of 83.85% rows.

Uncompressed	CSV	Pickle	H5	Feather
Size (KB)	9672	**5823**	5831	**5823**
Writing Time (s)	0.881	**0.007**	0.035	0.032
Reading Time (s)	0.193	**0.0038**	0.096	0.0043

Table 6.14: File size, Write- and Read times after activity classification on subject 4003601 without compression

Compressed	CSV	Pickle	H5	Feather
Size (KB)	1624	942	950	**941**
Writing Time (s)	0.146	**0.002**	0.032	0.026
Reading Time (s)	0.032	**0.002**	0.094	0.003

Table 6.15: File size, Write- and Read times after activity classification on subject 4003601 with compression

Chapter 7

Evaluation and Discussion

This chapter contains in-depth evaluations and discussions about the results presented in chapter 6.

7.1 Improved Accuracy by Extending the Features of the Meta Classifier

Both table 6.4 and 6.5 shows results from training the meta classifier, with a subset of the SNT dataset and the entire SNT dataset respectively with different sensor configurations. When analyzing the average accuracy for the different feature configurations, it reveals that, by adding the temperature memory and distance moved, the accuracy of the model increases. With the new features an increase of 23.6% in average accuracy is achieved when training on a small subset of the SNT dataset (see equation 7.1), while there is and increase of 7.3% accuracy when training on the entire dataset (see equation 7.2).

$$Increase = 0.892 - 0.656 = 0.236 = 23.6\% \tag{7.1}$$

$$Increase = 0.969 - 0.896 = 0.073 = 7.3\% \tag{7.2}$$

The more data the less relevant the features become, as the meta classifier manages to achieve 89.6% average accuracy when training on the entire dataset without the new features, but while training on the small subset with the new features, the accuracy drop to 89.2% on average.

The features are still helping the model learn, but the effect of the features are reduced to an accuracy increase of 7.3 % when giving the model 33 699 872 more rows of data to learn from, when migrating from table 6.4 to 6.5 and introducing new subjects, hence more rows of data is added. The increase in accuracy per row is defined in equation 7.3, and overall average accuracy achieved is 97.2% (see table 6.6), compared to Reinsve's (Reinsve, 2018) SNT classifier that achieved 95.6%. Thus proving research question 2.1.

$$
\begin{aligned}
&\text{Tot. Rows of Subjects Data In Table 6.5} = 52\,477\,924 \\
&\text{Tot. Rows of Subjects Data In Table 6.4} = 29\,889\,063 \\
&\text{Avg. Acc With Dist. Moved and Temp. Memory In Table 6.5} = 0{,}969 \\
&\text{Avg. Acc With Dist. Moved and Temp. Memory In Table 6.4} = 0{,}892 \\
&slope = (\text{\# Rows Table 6.5} - \text{\# Rows Table 6.4}) \\
&\qquad - (\text{Avg Acc Table 6.5} - \text{Avg Acc Table 6.4}) \\
&slope = (52\,477\,924 - 29\,889\,063)/(0{,}969 - 0{,}892) \\
&slope = 2{,}284\,875\,147\,300\,261\,3 \cdot 10^{-09} \\
&slope \approx 2{,}3 \cdot 10^{-09}\%
\end{aligned}
\tag{7.3}
$$

In table 7.1, Reinsve's (Reinsve, 2018) performance measures are presented, while the result from running the SNT dataset from his feasibility study through the proposed meta classifier are presented in table 6.3 on page 54. The meta classifier utilizing the new features show reduction in Precision, Recall, Specificity and F_1score for labels 1 (Both) and 3 (Back), presented in table 7.2. As Reinsve (Reinsve, 2018) did not managed to classify label 4 (None) in his feasibility study, it is therefor not present in table 7.1 and 7.2. The reason for lower performance measures is because the new meta classifier achieves low accuracy when testing on P1 datasets that does not contain any recorded data where only back or thigh sensors is attached to the subject. This is presented in table 6.2 on page 54, where both of the P2 datasets achieves an accuracy above 95% while P1 has lower than 85%. This result raises suspicion that there might be an error in the code for calculating the performance measure. The theory is that if dataset does not contain all the different sensor configurations, the performance measures are still calculated, but the score becomes zero for each performance measure where the sensor configuration is not present in the dataset. Thus appending the score of zero to the numerator and still counting the dataset in the denominator when calculating the average, thus incorrectly affecting the calculations. As the model is not performing bad since there are no such labels to calculate for the given dataset, the calculations affects the scores reported negatively.

Labels	Precision	Specificity	Recall	F_1 score
All(A)	0.954	0.964	0.954	0.954
Thigh(T)	0.952	0.984	0.952	0.952
Back(B)	0.960	0.983	0.961	0.961

Table 7.1: Precision, specificity, recall and F_1 score achieved by Reinsve

Label	Precision	Specificity	Recall	F_1 Score
A (1)	-0.185	-0.134	$+0.027$	-0.113
T (2)	$+0.036$	$+0.013$	$+0.038$	$+0.037$
B (3)	-0.24	-0.323	-0.301	-0.279

Table 7.2: Differences in performance measure between Reinsve and the meta classifier when training on Reinsve's SNT dataset

In section 3.2.2, hypothesis 2 states that adding features regarding previous windows and distance moved, the overall Precision, Recall and F_1 score is going to increase. Based on the performance measure results in table 6.7, the meta classifier achieves high results for labels 1 (Both) and 4 (None), but low on 2 (Thigh) and 3 (Back) when performing leave-one-out training on the entire new SNT dataset with the new features. This is proving hypothesis 2 wrong.

While analyzing the results from table 6.6 a few outliers is found. The lowest precision and F_1-score is achieved with 75% and 82.1% respectively when testing with recording 003. Additionally, recording 006 gives the lowest recall with 81.6% and recording 002.2 affects the overall performance measures with precision at 66.7%, recall at 0.002% and an F_1 score of 0.004% for label $3(Back)$. These findings could indicate that there is something wrong with the dataset, and should maybe be discarded from the comparison, but since the dataset scores high on the other performance metric, it is kept in the comparison. The other suspected main reason for the poor performance is as stated above, that the calculations of performance measures are wrong for precision, recall and F_1 score. Given the assumption that it is affecting the performance measures, looking at the different subjects data, it was clear that subjects $003, 004, 005$ and 006 does not have any $2 - Thigh$ and $3 - Back$ data, and could therefore affect the calculations as they report a precision, recall and F_1 score of 0 on those labels. Based on these findings, a new round of leave-one-out training was performed without recording $002.2, 003, 004, 005$ and 006. An average accuracy of 98.9% was achieved and the performance measures are presented in table 7.3.

The same behaviour appeared when performing leave-one-out training with data from Reinsve's (Reinsve, 2018) feasibility study, where P1 datasets reduced the performance measures of the meta classifier. Thus strongly supporting the theory of wrong performance calculations as shown in table 6.7, and not that the datasets are outliers, for precision, recall and F_1 score. When the datasets classified for sensor no-wear time are a mix of datasets using all possible sensor configurations and some datasets are not, the calculations are hurting the average calculation. Thus it would be wise to correct the performance measure calculations, so they handle the mixed sensor configuration datasets, and then confirm or contradict the theory. Thus hypothesis 4 is created for future work.

Hypothesis 4. The calculation of performance measures are being wrongfully punished, if at least one of the input datasets contains fewer sensor configurations than the other input datasets. For each missing sensor configurations, the performance measures for that configuration are given a performance score of 0, which is added to the nominator and the denominator is incremented by one, thus the average performance calculated is wrongfully punished.

The code used for evaluating the performance measures is presented below in listing 7.1, From the 'scikit-learn' python package, the function "classification_report" is used to calculate the performance measures. The function does in turn use a function called 'precision_recall_fscore_support' that might be the source for the problem, as it is suspected to not support the diversity of classes in the different datasets used for training, creating the problem stated in hypothesis 4.

A theory for the occurring problem is that the classifier has been trained on a dataset containing more sensor configurations than the dataset that is being tested. As the trained dataset have seen a configuration that is not in the test dataset, it is able to misclassify as

that configuration. The function raises an error as the classification contains values that are not in ground truth, thus combining the classification and the ground truth, and using it as input to the function. Hence the calculation problem as it sets sensor configurations that are not in ground truth to zero and utilizing them in the calculations.

```
# preds = prediction from RFC
# gt = Ground truth from the annotated data

# only use labels present in the data
        labels = []
        label_values = list(set(gt)) + list(set(preds))
        label_values = list(set(label_values))
        label_values.sort()

        for i in label_values:
            labels.append(target_names[i])

        report = classification_report(
            gt,
            preds,
            target_names=labels,
            output_dict=True
        )

        acc = accuracy_score(gt, preds)
```

Listing 7.1: Calcualation of performance measures

Label	Precision	Recall	F_1 Score
A (1)	0.967	0.991	0.978
T (2)	0.990	0.911	0.945
B (3)	0.932	0.956	0.941
N (4)	0.999	0.996	0.997

Table 7.3: Performance measures for the meta classifier with a reduced dataset (excluding 002.2, 003 , 004 , 005, 006)

Label	Precision	Recall	F_1 Score
A (1)	$+0.017$	$+0.006$	$+0.013$
T (2)	$+0.441$	$+0.383$	$+0.407$
B (3)	$+0.433$	$+0.588$	$+0.547$
N (4)	$+0.033$	$+0.043$	$+0.039$

Table 7.4: Comparison between performance measures from table 6.7 and 7.3

Table 7.4 presents the differences in performance measure between the entire SNT dataset and without the possible outliers. When analyzing the table, each performance measures increases with at least 38.3% for both 2(thigh) and 3(back) when excluding the outliers from the dataset. Having more balanced data by eliminating outliers for the new SNT dataset that does not have data for thigh and back, is actually proving hypothesis 2 correct as the performance measures increases, if hypothesis 4 is correct.

Label	Precision	Recall	F_1 Score
A (1)	$+0.013$	$+0.037$	$+0.024$
T (2)	$+0.038$	-0.041	-0.007
B (3)	-0.028	-0.005	-0.02
Total:	$+0.023$	-0.009	-0.003

Table 7.5: Comparison between Reinsve's RFC performance measures and the meta classifier with a reduced dataset (excluding 002.2, 003 , 004 , 005, 006)

Table 7.5 shows the differences in performance measures between Reinsve (Reinsve, 2018) and the performance measures achieved in table 7.3. In precision there in an increase of 2.3%, while there is an slight decrease in recall and F_1 Score. One of the main reasons for the differences in performance is in fact that the two different RFCs are trained on different datasets, which could explain why the meta classifier performs worse than the RFC developed by Reinsve (Reinsve, 2018). The new SNT dataset consists of out of lab data with more rapid transitions between sensor data streams and more changes in temperature, while the dataset from Reinsve's (Reinsve, 2018) feasibility study contains in-lab recordings. Further, the new SNT dataset consists of multiple days of recordings, versus recordings from the feasibility study which only last a couple of hours. Therefore the authors of this thesis states that the new gathered data are harder to learn and generalize, as there are more variations and unique factors affecting the recorded data and features.

7.2 Ensemble Classifier to Improve Accuracy of Activity Classification

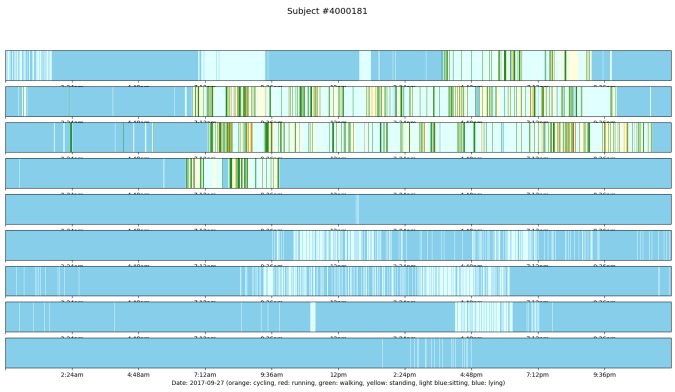
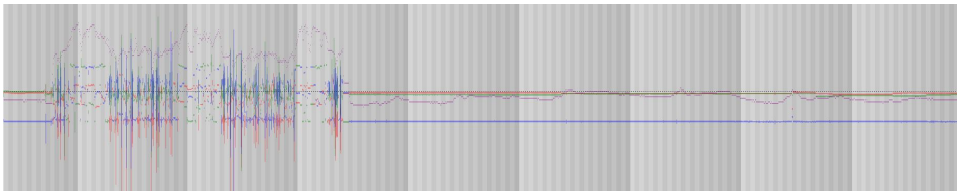


Figure 7.1: Activity classification for subject 4000181 from the existing HR system

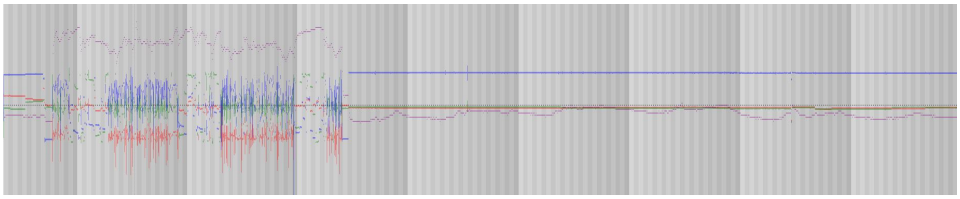
Figure 6.2 on page 58 is showing activity classification results from the proposed ensemble classifier. When analyzing and comparing the results against classification results

from the existing HAR system in figure 7.1. it becomes clear that the meta classifier predicts that both the sensors are detached from the subject before 5-6am on row one and after 9:36am, row four, counting from the top. Subject 4000181 is therefore identified as wrongly classified in these time periods, as there are only two activities classified by the existing classifier, lying and sitting. The activity lying is probably accurate between the late and early hours, but since entire days are classified as lying and sitting without any walking it gives an indication that sensors were detached from the subject. By analyzing the raw sensor streams in figures 7.2a and 7.2b, both streams start with almost no acceleration in either x, y or z direction and temperature readings is clearly showing temperatures below normal readings when wearing a sensor. Similar readings can be found later in the sensor stream. Thus, one can infer that both sensors are detached from the subject, as the meta classifier classified. Because there is no changes in acceleration and both sensors is aligned similar to when the thigh sensor is aligned when a person is sitting or lying, it is hard to differentiate between the two activities. This is also the reason why these are the only activities that are suspected misclassified.

