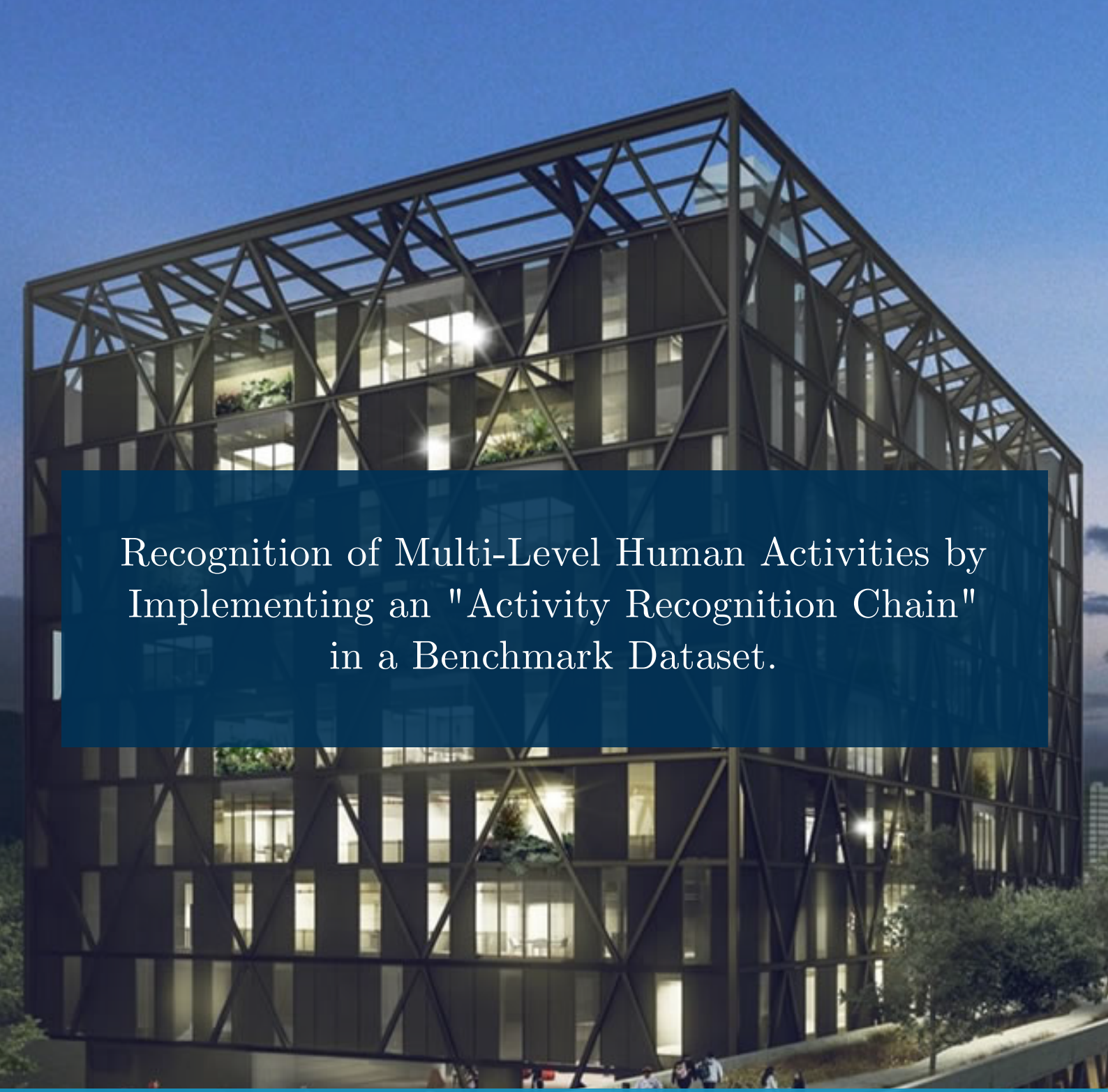


Recognition of Multi-Level Human Activities by Implementing an "Activity Recognition Chain" in a Benchmark Dataset.

Author:

Agustín VARGAS TORO

Supervisors:

Dr. Sebastian G.A. KONIETZNY

Dr. Alejandro MARULANDA TOBÓN

UNIVERSIDAD EAFIT

Fraunhofer IAIS

Universidad EAFIT

In cooperation with Fraunhofer IAIS

Bachelor Thesis

Recognition of Multi-Level Human Activities by Implementing an "Activity Recognition Chain" in a Benchmark Dataset

---

Author:
Agustín Vargas Toro

Supervisors:
Dr. Sebastian G.A. Konietzny
Dr. Alejandro Marulanda Tobón

A thesis submitted in fulfillment of the requirements for the degree of Physical Engineer

in the

Department of Physical Sciences
School of Sciences

July 10, 2018
Medellín, Colombia

Cover design: Juan Manuel Rico Londoño.
Cover image: **"Proyecto Escuela de Ciencias - Universidad EAFIT"**.
Author: **Estudio Transversal**.
(June 14, 2018) Retrieved from:
`http://www.eafit.edu.co/campus-eafit/acerca-de/Paginas/nuevas-construcciones.aspx`

Declaration of Authorship

I, Agustín Vargas Toro, declare that this thesis titled, "**Recognition of Multi-Level Human Activities by Implementing an "Activity Recognition Chain" in a Benchmark Dataset**" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"*Things only look pure if your look at them from far away. It is very important to know about our roots, to know where we come from, to understand our history. But at the same time, as important as knowing where we are from, is understanding that deep down, we are all from nowhere and a little bit from everywhere.*"

Jorge Drexler

Acknowledgements

First and foremost, I would like to profoundly thank my supervisor, Dr. Sebastian Konietzny for giving me the opportunity to work in this fascinating area of research as well as for the continuous guidance and infinite support through the development of this thesis. I would also like to thank my second supervisor, Dr. Alejandro Marulanda Tobón for all his help and supervision through this work.

To my parents, Ninfa Toro Herrera and John Jairo Vargas del Valle for their never-ending love and support, as well for instilling in me the values that today define me as a person. To Andres Felipe Uribe Peláez, who has been a father to me since the moment I met him and who has also continuously supported me through this adventure we call life. To my brother Tomás Vargas, whose kindness has always taught me how to be a better person.

Last but not least, to the Universidad EAFIT and the Deutscher Akademischer Austauschdienst (DAAD), who gave me wings and opened to me the doors of the world. Infinite thanks for giving me one of the most amazing experiences in my life.

Contents

List of Figures

List of Tables

List of Abbreviations

ADL	**A**ctivities (of) **D**aily **L**ife
ARC	**A**ctivity **R**ecognition **C**hain
AUC	**A**rea **U**nder (the) **C**urve
EAD	**E**vent **A**nalysis **D**iagram
EDA	**E**xploratory **D**ata **A**nalysis
FN	**F**alse **N**egatives
FP	**F**alse **P**ositives
FPR	**F**alse **P**ositive **R**ate
HAR	**H**uman **A**ctivity **R**ecognition
IMU	**I**nertial **M**easurement **U**nit
OC	**O**PPORTUNITY **C**hallenge
OCD	**O**PPORTUNITY **C**hallenge **D**ataset
OD	**O**PPORTUNITY **D**ataset
REACH	**R**esponsive **E**ngagement (of the Elderly promoting) **A**ctivity (and) **C**ustomized **H**ealthcare
RF	**R**andom **F**orest
ROC	**R**eceiver **O**perating **C**haracteristic
SET	**S**egment **E**rror **T**able
TN	**T**rue **N**egatives
TP	**T**rue **P**ositives
TPR	**T**rue **P**ositive **R**ate

A esos buenos pocos . . .

Chapter 1

Introduction

1.1 Motivation

The development of microelectronics and computer systems in the past decade has enabled the development of sensors with high computational power, small size, and low cost, allowing humans to interact with these devices as part of their daily living. The latter led to the creation of a research area, *Ubiquitous Sensing*, that focuses on extracting knowledge from data acquired by pervasive sensors to recognize activities that humans perform in their daily living, predict events that might happen, and prevent situations of risk **(Lara and Labrador, 2013)**. One of the key topics within this area is the automatic recognition of physical activities, commonly referred to as Human Activity Recognition (HAR), that aims to provide information about user behavior, allowing computing systems to recognize and assist users with their tasks **(Bulling et al., 2014)**.

The HAR topic has become a task of high interest in different fields, and it is a problem that is not feasible to be solved deterministically: the number of combinations of sensor measurements and activities can be very large and finding exact transition points between the activities becomes difficult, since the duration of each activity is generally unknown **(Lara and Labrador, 2013)**. Therefore, activity recognition systems are developed with a focus on machine learning algorithms and innovations of hardware architecture, achieving simultaneously a decrease in the cost of monitoring and an increase of the target subjects' safety.

According to **Ranasinghe et al. (2016)**, most of the HAR applications fit into one of the following four applications:

- Active and assisted living systems for smart homes.
- Healthcare monitoring applications.
- Monitoring and surveillance systems for indoor and outdoor activities.
- Tele-immersion and interactive applications.

Monitoring and surveillance systems pursue the automatization of the detection of anomalities in camera views, a task that has been mostly performed by human operators and whose productivity has dropped due to the increasing

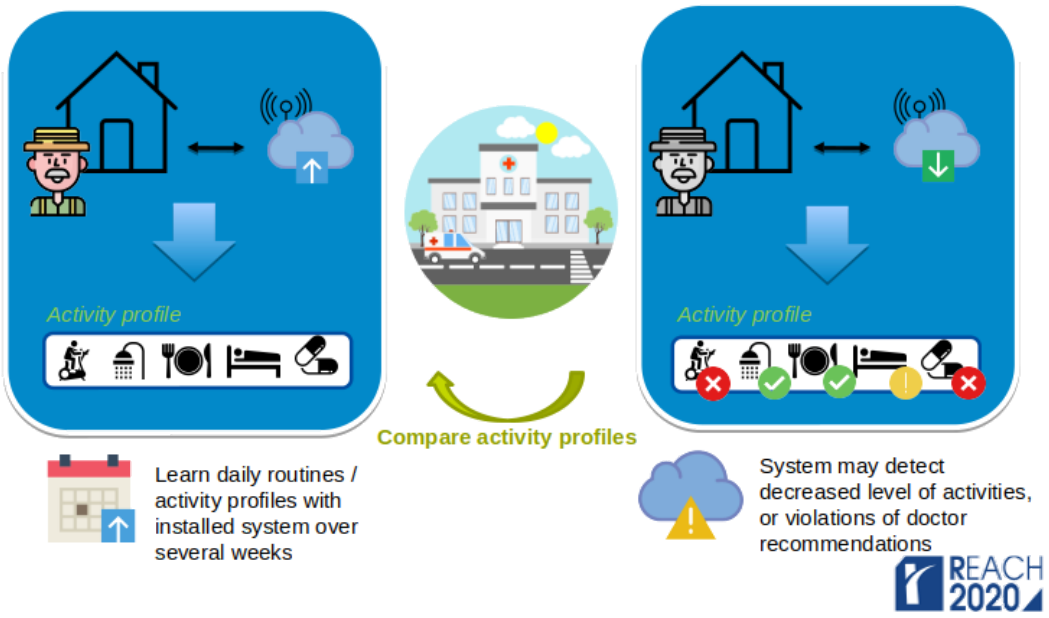


FIGURE 1.1: Example of a REACH use case: the REACH engine learns the daily routines from the patient over several weeks. After a visit to the hospital, where the patient underwent a medical treatment, the system checks the previous routines of the patient and is also updated with the medical recommendations, being able to recognize abnormalities and irregularities within the routines **(Konietzny et al., 2018)**

number of camera deployments **(Ranasinghe et al., 2016)**; ***tele-inmersion*** applications allow users to interact with virtual and real objects through the execution of physical actions that are interpreted as a command that is then executed by the system.

The two remaining HAR applications, ***active and assisted living systems for smart homes*** and ***health monitoring*** are addressed by the European Union project REACH (**Responsive Engagement** of the elderly promoting **Activity** and **Customized Healthcare**), a project that serves as a motivation to develop this work. The goal of the REACH project is to develop a service system that focuses on these HAR applications by "turning clinical and care environments into personalisable modular sensing, prevention, and intervention systems that encourage the elderly to become healthy via activity (physical, cognitive, mobility, personalized food, etc)"[1]. REACH intervenes in these two HAR applications as follows:

- **Goldstone (2010)** asserts that in 2050, due to the nowadays "baby boomers" aging and life expectancy increasing, the proportion of people aged 60 or older will increase dramatically. As populations age, there will be an increased demand on health care for longer periods of time, that will be unlikely to be supplied. Therefore, enhancing the medical monitoring

[1]Responsive Engagement of the Elderly promoting Activity and Customized Healthcare (REACH). *Short Description.* Available online at: http://reach2020.eu/

systems by making them able to handle a wide range of medical situations will shorten the visits of a patient to the hospital and will keep the doctors more informed about the medical status of their patients, allowing them to give a proper feedback that will improve their health condition **(Ranasinghe et al., 2016)**. Moreover, elderly care facilities could constantly monitor their patients to detect any abnormal activity (*e.g.* pain, elevated blood pressure, tachycardia) in order to prevent any undesirable consequences (*e.g.* fall, heart attack, faint).

- Equipping a home with sensing technology that improves the safety and independence of the residents while monitoring their health conditions, is the core of the active and assisted living systems **Ranasinghe et al. (2016)**. These systems would monitor routines of the patient, creating activity profiles allowing the system to do recommendations based on the patient's necessities. **Figure 1.1** shows a REACH use case: first, an accompanying system learns the habits and activities that are performed during a patient's daily routine. After a medical treatment, patients are often encouraged to follow a routine that improves their condition (*e.g.,* walking, staying regularly hydrated, taking a medicine, sleeping a certain amount of hours, etc.) and these recommendations are tuned with the system that will check if the activities are being performed and will report irregularities to the patient while recommending them new strategies to restore their health. Additionally, the activity profiles of the patient during the day become also a relevant source of information for the caregiver about the patients' health status.

1.2 Objectives

The HAR problem is commonly addressed by executing a sequence of tasks that transform the measured raw data and output the prediction of a set of activities. The following objectives resume the main goals aimed to achieve within this work:

1.2.1 General Objective

Develop a Human Activity Recognition system that recognizes locomotion, low, and middle level activities from a pool of activities contained in a public benchmark dataset.

1.2.2 Specific Objectives

- Select a publicly available dataset that can be used for the HAR problem by taking into account the use case settings, the database structure, and the data collection techniques performed when recording the dataset.

- Analyze and implement a method of missing data imputation that leads to the best representation of the missing values of the dataset.
- Implement a method of segmentation of time series, test several settings and analyze their impact on the results.
- Extract a collection of features from time series data that retains its internal characteristics in order to feed it to a classifier.
- Implement a methodology of feature selection that retains the most significant features that will be used in the classification.
- Implement a classifier and evaluate the performance of the complete system on the selected dataset with standard metrics and specialized metrics for HAR.
- Recognize locomotion activities, middle level activities (such as drinking), and low level activities (*e.g.* grab, release) from sensor data.

1.3 Outline

A HAR process involves a sequence of practices that begins with the collection of sensor data and finishes with the recognition of the performed activity. This document exposes a framework for activity recognition that has been implemented and tested in a publicly available dataset for activities of daily life (ADL). The framework includes:

1. An exploratory analysis of the data.
2. A pre-processing of the raw data that eliminates noise, handles missing values, calculates new predictors from the raw data, and removes high-correlated predictors
3. A segmentation strategy of the data and a feature extraction process that calculates the main characteristics of the data segments.
4. A dimensionality reduction and feature selection process that reduces the complexity of the data and the computational expense of the classification.
5. A classification process where machine learning algorithms identify to which of a set of categories an observation belongs.
6. An analysis of the results, based not only on classical performance metrics for any pattern recognition problem (accuracy, F-measure), but also with specialized metrics for activity recognition (SET2 and EAD).

The rest of this document is divided into the following: **Chapter 2** formulates the problem of human activity recognition, describes the categories in which an activity can be defined, and reviews the related work for the HAR tasks of medical monitoring and activities of daily life.

The concept of an 'Activity Recognition Chain', a sequence of techniques that implements a specific activity recognition system, is introduced in **Chapter 3**, along with the state-of-the-art in HAR.

The methodology followed for the implementation of the system is described in **Chapter 4**, and the obtained results are summarized in **Chapter 5**.

Finally, in **Chapter 6** the results are discussed, conclusions of the methodology are presented, and future work is proposed. Additionally, the appendix includes detailed information about the considered publicly available datasets **(Appendix A)**, an exploratory analysis of the chosen dataset **(Appendix B)**, and details about the implemented Activity Recognition Chain in the programming language `R` **(Appendix C)**.

Chapter 2

Background

2.1 Problem Formulation

Human Activity Recognition (HAR) aims to recognize the activities of a person based on sensor and/or video observation data and, in some applications, with sensors that provide information about the context (*e.g.,* location, temperature, time of the day) in which the activity is taking place. The latter is done by executing a series of data processing steps **(Section 3.1)** that transform the raw data from sensor streams and return the recognition of the performed activities **(Ranasinghe et al., 2016)**. This objective was defined by **Lara and Labrador (2013)** as the HAR and relaxed HAR problem as follows:

> ***Definition 1: the HAR problem***
>
> Given a set of n time series $T = \{T_0, T_1, ..., T_{n-1}\}$, where each time series represents the measurement of one sensor, commonly constrained within a time interval $I = [i_\alpha, i_\omega]$. The objective is to find a set of sequential partitions P_j, of I, $\langle P_0, ..., P_{m-1} \rangle$, based on the time series data (T) and a set of *labels* that represent the activity that was performed during each interval P_j.

This definition only holds if the performed activities are not simultaneous because each interval P_j will have assigned only one activity. It also implies that the intervals are consecutive, non-empty, non overlapping, such that the ordered arrangement of each partition describes all the activities that were done in the time interval (*i.e.,* $\bigcup_{j=0}^{m-1} P_j = I$) **(Lara and Labrador, 2013)**.

The segmentation of the time series into intervals that represent single activities is not straightforward, since activities can have different durations (*i.e.,* the length of each interval is unknown) and finding a transition point from one activity to the other can be a difficult task, since humans perform activities in a fluent way and consecutive activities blur into each other rather than being clearly separated by a pause **(Bulling et al., 2014)**. Therefore, a *relaxed* version of the HAR problem is introduced, where the time series are not segmented

into single activities, but they are partitioned into equally sized segments.

> ***Definition 2: the relaxed HAR problem***
>
> Given a set W of m equally sized windows ($W = \{W_0, W_1, ..., W_{m-1}\}$) that partition the interval I, where each window W_j contains a segment of the collection of time series (*i.e.,* $T_{i,j} = \{T_{0,j}, T_{1,j}, ..., T_{n_1,j}\}$) and a ground truth *label* of the set of **target** activities A, the objective is to find a mapping function $f : W_j \longrightarrow A$ that can be evaluated for all values of W_j, such that $f(W_j)$ is as similar as possible to the real activity performed during W_j.

In comparison with the HAR problem, the *relaxed* HAR problem introduces an error to the model by partitioning the time series into fixed segments, as it may happen that a person performs more than one activity during a time window, but the model can only associate this window to a single activity. However, in most HAR applications the length of a time window goes from 0.5 to 6 seconds **(Section 3.1.3)**, meaning that the number of activity transitions that can happen within one time window is expected to be considerably smaller than the total number of time windows, thus making the relaxation error insignificant for most of the applications **(Lara and Labrador, 2013)**. Therefore, this work is focused on solving the *relaxed* HAR problem.

The collection of data for HAR is a challenging and time consuming task that involves challenges on data collection, synchronization and multi-modal sensor arrangements **(Calatroni et al., 2011)** that are beyond the scope of this thesis. Instead, a dataset has been selected among the publicly available HAR datasets. The OPPORTUNITY dataset **(Chavarriaga et al., 2013)** has been selected selected to recognize locomotion (*i.e.,* walking, sitting, lying, and standing) and hand gestures (*e.g.,* drinking, open door, close door) from a series of simulated runs that were performed by several subjects in a data collection experiment. **Section 2.5** provides a description of some of the publicly available datasets for HAR and presents the arguments for selecting the OPPORTUNITY dataset as the target dataset.

2.2 Notion of an activity

Among the *Ubiquitous Sensing* community there is no consensus about a unique definition of what an "activity" represents, and the semantic use of this word is often mixed with "action" or "gestures", indiscriminately. Moreover, a categorization of activities is often made according to the target activities for each research project and therefore there is no homogeneous categorization of types of activities.

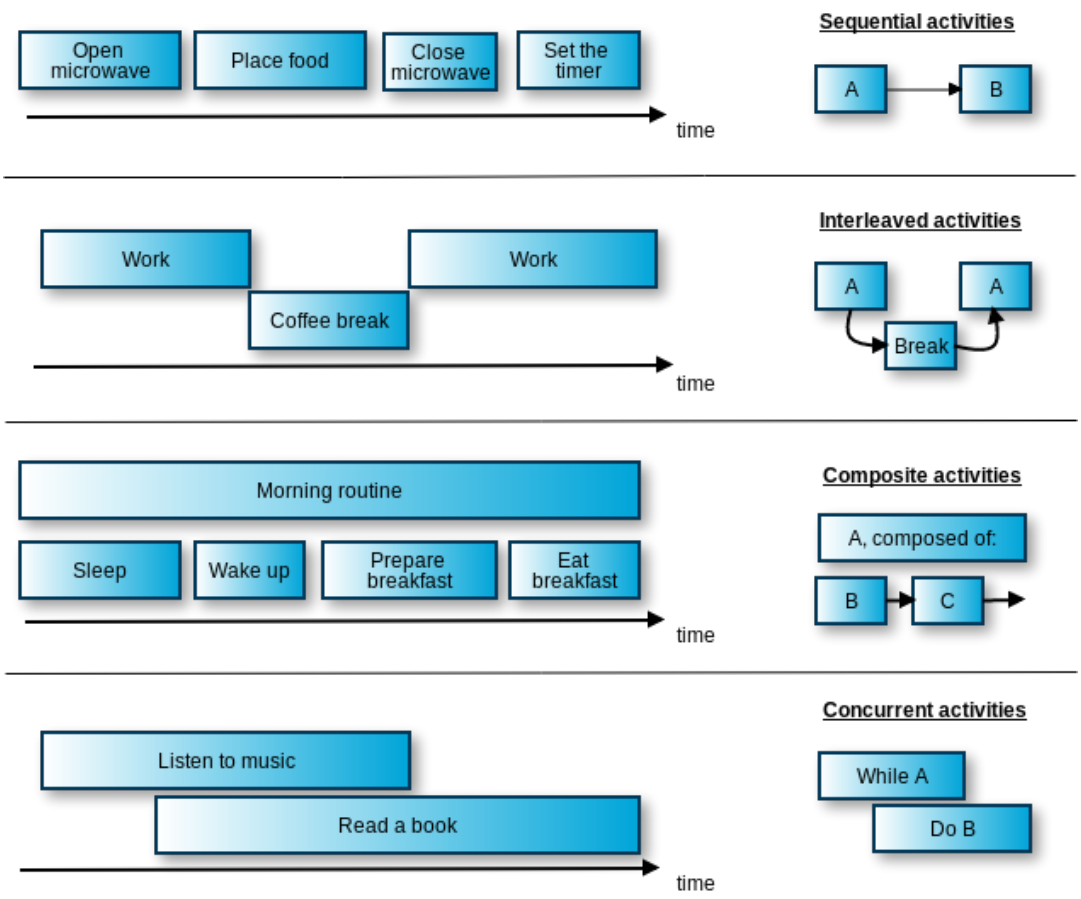


FIGURE 2.1: Relationships between activities: ***sequential activities*** are done one after another; ***interleaved activities*** can be interrupted and their interruption is often explained because of other activities; ***composite activities*** are high level description of a set of activities performed in a lapse of time; ***concurrent activities*** share time intervals, meaning that these activities can happen at the same time **(Ni et al., 2015)**.

This section defines the activities (depending on their granularity level and use case) that will be targeted for this thesis, facilitating the conceptualization and understanding of the problem.

- **Actions *vs* activities:** a notable differentiation between actions and activities was defined by **Liu et al. (2016)**: "actions are defined as primitives that fulfill a function or simple purpose", and "activities consist of a pattern of multiple actions over time".
- **Target activities:** there are two categories of activities that are within the interest of the thesis: activities of daily life (ADL) and locomotion activities. **Ni et al. (2015)** define ADL as the self-care and domestic activities that a patient performs in a daily living *e.g,* feeding oneself, bathing, dressing, grooming work, homemaking, and leisure. As mentioned in **Section 1.1**, monitoring ADLs allows both to have a reference for the estimation of the independent living level of the older adults, as well as being able to recommend activities that might improve a patient's

health status.
On the other hand, locomotion activities are those that involve the global mobility of an individual, and include walking, running, sitting, lying, etc **(Gonz et al., 2017)**. A monitoring of these activities allows to determine the psychological wellbeing and the physical activity level of a patient and can promote them by giving information to the patient about new exercises that could be beneficial.
The *Hôpitaux Universitaires de Genève* (HUG) conducted an ethnographic study for 6 weeks, with 20 senior people at their home places, and found out that there is a basic acceptance and adoption of simple and manageable technology to assist in elderlies' daily life. The authors noticed changes in patient's behavior and usage intention of tracking devices (the study was published in the report of **Konietzny et al. (2018)**).
Finally, monitoring locomotion activities can be useful for detecting hazardous situations such as falling **(Ni et al., 2015)**. Notice that local movements (*e.g.,* moving a hand, opening a door, grabbing an object) are considered as actions, whose sequence of events give information about an activity.

- **Categorization of activities:** Activities and actions may have temporal connections to each other. **Ni et al. (2015)** distinguish three situations: sequential, concurrent, and interleaved activities. Sequential activities happen one after another, concurrent activities happen at the same time, implying that these activities share time intervals. Interleaved activities have time intervals that "preempt" each other, which indicates that a long and complex activity has a long time interval that contains the shorter one. Examples of these situations are shown in **Figure 2.1**.
- **Composite activities:** It is also possible to define higher levels of activities by the composition of several activities. For example, **Figure 2.1** shows that a composite activity denominated "*Morning Routine*" is composed of several subactivities, such as sleeping, waking up, preparing breakfast, and eating breakfast. Notice that basic activities can be directly recognized with sensor data, while the composite activity often needs to be recognized by identifying temporal patterns among the lower level activities to capture patterns that represent the composite activities **(Liu et al., 2016)**.

2.3 Sensor configurations

Depending on how sensors are employed in an environment, there are two main categories of already existing systems for analyzing morning routines of human beings: fixed sensor settings and mobile sensor settings **(Gonz et al., 2017)**.

Fixed sensor settings use static sensors mounted at fixed locations. For example, acoustic, vibrational sensors, location tags, and static cameras are used

to retrieve information about the subjects' actions, activities and the context in which they are performed (*e.g.,* the place where the activity is taking place).

Mobile sensors settings, on the other hand, employ wearable sensors and allow to monitor the activity "from a first person perspective". Nowadays, mobile technologies are present in our daily lives in a non obtrusive way, making wearable technology culturally acceptable and affordable. Mobile sensing also does not restrict the monitoring to a single place, but makes it possible to monitor subjects whenever they may travel. Some mobile sensors commonly employed in recent work are accelerometers, magnetometers, gyroscopes, temperature sensors, electrocardiograms, photoplesthysmograms, electrooculographs, and electromyographs **(Gonz et al., 2017)**.

Camera technologies have been developed to the point that it is possible to capture images with non-obtrusive settings, making them useful for both fixed and mobile sensor settings **(Gonz et al., 2017)**. However, video sequences have several issues when implementing a real time HAR system: privacy, pervasiveness, and complexity **(Lara and Labrador, 2013)**. Not every potential subject for a HAR application is willing to be permanently recorded by cameras and HAR applications that handle sensitive data cannot involve camera recordings, as it increases the complexity of data anonymization techniques. Moreover, video cameras cannot easily be attached to target individuals to obtain images of their entire body during daily living activities. If cameras are used in fixed sensor settings, then the monitored subjects must stay within the perimeter defined by the position and the camera capabilities. Finally, video recordings are computationally expensive, hindering real-time HAR applications where an immediate answer from an event is desired (*e.g.,* predicting that a subject is going to fall down soon).

2.3.1 Time Series Analysis

Sensor data in HAR refer mostly to sets of time series recorded in an environment with body-worn, ambient, and/or object sensors. Time series are collections of n observations (o) taken sequentially (*i.e.,* observations $o_1, o_2, o_3, ..., o_n$ are taken at successive time stamps $t_1, t_2, t_3, ..., t_n$, equally spaced over time, *i.e.,* $|t_1 - t_2| = |t_2 - t_3| = ... = |t_{n-1} - t_n|$). This sequential ordering allows to observe the development of the state of a system's variable (*e.g.,* temperature, acceleration, position) through time. Typical sensor time series feature the following characteristics: large data size, high data dimensionality, and the measurements are continuously updated. Depending on the number of measured variables at each time stamp, time series can be univariate (one variable is measured for each time stamp) or multivariate (two or more variables are measured for each time stamp).

Some methods of time series representation have been developed to reduce the dimensionality (i.e. the number of data points) of raw sensor data. Among those are: re-sampling, *Piecewise Constant Approximation* (PAA), and *Perceptually Important Points* (PIPs).

Re-sampling a time series is the simplest method of dimensionality reduction, where a time series of length n is sampled m times ($m < n$). This method has a drawback of distorting the shape of the compressed time series if the sampling rate is too low. The Nyquist theorem states that if a time series $x(t)$ does not contain frequencies higher than f_n, then $x(t)$ can be completely represented by a sampling no less than $2f_n$.

PAA is an alternative method of sampling, where the series $x(t)$ is also split into m segments, but (in comparison to traditional sampling) the average value of each segment is calculated and used as the re-sampled value for the segment. PIPs are points that are visually important for the identification of patterns in time series data. Each PIP is identified by choosing the point with the maximum perpendicular distance to a line drawn between adjacent PIPs **(Fu, 2011)**.

Knowledge from time series is extracted mostly from two groups of time series analysis:

1. *Time series modeling* approaches decompose time series into their composing characteristics (*e.g.,* trend, seasonal component, and noise), and use them to fit mathematical models to time series (*e.g.,* auto-regressive models, ARIMA, Garch, moving-average) in order to predict future values of the series. Time series models are often used in financial applications and forecasting.

2. *Feature extraction* methods reduce the amount of data to be analyzed by dividing the time series into segments and calculating informative properties (*e.g.,* descriptive statistics, frequency-related characteristics) for each segment. These properties (or features) represent the important characteristics of the raw series and are mostly used as appropriate input for machine learning methods **(Zhu et al., 2017)**.
   Feature extraction is often used for HAR, where large streams of data are segmented and summarized with a collection of informative and non-redundant features that enable the recognition of the activities performed in the time series.

2.3.2 Machine learning backgrounds

Machine learning seeks to transform input data (also known as variables, predictors, features) into knowledge through statistical models that are aimed to understand data by discovering relations and patterns between the variables that conform them. The data that is used to develop models is denominated as **training** set, while the **test** set corresponds to the data that is used only for evaluating the performance of the model or a set of candidate models. The separation into training/testing stages allows to understand and quantify how well does a model perform and how well it might perform in the future with *yet-to-be-seen* data **(Kuhn and Johnson, 2013)**.

Machine learning tasks are divided into two categories *supervised vs. unsupervised*, depending on whether or not there are ground-truth labels associated to the **input** data used to train a model.

Supervised models produce specific outputs based on the inputs: the model learns to associate class labels (*i.e.,* ground-truth) and input features, meaning that there is a prior knowledge about what the output values should be. These models uncover relationships between the data and the desired outputs, allowing them to generalize and associate each feature to the most representative output. On the other hand, *unsupervised* models learn from data that are not labeled (*i.e.,* the input features have not an associated ground-truth). Since there are no labels to associate the input features, these models are destined learn the inherent structure of the data and discover hidden patterns that allow the model to group the input features according to the uncovered patterns **(Hastie et al., 2001)**.

The type of the **output** data in supervised problems differentiates the category of a machine learning problem that it is coped: when the output is a quantitative (numerical) value, then the model performs a *regression*. Contrarily, *classification* problems use models that learn from data whose output has a qualitative value, that is, each input data has associated a category out of a number of categories **(James et al., 2013)**.

2.4 Human Activity Recognition Challenges

HAR shares some challenges with other common pattern recognition problems. These challenges are associated to how robust (*i.e.,* how invariant with respect to the performance) the system is to ***intraclass variability***, ***interclass similarity***, and the ***NULL class problem*** **(Bulling et al., 2014)**.

Interclass variability is present in HAR, because different subjects may execute the same activity in a different way or at a different rhythm (*e.g.,* opening a door with the left of the right hand, drinking at a faster/slower pace), or the same subject could perform one activity in different ways (*e.g.,* the hand and head movement varies when a person drinks with a straw or directly from the glass).

A system that is robust to ***interclass similarity*** should be able to differentiate between activities that possess very similar characteristics of the recorded sensor measurements. For example, a system could confuse activities such as sitting, standing, or lying, since there is not much movement involved and sensor readings from an accelerometer could be similar.

NULL class refers to those activities that are not within the interest of the application. That is, according to **Definition 2**, an activity that does not belong to the set of target activities A. A system that is robust against the *NULL* class problem identifies relevant activities for the application of interest

without confusing it with irrelevant activities with similar patterns. For example, an activity such as *drinking* involves the movement of a hand to an area close to the face. If this is the target activity of a system, then other activities that involve taking the hand close to the face (*e.g.,* touching the face, eating, combing the hair) correspond to the *NULL* class and a robust system is able to not mistakenly confuse these activities with *drinking* events.

Moreover, HAR settings have an additional challenge: ***class imbalance***. For many HAR problems that include long-term monitoring, some target activities may occur often (*e.g.,* locomotion movements, sleeping, working), while others might occur infrequently (*e.g.,* drinking, fall down). Infrequent activities represent a challenge for a system in the training phase, because the classifier does not have enough samples to obtain good generalization capabilities and therefore an optimal performance.

2.5 Publicly available datasets

The aforementioned HAR challenges are addressed by collecting large amounts of data: the *NULL* class and the interclass similarity problems are coped by collecting large samples from the target activities, enhancing the learning stage of the models and allowing them to discover more complex patterns that effectively enables them to differentiate between similar classes and *NULL* class activities. Intraclass variability is addressed by collecting data from several subjects, so there exists a variability on the measurements that can be learned by the model. Nonetheless, the collection of large datasets represents an exhaustive effort that cannot always be attained. Moreover, the ground truth labeling (*i.e.,* the association of a ground truth label to each time stamp of the measurements) is a highly time-consuming task that is mostly done by recording the data collection experiments on video tapes and labeling afterwards the performed activities based on the recordings.

Several research groups in the past few years have recorded data from multimodal sensor environments, attempting to establish *benchmark* datasets in HAR. The creators of these datasets have attempted to simulate HAR environments that reflect real life scenarios, variabilities, and activities. These datasets serve as a standard point of reference for machine learning algorithms applied in HAR, allowing several research groups to apply their algorithms to a common, high-quality dataset and afterwards compare their performance with other groups.

Datasets for HAR are typically recorded in controlled environments, where subjects wear body sensors and perform a series of scripted activities. Several groups have expanded this approach and have included sensors that are attached to objects, as well as ambient sensors that serve to give context about the location of the subject and the characteristics of the surroundings (*e.g.,* time, temperature, humidity).

The choice of the public dataset to use for this thesis was based on the following criteria: quality of the data collection, amount and location of wearable sensors, relevance of the target activities in a real life setting, and quantity of test subjects that participated in the data collection runs. The following list enumerates the considered HAR datasets and gives a brief description. A more detailed summary can be found in the **Appendix A**.

- The DaLiAc and WARD datasets contain a collection of 13 locomotion-related activities using body-worn sensors **(Leutheuser et al., 2013; Yang et al., 2009)**.
- The OPPORTUNITY dataset has a more limited set of locomotion activities but includes hand gestures and high level activities in a setting with both body-worn and object sensors, simulating a scenario of activities of daily life (ADL) **(Chavarriaga et al., 2013)**.
- The PAMAP dataset focuses on locomotion activities using body-worn sensors and includes a heart rate monitor in the measurements to address the physical intensity of an activity **(Reiss and Stricker, 2012b)**.
- The HAR dataset focuses on recognizing some physical activities by measuring data from only one smartphone **(Anguita et al., 2013)**.
- The REALDISP dataset attempts to simulate some of the variability that may occur in the day to day usage of sensors by inducing in some measurements a degree of displacement of body-worn sensors. It focuses on recognizing physical activities **(Baños et al., 2012)**.
- The HHAR (Heterogeneity HAR) study analyzed various heterogeneities in motion sensor-based sensing (*i.e.,* sensor biases, sampling rate heterogeneity, and sampling rate instability) and their impact on HAR by sensing a set of activities with 13 different smartphones **(Stisen et al., 2015)**.
- The AReM dataset measures the Received Signal Strength (RSS) between body-worn sensors in an experiment focused on recognizing physical activities **(Palumbo et al., 2016)**.

The OPPORTUNITY dataset stands out above the rest. The dataset presents the richest sensor environment that was used in a data collection experiment, includes many different types of activities, activities of different complexity, and both the data collection process and the ground truth labeling were performed in a detailed an rigorous way[1] by recording each of the experimental runs to be labeled *a posteriori* with a labeling tool **(Chavarriaga et al., 2013)**. **Figure 2.2** shows an instance of the video recordings[2] corresponding to the OPPORTUNITY dataset.

[1]"A 30-minutes video footage required about 7-10 hours to be annotated" **(Roggen et al., 2010)**

[2]Roggen, Daniel. The OPPORTUNITY dataset. Video available at https://vimeo.com/8704668

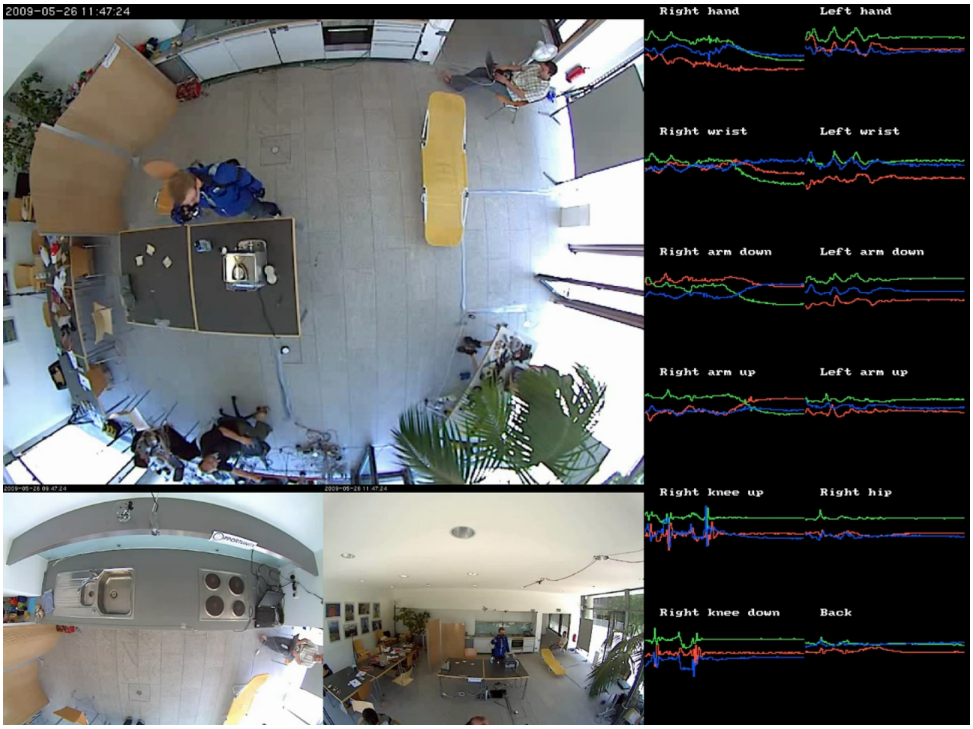


FIGURE 2.2: Video recordings of the OPPORTUNITY dataset. These recordings were used to label the activities after the experiment was performed. The subject is performing the middle-level activity "drinking" and sensor data is shown on the right.

Additionally, while some datasets were focused on simulating HAR challenges by inducing intentional displacements on sensors' positioning (REALDISP), measuring the intensity of a physical activity (PAMAP), introducing non-standard sensing methods of human actions (AReM), the OPPORTUNITY dataset simulates in a laboratory setting one of the four main research targets in HAR: activities of daily life. The OPPORTUNITY dataset simulates an ADL routine (specifically, a morning routine), which goes in line with the objectives established in the REACH project. Furthermore, this dataset includes a target activity, *drinking*, whose monitoring is relevant within the project in order to track the hydration habits of an elderly patient. This dataset can also serve to construct initial machine learning models for the target activities present in the dataset, that can be afterwards complemented with particular data collection experiments made for REACH. A summary of the OPPORTUNITY dataset is given in **Section 4.1**.

Chapter 3

State-of-the-art

3.1 The Activity Recognition Chain (ARC)

An Activity Recognition Chain (ARC) is a standard set of signal processing, pattern recognition and machine learning techniques that are implemented in a specific activity recognition system **(Bulling et al., 2014)**. This sequence of processing principles is followed by most researchers in HAR to interpret human activities from sensor data **(Roggen et al., 2011)**.

The ARC is illustrated in **Figure 3.1**: a set of sensors deliver a stream of unprocessed signals, which represent the magnitude measured. Disturbances are removed through a preprocessing step and new meaningful predictors can be calculated. The dynamics of the signals are captured by segmenting the data into fixed-size windows and a set of features are calculated for each window. Afterwards, a feature selection process selects the features that will be provided as input to a classifier that is responsible to recognize the performed activity within each time window, deciding for one of the k activities that are targeted. In the following, the stages that comprise an ARC are explained in more detail.

3.1.1 Sensor data acquisition

The raw data are acquired by sampling signals from sensors that measure the physical conditions of the subject who is target of the HAR application, and sometimes its surroundings. The sensors transform these samples into numerical values that are processed in a computer. *Wearable sensors* are attached to the body of the subject and measure the physical motions that the subject executes; *object sensors* are attached the surface of an object and are destined to give information about the interaction of the subject with the object; *ambient sensors* are sensors located in the environment and measure information that serve as context provider of the subject's activities or surroundings.

Missing values (abbreviated as ‘‘NA’’ or ‘‘NaN’’) occur when there is no measurement stored for an observation. These phenomena are not only tied to the sensor performance, but also to the communication between the sensor and the devices that are destined to collect the data **(Roggen et al., 2010)**.

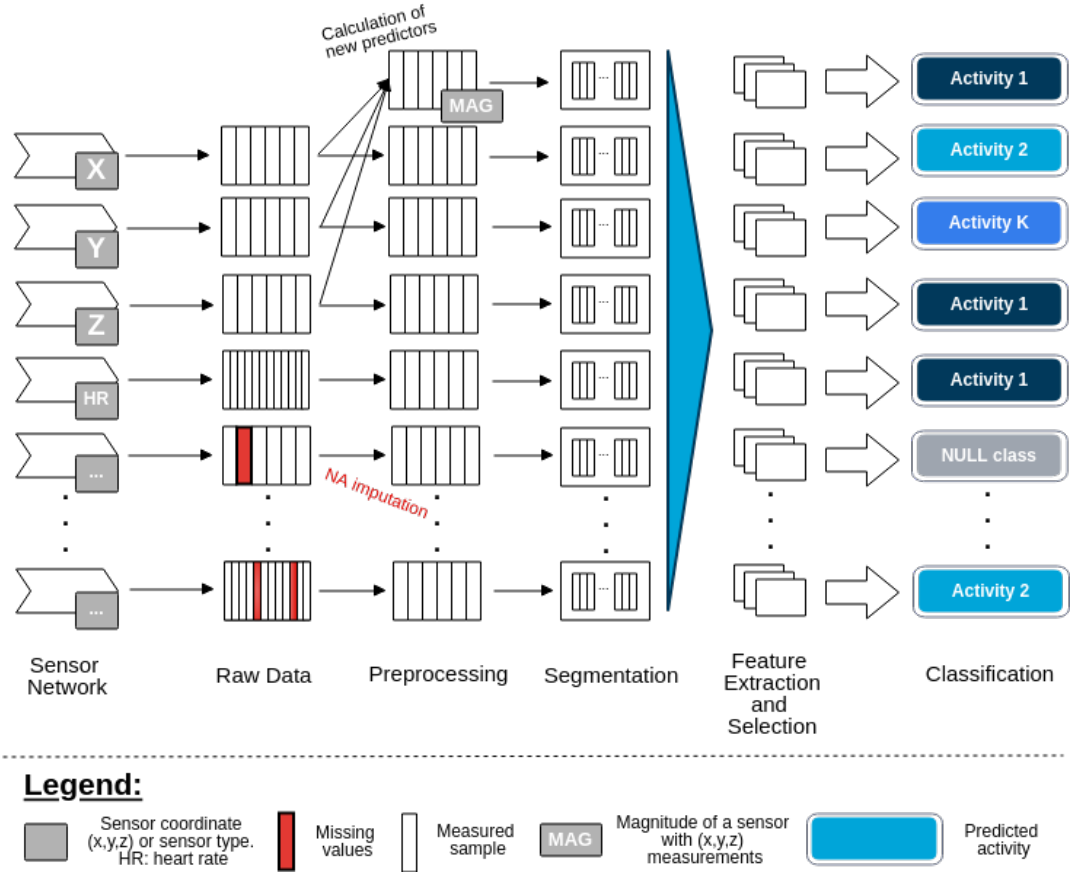


FIGURE 3.1: Activity Recognition Chain (ARC). It comprises stages for data acquisition, preprocesssing, segmentation, feature extraction, and classification.

3.1.2 Preprocessing

The representation and quality of the sensor data affect directly the performance of the ARC. In this stage, the sensor recordings may undergo several data transformation processes oriented to eliminate redundant information and improve the quality of the data **(Kotsiantis et al., 2006)**.

Figure 3.1 shows an example of some preprocessing steps done to raw data in a multi-sensor environment: raw data is obtained from a sensor network where several sensors possess different sampling rates (shown as individual squares in the *raw data* section of the figure) and missing values (highlighted in red). Once the preprocessing is done, sensors possess one common sampling rate by aggregating the data from those sensors with higher sampling rate, `NA` values are imputed, and the calculation of the magnitude vector from tri-axial measurements is performed.

Most of preprocessing techniques belong to one of the following groups:

- Filtering techniques are applied to the raw data recordings to remove any noise and artifacts present into the sensor recordings (*e.g.,* using a *Butterworth notch* filter to eliminate the power-line noise [60 Hz]).

- Data cleaning techniques impute the missing values present in the data. If several sensors gather data from a physical property (*e.g.,* acceleration, magnetic field), in this step the data is converted into a common metric system, so the units of measurement among the sensors are the homogeneous. Additionally, corrupt or irrelevant parts of the data that cannot be used in further stages in the ARC are removed.
- Sensor data fusion calculates new variables by gathering information from existing sensor data. For example, tri-axial measurements of a physical variable (*i.e., (x,y,z)*) can be combined to generate a magnitude vector.
- Synchronization and resampling transform sensor recordings from multi-sensor networks that possess heterogeneous sampling rates into homogeneous rates.

Missing value imputation

Many time series analysis methods require missing values to be replaced with reasonable values that describe the time series in such respective time points. This process of replacing missing values is called **imputation**. Missing value imputation techniques such as *Expectation-Maximization, Hot Deck, Nearest Neighbor, and Multiple imputation* rely on correlations between attributes in order to estimate the values from missing data **(Moritz and Bartz-Beielstein, 2017)**. However, univariate time series do not possess more than one attribute, preventing to use these algorithms directly. Furthermore, even though several sensors are capable of recording values along multiple reference axes *e.g., (x,y,z)*, if a sensor fails to record a measurement, it fails on doing it in each axis, making it impossible to use the aforementioned methods to impute `NA` values. Hence, univariate `NA` imputation methods are of interest within HAR, such as **(Moritz and Bartz-Beielstein, 2017)**:

- Interpolation.
- Kalman smoothing.
- Imputing `NA` values with the mean of the time series.
- Splitting the time series into seasons and perform interpolation separately on each resulting time series.
- Imputation using a weighted average method.
- Last observation carried forward.
- Next observation carried backwards.

Calculation of new predictors

From the raw signals it is also possible to calculate new predictors that might give additional information to the classifier. **Zhu et al. (2017)**, **Shoaib et al.**

(2013), and **Stisen et al. (2015)** suggested to calculate the magnitude vector from tri-axial acceleration sensors:

$$acc_{Mag} = \sqrt{acc_x^2 + acc_y^2 + acc_z^2} \tag{3.1}$$

Additionally, **Zhu et al. (2017)** and **Kavitha and Sijo Antony (2017)** calculated the jerk *i.e.,* the rate of change of acceleration, from accelerometers data.

3.1.3 Data Segmentation

The gathered sensor data form time series that represent the evolution of the measured physical parameters through the recorded time. During that time, the subjects whose sensor data were measured might have performed several activities and this stage of the ARC divides the time series into segments that might contain an action (or activity) of interest. This segmentation is not trivial, as humans fluently execute activities and sometimes they are not clearly differentiated. The latter is also tied to the strict definition of an activity: suppose for example that the movement of a human is being tracked and the goal of the ARC is to recognize such movements. If a person is standing up and then sits down, how to determine when did the person stop standing up and began sitting down? Does it happen in the moment the person leans towards the chair, or rather when the person is completely supported in the chair? Does this transition time belongs to either of those activities or should it be considered as *NULL* class? These questions are answered before performing the data collection experiments and the answers influence directly the data segmentation.