The new ensemble classifier is classifying sensor no-wear time, and the existing classifier is classifying the same time periods as lying or sitting, since the sensor is lying still and no-wear time is not taken into consideration. If multiple days of misclassified lying and sitting is used in public health research, it will give a wrong impression of how much time people are lying and sitting during a week. It is therefore really important to classify correctly when the sensor is not attached to the body as this lead to more valid contribution to public health research.



(a) Raw thigh sensor stream

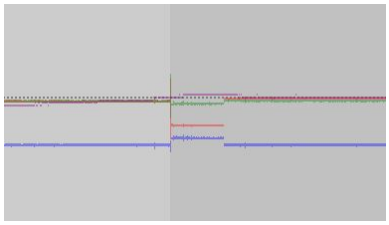


(b) Raw back sensor stream

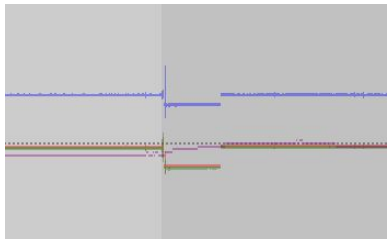
Figure 7.2: Subject 4000181 raw sensor streams for thigh and back sensor. Red X, Green - Y, Blue - Z, Purple - Temperature

When analyzing figure 6.2 on page 58 further on row eight (from the top), the new ensemble classifier is classifying a small interval as sitting. It is suspected that during that point of time, the sensor has been moved and therefore have some changes in acceleration and temperature. This is confirmed when looking at snippets from the raw sensor data,

shown in figure 7.3a and 7.3b. During this change a hand is most likely moving the sensor. The temperature change is a result of the subjects hand heating the sensor up when touching it, and the acceleration change in all directions is caused by the movement.



(a) Snippet of raw thigh sensor stream on row 8



(b) Snippet of raw back sensor stream on row 8

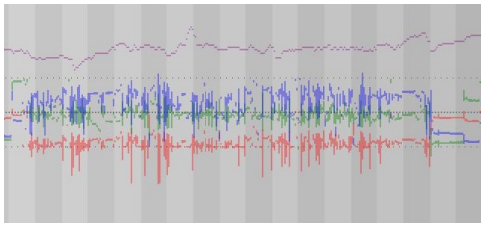
Figure 7.3: Snippets of subject 4000181 raw sensor stream on row 8.

Even though the changes is rather small, the acceleration changes in combination with a raising temperature affecting the new features introduced to the meta classifier and causing it to predict that the sensors are re-attached to the subject. When the subject stopped touching the sensor, it quickly stabilizes and the meta classifier is back to correctly classifying sensor no-wear time.

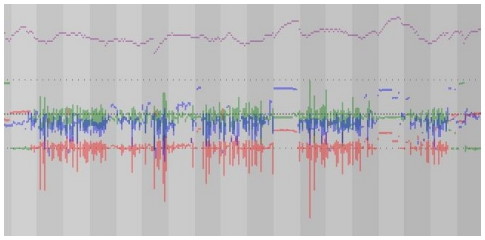
This is one of the drawbacks of using small windows for training and classification, as there might not be enough samples in the window to cancel out such small changes and unwanted movement of the sensors. This indicates that using bigger windows for training and classification for the meta classifier could improve the classification.

When analyzing the activity classification, where sensor no-wear time is not classified by the meta classifier, there is clear difference on activities that have been classified. The most obvious differences is that the ensemble classifier, classifies a lot more of cycling instead of walking for subject 400181. By looking at figure 6.3 and 6.2, subject 4003601 has been classified with a lot of walking each day while wearing the sensors, which indicates that the classification models have learned to classify walking. When analyzing raw sensors stream snippets for both subjects in figure 7.4, there is a clear indication that the back sensor for subject 400181 is moving a lot more on the z-axis (blue) than subject 4003601. This could either mean that the subject is actually cycling and moving the back more frequently, or walking in a specific pattern the classification model have not seen before. When comparing subject 4000181 against actual cycling recordings in figure 7.5, it indicates that the x-, y- and z-axis is moving similarly to acceleration when cycling. As the sensor stream is less intense, meaning lower frequency, and showing the same type of peaks as cycling, we can assume that subject 4000181 is walking and the ensemble classifier is misclassifying walking as cycling. The occurring misclassification between cycling and walking gives an indication that the training data contains little data about cycling and is probably overfitting on one way of cycling, as there are many different types of bikes which results in different body postures as well as different ways to cycle. To avoid these misclassifications in the future, more data where subjects are cycling on different types of bikes and different cycling styles, should be recorded. This should also be done for

walking and other activities that are being classified as one another.



(a) Snippet of raw back sensor stream for subject 4000181



(b) Snippet of raw back sensor stream for subject 4003601

Figure 7.4: Snippets of raw back sensor stream for subject 4000181 and 4003601

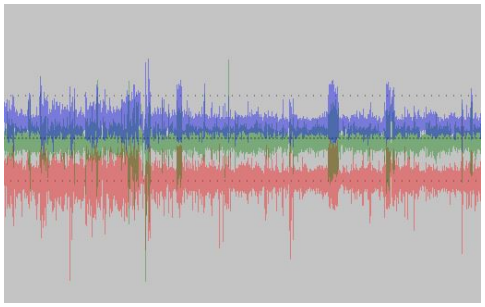


Figure 7.5: Snippet of raw back sensor stream for training subject with cycling

As there is no annotation of what kind of activities the subject are doing while wearing the sensors, the classification results from both classifiers are not reliable to use as an indication of ground truth. Activity classification comparisons should therefore not be analyzed to much. Identifying and analyzing the sensor no-wear time classifications is more comparable, as the raw sensor data is much more definite and easier to understand for humans, thus allowing us to be reasonably sure about sensor no-wear time for the subjects.

Analyzing the classification results from the existing classifier up against the results from the new ensemble classifier and the raw sensor streams, it is clear that the new ensemble classifier is improving the accuracy of HAR systems. The most valuable part of

the ensemble method for improving HAR systems, is the meta classifier as it eliminates a lot of misclassifications that occur when sensors are detached from the subjects body. This is proven in figure 6.2, as the meta classifier is eliminating approximately six days of misclassification. If individual models used for activity recognition in the ensemble method are well trained and scoring high on performance measures, there should be no doubt that the overall accuracy for HAR systems will be improved compared to a HAR system that is always expecting reliant input data. Such data streams are almost impossible to capture, since there are many factors involved when generating data. When using more than one sensor, more things can go wrong and corrupt a model that expects valid data, therefore it is important to have a HAR system that is capable to cope with these issues.

Since no one has researched and developed a ensemble classifier as the one proposed in this thesis where using a RFC for sensor no-wear time detection and incorporated in into a HAR system, it is hard to compare the proposed ensemble classifier to any of the existing state of the art classifiers. The big differences are the new meta classifier that detects sensors no-wear time, and the use of individually trained position specific models. No features for activity classification are extracted, manipulated or combined to try and improve the accuracy, as most of the relating state of the art papers solution propose. Seeing that the proposed solution is at the time of writing, the only one in its category, it is hard to reason, infer and compare what type of models the individual sensor specific models should be. State of the art does achieve good accuracy and clearly states that further experimentation and research should be done on the use of LSTM-RNN for activity classification, and thus no other types of machine learning models are experimented with.

7.3 Comparing Individual and Combined Sensor Models

Based on the results from the experiments in section 6.4.2, table 6.10 shows that the back model with Adadelta as optimizer achieves an average accuracy of 66.5% and is the model with the worst accuracy compared to the thigh- and both model that achieves average accuracy above 84% As suspected, one of the reasons for the low accuracy for the back sensor is that it is very difficult for it to differentiate between activities like sitting and standing. The reason for this is visualized in figure 7.6, where the back sensors is aligned in the same positions when performing both activities. This is also reflected in the confusion matrix in figure 7.7, where standing is misclassified as sitting 205 times and is the most misclassified activity. The same situation can occur for cycling, as the the sensors is approximately in the same position when a person is cycling as when the person is sitting still and walking. Additionally, it is even harder to differentiate between cycling and sitting when movement from the thigh sensor is not taken into consideration when classifying with the back sensor. The misclassifications that occur when classifying with the back sensor only affects the average accuracy considerably, as it is achieves approximately 20% lower accuracy than the model for both sensors combined.

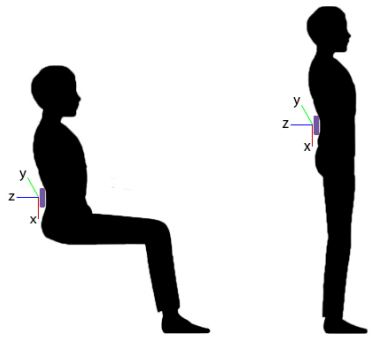


Figure 7.6: Sitting and Standing sensor axis for back sensor

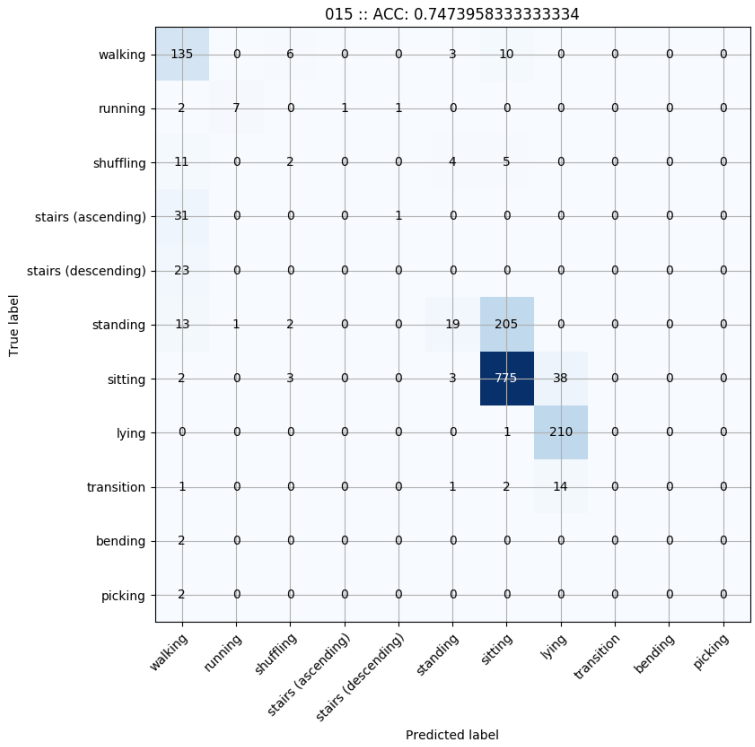


Figure 7.7: Confusion matrix back model for training subject 015

Table 6.9 shows that the thigh sensor achieves an average accuracy of 84.5% with Adadelta as optimizer, which is very similar to the model for both sensors that achieves an average accuracy of 85.1% with Adagrad, as shown in table 6.8. Even though the thigh sensors has an accuracy above 80%, it has some drawbacks which is similar to the back

sensor. The thigh sensor has some trouble to differentiate between sitting and lying, as sensor alignment is the same for both activities. This is visualized in figure 7.8. When analyzing the confusion matrix in figure 7.9 from leave-one-out training with the thigh model only, lying is misclassified as sitting 62 times and is the most frequent misclassification for thigh.

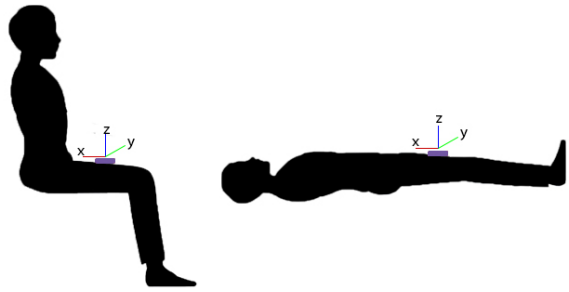


Figure 7.8: Sitting and Lying sensor axis for thigh sensor

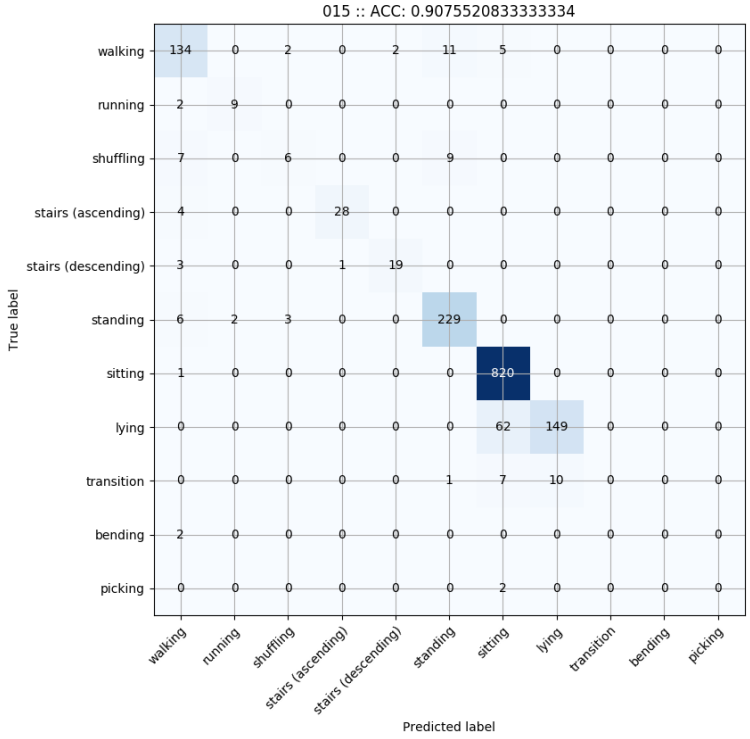


Figure 7.9: Confusion matrix thigh model for training subject 015

Statistics form the Norwegian Directorate of Health (Norwegian Directorate of Health, 2016) state that Norwegian adults are inactive 62% of the time they are awake. Since adults spend so much time being inactive, it is very important to classify static activities like sitting, standing and lying correctly to make valid contributions to public health research. As stated above, the back model alone does have some trouble differentiate between sitting and standing, and the thigh model has trouble with sitting and lying. To be able to differentiate between these activities, a model that combines acceleration signals from both sensors is necessary. The accuracy between the thigh- and both model is very similar, but the model for both is able to differentiate the static activities. This is confirmed in the confusion matrix in figure 7.10, where the misclassifications for lying, sitting and standing is now correctly classified.