Some methods have been implemented to segment the recorded time series in HAR problems:

- *Context-based* segmentation: segmentation is done using external context sources that indicate the beginning and the end of an activity. GPS recordings may estimate the velocity of displacement of a subject and that information might give an estimate of when a person is running, walking, riding a bicycle, driving a car, etc. Environment sensors (*e.g.,* microphones, switches attached to objects) may also indicate the start and end times of an activity.
- *Energy-based* segmentation: many activities in HAR have distinguishable intensity levels (*e.g.,* standing, walking, jogging, and running) that influence directly the amplitude of the recorded time series. This approach segments the activities by finding transitions between particular values of energy that indicate the end of an activity and the beginning of another one **(Bulling et al., 2014)**.
- *Sliding window*: a fixed-size moving window extracts several segments over the time series data, leaving no inter-window gaps. The window size

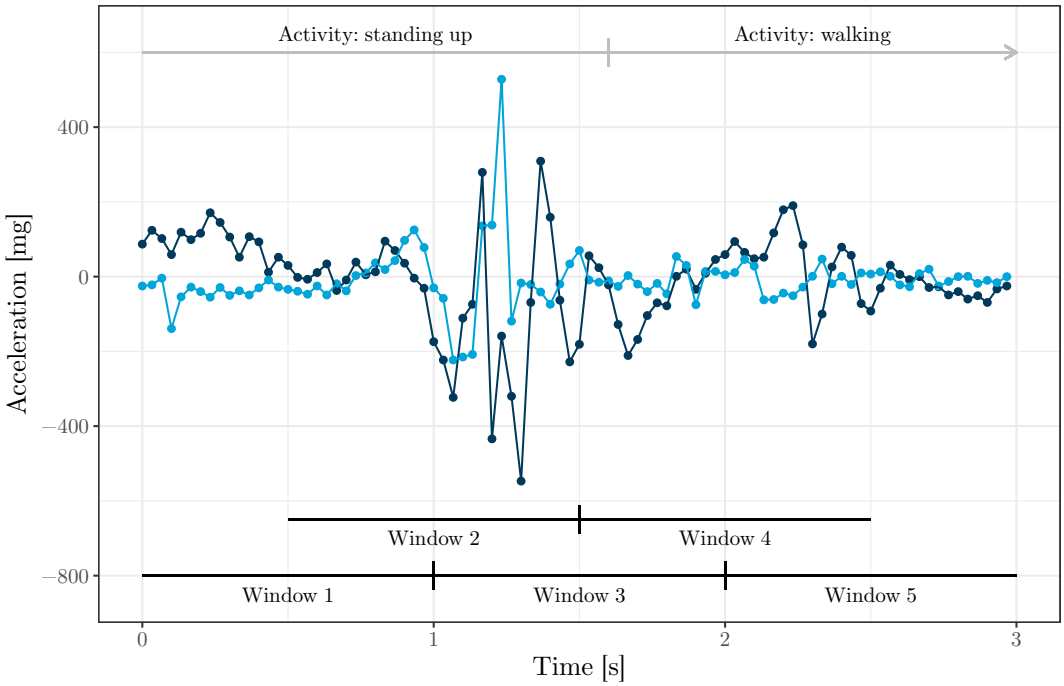


FIGURE 3.2: Segmentation using fixed sliding windows of 1 second length and 50% overlap.

determines on the amount of data per segment that will be used to perform the feature extraction, the recognition performance, and also the delay of the recognition system **(Bulling et al., 2014)**. *Overlapping* between two consecutive windows is done to capture the dynamics that might happen at the end of one window and the beginning of the consecutive window, leading to statistical dependence between consecutive windows that should be taken into account. For example, an overlap of 50% means that two consecutive windows will share 50% of the data. Notice that this approach corresponds to the method used to solve the *relaxed HAR problem* **(Definition 2)** and this approach is the most widely employed segmentation technique in activity recognition **(Banos et al., 2014)**. Therefore, the rest of this discussion will focus on this approach.

Figure 3.2 shows an example of segmentation with fixed time windows. Two time series are composed of sampled measurements (shown as bulletpoints in the figure) of two attached accelerometers to a subject while he was performing locomotion activities. The time series are segmented into fixed sliding windows of 1 second length (*i.e.,* 30 samples per window) and a 50% overlap between consecutive windows is introduced to capture the dynamics between two consecutive windows. Notice for example that the window 2 captures the decrease in acceleration in both sensors, while this behavior in window 1 and 3 can pass unperceived.

Deciding for a window length is not a trivial matter. Several approaches have been proposed in the literature and there is no clear consensus on which window size should be used. Small window sizes allow for a faster activity detection,

TABLE 3.1: Review of fixed sliding window approaches

Window Length	Overlap	References
-	50%	Guo et al., 2016
500 ms	50%	Chavarriaga et al., 2013
24 samples	50%	Plötz et al., 2011
1 s	0%	Bannach and Lukowicz, 2008 Bulling et al., 2014 Karantonis et al., 2006
1 s	50%	Liu et al., 2017
2 s	50%	Stisen et al., 2015 Zhao et al., 2014 Wen and Wang, 2017
2.56 s	50%	Kavitha and Sijo Antony, 2017 Anguita et al., 2013
3 s	66%	Zhu et al., 2017
5 s	50%	Leutheuser et al., 2013 Lara et al., 2012
5.12 s	4.12 s	Reiss and Stricker, 2012a
6 s	0%	Baños et al., 2012
6.7 s	50%	Bao and Intille, 2004
12 s - 20 s	50%	Lara et al., 2012

reduced resources and energy needs, because the system does not have to gather large amounts of samples to make a prediction. On the other hand, large windows are normally considered for the recognition of complex activities, as the system needs to gather more data to identify distinctive patterns that lead to a correct detection **(Banos et al., 2014)**.

Table 3.1 shows several combinations of window length and overlapping that have been used in HAR's related work. It can be seen that most of the related work used a window length between 1 and 2 seconds. Moreover, an overlap of 50% is mostly used among these works. **Banos et al. (2014)** tested different window lengths (from 0.5 s to 7 s in steps of 0.25 s) in the REALDISP dataset **(Section 2.5)** and found that the the interval $t_w = [1s, 2s]$ provides the best trade-off between recognition speed and accuracy for on-body activity recognition systems.

3.1.4 Feature Extraction

The feature extraction stage computes from the time series segments N key signal characteristics (*i.e.,* features), reducing a complete time series segment into a vector of length N, that is, a time series segment is now represented as one point in an N-dimensional space, known as the *feature space*. Representing time series as feature vectors has been one of the most common and convenient means

to represent data for classification problems **(Guyon and Elisseeff, 2006)**. Additionally, this technique reduces the amount of information (in terms of data size) that will be given as input to the classifier, making it possible to the machine learning module to learn the relevant information of raw signals by retaining the dynamics of the time series. **Table 3.2 (a)** shows the most used features in HAR. Statistical features, *e.g.,* mean, variance, kurtosis, interquartile range, are popular across HAR problems due to their simplicity as well as their performance **(Bulling et al., 2014)**. Notice that *mean* and *standard deviation* are used in all reviewed HAR works.

Ideally, the features that correspond to one activity should cluster together in the feature space (*i.e.,* the N-dimensional space where the features vectors are mapped into), while different activities should be located far apart. Note that the higher the dimensionality of the feature space, the more training data is needed for model parameter estimation and the more computationally intensive the classification becomes **(Bulling et al., 2014)**.

Feature Selection

Feature selection methods are destined to reduce the dimensionality of the feature vectors by choosing the most relevant features among the complete feature set. By selecting a reduced set of relevant features, the generalization capabilities of a classification model are enhanced, both the time that the learning process takes and the complexity of the problem are reduced, and the interpretability of a model is increased **(John et al., 1994)**.

Most feature selection methods can be categorized within one of the following two groups:

1. **Filter** methods use statistical techniques that evaluate the relationship between each feature and the ground truth labels. Then, only the features that pass all statistical criteria, or those that have the highest scores are selected. Filter methods are computationally efficient, although the selection criteria are not related to how well the model performs, but to statistical criteria. It might happen that if a feature passes all tests, then all its highly correlated features would be selected as well, decreasing the performance of the model **(Kuhn and Johnson, 2013)**.

2. Contrarily, **wrapper** methods evaluate the performance of a model by selecting *subsets* of the features in order to find the optimal number and combination of features that maximize the model's performance. These methods use information from models that allow to rank the *importance* of each feature and then several models are built for each subset of features and the subset with the highest performance is selected. Since one model is tested for each subset, wrapper methods are computationally expensive **(Kuhn and Johnson, 2013)**.

TABLE 3.2: Review of calculated features and classifiers in HAR

(a) Calculated features

Feature	Used in	Feature	Used in
Mean	[1 - 15]	Fourier coeff.	[4, 9, 10, 14]
St. Deviation	[1 - 15]	Cepstral coeff.	[4]
Median	[6, 7, 13, 14]	Spectral Entropy	[3, 4, 12, 13]
Root mean square	[1, 6, 7, 14]	Energy	[4, 12, 13, 14]
Range	[7]	Signal Magnitude Area	[3, 7, 15]
50% Quantile	[2]	Autocorrelation	[10]
Skewness	[2, 3, 10, 15]	Spectral centroid	[11]
Kurtosis	[2, 3, 10, 15]	Bandwidth	[11]
Max and min	[2, 3, 7, 10, 11]	Correlation	[3, 13]
Mean absolute diff.	[3, 5]	AR coefficients	[3, 15]
Interquartile range	[3, 15]	Zero-crossing rate	[1, 2, 4, 6, 7, 14]

(b) Classifiers

Classifier	Reference
Support Vector Machine	[1, 2, 4, 7, 11, 14]
K-nearest neighbors	[1, 5, 7, 8, 10, 11, 13, 14]
Random Forest	[2, 3, 15]
Naïve Bayes	[4, 7, 10, 13, 14]
Discriminative Analysis	[4]
Hidden Markov Models	[5, 9]
AdaBoost	[6, 11]
Decision Tree	[7 - 10, 12 - 15]

(c) References

#	Reference	#	Reference	#	Reference
1	Sano and Picard, 2013	2	Guo et al., 2016	3	Kavitha and Sijo Antony, 2017
4	Bulling et al., 2014	5	Yang et al., 2014	6	Wen and Wang, 2017
7	Liu et al., 2017	8	Baños et al., 2012	9	Rashidi and Mihailidis, 2013
10	Rahman et al., 2017	11	Leutheuser et al., 2013	12	Bao and Intille, 2004
13	Reiss and Stricker, 2012a	14	Shoaib et al., 2013	15	Zhu et al., 2017

3.1.5 Classification

In supervised machine learning tasks such as HAR, a machine learning algorithm associates every feature vector to one label corresponding to the performed activity during the time window. Initially, machine learning models are built with *training data*. In that process the model uncovers patterns within the data, associates these patterns with the corresponding ground truth labels, and generalizes these patterns in order to make predictions on new, unseen data (*i.e., testing data*). According to the machine learning model, the output of a classifier can be either discrete (*i.e.,* the classifier outputs one ground truth label per window) or probabilistic (*i.e.,* the classifier outputs the likelihood of the time window to belong to one of the target activities and the activity with the highest likelihood is selected) **(James et al., 2013)**. **Table 3.2 (b)** summarizes the implemented classifiers in several reviewed HAR works.

In general, model complexity is related to the number of free parameters (or degrees of freedom) used in a model: the more free parameters a model has, the more complex the model is. Since model building is done on the *training data*, highly complex models lead to a low error on the training data, but could incur an increase of the error with the *testing data*, since the high model complexity may cause the model not to generalize common patterns of the data, but to memorize the *training data* instances. This behavior is known as *overfitting* **(Kuhn and Johnson, 2013)**.

Decision trees

The feature extraction stage of the ARC yields a collection of $N-$dimensional vectors $X = \{X_1, X_2, ..., X_N\}$, where each dimension corresponds to a feature calculated on a single time window. The space where these vectors exist is known as the *feature space*.

A *decision tree* divides the feature space into regions, where each region will have assigned a ground truth label that will correspond to the *prediction* done for any feature vector that is located within that region. The division of the feature space is done by sequentially splitting the feature space into regions that minimize the classification error for the training data **(James et al., 2013)**.

Initially, the feature space is a homogeneous region and a partition on a predictor (feature) X_i is done by selecting a cutpoint s that splits the region into two parts: $R_1\{X|X_i > s\}$ and $R_2\{X|X_i \leq s\}$. Then, each region will have assigned a ground truth label that corresponds to the most frequent label among all the feature vectors contained in that region. Afterwards, the *Gini* index is calculated for each region:

$$G = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}) \tag{3.2}$$

where p_{mk} is the proportion of training observations in the m_{th} region corresponding to the k_{th} class[1].
This process is repeated by variating s and the predictor X_i, and the optimal cutpoint that generates the first split in the feature space is such that yields in the maximal decrease of the Gini index G **(Louppe et al., 2013)**:

$$\Delta G(s, R) = G(R_t) - p_{R1}G(R_1) - p_{R2}G(R_2) \tag{3.3}$$

where R_t represents a region of the feature space (without splits), R_1 and R_2 are the two parts in which the region R_t is split in the cutpoint s, and p_{R1} and p_{R2} are the proportion of samples in each split with respect to the total of the region R_t.
These steps are sequentially repeated by partitioning the already defined regions of the feature space, until a desired number of regions is reached, or a stopping criterion is satisfied.

Each split in the predictor space increases the complexity of the model. Trees with a high number of splits can lead to an overfit because the feature space would be partitioned to represent only the training data. There are techniques to avoid overfitting by applying a cost α to trees with large numbers of splits **(James et al., 2013)**.

Decision trees suffer from *high variance*, that is, that modeling two datasets that follow the same probability distribution might lead to decision trees with big differences in the partitions of the feature space. The contrary happens with *low variance* models, such as logistic regression. This problem can be addressed by aggregating many decision trees, as the *random forest* algorithm does.

Random Forest

As mentioned earlier, decision trees suffer from high variance. This limitation can be overcome by modeling several decision trees from the training data and taking a majority vote from the decision trees in order to make a prediction. Each tree requires its own training data and the modeling of multiple decision trees would involve extremely large datasets. However, the *bootstrap* overcomes this limitation and allows to generate several datasets by repeatedly sampling observations, with replacement, from the training dataset **(James et al., 2013)**.

A random forest algorithm initially generates D datasets via bootstrap and trains one decision tree per dataset. However, each random tree only considers a random sample of predictors when creating a split, and this is done to avoid

[1]The Gini index is preferred as a measure of error rate for classification problems rather than the classification error (*i.e.,* $E = 1 - \max_k(p_{mk})$) because the Gini index not only takes a small value when p_{mk} is close not only to 1.0 (as E does), but also when p_{mk} is close to 0. This characteristic is special because the Gini index does not only serve as a measure of error, but it measures the *purity* of a single region: a small value of G indicates most of all observations enclosed into one region belong to the same class **(James et al., 2013)**.

that strong predictors dominate how splits are done in the decision trees and therefore end up being similar within each other. Normally, each split considers the square root of the total predictors **(James et al., 2013)**.

Since bootstrap generates new training datasets by sampling with replacement, not all observations (*i.e.,* time windows) are considered for training. In fact, approximately one third of the observations are *left out* via bootstrap **(James et al., 2013)**. These observations are called *out-of-bag* (OOB) and can be used to estimate the test error without need of resampling methods such as cross-validation. OOB error is measured by predicting the ground truth label of each time window using **only** the trees that left such window 'out of bag'.

Additionally, **Equation 3.3** can be also used to measure the *importance* of each feature. Each time that a feature is selected to create a split in a tree, the amount of reduction in **Equation 3.3** is stored and averaged over the total number D of trees generated. The higher the total amount of reduction, the more the feature was used to create splits in the trees and the more relevant it is for the model.

Recursive feature selection for random forests

Random forests estimate the importance of each feature in the training process. This ranking of important features can be used to train models with only a subset of important features and estimate the test error.

Recursive feature selection is a wrapper method **(Section 3.1.4)** and it is implemented as follows **(Kuhn and Johnson, 2013)**:

1. Train a random forest with all features.
2. Rank the features F according to the variable importance: $VarImp(F_1) > VarImp(F_2) > ... > VarImp(F_N)$
3. Select M subsets of features of size $S_i, i = 1, 2, ..., M$, and train the model with the first S_i ranked features.
4. Calculate the OOB error as an estimate of the test error.
5. Select the subset with the lowest OOB error.
6. Apply the best performing model to the test set.

3.2 Related work

This section presents related work in the HAR field, specifically in ADL and healthcare monitoring applications.

Zhu et al. (2017) used the REALDISP dataset and OPPORTUNITY dataset to test several normalization and feature extraction strategies, aimed to find the best performance and robustness of a classifier, specifically, a random forest.

The authors extracted a collection of temporal-based features and frequency-based features and found that the system performed the best using temporal-based features only (followed by a system that used all the features and finally a system that used frequency-based features only).
Additionally, **Zhu et al. (2017)** studied the influence of the type of sensor on the system performance: the REALDISP dataset uses IMUs that output 17 signals from accelerometers, gyroscopes, magnetometers, and quaternion sensors. The authors tested systems using data from only one sensor and found that the magnetometer data provided the best performance, followed by the accelerometer data, gyroscope data, and finally the quaternion data. Moreover, the system achieved the best performance using data from all sensors.
However, **Zhu et al. (2017)** used an arbitrary window size of 3 seconds and an overlap of 66% and did not study the influence of these parameters on the system's performance.

Guo et al. (2016) collected data from motion sensors embedded into one smartphone (*i.e.,* accelerometer, gyroscope, magnetometer, gravity accelerometer, and linear accelerometer) in order to recognize physical activities. The authors presented a coordinate transformation method that rotated all data into the earth's coordinate system with the aim to manage problems due to the variation of the orientation of the smartphone when recording data.
Their system targets motion activities such as walking, running, going upstairs, going downstairs, standing, and sitting. On top of this system, the authors developed a self-learning scheme. This scheme works with a novelty detection algorithm that maps new samples from data into a space where the samples receive a 'novelty score' based on the distance between each sample and the known classes. If this score surpasses a threshold, then the sample is considered to be a 'new activity' and the data is imported into a pool of unknown activities. When the pool of activities accumulates enough data, a clustering algorithm creates groups based on the data and afterwards these groups become new classes that are used to update the system's classifier.
The limitation of this approach is that, even though unsupervised methods such as clustering are able to generate new classes based on a pool of unknown data, these new classes are labeled as 'new class 1', 'new class 2', etc., since there is no prior knowledge about the kind of activity that the data is representing. However, the 'novelty score' strategy would represent a good method to determine a-priori (*i.e.,* before classification) if a sample belongs to the *NULL* class.

Rahman et al. (2017) gathered data from tri-axial accelerometers located in the chest, wrist, ankle, and hip of eight volunteers. Additionally, the volunteers carried an ECG in order to measure heart rate data and performed five ambulatory activities. The author tested the performance of each accelerometer separately (without heart rate measurements)and found that the data measured by the sensor located in the ankle provided the best performance. However, when the heart rate data was included, there was an overall increase in the performance, showing that the classification stage can benefit from both physical and physiological signals. Nonetheless, the data collection setting is limited, as there was only one run recorded per volunteer and the pool of target

activities is very limited.

3.3 Performance measures

In the *training* stage, several configurations of the ARC and parameters for the involved classification methods are tested, and the best-performing configuration is then used with the *testing* data in order to estimate how well the model predicts the performed activities in a real setting. The testing dataset is used only once through this process and the performance in this dataset cannot be taken in consideration for choosing the best ARC and best parameters because this choice would be *biased* towards the testing dataset and not an actual real-life implementation. Besides, the testing dataset needs to be sufficiently distinct from the training dataset in order to reflect the variability that can be present in the measurements in a real-life implementation. If the testing set is similar to the training set, then the results might overestimate the actual performance in a real setting.

The performance of a model, both for *training* and *testing*, can be evaluated with metrics such as accuracy (*i.e.,* the ratio of the number of correct predictions relative to the total number of input samples.), F-measure, and Area Under the ROC Curve **(Hastie et al., 2001)**. Most performance metrics used in HAR come from the field of *pattern recognition*. Therefore, they judge the overall performance of the classifier but fail to capture common artifacts found in HAR; specifically, event fragmentation, event merging, and timing offsets. **Ward et al. (2011)** developed two new performance metrics specialized for HAR, overcoming these limitations of the classic methods, allowing to achieve a better interpretation of the results.

3.3.1 Confusion Matrix

The selection of the best performing ARC is done based on criteria extracted from a confusion matrix, an $A \times A$ matrix, where A indicates the number of classes to classify.

- *True negatives (TN)* indicate the number of feature vectors that did not correspond to the target class and that were not classified as the target class.
- *False Negatives (FN)* are the target class' feature vectors that were not classified as the target class.
- *False Positives (FP)* are feature vectors that do not correspond to the target class, but were classified as the target class.
- *True Positives (TP)* are the target class' feature vectors that were classified as the target class.

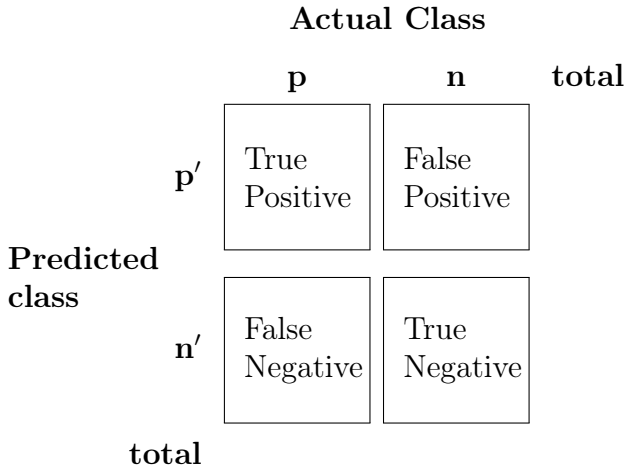


FIGURE 3.3: Confusion Matrix.

A 2-class confusion matrix is shown in **Figure 3.3**. Notice that the diagonal of this matrix indicates the number of cases when the classifier was successful. From this matrix, some important rates can be calculated:

- **Accuracy:** describes how often the classifier was correct, and it is calculated as:

$$Accuracy = \frac{TN + TP}{TN + TP + FP + FN} \tag{3.4}$$

- **Recall:** measures the proportion of elements belonging to the target class that were correctly classified:

$$Recall = \frac{TP}{FN + TP} \tag{3.5}$$

- **Precision:** measures how often the classifier is correct when it predicts that a feature vector belongs to the target class:

$$Precision = \frac{TP}{TP + FP} \tag{3.6}$$

- **False positive rate (FPR):** measures the proportion of negative cases predicted incorrectly (*i.e.,* false positives):

$$FPR = \frac{FP}{FP + TN} \tag{3.7}$$

3.3.2 F-measure

The F-measure (also named F1-score) takes into account the precision and recall for each class and can provide a better assessment of performance than computing the accuracy. The F-measure is mathematically the harmonic mean

between precision and recall. For multi-class problems, such as HAR, classes are weighted according to the sample proportion in order to counter class imbalance **(Chavarriaga et al., 2013)**:

$$F_1 = \sum_i 2 * w_i \frac{Precision_i \cdot Recall_i}{Precision_i + Recall_i} \tag{3.8}$$

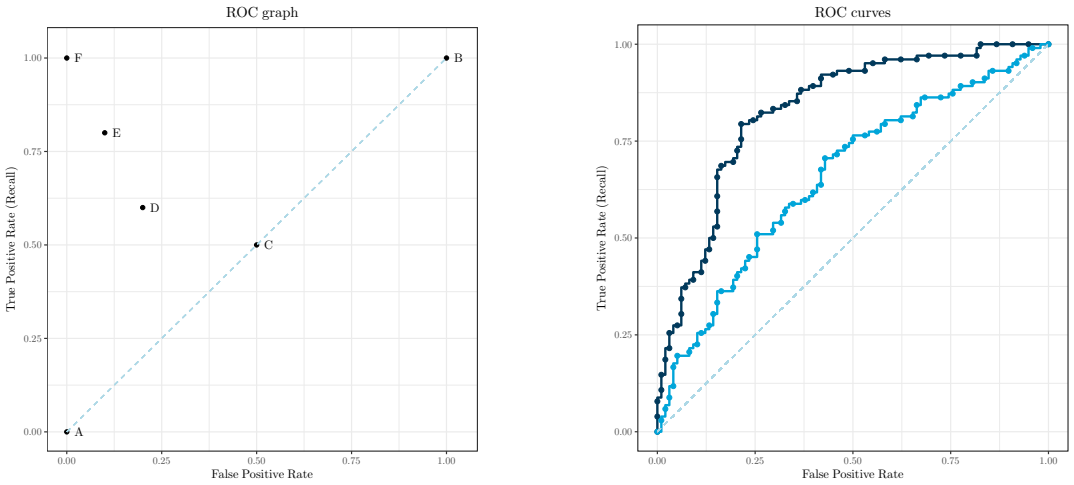
where i is the class index and $w_i = n_i/N$ is the proportion of samples of class i (with n_i being the number of samples of the ith class and N being the total number of samples).

3.3.3 Area Under the ROC Curve (AUC)

A receiver operating characteristics (ROC) graph is a two-dimensional plot in which *recall* is plotted on the Y axis and FPR is plotted on the X axis. These graphs are used for visualizing, comparing, organizing and selecting classifiers based on their performance **(Fawcett, 2006)**. ROC graphs make visible the trade-offs between benefits (TP) and costs (FP) between classifiers: on **discrete** classifiers, the performance of a classifier, summarized in a confusion matrix, can be mapped into the ROC graph as a point. **Figure 3.4a** illustrates a ROC graph and several classifiers (**A,B,C,D,E,F**). Considering these classifiers, it is possible to note several important points in the ROC graph: the classifier **A** represents the strategy of never issuing a positive classification, meaning that this classifier never commits false positive errors but it does not obtain true positives as well. The opposite strategy is done by the classifier **B**, which assigns positive classifications to *all* elements. The classifier **C**, as well as any classifier located through the blue dashed line, represent the strategy of randomly guessing a class. Classifiers located on the triangle above the blue dashed line perform better than random guessing. Classifier **F** has a perfect performance, and the closer a classifier is to this coordinate $(0, 1)$, the better the classifier is (**E** performs better than **D**). Conceptually, on **probabilistic** classifiers, by varying the threshold from $-\infty$ to $+\infty$ that allows to assign an observation to one class, it is possible to trace a curve that shows the performance of the classifier for each value of the threshold **(Fawcett, 2006)**. **Figure 3.4b** shows the performance of two classifiers, where the classifier depicted in dark blue color performs better than the other one.

ROC graphs are a two-dimensional description of a classifier's performance. The area under the ROC curve is often calculated to reduce the ROC performance to one scalar that represents the classifier's performance **(Fawcett, 2006)**. Note that a perfect performance yields an AUC of 1, while random guessing (the area under the blue dashed line in **Figure 3.4**) yields an AUC of 0.5). For multi-class problems, a weighted AUC can be calculated as:

$$AUC_{total} = \sum_i w_i AUC_i \tag{3.9}$$



(A) Discrete classifiers (B) Probabilistic classifiers

FIGURE 3.4: ROC graphs. Performance of discrete classifiers is shown on a ROC graph as points (A), while probabilistic classifiers are shown as curves (B)

where AUC_{total} indicates the overall performance, w_i is the weight for the i_{th} class (same as **Equation 3.3.2**), and AUC_i is the AUC for the i_{th} class **(Chavarriaga et al., 2013)**.