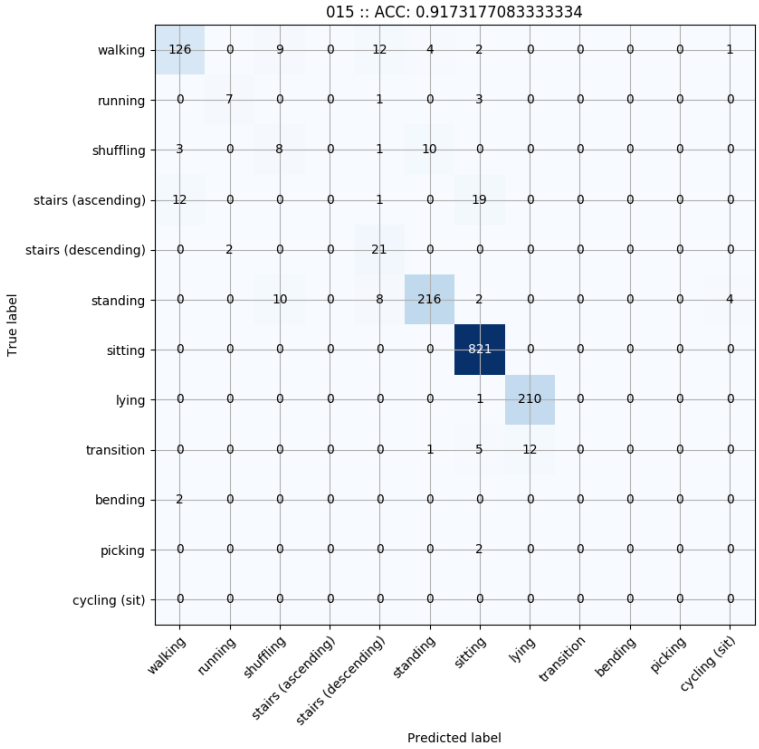


Figure 7.10: Confusion matrix both model for training subject 015

Further analyzation of the results from experiment in section 6.4.2, the LSTM models can achieve equally and higher accuracy than many of the state of the art HAR systems, as shown in the ”Best Accuracy”-column of the results tables. Even though the individual models can compete with the state of the art when looking at single testing instances, it does not mean that they are competing when looking at the average accuracy. The models are struggling to get above the 92% average accuracy mark, that seems to be the state of the art accuracy to beat, from the SLR research done in section 2.4. Most of the papers

included in the SLR do not state that what type of testing and evaluation method they use to get the accuracy reported, thus it is hard to say if the models in this thesis is better, worse or equally great. As it seems that state of the art is trying to combine spatial and temporal features to improve the accuracy, and using a combination of CNN and LSTM-RNN for the different feature types. It would be interesting to try and have the models used in the proposed ensemble classifier to have the same architecture as proposed by either Baccouche (Baccouche et al., 2011), Pigou (Pigou et al., 2015) or Song (Song et al., 2016), as all of them achieve accuracy above the state of the art accuracy mark previously mentioned.

In order to improve the accuracy for each individual model, it would be interesting to try and use the proposed "DNN bidirectional LSTM-RNN hidden layer swapping" solution presented by Hebbar (Hebbar, 2017) to see if it would help the back and thigh model to distinguish between activities. The solution has almost the same architecture base as the proposed ensemble classifier, as it uses position specific models. Hebbar's (Hebbar, 2017) bidirectional swapping technique in combination with the proposed "automatic feature extraction for deep learning" by Friday (Friday et al., 2018), would be interesting to see implemented in the new proposed ensemble classifier. As most state of the art HAR systems is researching feature extraction and suggest further experiments with LSTM-RNN models.

Furthermore Hebbar (Hebbar, 2017) and Stewart (Stewart et al., 2018) both studied on the use of multiple sensors and the efficacy of using two sensors relative to sensors individually. Both studies presents an overall improvement with the use of multiple sensors, and Stewart (Stewart et al., 2018) even states that using two sensors, one attached to the thigh and one to the lower back, achieved an accuracy higher than 98%. When using only individual models the accuracy drops with as much as 26% (72%). For this thesis, the back, thigh and both model achieved an average best accuracy of 66.5%, 84.5% and 85.1% respectively. This shows that the thigh model are inside the accuracy rate stated by Stewart (Stewart et al., 2018).

Alsheikh (Alsheikh et al., 2015) is the study from the SLR that stands out, as it uses HMM and RBM instead of LSTM-RNN and CNN, and reports 99.13% accuracy. Thus it would also be very interesting to see how the proposed ensemble classifier would perform if the combination of HMM and RBM are used. The advantages of using HMM and RBM is that they are more simple than RNNs and rely strongly on assumptions. There exists many assumptions that could prove useful for HAR, such as that in order to walk, the subject has to be in a standing position before hand. In order to lay down, the subject has to be in a standing or sitting position before hand, but not running as it is not possible to go from a running position to a full flat out lying position without jumping or falling, which can be added as activities. There are many other activities that rely on such assumptions and thus a order of activities could be formed to help the HMM better assume what the next activity could be.

Results presented in this section, answers research question 3.2, showing that the worst position specific model is the back model, with 20% less accuracy. The thigh model achieves almost as good accuracy (84.5%) as the model for both sensor (85.1%), thus indicating that using both sensors for classifications is still the best option, as the misclassifications can be reduced. The combined sensor model does also outperform the existing

HAR system (76.5%) with an increase of 8.6% accuracy, this does also apply for the thigh sensor which achieves 8.0% increase in accuracy. The suspected reason for the increase in accuracy is based on hyper-parameter tuning for the models, where the proposed models have an additional layer and better optimizer for the sensor data streams. It is suspected that further hyper-parameter tuning will increase the accuracy and performance measure even more, and seeing that the proposed system achieves better results than the existing HAR system with only using one sensor, meaning that the total cost and runtime of HAR systems could be reduced if a model is trained on a single sensor with high accuracy. Computational costs and pre-processing would be reduced by 50% if going from two sensors to one sensor, and hardware costs would be reduced as the amount of sensors is reduced. Memory allocation would also be reduces as there is less data to store. Individual models still perform well under the circumstances of static activities and achieves decent accuracies. The position specific models performs as suspected, and the performance measures are not surprising, as the confusion matrices for each individual model confirms that the models would misclassify sitting and standing for back and lying and sitting for thigh.

7.4 Minimizing File Size

The SLR conducted in chapter 2 reveled that there is done very little research on how to best store the results from HAR systems, and none of the articles and papers discussed how they store their results. Thus answering research goal 4, question 1, becomes quite simple as there is no standard or preferred way to store the classification results, and there is done little to none research on how it should be stored. When working on large datasets such as HUNT4, storing results becomes of great importance as equation 6.1 clearly shows. Therefor the results presented in section 6.5 is considered as great contribution to the HAR community.

Looking at the tables in section 6.5.2, one can clearly see that $Pickle$ has the best writing speed regardless of amounts of data. $Pickle$ has also the smallest file size when it comes to saving large amounts of data without compression, as seen in table 6.11, where $Pickle$ is able to compress 2.6 GB of data to 920 MB. $Feather$ is competing with $Pickle$ when saving small amounts of data, as seen in the tables for the small and large dataset. Analyzing the results further, $Pickle$ has clearly the best reading speed when reading small amounts of data, but is beaten by approximately one second by $Feather$ when the data grows larger. As research question 2 aims to find the best file format suited for saving the classification result with regards to memory allocation, choosing $Feather$ with compression for large HUNT4 datasets would be the best option, as it beats $Pickle$ in table 6.15, by 1 KB.

Equation 7.6 shows how using $Feather$ for HUNT4 subjects is 38 756 KB less in size, meaning that using $Feather$ insted of $Pickle$ saves nKB, where n is number of participants. If the dataset is small, then the $Pickle$ format is clearly the superior, as it has the same file size as $Feather$, but better writing and reading performance.

$$\begin{aligned} \text{\# participants} &= 38\,756 \\ \text{Pickle result file size} &= 942KB \\ Memory_{needed} &= 38\,756subjects * 942KB \\ &= 36\,508\,152KB = 0{,}036\,508\,152TB \end{aligned} \tag{7.4}$$

$$\begin{aligned} \text{\# participants} &= 38\,756 \\ \text{Feather result file size} &= 941KB \\ Memory_{needed} &= 38\,756 * 941KB \\ &= 36\,469\,396KB = 0{,}036\,469\,396TB \end{aligned} \tag{7.5}$$

$$\begin{aligned} \text{Pickle}Memory_{needed} &= 36\,508\,152KB \text{ (from eqn. 7.4)} \\ \text{Feather}Memory_{needed} &= 36\,469\,396KB \text{ (from eqn. 7.5)} \\ Memory_{saved} &= 36\,508\,152KB - 36\,469\,396KB \\ &= 38\,756KB = 0{,}000\,038\,756TB \end{aligned} \tag{7.6}$$

The equations below shows that choosing $Feather$ rather than $Pickle$ to get the best reduction in memory allocation (equal to saving $1KB$ for each HUNT4 subject), it costs 930.144 seconds (see equation 7.7) for writing and 38.756 seconds (see equation 7.8) for reading. Thus giving $Feather$ a writing speed of $6.709 * 10^{-07}$- and reading speed of $7.741 * 10^{-08}$ seconds per subject, compared to $Pickle$'s writing speed of $5.160 * 10^{-8}$ and reading speed of $5.160 * 10^{-8}$ seconds per subject, using equation 7.9

$$\begin{aligned} \text{\# participants} &= 38\,756 \\ \text{Pickle} &= 0{,}002s \\ \text{Feather} &= 0{,}026s \\ \text{Subject Cost} &= 0{,}026s - 0{,}002s = 0{,}024s \\ \text{Total Cost} &= 38\,756subjects * 0{,}024s \\ &= 930{,}144s = 15{,}502\,4min \end{aligned} \tag{7.7}$$

$$\begin{aligned} \text{\# participants} &= 38\,756 \\ \text{Pickle} &= 0{,}002s \\ \text{Feather} &= 0{,}003s \\ \text{Subject Cost} &= 0{,}003s - 0{,}002s = 0{,}001s \\ \text{Total Cost} &= 38\,756subjects * 0{,}001s \\ &= 38{,}756s = 0{,}645\,9min \end{aligned} \tag{7.8}$$

$$\text{Performance per subject} = \frac{performance}{\#subjects} \tag{7.9}$$

The reason for the improved performance when using both $Feather$ and $Pickle$ is that $Feather$ is created to store dataframes as binary text with the main goal of writing and reading as fast as possible and $Pickle$ is serializing objects by converting (flattening) the

object hierarchy into a byte stream. This results in smaller file sizes as csv stores results in plain text.

To answer the research question, the most memory efficient in terms of file size is $Feather$ when classifying datasets with the same format and size as HUNT4 data. The writing and reading speed performance is better than csv by far, but not the fastest as it takes 15.5 minutes longer to save all the subject as $Feather$ instead of $Pickle$. As $Feather$ needs 38.756 MB less than $Pickle$, it is the most memory efficient for storing the classification results, but yielding a probability of becoming file format version dependent. This results in a total reduction of 99.96% compared to the existing HAR system, as necessary storage space is reduced from 96.89 TB to 0.036469396 TB to store all HUNT4 participants using $Feather$ rather than csv.

Chapter 8

Conclusion and Future Work

This chapter gives a brief summary in the form of conclusion for the different experiments conducted and recaps the research goals, research questions and answers found. Afterwards suggested future work is presented and shortly discussed.

8.1 Conclusion

Four goals were defined in section 1.1 for this thesis. In goal 1, exploration of state of the art HAR systems regarding choices of deep learning models and system architecture was performed and presented in the SLR in chapter 2. Goal 2 aimed to create a brand new SNT dataset with real world data, and improve the sensor no-wear time accuracy from Reinsve's (Reinsve, 2018) feasibility study by introducing more features. Goal 3 was to improve the accuracy of the activity classification by developing an ensemble classifier of three LSTM models, one for back, one for thigh and one for both of them combined. Goal 4 aimed to improve how classification results are stored with regards to memory allocation by experimenting with different file formats.

The SLR conducted in chapter 2 answers both research question for the first research goal, as it identifies knowledge gaps for both questions. State of the art HAR systems do not iteratively choose between position specific models, therefore an ensemble classifier is created and experimented with, where input data is segmented into windows where each window is given to a position specific model chosen by the new meta classifier. Furthermore the SLR also revealed that there is no research and resources targeted at HAR system scalability and storage, which should be a separate research field as HAR systems processes huge amount of data.

For the developed meta classifier that aims to detect sensor no-wear time by identifying faulty or missing sensor streams, a whole new SNT dataset with real world data was created. The new dataset consist of recordings with many sensor transition and sensor configurations from the subjects and is used to train the developed meta classifier. Adding

distance moved and temperature memory as new features has proven to greatly improve the average accuracy of the meta classifier. The results presented in section 6.2 on page 52, shows that by adding the new features, the meta classifier achieves an average accuracy of 96.9% when training on the entire new SNT dataset and 89.2% with a small subset of the SNT dataset. Without the new features the meta classifier achieved 89.6% and 65.6% respectively, which is an increase of 7.3% for the entire dataset

In table 6.6, the meta classifier achieves an average accuracy 97.2%, which is an improvement compared to 95.6% that Reinsve's (Reinsve, 2018) achieved in his feasibility study with the original SNT dataset. When removing possible outliers from the dataset, where no back or thigh data is collected and the performance calculations are suspected to be correct, the meta classifier achieved and average accuracy of 98.4%.

An ensemble classifier was developed to improve the accuracy by introducing position specific LSTM models that is used for activity classification. The meta classifier iteratively choose which model used for activity classification based on which sensors record valid data and are attached to the subjects body. As the SLR revealed, there has not been done any research or experiments on this topic and an annotated dataset with information about which sensors are attached or not and which activities are being performed does not exist. As an annotated dataset is necessary to measure accuracy, this thesis proved that the developed ensemble classifier works as a proof of concept instead. Figure 6.2 shows the results after activity classification has been performed on subject 4000181. The results shows that the developed meta classifier is able to predict that six days of sensor no-wear time where the existing HAR system predicts lying and sitting, as sensor no-wear time is not take into consideration. This proves that the meta classifier is a big and important contribution to HAR research, as it improves accuracy by potentially eliminate days of misclassification.

Comparing the performance for the position specific models against the model for both combined. The results from the experiments corresponds with the initial thoughts that the back sensor model is struggling with differentiating between sitting and standing, and the thigh sensor model is struggling with lying and sitting. These activities is hard to differentiate as the sensor alignment is in the same position when they are performed, thus the position specific models does not achieving the highest accuracy. The experiments showed that the back model is the one with the lowest performance regarding accuracy. One of the reasons for this is that the thigh sensor data stream is more diverse, and the different activities has enough of difference in acceleration recorded while performing activities than the back sensor stream. The experiment has proven that the model for both sensors combined is the best, as it achieves the best accuracy and is able to differentiate between sitting, standing and lying.

Improving how the classification results are stored with regards to memory allocation consisted of experimenting with different file formats, as the SLR shows that there has been done no research on how state of the art HAR systems store or should store their results. Experiments in section 6.5 shows that $Feather$ and $Pickle$ are the best options when it comes to choosing file format for datasets with the same size as HUNT4. $Feather$ is the file format that stores the smallest result files, but it is not the fastest to write to disk or read from disk, as $Pickle$ is the fastest. The experiment also shows that utilizing a simple compression method, that concatenates sequential windows with the same classification, and only saves the starting time, end time, average confidence, activity classified and what

position specific model used for classification works better then the method used by the existing HAR system, as the result file size is smaller and stores more information. Saving classification results with the new compression method and $Feather$, the result file size for each participant is reduced to 941KB compared to 2.5GB if writing out each recorded time stamp and the corresponding classification to a CSV file. Hence resulting in a overall reduction of 99.96% memory allocation needed to save classification results for entire HUNT4 dataset.

In conclusion, all the goals defined in this thesis has been reached. There has been created a new SNT dataset and developed a new meta classifier with two new features, distance moved and temperature memory. With the these features the meta classifier is able to predict with high accuracy when sensors are attached to the subject or not during recording, thus proving hypothesis 2 and 3. As the new meta classifier predicts sensor no-wear time with high accuracy, it is an important contribution tho the HAR community as it is able to cope with faulty sensors and eliminate large amounts of misclassifications, and therefor making more valid contributions towards public health research. The developed ensemble classifier answers hypothesis 1 to the extend it is possible since there are no existing annotated dataset that is necessary to measure the performance, but together with meta classifier the results looks promising. The ensemble classifier should therefore be further experimented with to achieve best possible results for the position specific models. With regards to storing classification results, the compression method described in section 5.2.4 step 8 on page 50, together with the $Feather$ file format will aid in minimizing storage space that is needed for the HUNT dataset, as classification results for recording with seven days of data is compressed with the developed compression method to 941 KB. The knowledge gaps identified has therefore been closed as a results of the conducted experiments in this thesis and the conclusion above.

8.2 Future Work

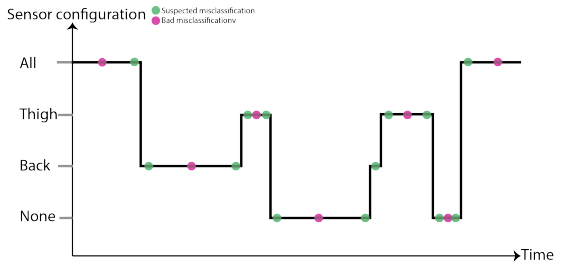
This section presents the authors thoughts about future work that was identified throughout the work on this thesis. The authors also proposes further research on the problem of overcoming faulty sensors without manipulating the raw data and rather eliminate the invalid features from the faulty sensor. When the invalid features are eliminated and passed to the position specific models in the ensemble classifier, further research is needed to achieve better activity classification performance.

8.2.1 Correct Calculation Error

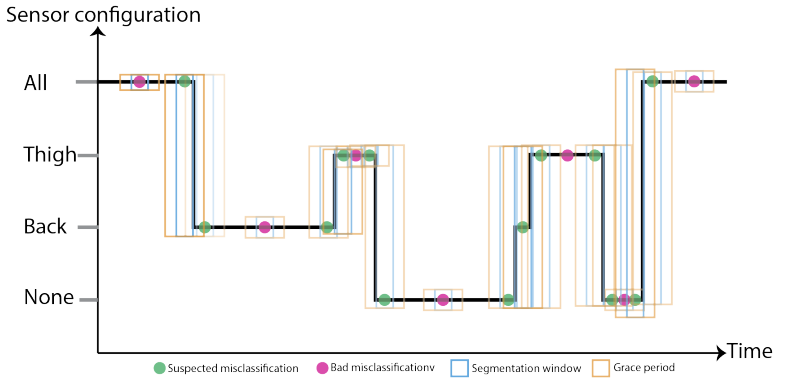
Hypothesis 4 states a suspicion that is raised when looking at the results from the experiments conducted for the meta classifier. It is suspected that the resulting performance measures are actually lower then what they are, and thus there should be dedicated some resources to correcting the calculation error. Then the experiment should be executed again to see if the results actually gets better, and confirms the hypothesis. If the results stays the same or drops in performance, it would surprise the authors, and further research on why they drop should be prioritized as the new proposed ensemble classifier with position specific models shows great potential.

8.2.2 Grace Period and Bigger Segmentation Windows

A Grace period is a window that does not count during training and classification, and should be experimented with. If the Grace period is set to a window of 60 seconds, and the classification window is representing 30 seconds of data, the Grace period allows the model to detect changes within a larger window without ”punishing” the accuracy. In context of HAR and the meta classifier, it is important to detect the sensor transitions and sensor configuration. The Grace period thus works as a 1 minute delay for the meta classifier to be more confident that the sensor was attached or detached. The reason the authors suspect grace period to help the meta classifier achieve higher accuracy is because the suspected reason for misclassifications happen directly around the sensor configuration changes. Figure 8.1a shows an illustration of when the suspected misclassification occurs (green) and the worst possible time it could happen(red), and figure 8.1b shows the effect of Grace period (orange). Thus utilizing Grace period is suspected to look at a larger frame than what it is actually classifying, thus removing the suspected misclassifications because the classifier actually sees more of what is happening than what it is classifying.



(a) Illustration of when misclassifications occur



(b) Illustration of with Grace period and without

Figure 8.1: Illustration of when the suspected misclassifications occur and potential Grace period

Implementation and evaluation of Grace period should take different window sizes into account, and in order to test the theory, one could raise the amount of records in each

window for the ensemble classifier from 5 seconds (250 records) to perhaps 60 seconds (3000 records) per window. Thus introducing the idea of bigger segmentation windows for the ensemble classifier to see if the misclassifications near the sensor configuration transitions decreases. The authors suspects that increasing the amount of data in a window will help neutralize misclassification due to short sensor movements of only a few seconds, as detached sensors is moved without being attached to the subject and is only held in the subjects hand. Thus if the classifiers sees more of the transition it can learn to classify the transition as the position specific sensor transition to, and the activity classifiers can learn that the activity to classify is the activity transitioned to. Figure 8.1b shows the effect of larger segmentation windows, when changing from small segmentation windows (blue), to larger (orange) if the Grace period is thought of as the larger segmentation windows.

8.2.3 Create Better Training Data for Activity Classification

If position specific models are going to be utilized for activity classification in the future, a better training dataset has to be created. The discussion in section 7.2 shows that the models are struggling to distinguish between walking and cycling, because there are little training data with cycling and they have similar peak-signal-signatures. Hence learning with more variations in the acceleration features is suspected to help the models become better at distinguishing the misclassified activities, as subjects perform the same activities with different techniques and postures. A clear example of how the supplementation of thigh and back sensor configuration is suspected to help is more data of cycling and walking. This highlights another aspect of the proposed ensemble classifier that needs to be further researched, improving the position specific models. One can try to incorporate the current state of the art HAR models into the ensemble classifier and even implement individual position specific feature manipulations to possibly achieve the same accuracy as state of the art models. for each position specific model. Another potential reason for misclassification is the resampling that is needed, because the activity training data is recorded at $100Hz$ and not $50Hz$ as the testing data. Appendix C presents acceleration signals for back and thigh sensors before and after resampling, using the resampling functionality in the proposed pipeline. The figures shows how the resampling manipulates the original raw sensor stream of acceleration data and reduces the peaks in terms of height. The results of the manipulation could have an effect on the sensor stream and reduces the variation in acceleration that distinguishes the activities that are most similar and therefore harder to classify. Such activities that might be affected are cycling, running, walking, lying and sitting. Therefor it could be interesting to capture new data that is recorded in $50Hz$, to see if the individual position specific models achieves better accuracy and that confusion matrices verifies that the resampling are affecting the model's accuracy.

8.2.4 Create Annotated Dataset

The authors would emphasize that the creation of a sensor configuration and activity annotated HUNT dataset would contribute to the HAR research community and ease the further research on ensemble classifier with position specific models for activity classification, which was proved to work and improve the overall HAR system accuracy. When

created, the annotated dataset should be tested on the ensemble classifier in order to measure the accuracy. Another advantage is that such a dataset makes it easier to experiment and try different position specific models and methods.

8.2.5 Features Manipulation

Additionally allowing for feature manipulation for each model, resulting in a cross over with the current state of the art and position specific models for activity classification, hopefully resulting in a new type of HAR systems that achieves better performance measures than state of the art. Hence future research should be conducted on finding the best sensor specific models and a good place to start would be with individual position specific model feature manipulation, as state of the art shows great results using feature manipulation. The authors would propose to start with testing CNN-LSTM, as it occurs rapidly in state of the art research and achieves good results.

Bibliography

Alsheikh, M. A., Selim, A., Niyato, D., Doyle, L., Lin, S., Tan, H.-P., nov 2015. Deep Activity Recognition Models with Triaxial Accelerometers. "arxiv.org".
URL http://arxiv.org/abs/1511.04664

Axivity, 2019. Axivity ax3 accelerometer. https://axivity.com/product/ax3, [Online; accessed 14-January-2019].

Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A., nov 2011. Sequential Deep Learning for Human Action Recognition. In: Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 29–39.
URL http://link.springer.com/10.1007/978-3-642-25446-8_4

Banos, O., Toth, M. A., Damas, M., Pomares, H., Rojas, I., jun 2014. Dealing with the effects of sensor displacement in wearable activity recognition. Sensors (Basel, Switzerland) 14 (6), 9995–10023.
URL http://www.ncbi.nlm.nih.gov/pubmed/24915181http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4118358

Bulling, A., Blanke, U., Schiele, B., jan 2014. A tutorial on human activity recognition using body-worn inertial sensors. ACM Computing Surveys 46 (3), 1–33.
URL http://dl.acm.org/citation.cfm?doid=2578702.2499621

Chen, W.-H., Betancourt Baca, C. A., Tou, C.-H., oct 2017. LSTM-RNNs combined with scene information for human activity recognition. In: 2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom). IEEE, pp. 1–6.
URL http://ieeexplore.ieee.org/document/8210846/

Crockford, D., 2019. Multimodal Multi-Stream Deep Learning for Egocentric Activity Recognition.
URL https://www.json.org/

Docker, 2019. Docker: Enterprise container platform for high-velocity innovation. https://www.docker.com/, [Online; accessed 2-May-2019].

Friday, N. H., Al-garadi, M. A., Mujtaba, G., Alo, U. R., Waqas, A., mar 2018. Deep learning fusion conceptual frameworks for complex human activity recognition using mobile and wearable sensors. In: 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET). IEEE, pp. 1–7.
URL https://ieeexplore.ieee.org/document/8346364/

GitHub, Inc, 2019. Github about.
URL https://github.com/about

Hebbar, A., nov 2017. Augmented intelligence: Enhancing human capabilities. In: 2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN). IEEE, pp. 251–254.
URL http://ieeexplore.ieee.org/document/8234515/

Hessen and Tessem, Hans-Olav, A. J., 2016. Human Activity Recognition With Two Body-Worn Accelerometer Sensors. Master's thesis, "Norwegian University of Science and Technology".

HUNT Research Center, 2019. About HUNT. https://www.ntnu.no/hunt/om, [Online; accessed 19-November-2018].

Jackson, D., 2016. Timesynch. https://openlab.ncl.ac.uk/gitlab/dan.jackson/timesync/tree/master, [Online; accessed 8-May-2019].

Keras, 2019. Keras: The python deep learning library. https://keras.io/, [Online; accessed 2-May-2019].

Ketkar, N. (Ed.), 2017. Recurrent Neural Networks. Apress US, Ch. 6, pp. 77–96.
URL http://dx.doi.org/10.1007/978-1-4842-2766-4

Lara, O. D., Labrador, M. A., 2013. A Survey on Human Activity Recognition using Wearable Sensors. IEEE Communications Surveys & Tutorials 15 (3), 1192–1209.
URL http://ieeexplore.ieee.org/document/6365160/

Mitchell, T. M., 1997. Machine learning. McGraw-Hill series in computer science, Artificial intelligence. McGraw-Hill, New York, Ch. 1, p. 2.