3.3.4 Event and Frame analysis

Standard evaluation metrics used in HAR do not characterize adequately the performance of a classifier, since the information about typical characteristics of activity events are ignored in order to fit the results into metrics such as accuracy, F-measure or AUC. In order to overcome that, **Ward et al. (2011)** developed specialized HAR performance metrics that lead to an unambiguous explanation of the performance of a classifier.

These metrics analyze multi-class problems by considering one class at a time. That is, the problem is transformed to a 2-class problem where the positive class is represented by the target activity and the negative class is represented by the rest of the classes. Moreover, there are two analyses that are done: event analysis and frame analysis.

Frame analysis

Following the definitions by **Ward et al. (2011)**, a *frame* is "a fixed-length, fixed-rate unit of time, which is often the smallest unit of measure defined by the system". In the ARC setting, a frame is equivalent to a *window*. The frame analysis is an expansion of the confusion matrix metrics by extending the

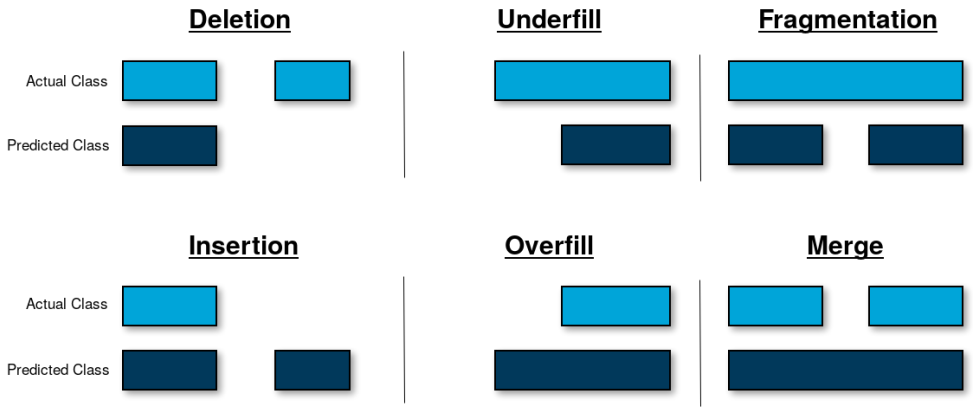


FIGURE 3.5: Error assignments. The different categories in which FN and FP can be classified are displayed. FP errors can be designated as insertion, overfill, and merge, while FN errors can be designated as deletion, underfill, and fragmentation.

existing categories in the confusion matrix (TP, FP, FN, and TN) by introducing a more detailed notion of the misclassification errors (FP, FN), which are categorized as:

False Positives:

1. **Insertion (I):** the classifier predicts the target activity when it was not being performed.
2. **Overfill (O_α or O_ω)):** the classifier either predicts the start of the target activity earlier than the actual time that it started happening (O_α), or predicts the end of the activity later than the actual time when the activity ended (O_ω).
3. **Merge:** (M) the classifier does not recognize successive activities of the target class and returns them as a single activity.

False Negatives:

1. **Deletion (D):** the classifier does not predict the target activity when it was being performed.
2. **Underfill (U_α or U_ω):** the classifier either predicts the start of the target activity later than the actual time that it started happening (O_α), or predicts the end of the activity earlier than the actual time when the activity ended (O_ω).
3. **Fragmenting (F):** the classifier does not recognize a single event as a whole, but partitions the event into successive activities.

Figure 3.5 illustrates the aforementioned categories by evaluating the predictions of the classifier (*i.e.,* predicted class). Notice the inverse relationship between *deletion-insertion*, *fragmentation-merge*, and *overfill-underfill*. A deletion, from the perspective of the **predicted class**, represents an insertion from

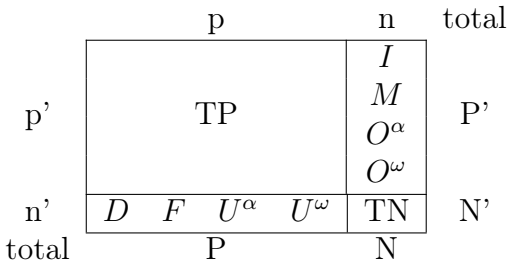


FIGURE 3.6: 2SET table. An expanded version of the classic confusion matrix, where FP and FN have new categories. Note that p and n denote the actual class, while p' and n' denote classifier returns.

the perspective of the **actual class**. The same applies to underfill-overfill, and fragmentation-merge errors.

These categories build the expansion of the classical confusion matrix, that is, the 2-class Segment Error Table (2SET) **(Figure 3.6)** and from it new metrics can be calculated:

P metrics:

- Deletion rate $\rightarrow dr = D/P$
- Fragmenting rate $\rightarrow fr = F/P$
- Start underfill rate $\rightarrow u^\alpha = U_\alpha/P$
- End underfill rate $\rightarrow u^\omega = U_\omega/P$
- TPR $\rightarrow tpr = 1 - dr - fr - u^\alpha - u^\omega$

N metrics:

- Insertion rate $\rightarrow ir = I/N$
- Merging rate $\rightarrow mr = M/N$
- Start overfill rate $\rightarrow o^\alpha = O_\alpha/N$
- End overfill rate $\rightarrow o^\omega = O_\omega/N$
- FPR $\rightarrow fpr = ir + mr + o^\alpha + o^\omega$

These metrics are the basis for the true positive rate (TPR, fpr, recall) and the false positive rate (FPR, fpr), maintaining class skew invariance: P-metrics are calculated as a percentage of the total experiment positive frames (the total positive class: **Figure 3.6**: P), while N-metrics are calculated as a percentage of the total experiment negative fragments (the total negative class: **Figure 3.6**: N).

Event analysis

Events are defined by **Ward et al. (2011)** as "a variable duration sequence of positive frames within a continuous time-series". In the ARC setting, an event is equivalent to a sequence of windows that represent the same activity. For a series of n ground truth events $G = \{g_1, g_2, ...g_n\}$, a classifier outputs a series of m returned events $R = \{r_1, r_2, ..., r_m\}$. Notice that there is no one-to-one relationship between these two series due to misclassification errors (*e.g.,* one event g_i can be recognized by the classifier as two fragmented events r_j, r_{j+1}).

The event analysis process consists of assigning a category to each ground truth event g_i and returned event r_j based on the following criteria:

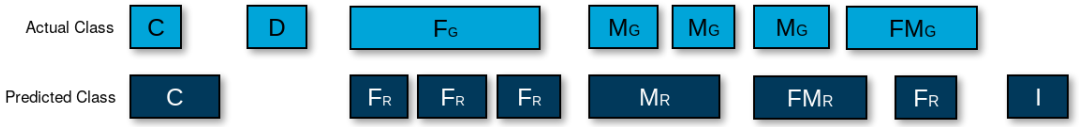


FIGURE 3.7: Event assignments. The possible categories, in which an event can be classified, are displayed.

- **Ground truth events** can be denoted as deleted (D), when the classifier does not recognize the activity at any moment that the activity took place; merged (M_G) events are events that were recognized by the classifier as a single event only. A fragmented (F_G) event is one that was recognized by the classifier as several events r_j, and a fragmented and merged event (FM_G) is a ground truth event that is associated to both merged and fragmented events in the output of the classifier.
- **Returned events** are denoted as inserted (I), when the classifier returns an event in a time lapse where there was no ground event happening; fragmented returns F_R are those events in which a continuous ground truth event was split, merged returns M_R are those events in which several ground truth events were combined into one, and fragmented and merged returns FM_R are those events that both merge and fragment ground truth events.
- **Correct classifications** (C) are defined as predictions of the classifier that do not fit into any of the aforementioned categories. The latter implies that correct classifications share a one-to-one relationship between returned and ground truth events, but do not necessarily match in terms of start time and end time (note that these mismatches are visible in frame metrics as overfill and underfill). Examples of each assignment are shown in **Figure 3.7**.

Event metrics provide additional information to consider when choosing an optimal classifier for a specific HAR problem. Take for example the case shown in **Figure 3.8**: two classifiers output the predictions of a specific activity. Judging both classifiers with standard metrics such as F-measure, the best performing classifier would be the classifier 2, since the rate of true positives is evidently higher than the one for the classifier 1. However, if a HAR application aims for instance to give a day-summary of the activities performed by a subject, regardless of its duration, then the classifier 1 would be a better choice.

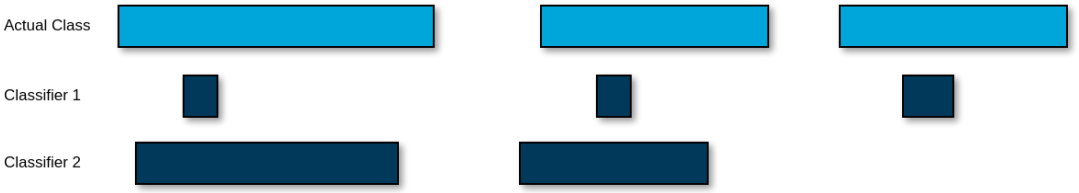


FIGURE 3.8: A ground truth activity (light blue) is recognized by two different classifiers. The first one recognizes the three ground truth events in short times, and the second one recognizes two events during a broader time lapse.

Visualization of the frame and event metrics

The HAR metrics proposed by **Ward et al. (2011)** are composed of three parts: P-Metrics and N-Metrics (belonging to the frame metrics), and event metrics. A proposed visualization of these metrics (inspired in the work of **Ward et al. (2011)**) is shown in **Figure 3.9**. The pie charts show the P-Metrics (*i.e.,* the metrics related with the positive ground truth class in the 2SET) and N-metrics (*i.e.,* the metrics related with the negative ground truth class in the 2SET). The bar plot the pie chart shows the event metrics that display the number of returned (*i.e.,* predicted) events on the left, and the number of ground truth events on the right. Additionally, the type of event is associated to a color and the color labels are shown on top of the color bars.

The frame analysis has a one-to-one relationship between the ground truth and the predictions, while *only* the correct events (C) in the event metrics conserve a one-to-one relationship[2].

An example of the application of these metrics and their visualization was done based in an example made by **Ward et al. (2011)**. This example is shown in the **Figure 3.10** and presents the frame and event analysis made to a simulated activity and recognition system. The yellow rectangles show the ground truth state of the target activity, while the light blue rectangles show the returned output. This example is composed of 28 frames, 8 ground truth events and 10 returned events.

The frame errors and event assignments are shown directly in the **Figure 3.10**. However, this visualization becomes unfeasible in a real implementation with a large number of frames and events. Following the methodology of visualizing the P-Metrics and N-Metrics in pie charts and the event assignments in a bar plot, the metrics for this example **(Figure 3.10)** are displayed in **Figure 3.11**. The pie charts summarize the frame analysis shown in **Figure 3.10** and displays the proportion of each of the errors over the total of frames. For instance, in **Figure 3.10**, four fragmented frames are shown. This information is summarized in **Figure 3.11** as the fragmentation rate ($fr = F_s/P = 4/18 = 22.2\%$. On the

[2]For example, one ground truth event can be fragmented into 3 parts, which means that the number of predicted events is already greater than the number of ground truth events. In the frame analysis, however, only the respective time windows that cause the fragmentation are highlighted, conserving the one-to-one relationship in this analysis.

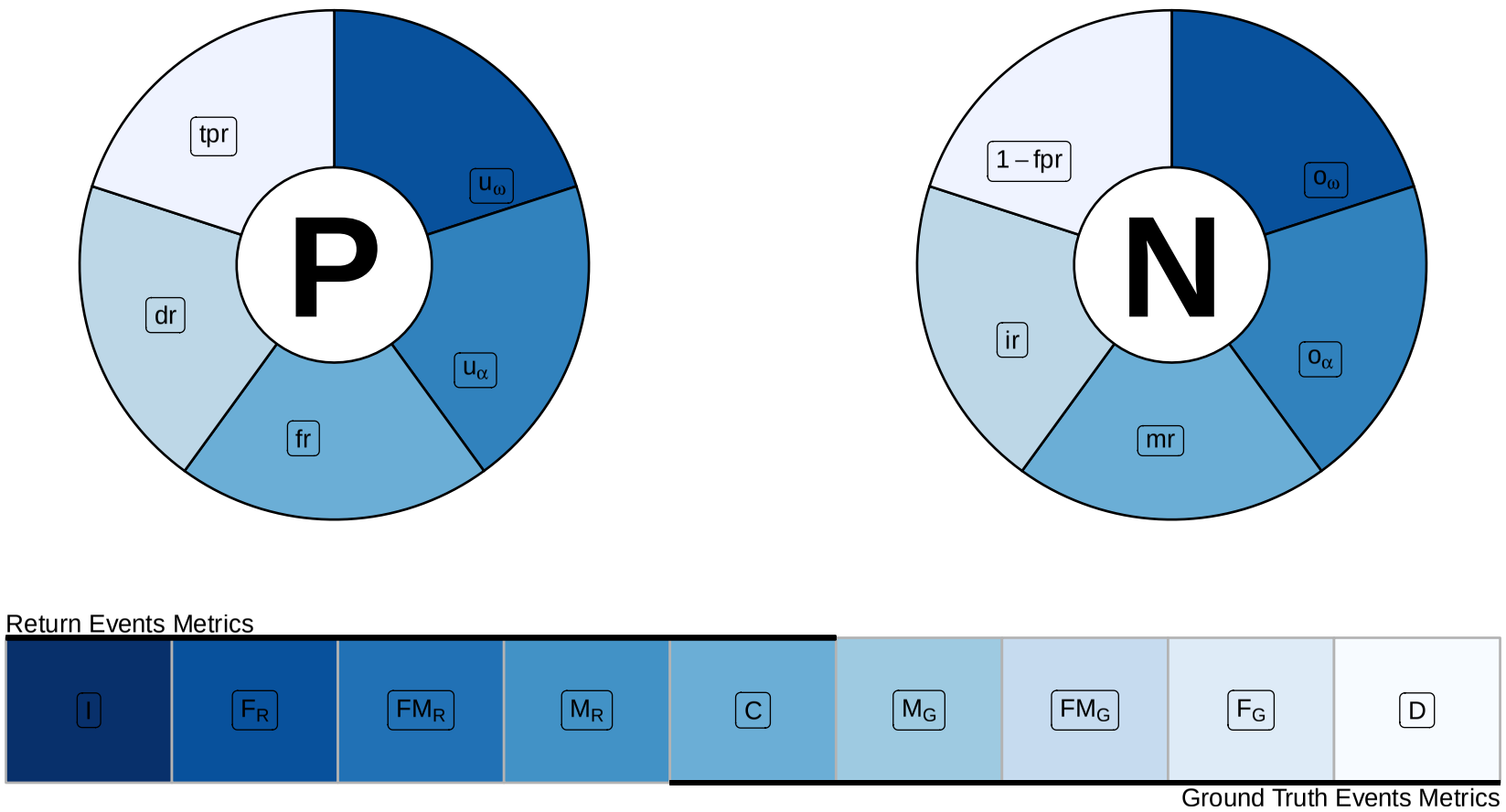


FIGURE 3.9: Visualization of event and frame metrics. P-Metrics and N-Metrics are visualized in pie charts where the each circle represents 100% of positive (P) and negative (N) ground truth classes respectively (See 2SET table in **Figure 3.6**). A bar plot is used to visualize the event metrics. The plot shows the number of ground truth and returned (predicted) events and the category of each event is differentiated with a color (possible event assignments were shown in **Figure 3.7**).

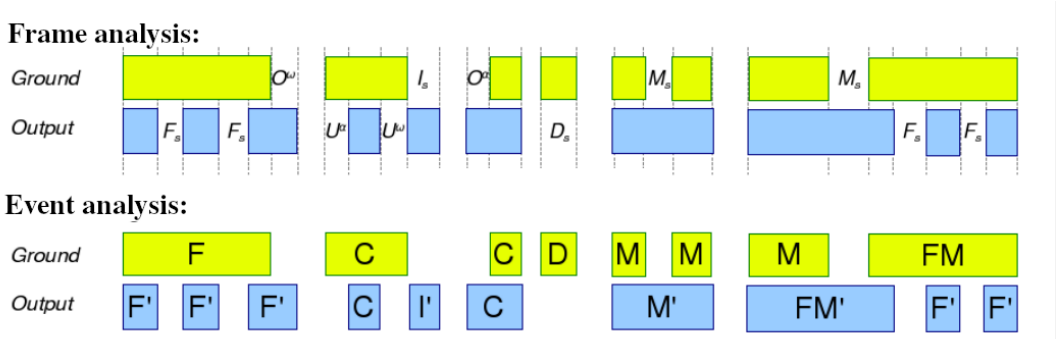


FIGURE 3.10: Example of the ground truth of a simulated activity and the simulated recognition output of a system **(Ward et al., 2011)**. Yellow rectangles represent the moments when the activity was being done (positive class), while the gaps between each rectangle represent the time where the activity was not being executed (*i.e.,* negative class). The blue rectangles show the output from a system that attempts to recognize when the ground truth events took place.

other hand, the count of each type of event in the example **(Figure 3.10)** is shown in the bar plot in **Figure 3.11**. For example, three ground truth events are categorized as merged (M). This information is shown in the bar plot as well as the proportion of events through the total ground truth events ($M_G/Total_G = 3/8 = 37.5\%$).

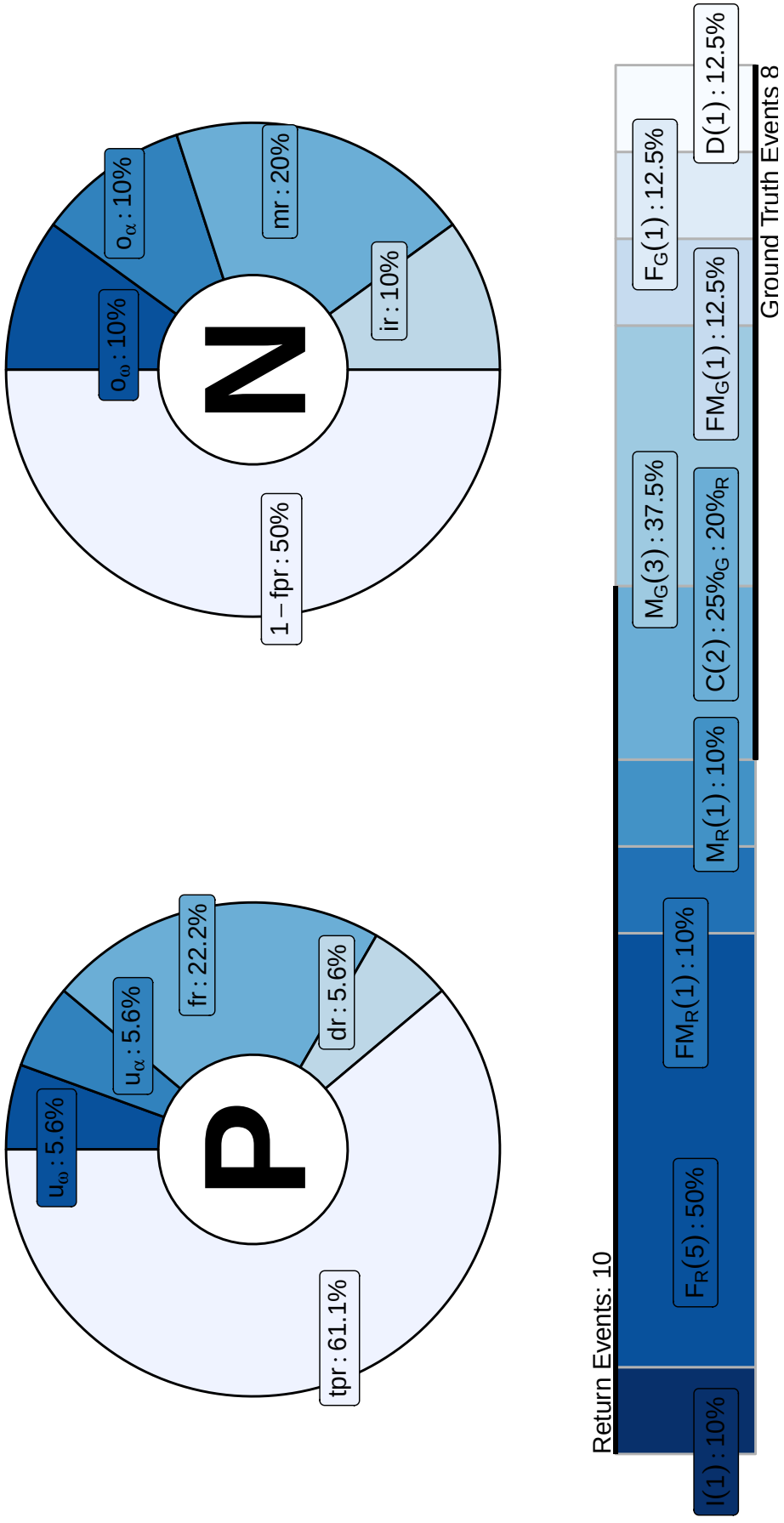


FIGURE 3.11: Frame and event metrics based on the example shown in **Figure 3.10**.

Chapter 4

Methodology

4.1 The OPPORTUNITY dataset

Roggen et al. (2010) recorded a dataset devised to benchmark human activity recognition algorithms. The authors published the data from four subjects that participated in the experiment[1], where each one wore 7 inertial measurement units (IMUs), 12 tri-axial acceleration sensors, and 4 3D-coordinates from a location system[2]. Additionally, the experiment setup included 12 object sensors (12 objects that were instrumented with wireless sensors measuring tri-axial acceleration and bi-axial rate of turn), and 21 ambient sensors (13 switches and 8 tri-axial acceleration sensors in kitchen appliances and furniture). Their main focus was to obtain a dataset that represents realistic data, which was achieved by giving to the subjects that participated in the data collection experiment loose high level instructions with respect to the activities that they needed to perform, ensuring that those activities would be done as naturally as possible **(Roggen and Tröster, 2010)**. This dataset was recorded in a highly multimodal sensor setup, where the four subjects participated in 6 experimental runs: 5 activities of daily life (ADL) runs and 1 *'drill'* run.

4.1.1 ADL Run

This type of run consists of a sequence of activities that simulate a morning routine of a human. These runs represent the target scenario of the dataset and consists of the following tasks **(Chavarriaga et al., 2013)**:

1. *(Start)* Lying on a deckchair, get up.
2. Groom: move in the room, check that all the objects are in the right places in the drawers and on shelves.
3. Relax: go outside and have a walk around the building.

[1]The OPPORTUNITY Challenge dataset is publicly available here:
https://archive.ics.uci.edu/ml/datasets/opportunity+activity+recognition

[2]Four tags for an ultra-wideband localization system were placed on the left/right front/back side of the shoulder. These deliver 3D-coordinates of the tag in a coordinate system that is fixed for the room in which the experiments took place.

4. Prepare coffee: prepare a coffee with milk and sugar using the coffee machine.
5. Drink coffee: take coffee sips, act naturally in the environment.
6. Prepare sandwich: include bread, cheese and salami, using the bread cutter and various knifes and plates.
7. Eat sandwich.
8. Cleanup: put objects used to original place or dish washer, cleanup the table.
9. *(End)* Lie on the deckchair

4.1.2 *Drill* Run

Several activities of the ADL run are scheduled to be performed only once per run. In order to generate more instances of those infrequent activities, the subjects needed to perform an additional run where they were instructed to execute 20 repetitions of the following sequence of tasks **(Chavarriaga et al., 2013)**:

1. Open and close the fridge
2. Open and close the dishwasher
3. Open and close 3 drawers
4. Open and close door
5. Open and close door
6. Turn on and off the lights
7. Clean table
8. Drink (standing)
9. Drink (sitting)

4.1.3 Multi level activities

The OPPORTUNITY dataset (OD) has a collection of labeled actions and activities that are divided into four categories:

- Modes of locomotion, *i.e.,* activities such as stand, walk, sit, and lie.
- Low-level actions, *i.e.,* actions made with one hand (unlock, stir, lock, close, reach, open, sip, clean, bite, cut, spread, release, and move) and objects held by one hand (bottle, salami, bread, sugar, dishwasher, switch milk, lower drawer, spoon, knife cheese, middle drawer, table, glass, cheese, chair, door 1, door 2, plate, top drawer, fridge, cup, knife salami, and 'lazychair').
- Middle-level actions (also defined as "hand gestures"), *i.e.,* actions made with both hands: open door, close door, open fridge, close fridge, open dishwasher, close dishwasher, open drawer, close drawer, clean table, drink from cup, and toggle switch. This corresponds to an aggregation of the low level activities.

- High-level activities, *i.e.,* relaxing, coffee time, early morning, cleanup, and sandwich time).

An additional "*NULL* class" represents any other activity that is not of interest within the data collection (*i.e.,* that is not present in the previous list). This *NULL* class is present in all activity levels.

4.1.4 Experimental design

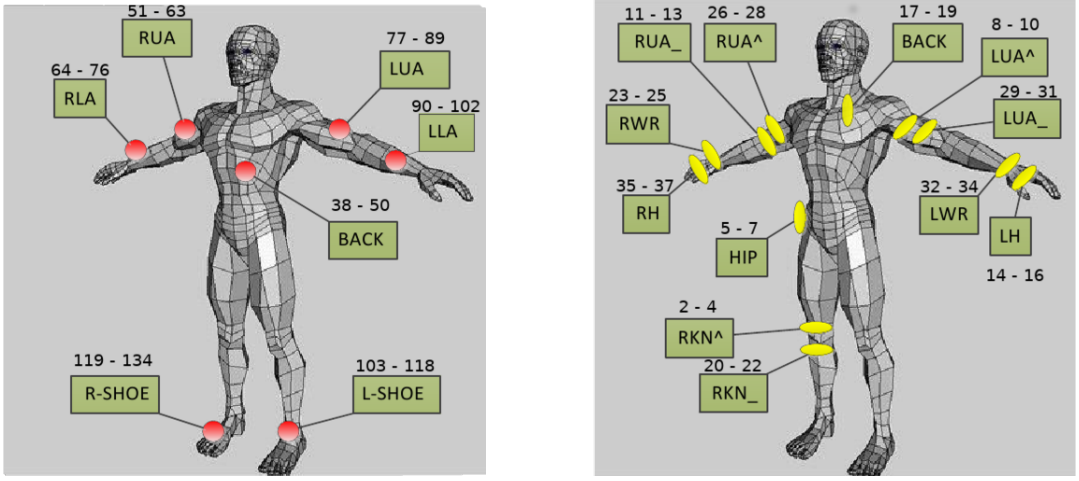
The goal of this thesis was to recognize the *modes of locomotion* (activities) and *middle-level actions*. The implemented activity recognition chain focuses on recognizing atomic activities such as movement, gestures with hands (drinking, open door) and several other short actions. High-level activities are composed of the execution of several low-level actions (*i.e.,* composite activities **Section 2.2**) and can only be recognized by implementing hierarchical recognition methods on top of the previously recognized actions **(Roggen and Tröster (2010); Liu et al. (2016))**.