Mohammad, Y., Matsumoto, K., Hoashi, K., 2018. Deep feature learning and selection for activity recognition. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing - SAC '18. ACM Press, New York, New York, USA, pp. 930–939.
URL http://dl.acm.org/citation.cfm?doid=3167132.3167234

Norwegian Directorate of Health, 2016. Statistikk om fysisk aktivitetsnivå og stillesitting. https://www.helsedirektoratet.no/tema/fysisk-aktivitet/statistikk-om-fysisk-aktivitetsniva-og-stillesitting.

NTNU, 2019. Institutt for datateknologi og informatikk.
URL https://www.ntnu.no/idi

Nvidia, 2019a. Nvidia. https://www.nvidia.com/object/doc_gpu_compute.html, [Online; accessed 2-May-2019].

Nvidia, 2019b. nvidia-docker. https://github.com/NVIDIA/nvidia-docker, [Online; accessed 2-May-2019].

Oates, B., 2006. Researching Information Systems and Computing. SAGE Publications.
URL https://books.google.no/books?id=ztrj8aph-4sC

Open Lab, 2018. Ax3 omgui. https://github.com/digitalinteraction/openmovement/wiki/AX3-GUI, [Online; accessed 2-May-2019].

Ordóñez, F., Roggen, D., Ordóñez, F. J., Roggen, D., jan 2016. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. Sensors 16 (1), 115.
URL http://www.mdpi.com/1424-8220/16/1/115

Pavlov, I., 2010. 7-Zip [64] 9.20 Copyright (c) 1999-2010 Igor Pavlov 2010-11-18 p7zip Version 9.20.

Pigou, L., van den Oord, A., Dieleman, S., Herreweghe, M. V., Dambre, J., 2015. Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video. CoRR abs/1506.01911.
URL http://arxiv.org/abs/1506.01911

Reinsve, Ø., 2018. Data analytics for hunt: Recognition of physical activity on sensor data streams. Master's thesis, Norwegian University of Science and Technology.

Russell, S. J., 2016. Artificial intelligence : a modern approach, 3rd Edition. Prentice Hall series in artificial intelligence. Pearson Copyright © 2010, Boston, Ch. 18, 21, 26, pp. 694–695,830,1025.

Sammut, C., Webb, G. I. (Eds.), 2010a. Accuracy. Springer US, Boston, MA, Ch. 1, pp. 9–10.
URL https://doi.org/10.1007/978-0-387-30164-8_3

Sammut, C., Webb, G. I. (Eds.), 2010b. F1-Measure. Springer US, Boston, MA, Ch. 1, pp. 397–397.
URL https://doi.org/10.1007/978-0-387-30164-8_298

Sammut, C., Webb, G. I. (Eds.), 2010c. Recall. Springer US, Boston, MA, Ch. 1, pp. 829–829.
URL https://doi.org/10.1007/978-0-387-30164-8_702

scikit learn, 2019. sklearn.ensemble.randomforestclassifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, [Online; accessed 2-May-2019].

SciPy.org, 2019. scipy.signal.resample, v0.16.1. https://docs.scipy.org/doc/scipy-0.16.0/reference/generated/scipy.signal.resample.html, [Online; accessed 8-May-2019].

Song, S., Chandrasekhar, V., Mandal, B., Li, L., Lim, J.-H., Sateesh Babu, G., Phyo San, P., Cheung, N.-M., 2016. Multimodal Multi-Stream Deep Learning for Egocentric Activity Recognition.
URL https://www.cv-foundation.org/openaccess/content_cvpr_2016_workshops/w13/html/Song_Multimodal_Multi-Stream_Deep_CVPR_2016_paper.html

Stewart, T., Narayanan, A., Hedayatrad, L., Neville, J., Mackay, L., Duncan, S., dec 2018. A Dual-Accelerometer System for Classifying Physical Activity in Children and Adults. Medicine & Science in Sports & Exercise 50 (12), 2595–2602.
URL http://www.ncbi.nlm.nih.gov/pubmed/30048411http://insights.ovid.com/crossref?an=00005768-201812000-00025

Tensorflow, 2019. Tensorflow install gpu support. https://www.tensorflow.org/install/gpu, [Online; accessed 2-May-2019].

Ting, K. M., 2010a. Confusion Matrix. Springer US, Boston, MA, Ch. 1, pp. 209–209.
URL https://doi.org/10.1007/978-0-387-30164-8_157

Ting, K. M., 2010b. Precision. Springer US, Boston, MA, Ch. 1, pp. 780–780.
URL https://doi.org/10.1007/978-0-387-30164-8_651

Ting, K. M., 2010c. Sensitivity and Specificity. Springer US, Boston, MA, Ch. 1, pp. 901–902.
URL https://doi.org/10.1007/978-0-387-30164-8_752

Vågeskar, E., 2017. Activity recognition for stroke patients. Master's thesis, Norwegian University of Science and Technology.

Appendix A

Segmentation

Section A.1 describes the different data formats and feature information during window segmentation and feature extraction for the meta classifier. Section A.2 describes the different data formats and feature information during window segmentation for the LSTM models.

A.1 Meta Classifier Data Segmentation

Figure A.1 shows the input for function ”segment_acceleration_and_calculate_features” that is used by the meta classifier to segment and extract features for training and classification. Figure A.2 show the input used to represent an extracted window with raw features from the input data showed in figure A.1. The sequence length, number of raw features, are now available to be used for window feature extraction, to represent the window features. Figure A.3 shows the result after extracting the wanted features from the raw window features showed in figure A.2. Then the extracted features are appended to another list, to keep track of all the new windows when all the windows with extracted features have been created. Figure A.4 shows the result format of the meta classifier's segmented data for one sensor stream.

```
[
    [-0.952637 -0.109375  0.202637]
     [-0.955322 -0.109375  0.205322]
     [-0.970703 -0.108887  0.220703]
     ...
     [-0.063232 -0.278564 -1.09375 ]
     [-0.075195 -0.283203 -1.09375 ]
     [-0.078369 -0.267822 -1.09375 ]
]
Array shape:  (1740050, 3)
```

Figure A.1: Raw input format of back feature to the meta classifier

```
[
    [-0.952637 -0.109375  0.202637]
     [-0.955322 -0.109375  0.205322]
     [-0.970703 -0.108887  0.220703]
     [-0.965576 -0.112305  0.215576]
     [-0.953857 -0.124023  0.203857]
     ...
     [-0.986572 -0.125977  0.221436]
     [-0.973145 -0.137451  0.218262]
     [-0.955322 -0.140869  0.220459]
     [-0.952637 -0.140625  0.205322]
]
Array shape: (250,3)
```

Figure A.2: Extracted window features format of back feature inputs to the meta classifier

```
[29.6875, 29.6875, 0.0, 0.0, -3.4313636944000003,
0.3803171944, -3.5170350954000003, 0.0, 0.0]
List Shape: (9)
```

Figure A.3: Format of extracted features from the window features of back feature inputs to the meta classifier

```
[
    [3.29101562e+01, 3.29101562e+01, 0.00000000e+00 ...
    1.05788287e+00,  0.00000000e+00, 0.00000000e+00]

    [3.29101562e+01, 3.29101562e+01, 0.00000000e+00 ...
    8.76227367e-01, 0.00000000e+00, 0.00000000e+00]

     [3.32031250e+01, 3.29101562e+01, 2.92968750e-01 ...
    9.67753729e-01, 2.92968750e-01, 2.92968750e-01]
     ...
     [3.29101562e+01, 3.29101562e+01, 0.00000000e+00 ...
    1.91218494e-02,  5.85937500e-01, 2.92968750e-01]

    [3.29101562e+01, 3.26171875e+01, 2.92968750e-01 ...
    5.50091153e+00, 5.85937500e-01, 5.85937500e-01]

    [3.26171875e+01, 3.26171875e+01, 0.00000000e+00 ...
    5.50156250e+00, 5.85937500e-01, 2.92968750e-01]
]
Array shape: (6960, 9)
```

Figure A.4: The format of result array returned from segmentation of back sensor input data to the meta classifier

A.2 LSTM Data Segmentation

Figure A.5 shows the input format for the segmentation code that is described in section 5.2.3. Figure A.6 shows the result array after segmentation of the input data stream. The model that uses both sensor data streams does the segmentation twice, one for the back features and once for the thigh features. Then it passes each segmented window array into the training or classification function, since it expects two different input sources. The LSTM models that only uses one sensor data stream, does the segmentation only once,

and passes in the resulting array as the single input source to the training or classification function.

index=Timestamp	bx	by	bz	tx	ty	tz	label
2019-05-07 14:28:49.936877056	-3.317336	0.198589	0.178875	-3.054523	-0.889009	-0.849577	6
2019-05-07 14:28:49.956879104	-2.164232	-0.053542	0.272944	0.170010	-0.617006	1.005296	6
2019-05-07 14:28:49.976881152	-1.417694	0.521499	0.278008	-1.416422	-0.078709	0.238333	6
2019-05-07 14:28:49.996882944	-1.247310	0.350846	-0.337431	-1.063031	0.073647	-0.450256	6
2019-05-07 14:28:50.016884992	0.718958	-0.291272	-0.118643	-0.598168	1.405616	-0.645643	6
...							
2019-05-07 16:52:26.496632064	-0.950248	-0.163437	-0.175370	-0.985232	-0.146119	-0.093409	3
2019-05-07 16:52:26.516655872	-0.968631	-0.160248	-0.194454	-0.975029	-0.136298	-0.077665	3
2019-05-07 16:52:26.536679936	-0.979692	-0.174562	-0.204234	-0.985752	-0.170817	-0.059147	3
2019-05-07 16:52:26.556703744	-0.987859	-0.157429	-0.202106	-0.974281	-0.152641	-0.047847	3
2019-05-07 16:52:26.576727552	-0.989634	-0.153336	-0.215740	-0.998599	-0.121618	-0.028792	3
2019-05-07 16:52:26.596751360	-0.985901	-0.136210	-0.210973	-0.991204	-0.082875	-0.065771	3
2019-05-07 16:52:26.616775424	-0.994026	-0.134252	-0.219491	-0.997643	-0.092655	-0.056850	3
2019-05-07 16:52:26.636799232	-1.003030	-0.105652	-0.217686	-0.982686	-0.051886	-0.067491	3
2019-05-07 16:52:26.656823040	-1.000200	-0.117214	-0.210253	-0.996057	-0.044920	-0.064276	3
2019-05-07 16:52:26.676847104	-1.009537	-0.106673	-0.196176	-0.975586	-0.009792	-0.080069	3
2019-05-07 16:52:26.696870912	-0.993653	-0.150775	-0.215815	-1.023444	-0.035857	-0.070622	3

[440838 rows x 7 columns]
Number of dataframes in input list: 1

Figure A.5: The format of INPUT array for segmentation of back sensor input data to the LSTM

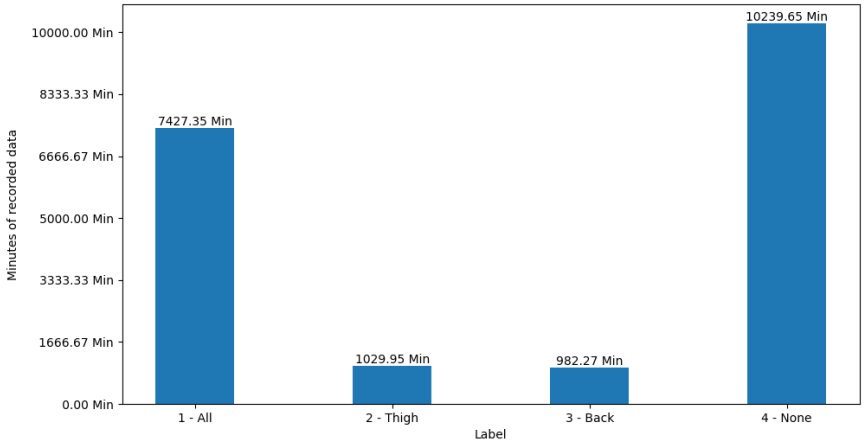
```
[
    [
        [-3.31733576  0.19858909  0.17887465]
        [-2.1642322  -0.05354198  0.27294394]
        [-1.41769375  0.5214987   0.27800779]
        ...
        [-0.99892955  0.06343103 -0.0219557 ]
        [-1.00741534  0.06014781 -0.04058109]
        [-1.01776431  0.07111984 -0.03247158]
    ]
    ...
    [
        [-1.0017032   0.06281857 -0.27206858]
        [-0.990159    0.06213143 -0.26350186]
        [-0.98886532  0.06285155 -0.26744651]
        ...
        [-0.99309492  0.06234614 -0.28381926]
        [-1.00072998  0.06234943 -0.27158172]
        [-0.98432787  0.06275191 -0.2654673 ]
    ]
    [
        [-0.98409346  0.06223443 -0.26471542]
        [-0.98473634  0.0627619  -0.26675166]
        [-0.99917809  0.06224416 -0.28006439]
        ...
        [-0.98594638  0.0623323  -0.28021695]
        [-0.98325941  0.06170937 -0.28209383]
        [-0.98527795  0.05432499 -0.28055647]
    ]
    ...
]
Array shape:  (1536, 250, 3)
```

Figure A.6: The format of result array returned from segmentation of back sensor input data to the LSTM

Appendix B

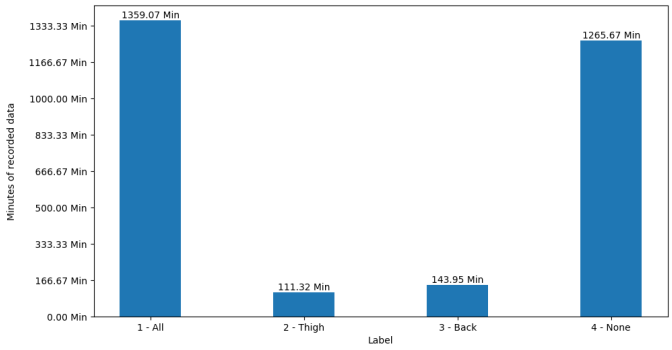
SNT Dataset Label Distribution

This chapter presents bar plots of the labeling distribution in the SNT dataset, one for the entire dataset and one for each recording.

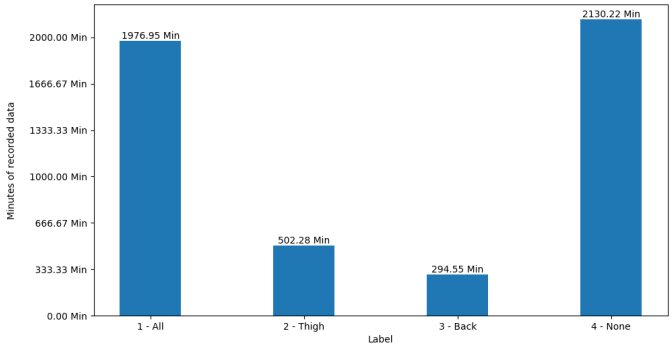


(a) Entire SNT dataset

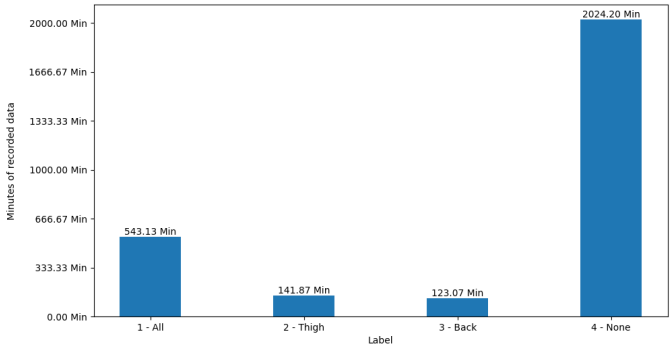
Figure B.1: SNT dataset Label Distribution



(b) Recording 001.1

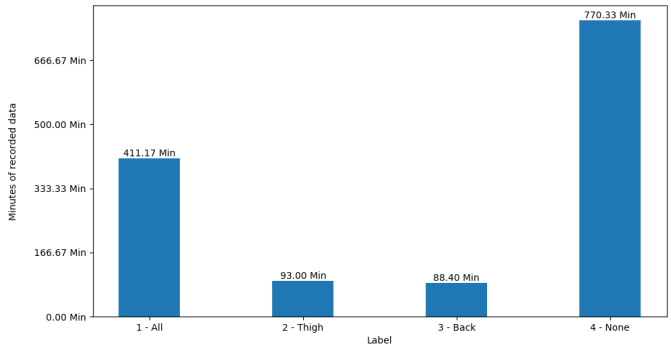


(c) Recording 001.2

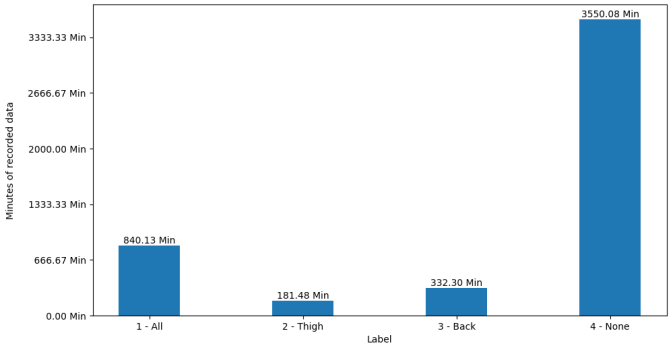


(d) Recording 002.1

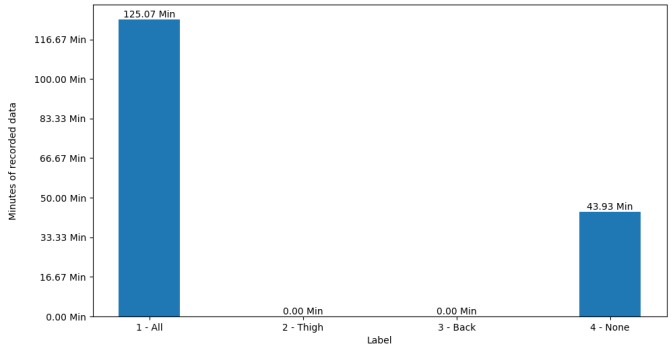
Figure B.1: SNT dataset Label Distribution



(e) Recording 002.2

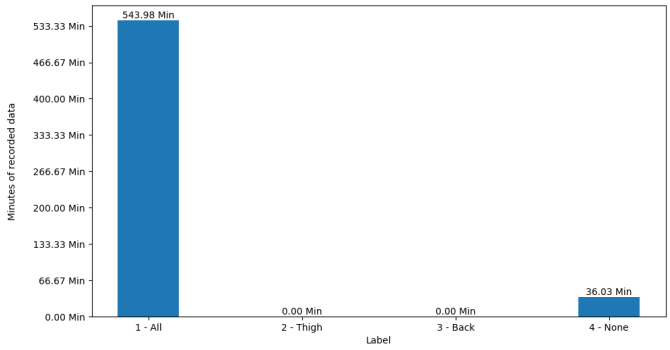


(f) Recording 002.3

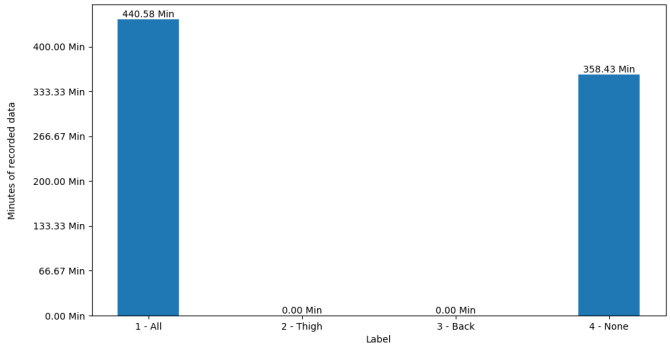


(g) Recording 003

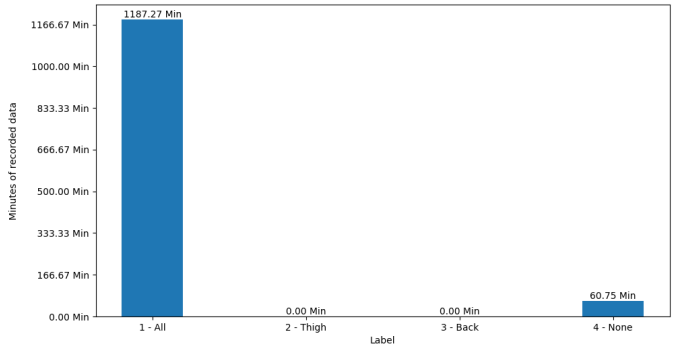
Figure B.1: SNT dataset Label Distribution



(h) Recording 004



(i) Recording 005



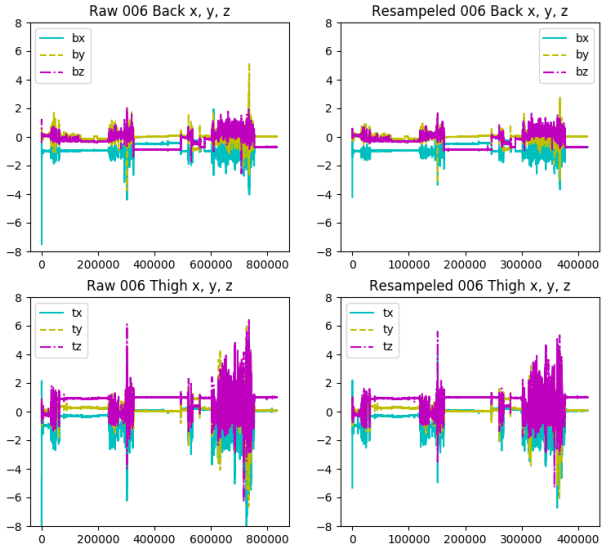
(j) Recording 006

Figure B.1: SNT dataset Label Distribution

Appendix C

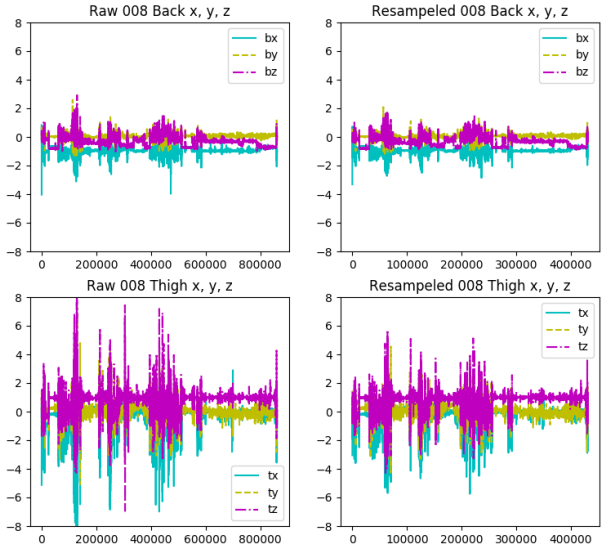
Resampling

This appendix presents acceleration signals for back and thigh sensors before and after resampling from $100Hz$ to $50Hz$.