Additionally, object sensors were not used because we envisioned use case settings that are independent of external sensors except from *body-worn* ones. Object and ambient sensors are much more useful to give *context* to the activity that the subject is performing in hierarchical and rule-based methods that focus their efforts on recognizing high level activities. For example, *sitting* is an activity that can be recognized with body-worn sensors, as this activity is represented in its totality by the movements of the human body. On the other hand, in order to recognize if the person is sitting because he/she is in the bathroom, reading a newspaper in a living room, or having breakfast in the dining room, it is necessary to obtain context information that can be obtained from object sensors and location tags that give information about the room in which a person is.

Finally, special attention is given to the *drinking* activity within the middle-level actions of the OD. A proper monitoring of a person's drinking events is relevant for the REACH project **(Section 1.1)**, as it gives a good estimate of the hydration routines that a patient follows at home. These insights are relevant for patients who might not be used to a specific routine, as well as for care givers who need to know the development of the health status of a patient after a specific medical treatment.

4.1.5 Sensor arrangement

The data measured from body-worn sensors provide information about the movements that a people executes with their body *(e.g., modes of locomotion and hand gestures)*. There are 19 *body-worn* sensors in the OD and are divided into two groups that are shown in **Figure 4.1**: Inertial Measurement Units (IMU) and tri-axial accelerometers. The five upper-torso IMUs **(Figure 4.1a)**



(A) Inertial Measurement Units (B) Triaxial accelerometers

FIGURE 4.1: Body-worn sensors in the OPPORTUNTIY dataset. IMU (red) contain triaxial accelerometer, magnetometer, gyroscope and 4D quaternion data. Individual triaxial accelerometers (yellow) were simultaneously used. Column numbers and column names for each sensor are also displayed **(Chavarriaga et al., 2013)**.

are composed of an ensemble of a tri-axial accelerometer, a tri-axial gyroscope, a tri-axial magnetometer, and a 4D (quaternion) location measurement.

- *Accelerometers* consist of a damped mass system on a spring etched in silicon and measure the acceleration of the body by measuring with electrodes the relative movement of the proof mass with respect to the casing[3]. An accelerometer measures the acceleration with respect to one axis, meaning that triaxial accelerometers are devices that have three accelerometers attached, one per axis (x, y, z).
- *Gyroscopes* are active sensors in the sense that the proof mass is pushed back and forth by driving combs and rely on the Coriolis effect in order to measure angular velocity. That is, when a rotation is done, a Coriolis force is acting on the proof mass and changes the direction of the motion, which is measured by electrodes[4].
- *Magnetometers* detect and measure the magnetic field, and, depending on the magnitude of the measured field (*i.e.,* how sensitive should be the sensor) these devices rely on different phenomena: Hall effect, Lorentz force, Electron Tunneling, etc. Most of the magnetometers in IMUs use the Lorentz force **(Lenz and Edelstein, 2006)**.
- *Quaternions:* 3-D rotations are often represented Euler angles (*i.e.,* yaw, pitch, roll). However, these representation has a limitation called *gimbal lock,* that is the loss of one degree of freedom in 3-D rotations. A *quaternion* is a four-element vector that can be used to encode any rotation in a

[3]Xsens Sensors. *Accelerometers:* https://www.xsens.com/tags/accelerometers/
[4]Xsens Sensors. *Gyroscopes:* https://www.xsens.com/tags/gyroscopes/

3-D coordinate system, overcoming phenomena like *gimbal lock*. Quaternion data comes from algorithms that have been developed to estimate the orientation of an IMU by fusing information from accelerometers and gyroscopes **(Kim and Golnaraghi, 2004)**.

IMUs located on the feet **(Figure 4.1a:** R-SHOE, L-SHOE**)** contain a tri-axial accelerometer, gyroscope, and a compass.

4.2 The proposed implementation of the ARC

4.2.1 Preprocessing

`NA` imputation

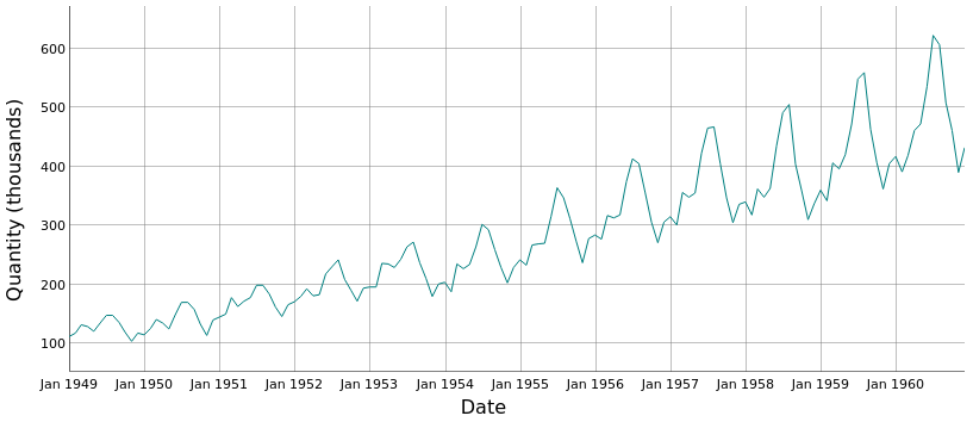
Several `NA` imputation methods were tested on the "Air Passengers" time series (provided by **Box et al. (1994)**) to evaluate their performance. This is a classic time series, containing the monthly totals of international airline passengers between 1949, to 1960.

Five univariate time series imputation methods were tested: imputation by linear interpolation, Kalman smoothing, imputing the mean of the time series, splitting the time series into seasons and perform interpolation separately on each resulting time series, and imputation using a weighted average method. These methods were already implemented by **Moritz and Bartz-Beielstein (2017)** in the programming language `R`.

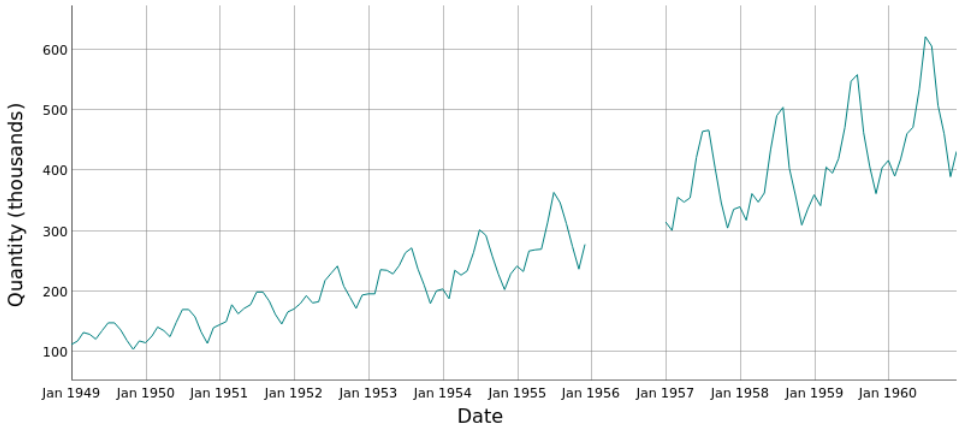
For the purpose of testing the `NA` imputation methods, `NA` values were simulated by removing `NA` values along the series. Two scenarios were tested:

1. **A large continuous gap within the time series:** the data corresponding to the year 1956 were removed (the resulting time series is shown in **Figure 4.2a**). **Figure 4.3a** shows the imputed values for this scenario and **Table 4.1** summarizes the mean error rate of the method. The running time in `R` is also shown in order to give an idea of the computational expensiveness of the method. The "Air Passengers" dataset presents a high seasonality, a characteristic that is exploited by the seasonally splitting imputation method, achieving the lowest error.

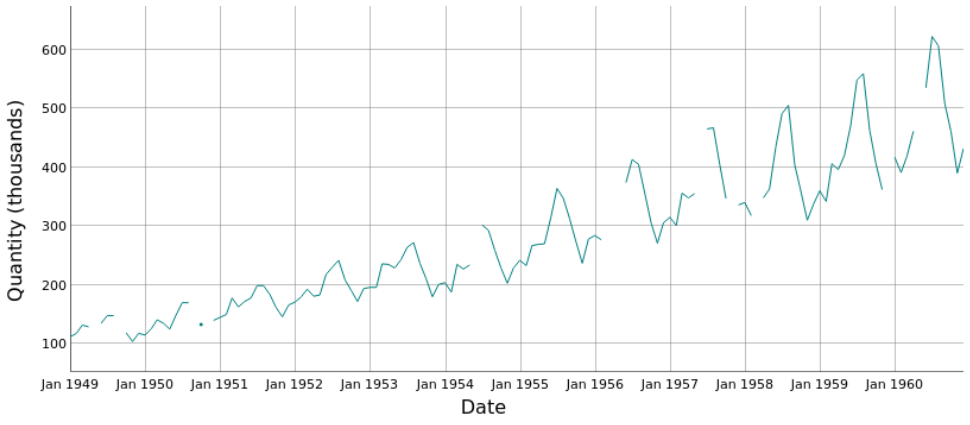
2. **Dispersed `NA` values throughout the time series:** several monthly measurements were removed **(Figure 4.2c)**. **Figure 4.3a** shows the imputed values for this scenario and **Table 4.1** summarizes the mean error rate of the method. The results show that imputing `NA` values with the mean value of the complete time series is highly error-prone, specially with time series with high trend. Kalman smoothing achieves the best performance, indicating that this method performs well for short gaps of `NA` values (in comparison with the scenario 1, where this method produced the highest error). Nonetheless, the running time of this method is considerably higher than for the rest of the methods, making it not feasible to use it with big datasets. Linear imputation, despite being a simple method, achieves the second place in both scenarios in terms of error rate and runtime.



(A) Original data



(B) Data from 1956 removed (big gap of `NA`)



(C) Data points throughout the time series were removed (dispersed `NAs`)

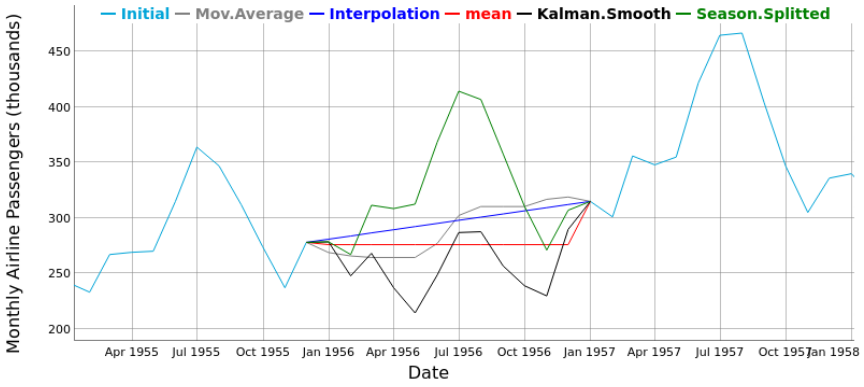
FIGURE 4.2: "Air Passengers" time series. Monthly totals of international airline passengers between 1949 and to 1960.

TABLE 4.1: Results of `NA` imputation for the scenario 1.

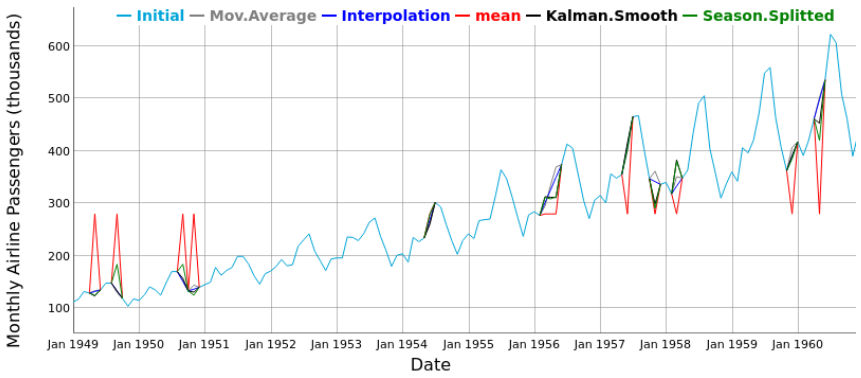
Imputation method	Error	Runtime in R
Linear Interpolation	11.2%	0.007s
Kalman Smoothing	20.7%	0.346s
Mean	14.7%	0.006s
Seasonally Splitted Imputation	1.28%	0.01s
Weighted Moving Average	14.1%	0.01s

TABLE 4.2: Results of `NA` imputation for the scenario 2.

Imputation method	Error	Runtime in R
Linear Interpolation	6.72%	0.006s
Kalman Smoothing	3.25%	0.187s
Mean	48.9%	0.005s
Seasonally Splitted Imputation	8.12%	0.009s
Weighted Moving Average	8.52%	0.007s



(A) Big `NA` gap imputation



(B) Dispersed `NA` imputation

FIGURE 4.3: Imputed "Air Passengers" time series. Data imputed by moving average, linear interpolation, mean, Kalman smoothing, and seasonality splitting.

Linear interpolation has been selected as the imputation method that will be used on the OD since this method presents the best trade-off between performance and computational running time. Moreover, the sensor time series from the OD does not show a decisive trend and seasonality, limiting the performance of methods such as *seasonally splitted imputation* and *Kalman smoothing*.

After performing `NA` imputation, the magnitude vector **(Equation 3.1)** was calculated for acceleration, gyroscope, and magnetometer measurements.

4.2.2 Data segmentation

Window length

The window length determines the size of the segments into which the time series are split. Several experiments were done to test three window lengths:

- **500 milliseconds:** proposed by the authors of the OPPORTUNITY dataset.
- **1.5 seconds:** according to **Banos et al. (2014)**, the best trade-off between recognition speed and accuracy for on-body activity recognition systems is done with window lengths within the interval $t_w = [1, 2]s$.
- **3 seconds:** this window length is proposed by **Zhu et al. (2017)** in a HAR setting where a large collection of features were extracted from the data.

Window overlap

The three methodologies proposed in the literature **(Table 3.1)** were tested, that is, an overlap of 0%, 50%, and 66%.

4.2.3 Feature Extraction

This section presents the features that were extracted for the analysis. The selected features are mostly statistical measures, since these features outperform frequency-related features in HAR **(Bulling et al., 2014; Zhu et al., 2017)**.

Mean

The arithmetic mean is the most commonly used and readily understood measure of central tendency. Moreover, it is the most commonly extracted feature in HAR **(Section 3.1.4)**. It is defined as the sum of a set of measurements, divided by the amount of measurements in that set. It is calculated as follows:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n} \tag{4.1}$$

Standard Deviation

The standard deviation is a measure of the amount of variation or dispersion of a set of measurements with respect to the mean. The sample standard deviation is calculated as:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{4.2}$$

Median, maximum and minimum

The set of measurements $X = \{x_1, x_2, \ldots, x_n\}$ is sorted so that $x_j \geq x_{j+1} \geq x_{j+2} \geq \ldots \geq x_{j+n}$. From this ordered set, the maximum value (x_j) and the minimum value (x_{j+n}) are extracted.

If n is odd, the median represents the middle value of the ordered set. That is, the value $(x_{j+\frac{n+2}{2}})$ that separates the higher half from the lower half of the set. If the set has an even number of measurements, the median is defined as the mean of the two middle values: $\frac{x_{j+\frac{n}{2}} + x_{j+\frac{n+2}{2}}}{2}$.

Root mean square

The root mean square is a measure of the magnitude of a set of numbers. It is calculated as the square root of the arithmetic mean of squares of a set of measurements:

$$x_{rms} = \sqrt{\frac{1}{n}(x_1^2 + x_2^2 + \ldots + x_n^2)} \tag{4.3}$$

Mean absolute deviation

The mean absolute deviation (MAD) of a set of measurements is the average of the absolute deviations from a central point such as the median, mean, or mode. Calculating the MAD around the median is done as follows:

$$\frac{1}{n}\sum_{i=1}^{n}|x_i - med_X| \tag{4.4}$$

Interquartile Range

The interquartile range (IQR) is a measure of statistical dispersion and it is calculated as the difference between 75th and 25th percentiles of the set of measurements.

$$IQR = Q_3 - Q_1 \tag{4.5}$$

Energy

In signal processing, the energy of a discrete signal is defined as:

$$Energy = \sum_{i=1}^{n} |x(i)|^2 \tag{4.6}$$

Zero-crossing rate

The zero-crossing rate (ZCR) is the rate of sign-changes along the time window, that is, the rate at which the measurements change from positive to negative or vice-versa **(Guo et al., 2016)**. The ZCR is defined by:

$$zcr = \frac{1}{n-1} \sum_{i=2}^{n} sgn\{x_i x_{i-1} < 0\} \tag{4.7}$$

where the function $sgn\{w\}$ is equal to 1 or 0 when the value of w is true or false, respectively.

This feature was not extracted to the new calculated predictors (*i.e.,* magnitude vectors **(Equation 3.1)**), since the calculation of the magnitude always yields a positive value, meaning that $zcr = 0$ for all possible windows.

4.2.4 Classification

The selected classifier for this thesis' ARC is the *Random Forest* (RF). **Zhu et al. (2017)** used several classifiers in a HAR problem with a high dimensional feature space and reported that the random forest classifier yielded the best results. Additionally, RFs have several advantages over other classifiers:

- RFs are ensembles of classifiers that use the predictions of several decision trees. A consensus is made by selecting the most frequent prediction.
- RFs calculate the *out-of-bag error*, an estimate of the test error without the need of resampling methods such as cross validation, reducing significantly the time spent in modeling.
- RFs can estimate the *"importance"* of each predictor, a capability that can be used as a feature selection method or as a way to interpret which are the most important features (and even sensor locations) in a HAR problem.

This work used 150 trees for every random forest modeled and considered the square root of the total number of predictors for each split.

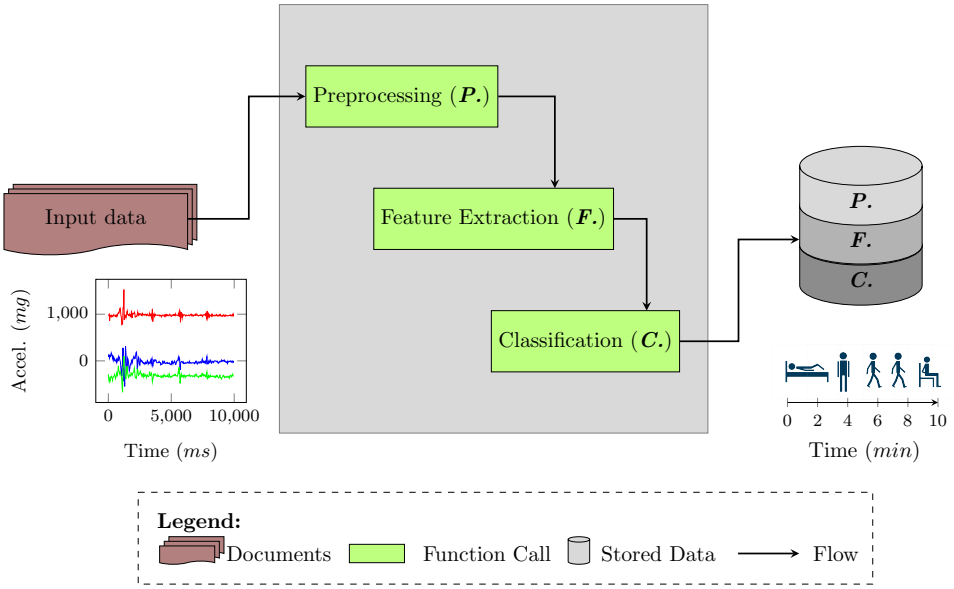


FIGURE 4.4: Diagram of the implemented *Activity Recognition Chain*. The input data contains a collection of sensor data that undergoes several transformations through three modules, obtaining finally a prediction of the performed activities from the underlying input data from the sensors.

4.3 Implementation

The Activity Recognition Chain (ARC), **Section 3.1**, has been implemented in `R`, a free software environment for statistical computing. `R` facilitates data manipulation, calculation and graphical display, includes effective ways of handling data that can be extended with a collection of packages dedicated to data manipulation and data analysis [5].

The implementation consists of three major modules, illustrated in **Figure 4.4**, that are applied sequentially once the data is loaded in `R`, and outputs predictions about the performed activities underlying the data. The first module, *Preprocessing* **(Section 4.3.2)**, manipulates the data in order to filter out non-valid sensors, impute missing `NA` values, and calculate new predictors such as the magnitude vector of accelerometers. Preprocessed data is then segmented into fixed-size windows within the module *Feature Extraction* **(Section 4.3.3)**, and then a collection of features and the ground truth (*i.e.*, label) are calculated for each time window. Finally, the *Classification* module **(Section 4.3.4)** splits the data into training set and testing set, and builds a random forest algorithm with the training set. Afterwards, the performance of the model is evaluated by comparing the predictions of the model in the test set with the ground truth labels. A feature selection process can optionally be performed using a recursive feature elimination algorithm on the training set.

[5] The `R` Foundation. *`R`: What is `R`?*, available online at `https://www.r-project.org/about.html`

```
> S1_ADL1 %>%          # Data of the Subject 1, Run 1
+     head(n = 12) %>% # Printing the first 12 rows
+     select(1:5)      # Selecting the first 5 columns
# A tibble: 12 x 5
    t_ms ACCax_RKNU_B ACCay_RKNU_B ACCaz_RKNU_B ACCax_HIP_B
   <dbl>        <dbl>        <dbl>        <dbl>       <dbl>
 1    0.           87.          975.        -287.         11.
 2   33.          124.          978.        -389.         -7.
 3   67.          102.          996.        -440.        -49.
 4  100.           59.          861.        -384.         -9.
 5  133.          119.          946.        -426.        -22.
 6  167.           99.          972.        -365.         -3.
 7  200.          116.          960.        -344.         30.
 8  233.          171.          945.        -330.         -4.
 9  267.          145.          971.        -355.        -11.
10  300.          106.          950.        -356.         12.
11  333.           52.          962.        -416.         16.
12  367.          107.          951.        -359.         10.
```

FIGURE 4.5: Subset of the raw data. Every column corresponds to a variable and every row corresponds to an observation. The column names (*i.e.*, `t_ms, ACCax_RKNU_B`, etc.) are used to identify each variable from the dataset, allowing to subset the data depending on the analysis.

4.3.1 Data importing

As input for the ARC, data should be similarly structured as the data of the OPPORTUNITY dataset (OD)[6]. The input files are loaded into R and are formatted into tibbles (*i.e.,* a "tabular" data object whose structure represents cases (rows), each of which consists of a number of observations or measurements (columns)) that are afterwards grouped into a common list. **Figure 4.5** illustrates a subset of one of the runs of the OD, once it is loaded in R.

The column names have been created considering the type sensor and variable measured in the dataset. It has been ensured that each column has a unique and identifiable name that facilitates subsetting tasks of the dataset. For that, column names have been standardized following this pattern:

`SENSORTYPEtypeandmeasuredcoordinate_LOCATION_SENSORSET`

- `SENSORTYPE`: differentiates between accelerometers `(ACC)`, inertial measurement units `(IMU)`, reed switches `(RSW)`, labels `(LAB)`, and location tags `(LTAG)`.
- `typeandmeasuredcoordinate`: specifies the measured coordinate and the type of measurement, according to the sensor. For example, in tri-axial accelerometers one can find: `ax` (acceleration on the x-axis), `ay` (acceleration on the y-axis), `az` (acceleration on the z-axis).

[6]The OD is a collection of 24 files, where each file corresponds to one experimental run.

```
# Counting number of rows and columns
> nrow(S1_ADL1) # Number of rows
[1] 51116
> ncol(S1_ADL1) # Number of columns
[1] 250
```

FIGURE 4.6: Dimensions of the data records in the first run.

- `LOCATION`: indicates the placement of the sensor. For example, the acronyms of body locations shown in **Figure 4.1a** and **Figure 4.1b** are used.
- `SENSORSET`: differentiates between body sensors `(B)`, location tags `(L)`, ambient sensors `(A)`, object sensors `(O)`, and labels `(L)`.

Since the experimental runs did not have a fixed time of execution, the length of the tibbles are variable. For example, inspecting the dimensions of the dataset's first run **(Figure 4.6)**, it is possible to see that there are 51116 measured instances of this run. The number of columns is conserved through the dataset (250), since the same collection of sensors and labels were included within each run. Additional details about the number of instances per run can be found in the **Appendix B**.

4.3.2 Preprocessing

The preprocessing is the second step of the *ARC*, responsible of performing different data transformations to prepare the data for the feature extraction **(Section 4.3.3)**. These methods have been wrapped into one function called `preprocessing()` and the flowchart of this function is shown in **Figure 4.7**.

The function transforms data by applying (if desired) operations of subsetting, `NA` imputation and calculation of the magnitude vector of sensors. Depending on the operations performed, the function outputs the preprocessed data, along with information about the modifications done to the data such as: column names filtered (*Subsetting*), method of imputation (*NA Imputation*), and column names of the new additional predictors (*Magnitude calculation*).