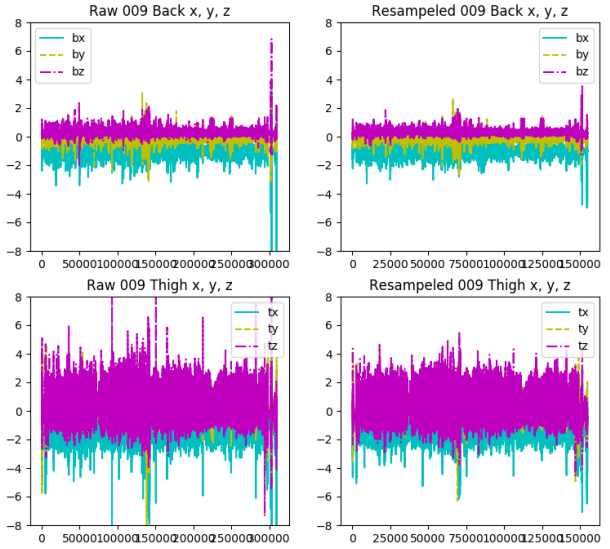


(a) Subject 006

Figure C.1: Resampling for back and thigh sensors for subject 006

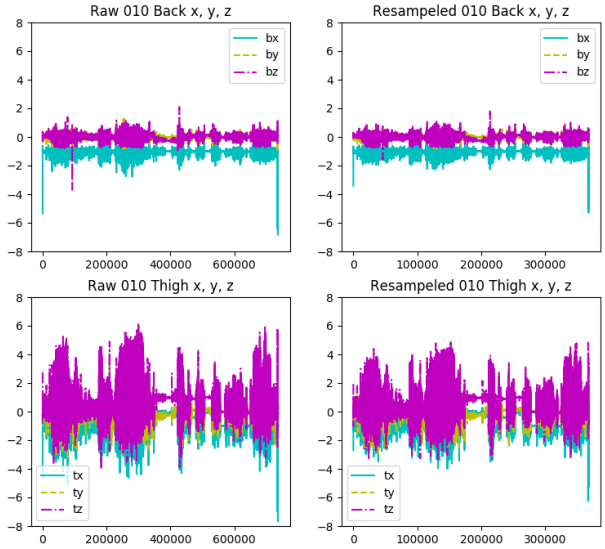


(a) Subject 008

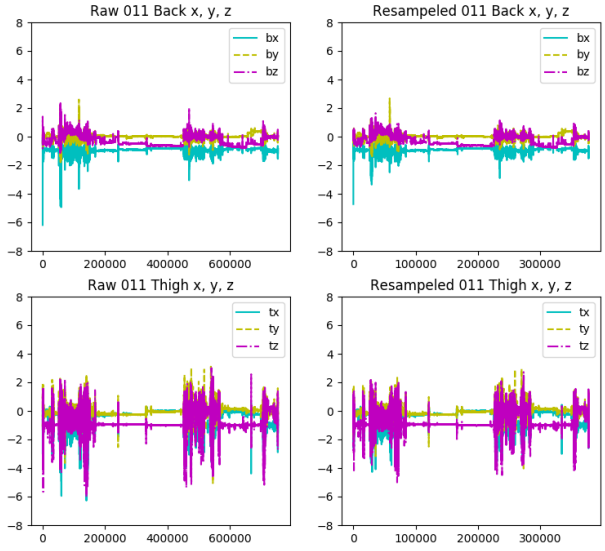


(b) Subject 009

Figure C.2: Resampling for back and thigh sensors for subjects 008 and 009

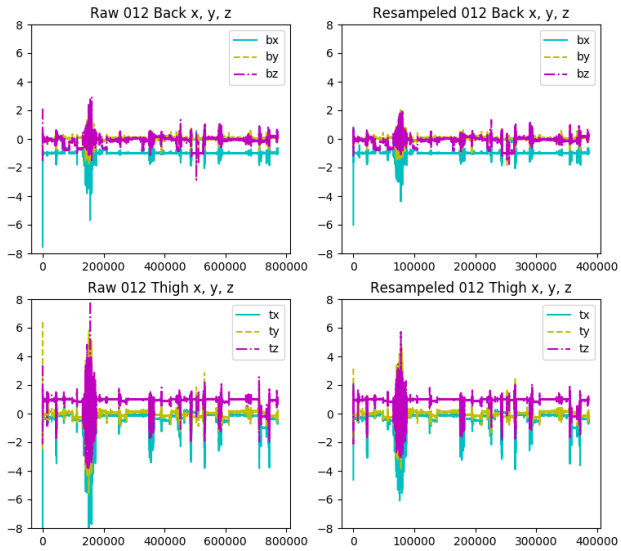


(a) Subject 010

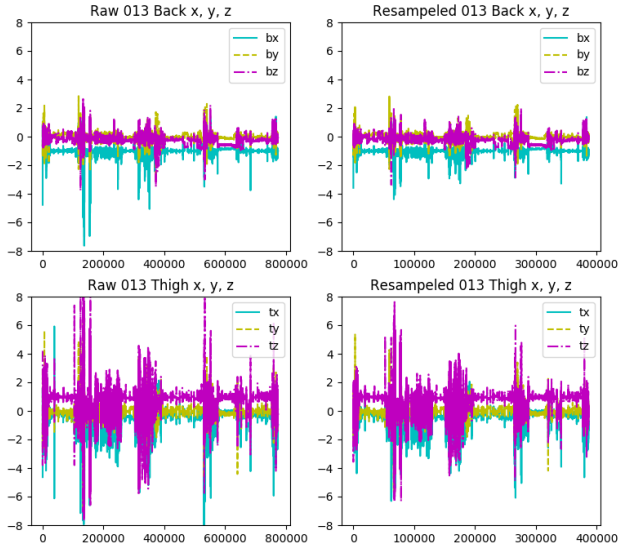


(b) Subject 011

Figure C.3: Resampling for back and thigh sensors for subjects 010 and 011

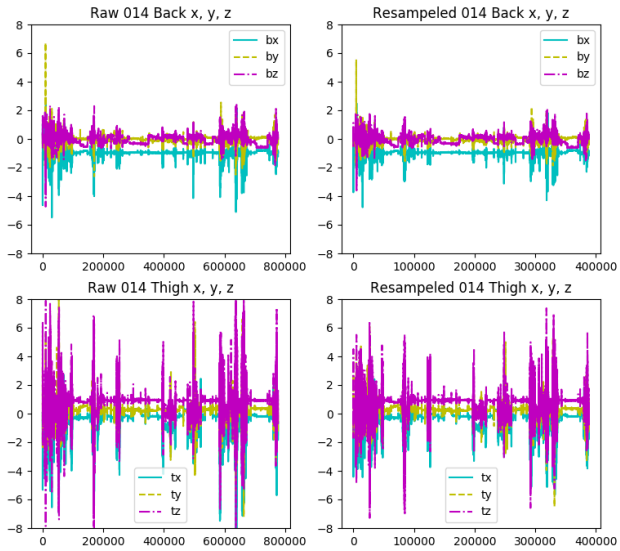


(a) Subject 012

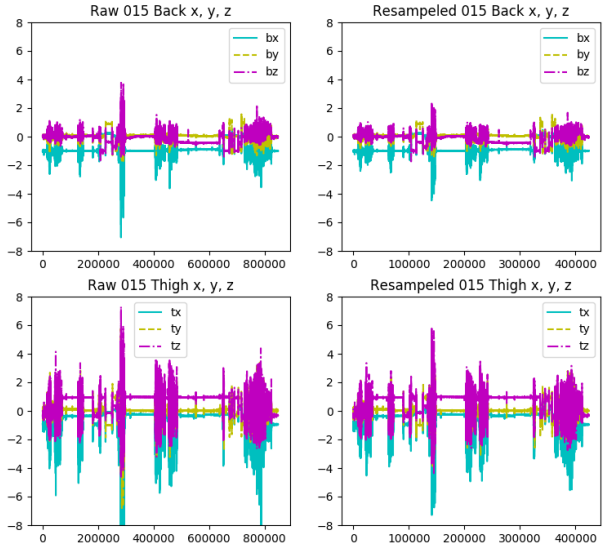


(b) Subject 013

Figure C.4: Resampling for back and thigh sensors for subjects 012 and 013

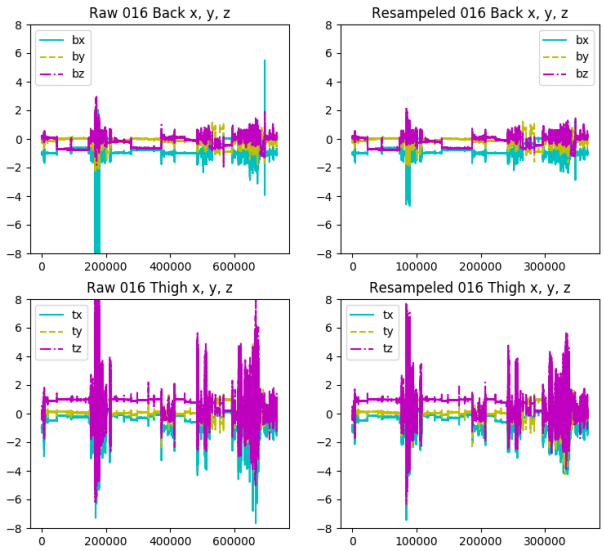


(a) Subject 014

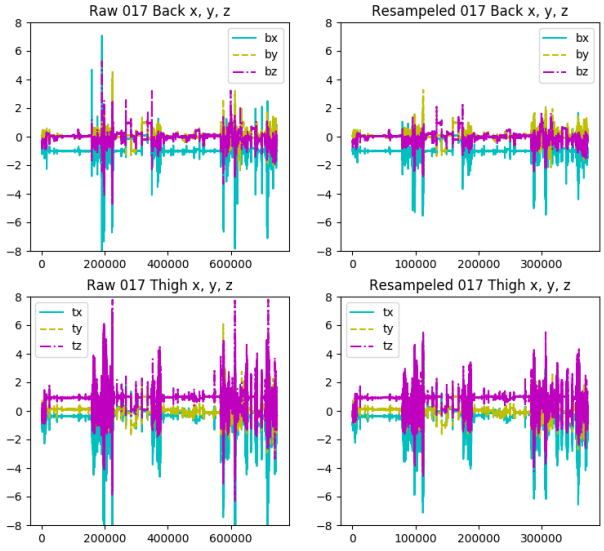


(b) Subject 015

Figure C.5: Resampling for back and thigh sensors for subjects 014 and 015

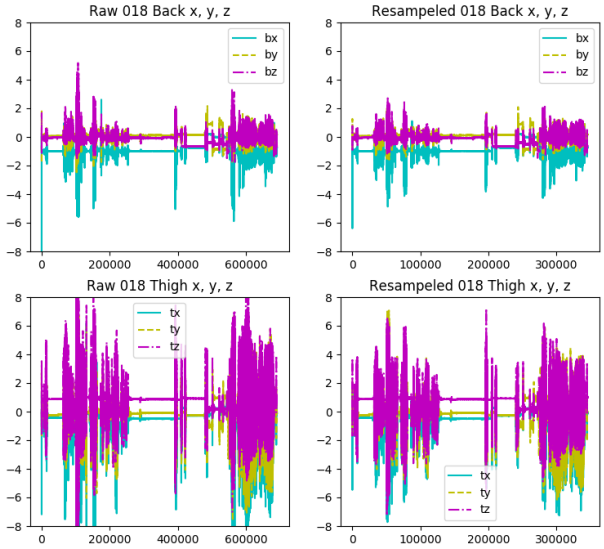


(a) Subject 016

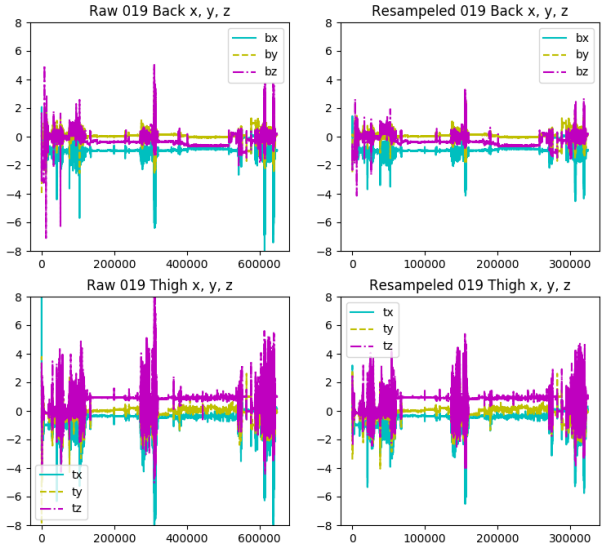


(b) Subject 017

Figure C.6: Resampling for back and thigh sensors for subjects 016 and 017

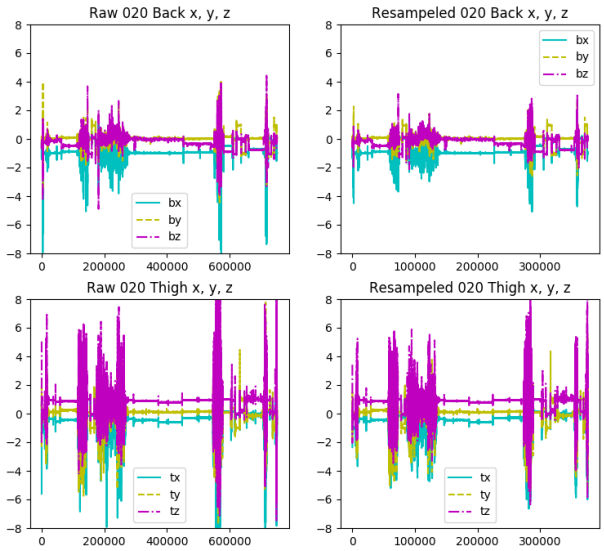


(a) Subject 018

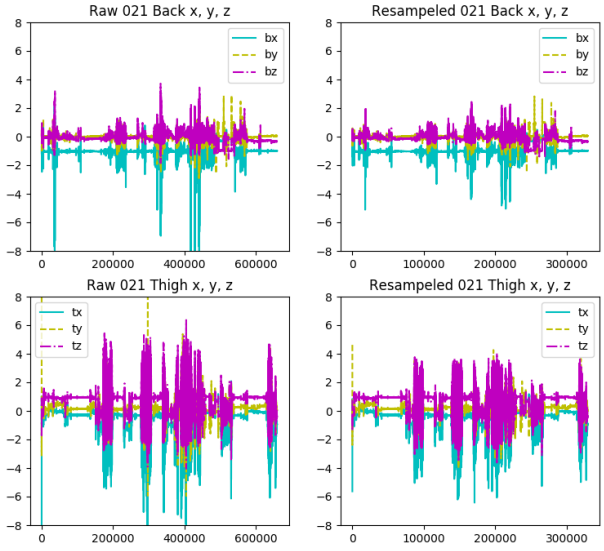


(b) Subject 019

Figure C.7: Resampling for back and thigh sensors for subjects 018 and 019

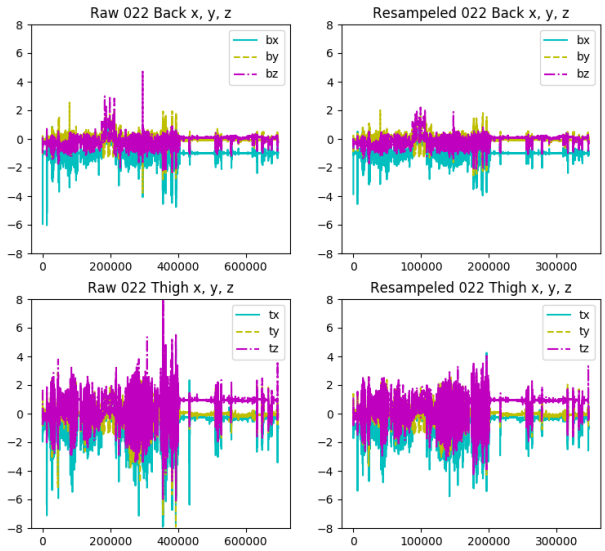


(a) Subject 020



(b) Subject 021

Figure C.8: Resampling for back and thigh sensors for subjects 020 and 021



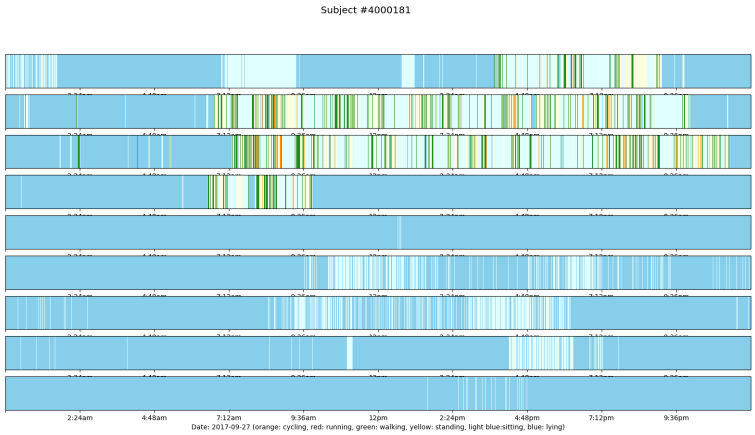
(a) Subject 022

Figure C.9: Resampling for back and thigh sensors for subject 022

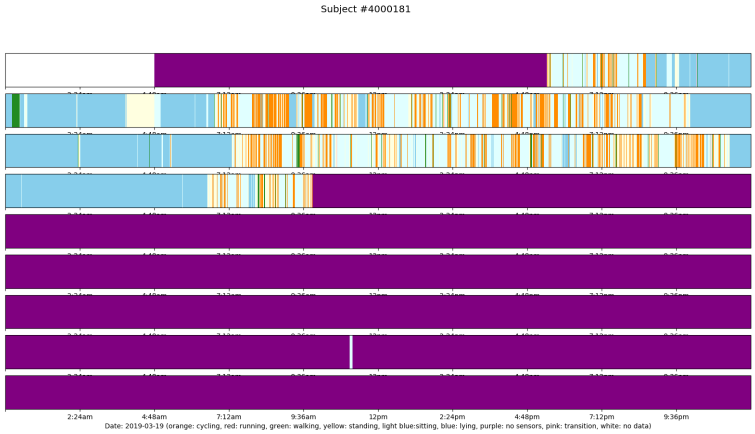
Appendix D

Suspected Wrongly Classified Subjects

This appendix presents all the identified subjects from the HUNT4 dataset that is suspected to contain sensor no-wear time and therefore be wrongly classified because of faulty data sensor streams caused by faulty sensors or sensors being detached from the subject. There are five subjects identified, and each subject is presented with classification results from the existing HAR system and the proposed ensemble classifier. Both classifications are presented in forms of a daily overview chart, where each row starts at midnight, and spans through an entire day, consisting of 24 hours.