4.3.3 Feature extraction

The preprocessed data are fed to the feature extraction module. This module wraps several processes into the `feat_extraction()` function, whose flowchart is shown in **Figure 4.8**. The function initially segments the data into fixed time windows, with or without overlapping (*Segmentation*), and subsequently splits the data into two tibbles: *sensor data* and labels (*Splitting*); computes a user-specified collection of features on the *sensor data* tibble (*compute features*) and determines the ground truth labels per window by selecting the most frequent label of each window (*select labels per segment*). Finally, features with zero variance are removed (*Feature selection: filter methods*) and afterwards the kept features' and labels' tibbles are merged together (*Merging*).

The collection of features that are computed with this function and that represent the total of features computed for this work are:

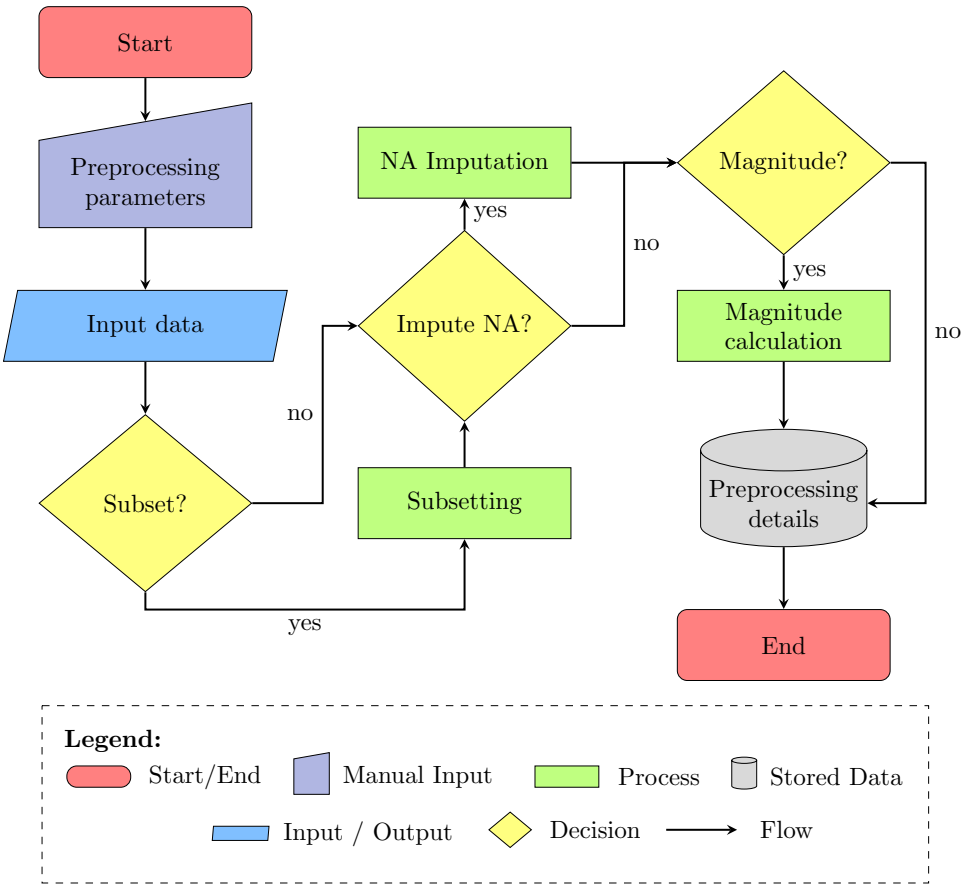


FIGURE 4.7: `preprocessing()` flowchart. The diagram shows the steps followed within the preprocessing steps of the implemented ARC.

- Mean
- Standard Deviation
- Median
- Interquartile Range
- Zero crossing rate
- Median absolute deviation (MAD)
- Root mean square (RMS)
- Maximum and minimum values
- Energy

The `feat_extraction()` function outputs the collection of features calculated per window, the ground truth labels per window, and a list of names of the extracted features.

4.3.4 Classification

The final step in the ARC, the classification stage, creates models from the collection of features generated by the `feat_extraction()` function. Initially, as it is shown in **Figure 4.9**, a partition of the data is done into training and testing sets (*Partition*). Then, a feature selection process is performed to remove highly correlated features (*Feature selection: wrapper methods*) for every target level (*e.g.,* locomotion activities, low-level actions, middle-level actions). Afterwards, a random forest model is created for every target level based on the training data (*Classification*). Sequentially, the

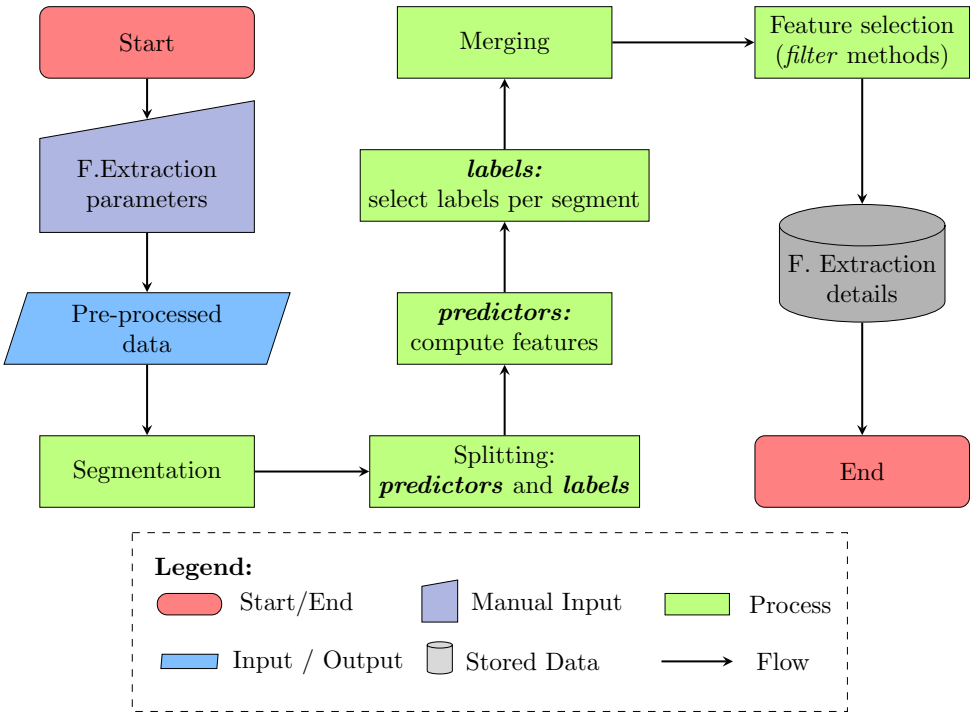


FIGURE 4.8: `feat_extraction()` flowchart. The diagram shows the steps followed within the segmentation and feature extraction steps of the implemented ARC.

model is executed on the test data to validate its performance and several performance measures are calculated based on the predictions and the true labels of the testing set (*Performance measures*). Finally, a list of the implemented classifiers and their performance measures are stored in the system.

4.3.5 Wrapping it all together

A set of *control* functions (explained in detail in **Appendix C**) were developed to manage and validate the parameters of the three functions that compose the ARC. With this validation, the three functions described previously were wrapped into a unique function, `activity_recognition`, that performs the whole chain and outputs detail of every subprocess. **Figure 4.4** is, in this sense, a gray box diagram of this function, and a detail flowchart of its processes is displayed in **Figure 4.10**. Notice that this function serves only as a connection between the three modules and does not include any new processes. This function enables a more structured workflow and experimental design.

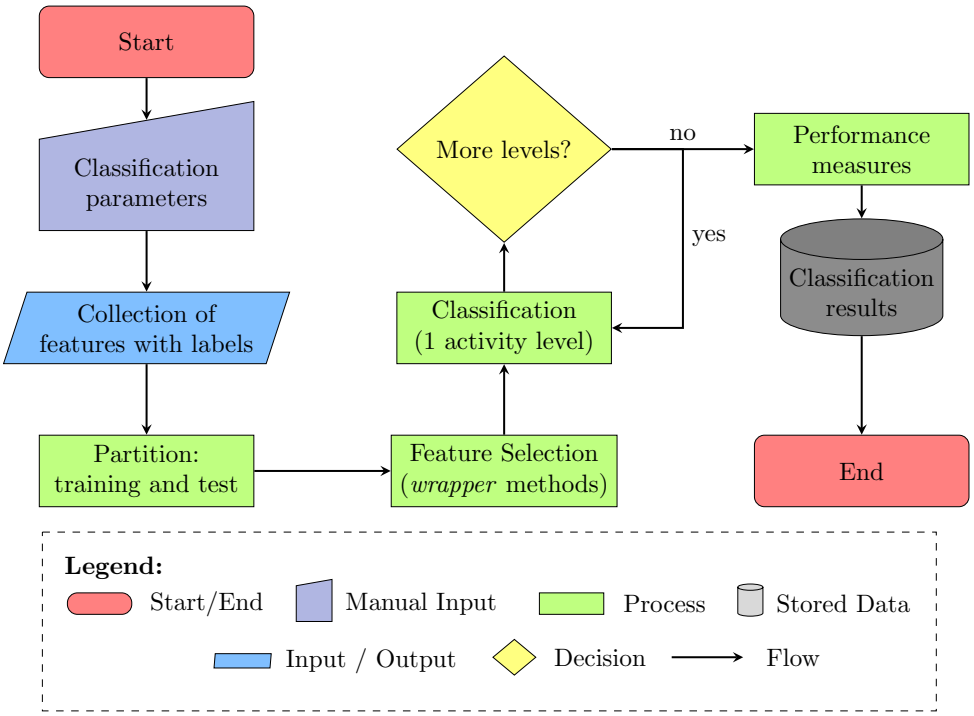


FIGURE 4.9: `classification()` flowchart. The diagram shows the steps followed within the classification and evaluation steps of the implemented ARC.

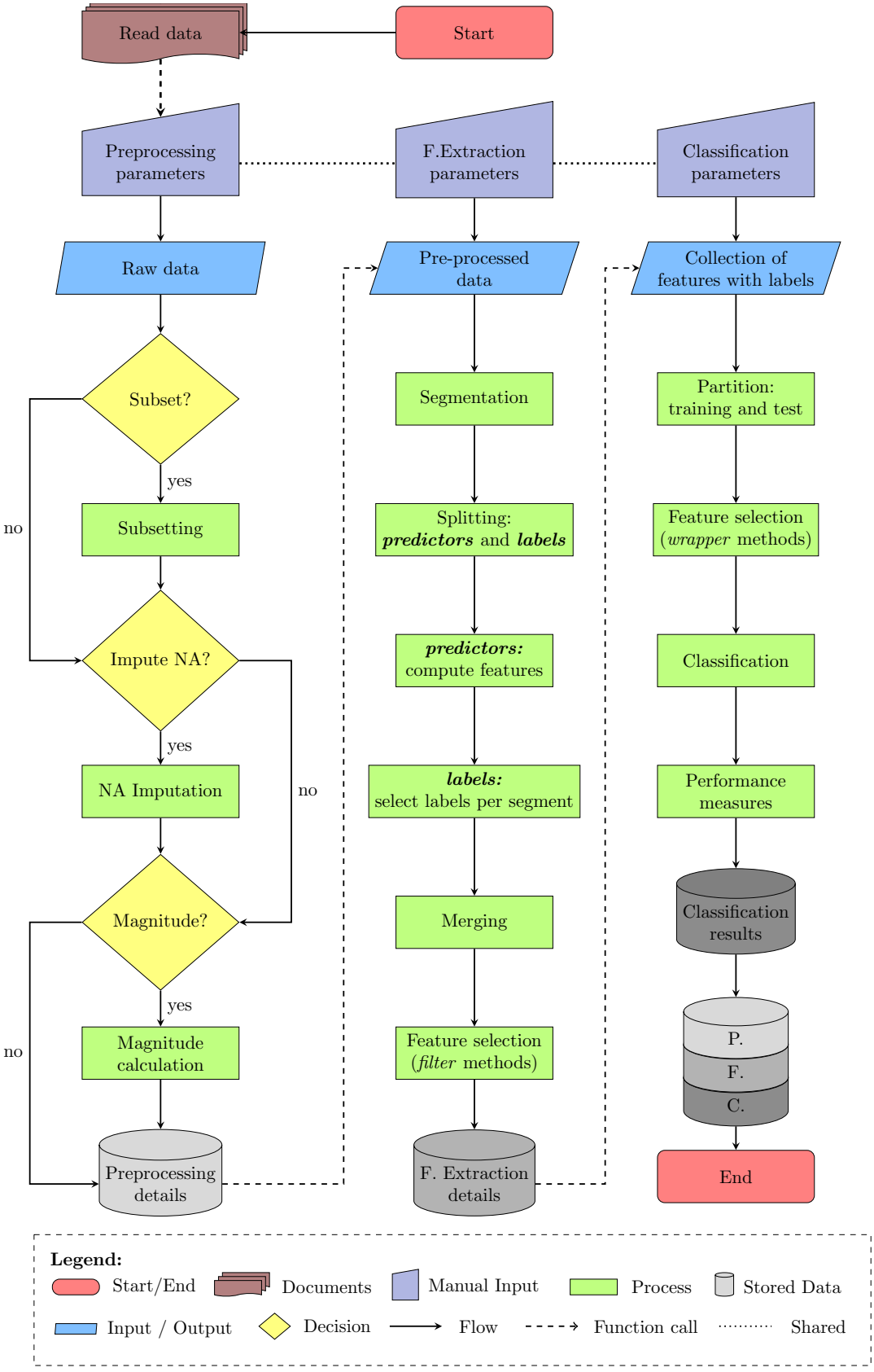


FIGURE 4.10: `activity_recognition()` flowchart. The diagram shows the implemented ARC. The `preprocessing`, `feature_extraction`, and `classification` functions are piped together and the collection of parameters are organized by *control* functions. `activity_recognition()` allows, in one call, to run a complete experiment and generate variations of it by adjusting only the parameters set in the control functions.

Chapter 5

Evaluation

In **Section 3.1**, it was shown that each of the steps followed in the Activity Recognition Chain (ARC) conveys a choice of parameters oriented to optimize the performance of the system. Moreover, in **Section 4.2**, several parameters were proposed for the ARC, and these were tested in order to choose the best-performing system. These tests were done under the benchmarking conditions proposed for the OPPORTUNITY challenge **(Section 5.1)** and further experiments were done using only the best-performing system.

5.1 The OPPORTUNITY Challenge

Prior to the release of the OPPORTUNITY Dataset (OD), **Chavarriaga et al. (2013)** released a subset of the dataset as a basis for a public challenge: the OPPORTUNITY Challenge (OC). The OPPORTUNITY Challenge Dataset (OCD) consists of a set of 18 experimental runs: five ADL runs and one drill run per subject, and data from 3 subjects is given. The authors provided only the complete **body-worn** sensor data, except for quaternion orientation data from IMUs.

The **testing** dataset for the OC is composed of two ADL runs **(Section 4.1.1)** from subject 2 and subject 3 (specifically, ADL4 and ADL5), while the **training** dataset is composed of the rest of the runs of the three subjects. That is, 14 runs are used as training, and 4 runs are used as testing. Two classification challenges were proposed: the classification of the modes of locomotion and middle level actions (hand gestures) **(Section 4.1.3)**.

Chavarriaga et al. (2013) implemented an ARC using a time window of 500 ms, a 50% overlap, and 'mean' as a feature. The authors tested four classification methods and these results served as the baselines of the OC:

- k-Nearest-Neighbors (k-NN) classifier: The Euclidean distances between a test sample and the samples of the training set are computed and the most frequently occurring label among the k closest samples is selected as output label. **Chavarriaga et al. (2013)** used two different classifiers, setting either $k = 1$ or $k = 3$.
- Nearest-Centroid Classifier (NCC): In this method, the Euclidean distance between the test sample and the centroid of each class is calculated. Afterwards, the classification is done by assigning a sample to the closest centroid.

TABLE 5.1: Summarized experiments

ARC step	Comments
NA imputation	Linear interpolation
Magnitude	Yes
Window length	0.5s, 1.5s, 3s
Window overlap	0%, 50%, 66%
Features	All **(Section 4.2.3)**
Classifier	Random forest

- Linear Discriminant Analysis (LDA): This is a Gaussian classifier which classifies on the assumption that the features are normally distributed and all classes share the same covariance matrix.
- Quadratic Discriminant Analysis (QDA): Similar to LDA, this technique assumes a normal distribution for the features but the class' covariance may differ.

A summary of the experiments done in this work (with the conditions given in the OC) are displayed in **Table 5.1**. There are in total nine ARCs that were tested, which represent the possible combinations of window lengths and overlap sizes. If the choice of the best ARC was done based on the results on the **testing** set, then this choice would be *biased* in the results of a testing set and not any real case scenario[1]. In order to prevent this, the selection of the best configuration of parameters is done based on the out-of-bag (OOB) set generated by the random forest **(Section 3.1.5)**, and the best performing system is afterwards tested using the testing set for a comparison with the baseline results. Since there are two target activities to recognize within the OC (locomotion and hand gestures), this procedure is repeated twice, one for each group of target activities.

Figure 5.1 shows the accuracy obtained with the OOB samples for each ARC. The results show an overall increase of the accuracy, the higher the window overlap is, and the smaller the window size is. For this reason, the selected ARC (for both gestures and locomotion activities) uses a window length of 500 ms and an overlapping of 66% between two consecutive windows. The proposed random forest model uses 150 trees and a plot of the OOB error *vs.* the number of considered trees is shown for the selected ARC in **Figure 5.2**

The random forest for each ARC is modeled in a feature space of 1220 dimensions (122 sensor readings $\times$ 10 extracted features per reading). Several sensors are highly correlated between each other, meaning that the extracted features are correlated as well. The more dimensions the feature space has, the more degrees of freedom the model possibly has, and this could lead to overfitting. Eleven subsets of features[2] were tested on the best segmentation configuration (500ms and 66% overlap) and the results are shown in **Figure 5.3**. The results show initially a steep increase in the accuracy when adding more predictors to the model. However, after 244 predictors,

[1] If the choice of parameters is based on how well does the system perform with the test set, then maybe these combination of parameters work well with that particular dataset and not with any other dataset that the system might test.

[2] The subsets contain 2.5%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90% of the total features.

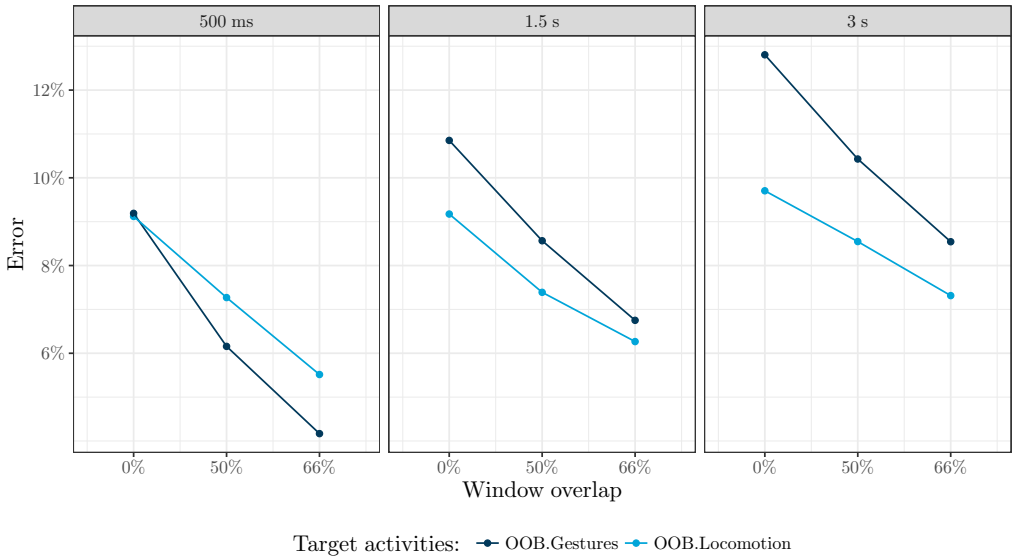


FIGURE 5.1: OOB error of the initial experiments. The results of the nine proposed ARCs **(Table 5.1)** are shown in the figure for both locomotion and gesture activities.

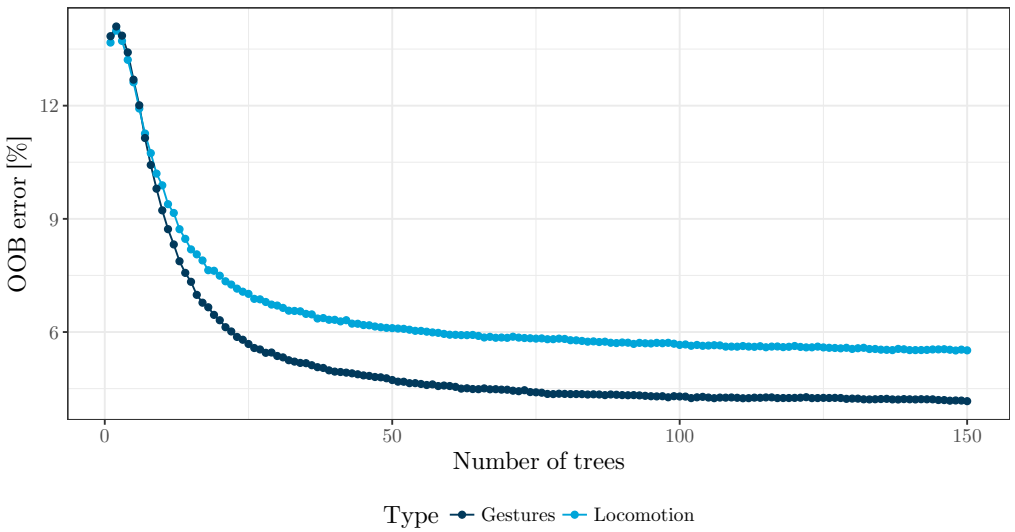


FIGURE 5.2: OOB error *vs.* number of trees considered. The plot shows the OOB error for the selected ARC ($w_{len} = 500ms$ and overlap = 66%). The performance of the random forest is increased by considering more trees in the model. The first 75 trees show a strong increase in the performance and at 150 trees the performance converges.

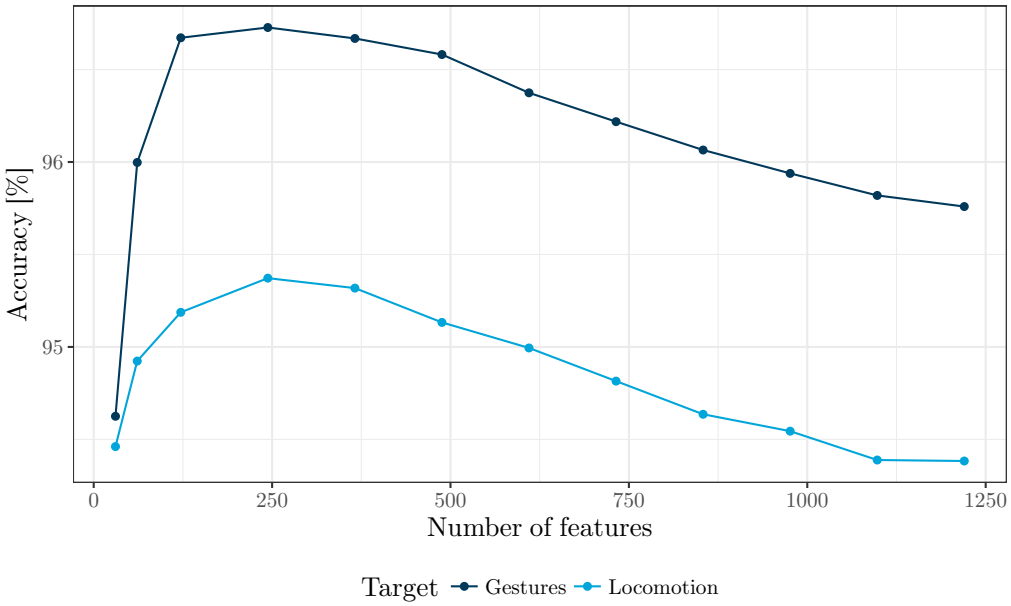


FIGURE 5.3: Feature selection results. The accuracy of models that consider the k most important predictors according to the random forest model is displayed.

the performance of the model starts to decrease for both locomotion and gestures models. This result shows that the model performs better when considering only 20% of the total predictors.

The F-measure **(Section 3.3.2)** calculated for the testing set is shown in **Table 5.2**. The table summarizes the results obtained by the participants of the challenge, the baselines established by **Chavarriaga et al. (2013)**, and the results obtained in this work with and without using feature selection (FS). The results show that the feature selection in the proposed ARC outperforms the rest of the models.

There is no increase in the performance, when using feature selection compared with the setting that uses all predictors in the locomotion experiment. However, the performance increases moderately by 2%, when using feature selection in the model that targets hand gestures. This difference might be due to the involvement of extremities when performing either locomotion activities or hand gestures: locomotion activities are performed with the whole body, meaning that more extremities are involved in the execution of the activity compared to hand gestures, where the sensors that are involved are located only in the upper torso. Therefore, feature selection is more effective in the case of hand gestures model, where there is more noise in the sensor measurements, as there are extremities that do not contribute to any of the set of the target activities.

As mentioned in **Section 3.3.4**, even though the F-measure is a standard and widely used performance metric for HAR, it has the great limitation that the interpretation of the results becomes very limited, and it is not possible to analyze the performance of the classifier on each activity in a more detailed way than a confusion matrix offers. To cope with this limitation, the HAR-focused performance metrics proposed by **Ward et al. (2011)** were implemented. These metrics extend the categories of a confusion matrix (derived from the frame analysis **(Section 3.3.4)**) and analyze the

TABLE 5.2: Baseline results of the OPPORTUNITY challenge **(Chavarriaga et al., 2013)**. The table summarizes the baselines of the challenge and the score obtained by the best ARC configuration proposed in this work.

OPPORTUNITY challenge		
Method	F - measure	
	Locomotion	Gestures
LDA	0.59	0.69
QDA	0.68	0.53
NCC	0.53	0.51
1-NN	0.84	0.87
3-NN	0.85	0.85
UP	0.60	0.64
NStar	0.61	0.84
SStar	0.64	0.86
CStar	0.63	0.88
NU	0.53	-
MI	0.83	-
MU	0.62	-
UT	0.52	-
NAGS	-	0.71
This work (no FS)	0.87	0.88
This work (FS)	0.87	0.90

quality of the predicted events with respect to the ground truth events (derived from the event analysis **(Section 3.3.4)**). Although these metrics give more detail about the performance of a system, there are many metrics to consider in the analysis, and it requires to observe both the event and frame metrics simultaneously, with the aim of delivering proper conclusions. It should be noted that this is a *one-vs-all* analysis, meaning that the one target class represent the *positive* class in a confusion matrix and the other classes are grouped together as the *negative* class.

HAR-focused performance metrics were calculated for the potential activities in a REACH setting: locomotion activities and drinking. A focused analysis will be made to the sitting activity **(Figure 5.4)** to show how the frame and event analysis are done, and afterwards a summary will be given for the other locomotion activities. The analysis of drinking events is done in **Section 5.4**, as the performance of this setting, along with a model focused with drinking events are compared.

The Frame Metrics (FM) and Event Metrics (EM) with respect to the *sitting* activity is shown in **(Figure 5.4)**. The frame analysis shows that, most of the time, windows were correctly classified, since the true positive rate (*tpr*) is 97% and the false positive rate (*fpr*) corresponds to 0.6% *(FM)*. All sitting events were correctly identified since there were no deleted ground truth events (EM). 23 events were wrongly predicted by the model (insertion EM), and correspond to 43.4% of the total returned events. However, the length of those events is minimal, since the total insertion rate (*ir*) is about 0.4% of the total number of windows that did not have sitting as the ground truth label (FM). Four events were fragmented into 18 events (EM), but once again the duration of these fragmented events is short, since the fragmentation rate (*fr*) is 1.2%. Overall, the system recognizes effectively the sitting events and only few windows are mistakenly confused as other activities.

Lying activities **(Figure 5.5)** are the best identified activities with a *tpr* of 97.4% and a *fpr* of 99.9%. Fragmentation events are the biggest cause of error for this model but they represent 1.1% of the total P frames (the EM shows that the model fragmented two lying events into four predicted events).

Standing up **(Figure 5.6)** has much more errors compared with the former two locomotion activities. The Frame Metrics (FM) shows that some transitions between standing up and another locomotion activities are not immediately captured by the system, since the overfill and underfill rates correspond to 4% of the P and N frames, respectively. 36% of the returned events correspond to insertions, which is equivalent to 4.7% of the N frames. Finally, most of the fragmented events were split into two parts, since 30 ground truth events were split into a total of 69 events. The *fr* is not significant and corresponds to 1.4% of the P frames, indicating that the gap between the fragmented events is not long.

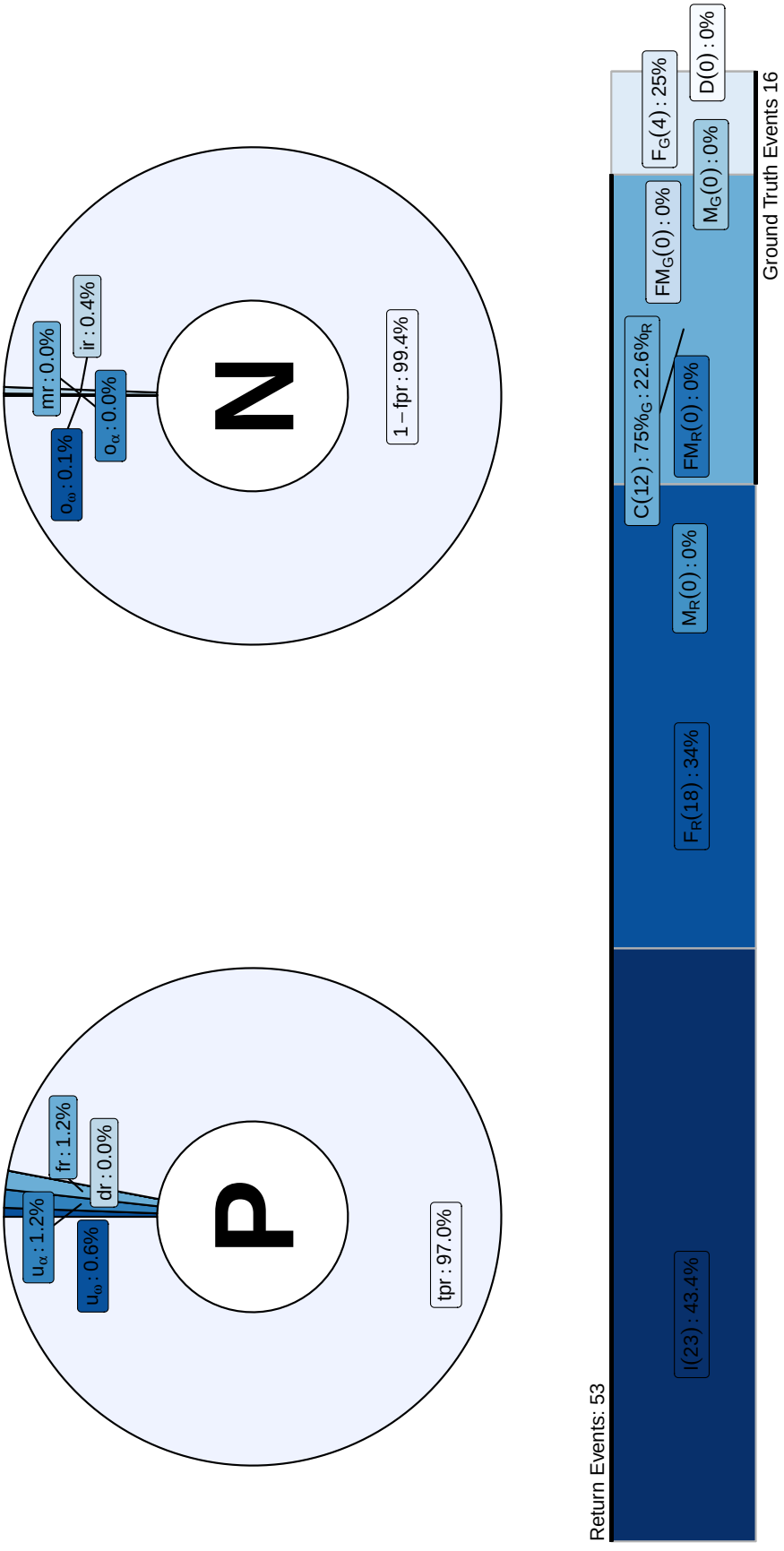


FIGURE 5.4: HAR metrics (frame and event analysis) for the *sitting* mode of locomotion.

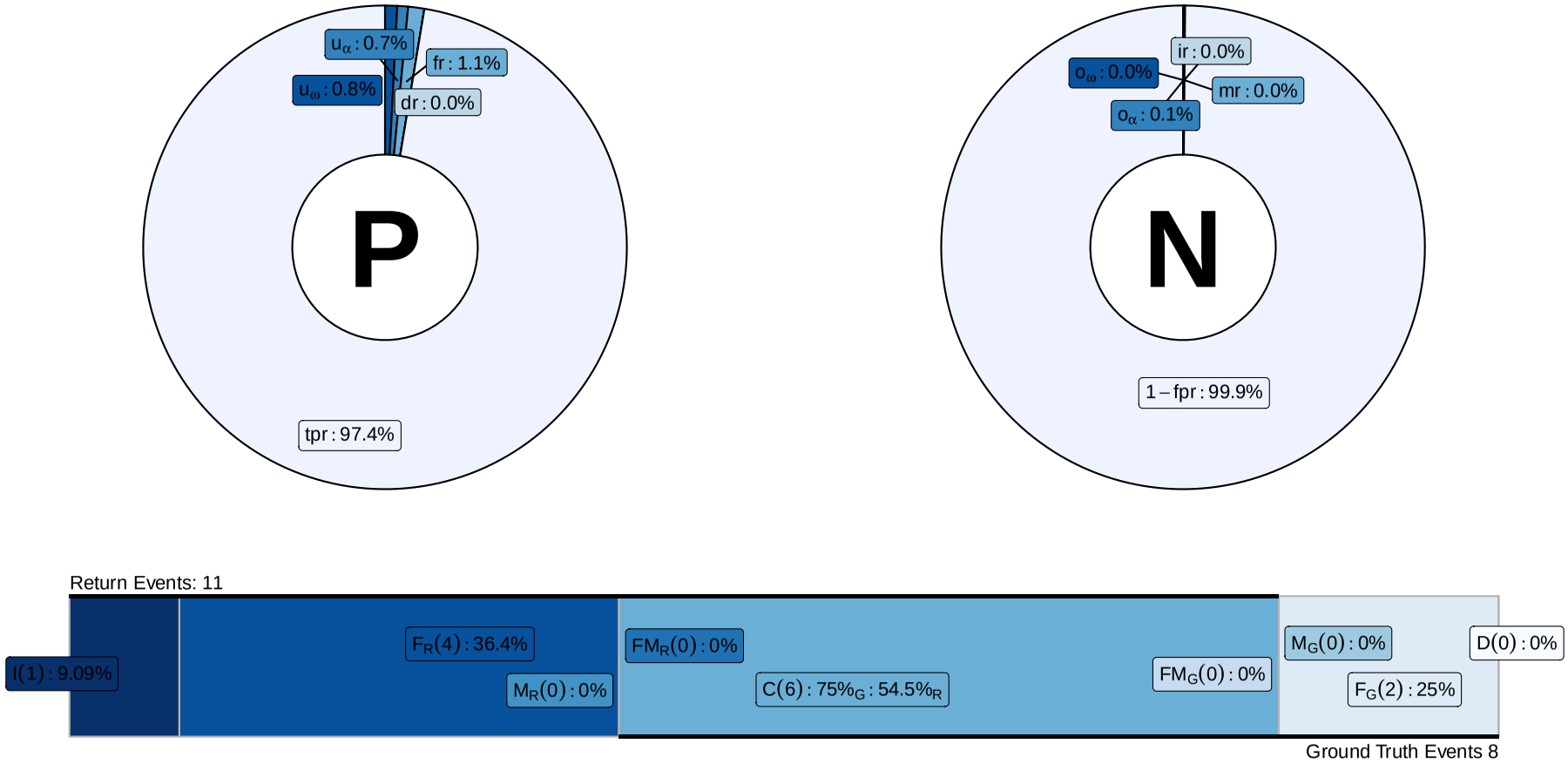


FIGURE 5.5: HAR metrics (frame and event analysis) for the *lying* mode of locomotion.

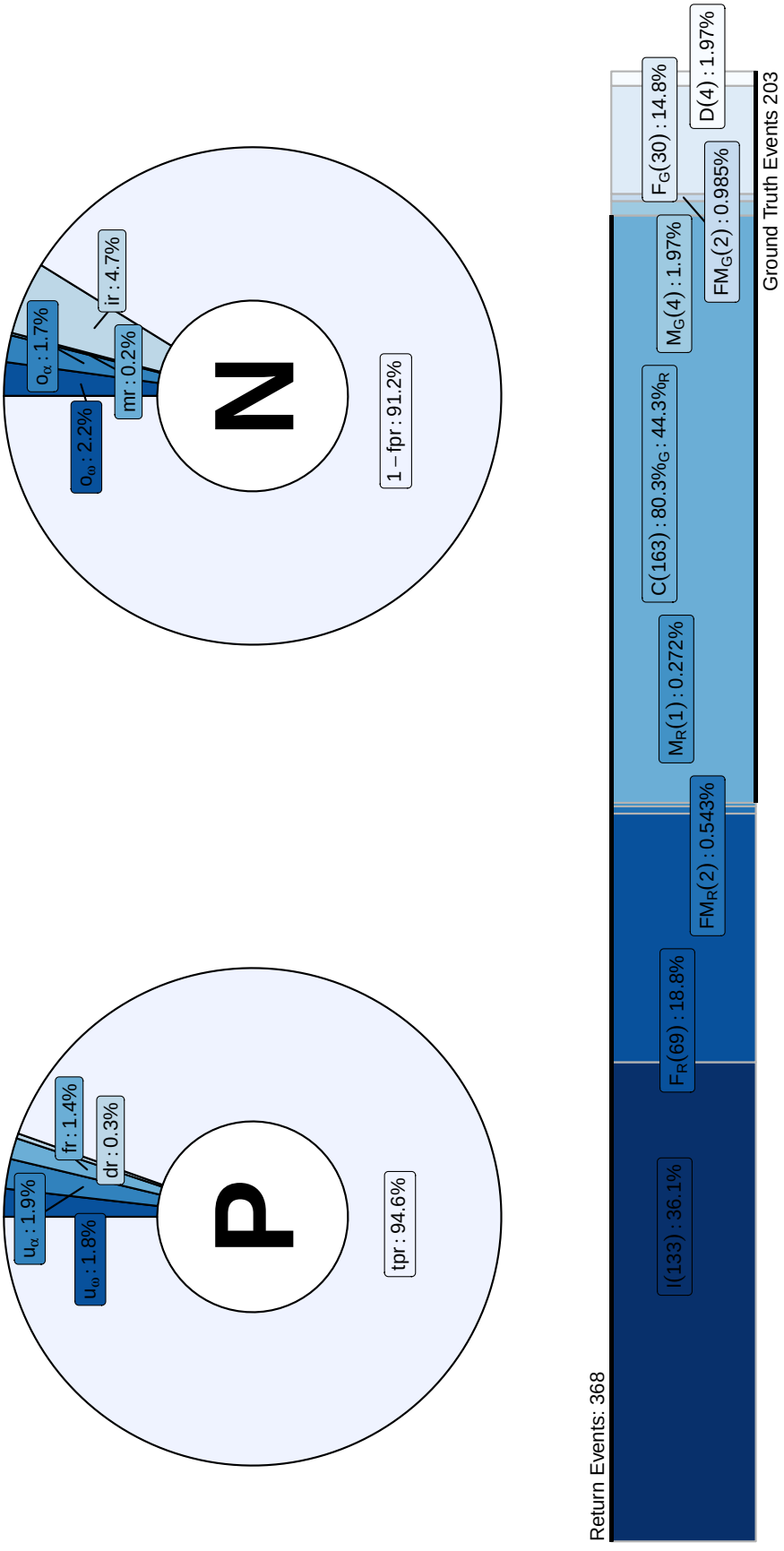


FIGURE 5.6: HAR metrics (frame and event analysis) for the *standing* mode of locomotion.

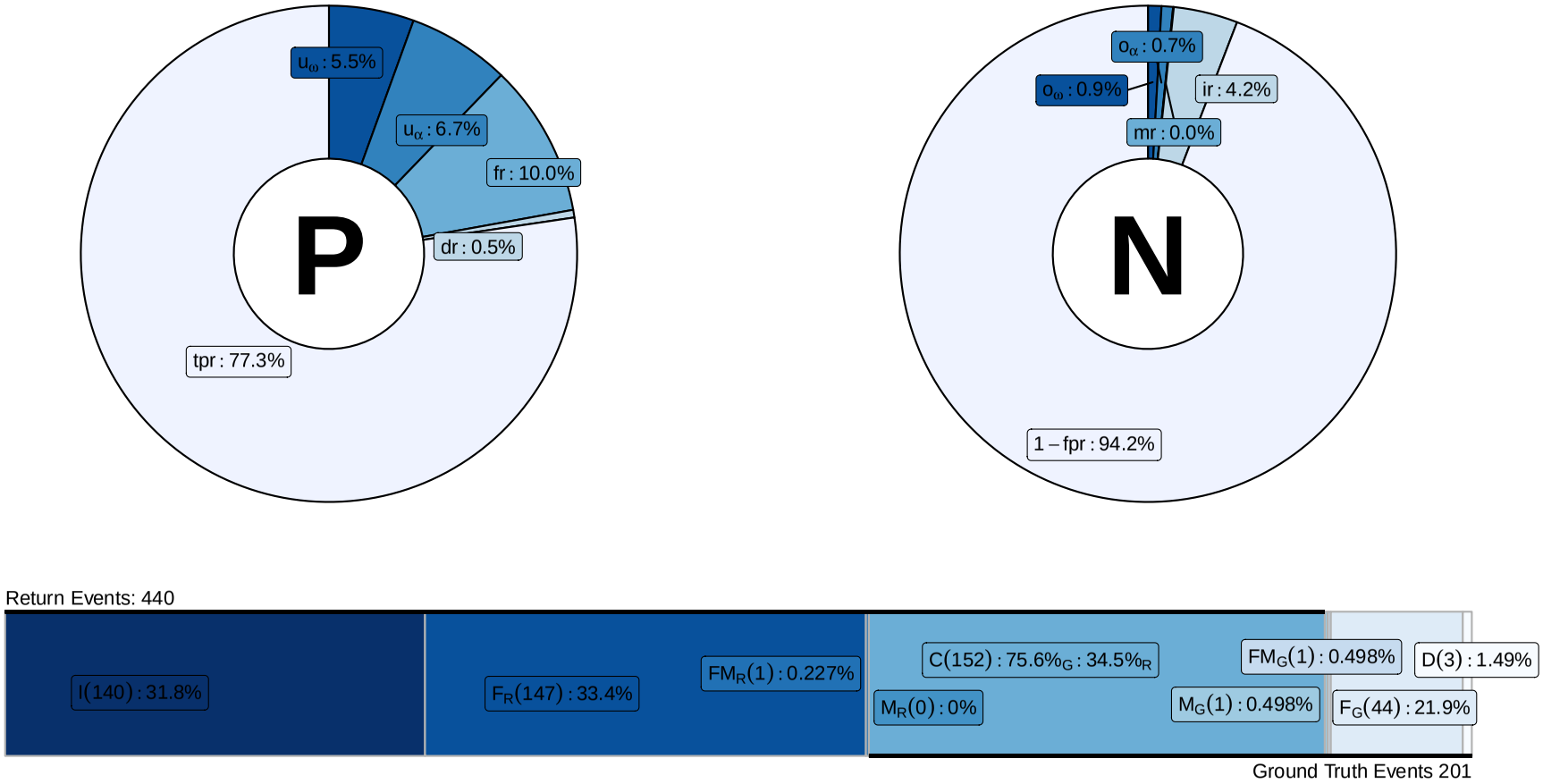


FIGURE 5.7: HAR metrics (frame and event analysis) for the *walking* mode of locomotion.

The model cannot optimally predict transitions between an arbitrary locomotion activity and a successive walking activity. This is evidenced by underfill errors that represent 12.2% of the positive frames. The overfill errors, on the other hand, represent only 1.6% of the negative frames and show that the system performs well when the subject walks and changes to another locomotion activity. 33% of the returned events are fragmented, and this is equivalent to 10% of the positive frames (fr).

5.2 User-dependent scenario

The OCD scenario suggested to use data from three subjects as training set, and a subset data (not used in training) from two of these subjects as testing. That is, the OCD model learns not only with data from the subject in which the model will be tested, but also combines knowledge from 'external' sources (*i.e.,* two subjects).

Another scenario was tested, in which the models are trained using only data from the subject in which the model will be applied. This setting is known as the *user-dependent* scenario. User-dependent scenarios resemble application settings where a HAR system is targeted to be used by only one person.

User-dependent challenges involve the variability that exist in the way people can perform the same activity **(Section 2.4)**. Four models were built, one per subject, in order to see how well the model can generalize the performed activities using a reduced training dataset: each model used the data of only one subject and was distributed in the following way: the drill run and three ADL runs (ADL1, ADL2, and ADL3) were used as training set. The remaining ADL runs (ADL4 and ADL5) were reserved as testing set.

The results for locomotion activities are shown in **Table 5.3**, and for hand gestures in **Table 5.4**. The ARC framework implemented in the course of this thesis outperforms the other approaches in three of the four subjects for locomotion activities and one subject for the hand gestures. User-dependent models for locomotion activities show an increase of the F-measure in comparison with a setting such as the OC, where data from several subjects served as training set. This shows that locomotion activities have not a high variance within a subject and user-specific models would have a high performance, while the decrease in the F-measure in the OC setting might be due to variances in the way the subjects execute locomotion activities (*e.g.,* the rate of steps per second while walking, and the movements done with the extremities when standing, lying, and sitting). Nevertheless, in a real setting, user-centered models demand a data collection for each person and an optimization of all parameters within the ARC, making the task time-consuming and sometimes infeasible.

Hand gestures, on the other hand, can have a higher variance, and the model benefits from data from the other subjects to improve the performance on a specific subject. For example, that the OC experiment yielded in a F-measure of 0.87 (no FS) for hand gestures in which data from the subject 2 and 3 were part of the test set and the model was trained with data from these two subjects, as well with the complete data from subject 1. Subject 3 performs the same hand gestures in different ways, and this becomes evident by comparing the OC results with the user-dependent scenario (F-measure: 0.78). On the other hand, the performance of subject 2 is much better in the user dependent scenario, revealing that the subject performed the activities with less variability.

TABLE 5.3: User-dependent experiments (locomotion). The table summarizes the results obtained by the groups that participated in the OC, the work of **Zhu et al. (2017)** and the implementation made in this thesis. The highest F-measure is highlighted in dark blue.

User dependent (locomotion)				
Method	F-measure			
	Subject 1	Subject 2	Subject 3	Subject 4
LDA	0.62	0.64	0.68	0.43
QDA	0.67	0.66	0.71	0.45
NCC	0.60	0.58	0.56	0.45
1-NN	0.84	0.85	0.83	0.76
3-NN	0.85	0.86	0.83	0.77
UP	-	0.58	0.62	-
NStar	-	0.58	0.66	-
SStar	-	0.61	0.68	-
CStar	-	0.60	0.65	-
NU	-	0.54	0.49	-
MI	-	0.85	0.81	-
MU	-	0.57	0.68	-
UT	-	0.48	0.55	-
Zhu	0.88	0.88	0.80	0.85
This work	0.90	0.91	0.81	0.88

TABLE 5.4: User-dependent experiments (gestures). The table summarizes the results obtained by the groups that participated in the OC, the work of **Zhu et al. (2017)** and the implementation made in this thesis. The highest F-measure is highlighted in dark blue.

User dependent (gestures)				
Method	F-measure			
	Subject 1	Subject 2	Subject 3	Subject 4
LDA	0.65	0.63	0.70	0.62
QDA	0.60	0.57	0.69	0.64
NCC	0.48	0.48	0.51	0.35
1-NN	0.85	0.89	0.86	0.84
3-NN	0.85	0.89	0.86	0.88
UP	-	0.64	0.64	0.64
NStar	-	0.84	0.83	-
SStar	-	0.87	0.84	-
CStar	-	0.88	0.87	-
NAGS	-	-	-	0.71
Zhu	0.87	0.89	0.87	0.85
This work	0.84	0.91	0.78	0.84

5.3 User-independent scenario

The *user independent* experiment involves the evaluation of a model that is trained on the data of three subjects and tested on the fourth one. This scenario simulates an application, where the implemented HAR system is used for a new a person, for whom the model had no previous training information and its performance relies on the generalization capabilities that the model obtained during the training with the other people. To test the user-independent scenario, a leave-one-subject-out cross-validation is performed: three subjects are used for training and one subject is used for testing. This procedure is repeated four times, variating each time the testing subject. Finally, the overall recognition performance is calculated as the average performance across all cross-validation rounds **(Bulling et al., 2014)**.

The results of this experiment are shown in **Table 5.5**. The table represents only the results obtained in this work, as no other references were found in the literature that evaluated a user-independent scenario. The intraclass variability challenge (*i.e.,* people can perform the same activity in different ways **(Section 2.4)**) becomes evident, as there is an overall dropout in the performance of the system. Even though a model benefits from data from other subjects to increase its generalization capabilities, this experiment shows that, in experiments where few subjects are involved, it is as well necessary to train a model with data from the target subject in order to obtain a better performance. This need certainly becomes negligible as more subjects are involved in a data collection experiment.

TABLE 5.5: User-independent experiments. Individual results using one subject as a test set are displayed, as well as the overall performance.

User independent		
Test set	**Locomotion**	**Hand gestures**
Subject 1	0.81	0.78
Subject 2	0.82	0.68
Subject 3	0.74	0.62
Subject 4	0.78	0.78
Overall	0.79	0.72

5.4 Use case: drinking activities

Among the hand gestures targeted in the OC, especially the *drinking* events represent a target activity that can be relevant in most ADL and health monitoring systems (*e.g.,* the REACH project **(Section 1.1)**). The accurate recognition of this activity can give a profile of the hydration habits of a patient, and this information, combined with the monitoring of other activities[3], can be useful for care givers to track the rehabilitation of a patient, and to take actions if the activity patterns of the patient are not the ones that were expected (*e.g.,* the patient maintains a sedentary life style, the patient's hydration levels are low, the patient skipped a drinking event at a certain time where he needed to drink a pill **(Figure 1.1)**).

The results obtained for the drinking events with the implemented ARC on the OCD **(Section 5.1)** are shown in **Figure 5.9**. Even though the overall F-measure for the hand gesture activities was of 0.9, the metrics shown in the figure reveal that the drinking activity was not correctly recognized by the system, since 35.3% of the returned activities were fragmented and correspond to 11.6% of the total P frames. Moreover, the underfill represents 32.9% of the positive frames, and reveals that the system does not recognize immediately when a drinking activity starts. This behavior can be explained by decomposing the drinking activity into three consecutive actions:

1. A person has a cup in his hand and moves it towards the mouth.
2. The person holds the cup near to the mouth while drinking.
3. The person finishes drinking and moves the cup away from the mouth.