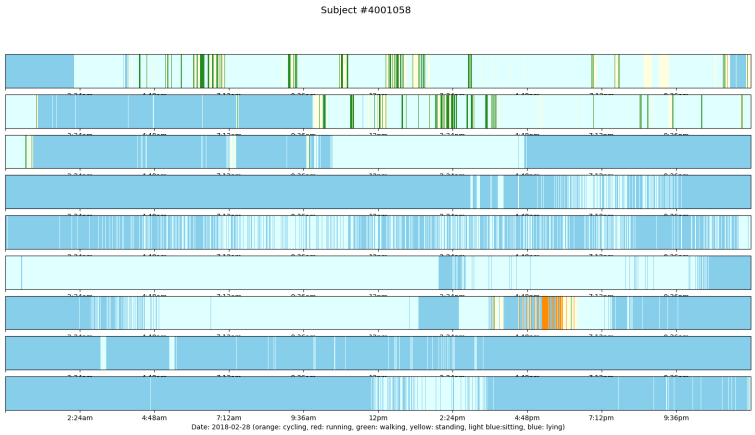


(a) Classification done by existing HAR system on subject 4000181

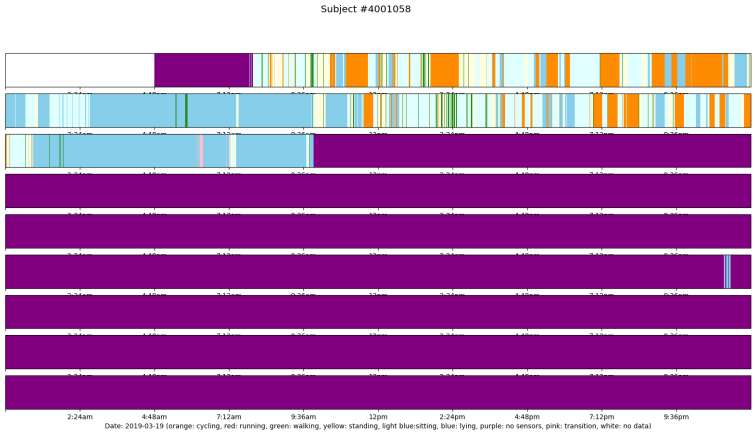


(b) Classification done by the proposed ensemble classifier on subject 4000181

Figure D.1: Activity classification comparison for subject 4000181

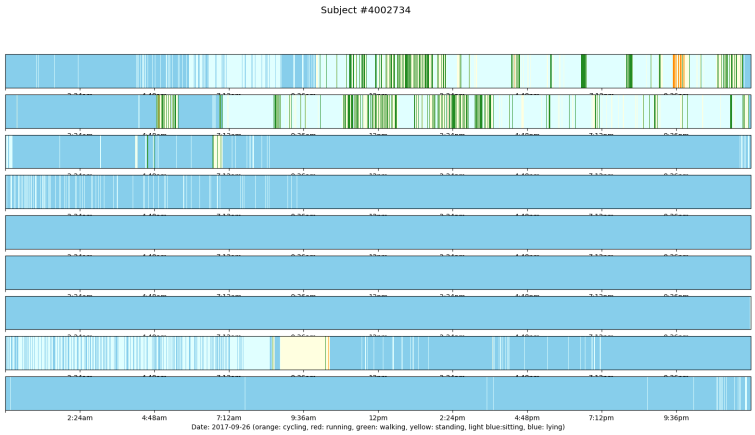


(a) Classification done by existing HAR system on subject 4001058

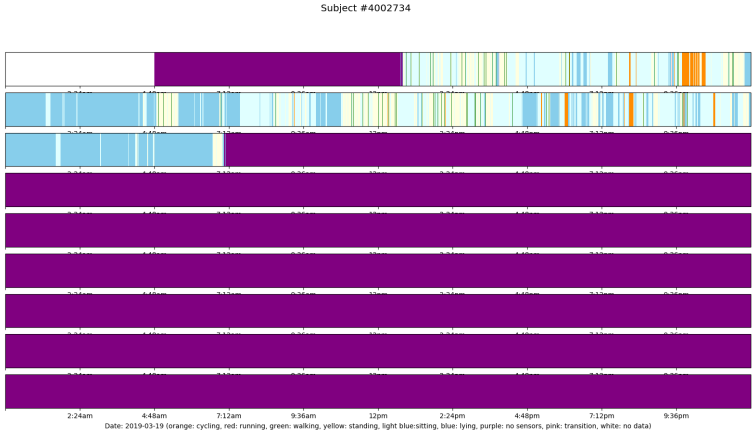


(b) Classification done by the proposed ensemble classifier on subject 4001058

Figure D.2: Activity classification comparison for subject 4001058

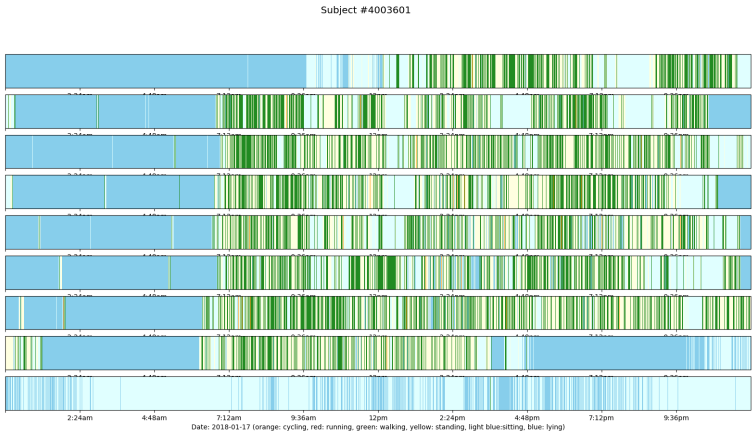


(a) Classification done by existing HAR system on subject 4002734

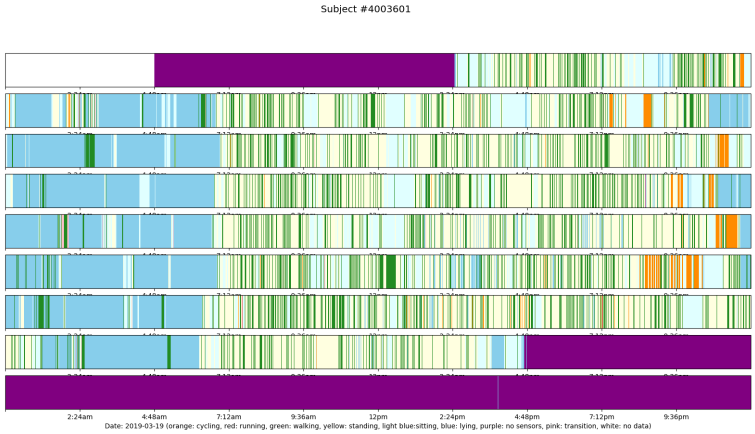


(b) Classification done by the proposed ensemble classifier on subject 4002734

Figure D.3: Activity classification comparison for subject 4002734

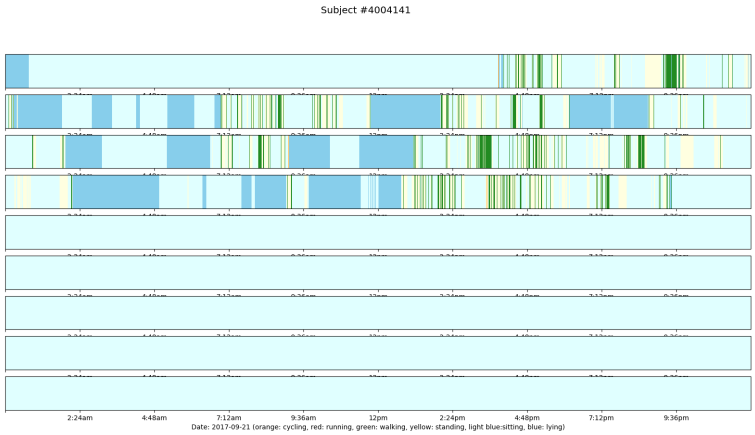


(a) Classification done by existing HAR system on subject 4003601

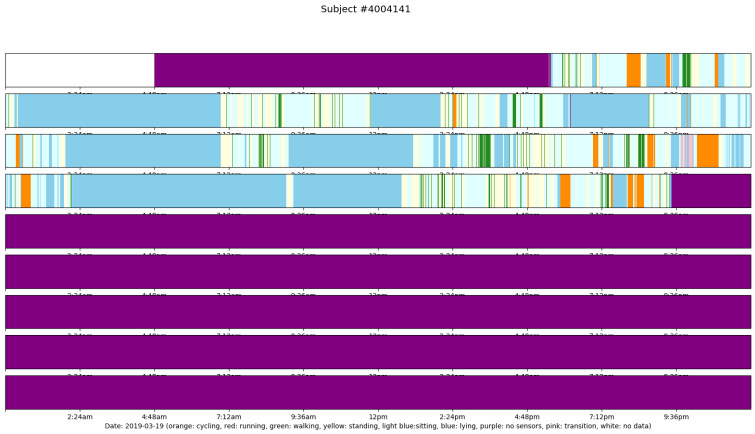


(b) Classification done by the proposed ensemble classifier on subject 4003601

Figure D.4: Activity classification comparison for subject 4003601



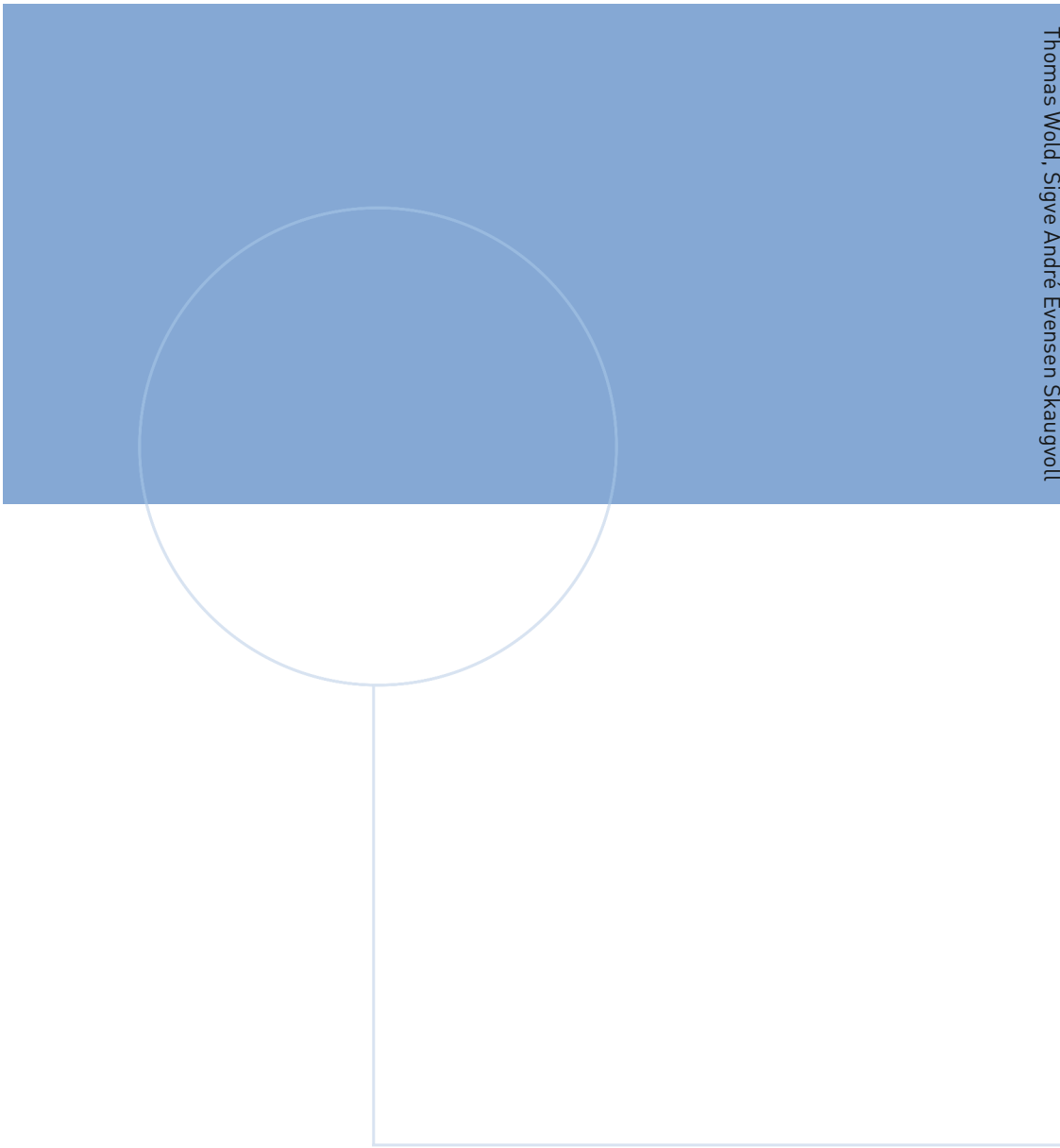
(a) Classification done by existing HAR system on subject 4003601



(b) Classification done by the proposed ensemble classifier on subject 4004141

Figure D.5: Activity classification comparison for subject 4004141

Thomas Wold, Sigve André Evensen Skaugvoll



NTNU
Norwegian University of
Science and Technology