A visual inspection of sensor time series for this activity is displayed in **Figure 5.8**. The figure shows that the three steps of a drinking event are clearly identified in the *x-axis* acceleration recordings of the right lower arm (RLA) IMU: the acceleration increases as the arm goes up, then it stays still as the person drinks and then accelerates in the opposite direction as the person lowers the arm[4]. Additionally, the drinking

[3]For example, monitoring locomotion activities can give an estimate of the physical activity levels of a patient through the day and track the development of these activities through time.

[4]This inspection is in-line with the analysis of drinking events made by **Shokoohi-Yekta et al. (2015)**

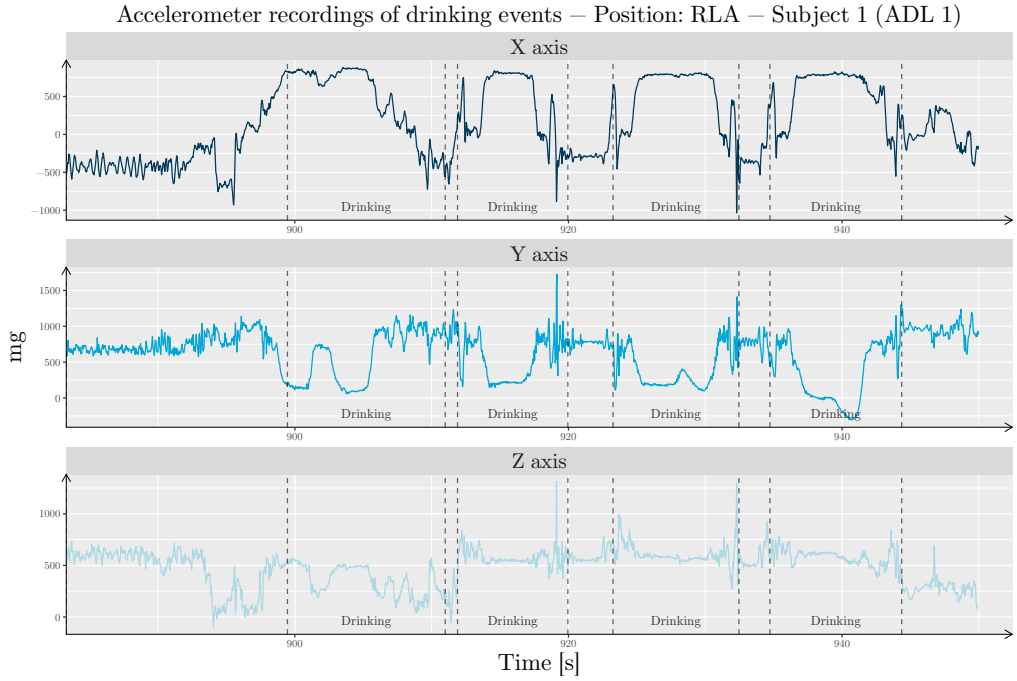


FIGURE 5.8: Tri-axial accelerometer recordings of drinking events.

activities were inspected and it was found that an average drinking event lasts approximately 6 seconds. The implemented ARC failed to recognize drinking events effectively, because the defined time window had a length of 0.5 seconds, explaining both the underfill and fragmentation errors:

- Underfill events are explained because the actions *1)* and *3)* of the drinking event can be mistakenly recognized as another hand gesture when the time window only gathers 0.5 seconds of the recordings.
- Fragmentation events are explained because there is no much movement involved with the arms in the action *2)* of the drinking event. This lack of movement in the arms might be recognized as the *NULL* class, thus fragmenting the drinking activity.

Based on these considerations, an alternatively configured ARC was tested, focused **only** on recognizing the drinking events, and using the same training and testing sets as the OCD. This served to compare the results with the ones in **Figure 5.9**. This ARC follows the same practices as shown in **Table 5.1**, except the window length that was set to 6 seconds (as it is the average length of a drinking event), and the overlap was set to 66% (as it had shown the best results in **Figure 5.1**).

Additionally, it was desired to test a real-life scenario in which a person would not have to carry a high amount of sensors as it was the case for the collection of the OCD. For that reason, only two IMUs were selected: RLA and LLA (right and left lower arms, **Figure 4.1a**). All embedded sensors in these IMUs (accelerometers, gyroscopes, magnetometers, and quaternions) were considered.

The results of the *drinking-focused* ARC are shown in **Figure 5.10**. They reveal that the window length influences the fragmentation rate, as no fragmentation events are

present in the predictions of this ARC. Since the window length is now increased, the system acts in a more conservative way, and the number of return events is lower than in the 0.5s setting (from 28 returned events to 12). 11 of the returned events correspond correctly to a drinking event (91.7%) and the inserted event is insignificant (in terms of the length of the event) as the N metrics shows that insertion errors represent only 0.1% of the N frames (*ir*). However, the deletion rate (*dr*) increases, and shows that the system fails to recognize 6 of the total drinking events (35.3%). Finally, the underfill errors are also reduced with a longer window length.

These results show that the predictions of the *drinking-focused* ARC are more reliable than the predictions of the 'OCD-ARC' since the fragmentation rate was reduced from 11.8% to 0% and the overall timing errors (underfill and overfill) were reduced from 32.9% to 26.3%. However, this does not mean that all drinking events are predicted by this system, as the deletion rate increases from 8% to 35.4%. Both the improvements and the decline in performance are highly influenced by the window length: longer window lengths serve predict complex target activities better, but the length of the activity needs to be similar to the window length. If the latter is not fulfilled, then the features calculated from the time window might get 'confused' with other activities or the *NULL* class (leading to an increase of the deletion rate). As mentioned before, drinking is a composite activity that can be divided into three actions. Another approach to recognize this activity would be to model an ARC that focuses on recognizing these three actions that are afterwards used as an input to a rule-based classifier that focuses on recognizing the drinking events. This consideration is part of the future work, mentioned in **Section 6.2**.

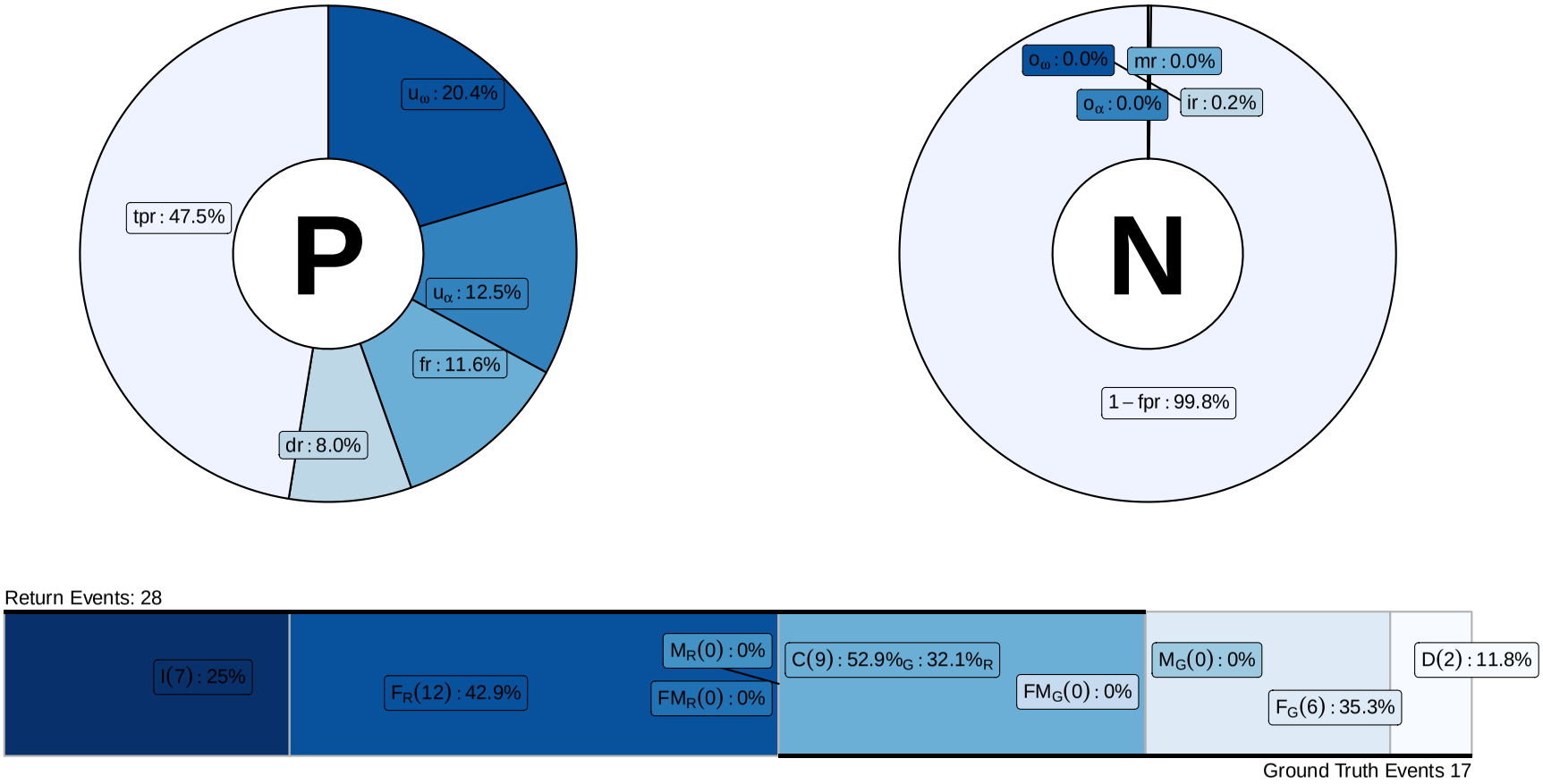


FIGURE 5.9: HAR metrics (frame and event analysis) for drinking events obtained with the optimal ARC on the OCD.

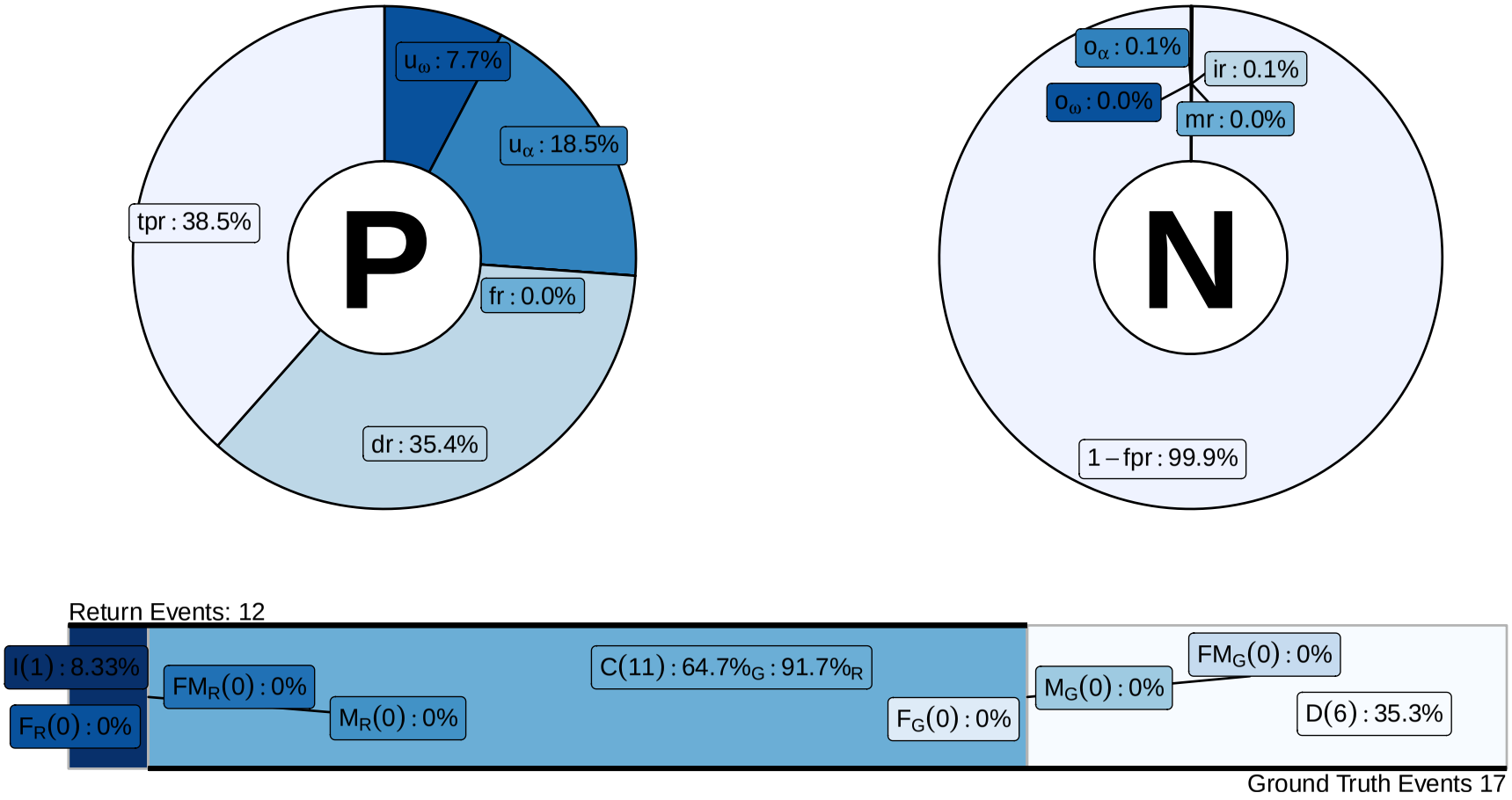


FIGURE 5.10: HAR metrics (frame and event analysis) for drinking events in a *drinking-focused* ARC.

Chapter 6

Discussion, Future Work and Conclusions

6.1 Discussion

6.1.1 Optimal sensor locations

The application of a recursive feature selection for the classifiers modeled for the OPPORTUNITY Challenge Dataset **(Section 5.1)** has shown that locomotion activities can be explained with the same performance using 20% of the total generated predictors, compared with a setting where all predictors are used. Additionally, feature selection reduces noise in predictors for the gesture recognition model, and increases its performance by 2%. Besides the performance improvement, feature selection reduces the complexity of the models and the time taken in training and testing.

Compared with decision trees, a random forest loses interpretability of the results, because it combines the decision of several trees, thus impeding the analysis of a single decision tree to see which splits were made, and based on which predictor. However, the *variance importance* calculation **(Section 3.1.5)** allows to observe the most relevant features among all trees. The implemented ARC calculated ten features per sensor time series and per window, meaning that the feature importance also gives information about the sensor locations that were more relevant to the model. **Figure 6.1** shows the most important sensor locations, based on the 244 most important features, and highlights the IMU located at the back of the subject as the most important location (BACK), followed by the right and left shoes' IMUs (RSH/LSH), and those at the left and right upper parts of the arms (LUA, RUA). In contrast, the most important locations for the hand gestures are precisely the IMUs located in the lower and upper parts of both arms, and the IMU located in the back. The results in total highlight the effectiveness of the feature selection approach, where the locomotion model gives importance to the four extremities and the back, and the gesture model gives importance to the back and the lower and upper parts of the arm.

Moreover, the results of feature selection show that from the ten type of extracted features per sensor measurement, only six were relevant for the gesture recognition model: mean (20,9 % of the total selected features), median (18,85%), min (18,44%), max (17,21%), energy (12,70%), and RMS (11,89%). For the locomotion recognition model, nine features were relevant: median (15,57%), mean (14,75%), max (13,52%),

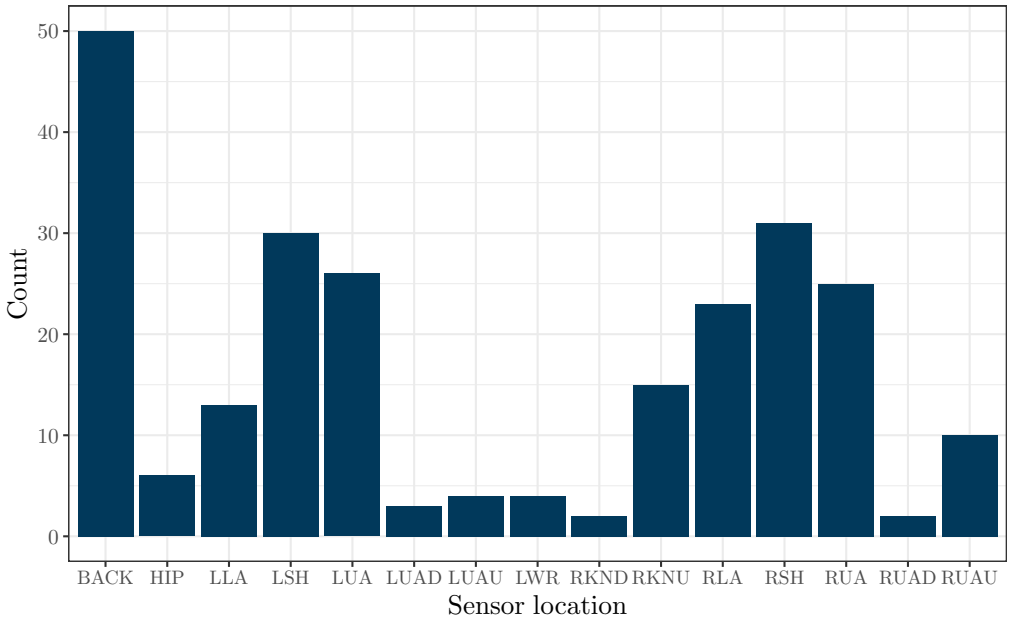


FIGURE 6.1: The most relevant sensor locations for the 'modes of locomotion model'. The model assigns most importance to the feet, the upper parts of the arms and the back. This behavior is expected, as these activities involve the movement of the whole body.

RMS (12,70%), min (12,30%), energy (11,07%), SD (1,22%), MAD (0,41%), and IQR (0,41%). The zero crossing rate feature was not relevant for either one of the models.

6.1.2 Recognizing composite activities

The results for the *drinking* events **(Section 5.4)** show a visible trade-off between underfill/fragmentation errors and deletion errors when choosing a window length for the system. This highlights the limitation of an activity recognition system that seeks to recognize composite activities based only on classification of feature vectors of time windows. Locomotion activities are well recognized by the system, because the sensors capture the characteristic motions of the body, as well as low-level actions such as hand gestures like "open door", "close door", "clean table", etc. Drinking events can be decomposed into three successive actions (reaching for a cup, taking a sip, putting away the cup), and the ARC would perform better by recognizing these actions that afterwards could be fed as input to a rule-based classification.

6.1.3 Target activities for a classifier

Two groups of target activities were evaluated by implementing a dedicated classifier for each group (*i.e.,* locomotion activities and hand gestures), as it was proposed in the OPPORTUNITY Challenge baselines and the related work of **Zhu et al. (2017)**. This approach however has some limitations because the fixed sliding window size restricts the amount of data that the model "sees" in order to make a prediction. Notice that three modes of locomotion were recognized with a low fragmentation

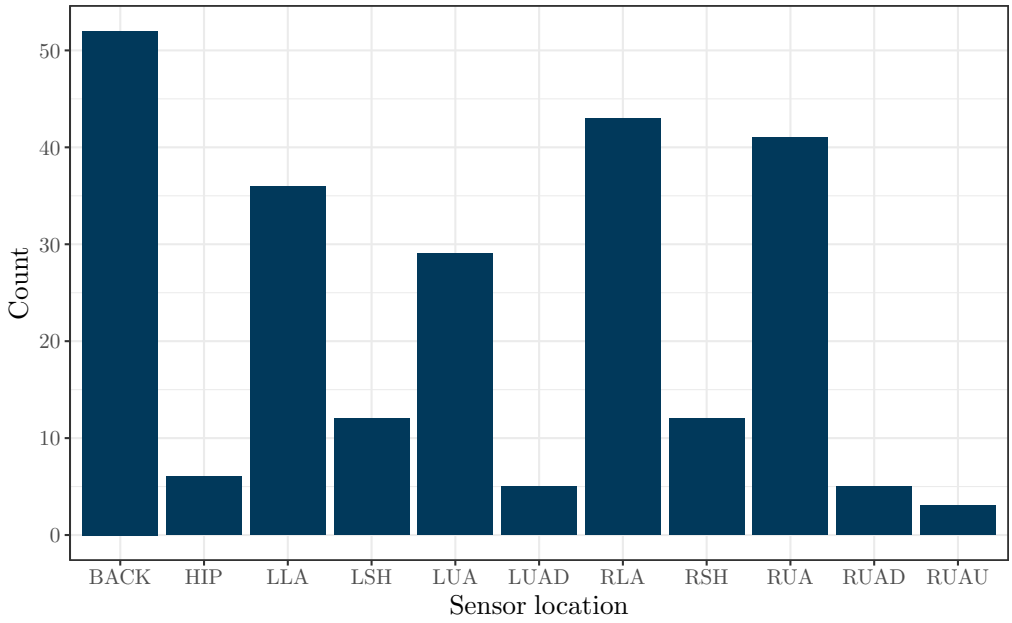


FIGURE 6.2: The most relevant sensor locations for the 'hand gesture model'. The model assigns most importance to the back and the lower and upper parts of the arm, which are precisely the most relevant body parts for the execution of this kind of activities.

rate (lying (1.1%), standing (1.4%), and sitting (1.2%)), while walking has a 10% fragmentation rate. Walking events can be described as the sequential movement of the legs, and a complete two-step cycle is desired within each time window, which on average lasts 1.33 seconds **(Anguita et al., 2013)**. The increase of the window length has shown benefits by reducing the fragmentation rate of the drinking events since this event takes longer than the rest of the activities contained in the hand gestures group. Therefore, real-life applications should not rely on a single ARC to deliver all predictions, but instead on an ensemble of specialized ARCs, which could provide better results. Each ARC needs to be tuned for specific activities that share similar execution lengths.

6.2 Future Work

This work has implemented several ARCs using data from a HAR benchmark dataset: the OPPORTUNITY dataset (OD). The proposed methodology outperforms the baselines established by the creators of the OPPORTUNITY challenge, the groups that participated in the OPPORTUNITY challenge, and several state-of-the-art results achieved by **Zhu et al. (2017)** [1].

[1]These improvements might be explained by the window length. **Zhu et al. (2017)** used a window length of 3 seconds and did not test the performance of their system with other window lengths. **Figure 5.1** showed that a shorter window length increases the performance of the system and the implemented ARC used a window length of 0.5 seconds.

The presented work could represent a basis for further research and improvement, as for example:

- Explore methods that cope with the user-independent HAR challenge, improving the system's performance in scenarios, where data from new subjects have to be processed without prior training.
- Test the performance of the ARC with other classifiers, as well as enhancements to the implemented classifier (*e.g.,* boosting).
- Extend the methodology of activity recognition to deep learning approaches **(Wang et al., 2018)**.
- Construct *rule-based* classifiers that take as input the predictions from the ARC. These classifiers should address high-level activities and their performance could also be tested with the OD, as several high level activities were also annotated **(Liu et al., 2016)**.
- Construct data collection experiments that address target activities within the REACH scenario and test the performance of the proposed methodology. This is an already planned phase, proposed by the directives of the REACH project. Due to the limitations in the availability of large annotated datasets, techniques of semi-supervised learning are as well within the scope of future research.

6.3 Conclusions

At the core of this thesis, the author has implemented an activity recognition methodology (*i.e.,* Activity Recognition Chain, ARC) to recognize human activities using sensor data **(Bulling et al., 2014)**. The goal was to test the proposed ARC on a benchmark dataset, allowing to compare results from previous works in the area.

This work combines a review of the publicly available datasets focused on Human Activity Recognition (HAR), as well as a description and an exploratory data analysis of the selected dataset, the OPPORTUNITY challenge dataset, a dataset that stood above other considered datasets as it provided the richest sensor environment, the most detailed data collection procedures and a collection of target activities relevant for surveillance and monitoring systems for Activities of Daily Life (ADL). Two independent ARCs were evaluated, one for each group of target activities in the dataset: locomotion activities and hand gestures.

Missing values were present in the OPPORTUNITY dataset, due to mechanical failures and connection errors **(Chavarriaga et al., 2013)**. Several missing value imputation methods were tested and the linear interpolation method was selected as it presented the best trade-off between performance and computational running time.

An ARC is composed of sequential steps (*i.e.,* data acquisition, preprocessing, data segmentation, feature extraction, and classification) and each step includes several methodologies and parameters that need to be addressed **(Bulling et al., 2014)**. Therefore, nine ARCs were implemented to test the influence of the ARC parameters on the system's performance.

The evaluation of the results followed standard metrics used also in pattern recognition problems (*i.e.,* accuracy, precision, recall, and F-measure). In addition, a HAR-specific performance metric was implemented following the work of **Ward et al. (2011)**. The results revealed that short time windows yield a better performance than longer time windows. However, if the time window's length is considerably shorter than the time it takes to execute the target activity, then fragmentation errors will incur in the performance of the system. The overlap size of sliding windows captures the dynamics that happen at the end of a time window and the beginning of the consecutive window. This becomes evident with the overall increase of the performance in the ARCs that used an overlap of 66% between two consecutive windows. Additionally, the segmentation of the data in time windows of 0.5 seconds yielded the best results in the OPPORTUNITY challenge setting.

A review of the most frequently used features in HAR was done, showing that statistical and time-related features perform better over frequency-related features **(Zhu et al., 2017)**. Ten features were extracted for each sensor's measurement, yielding a total of 1220 features per time window.

A random forest algorithm was selected for the classification stage of the ARC, as it present several advantages over other classifiers, such as being an ensemble of decision trees, a built-in estimate of the test error, and a measure of the "importance" of each extracted feature **(Section 4.2.4)**. The "measure of variable importance" allowed to implement a recursive feature selection method that reduced the amount of features to 244, yielding the same results as when using all (1220) features in the dedicated ARC for locomotion activities, and an improvement of the performance in the ARC for hand gestures. Additionally, the feature importance measure demonstrated that the classifier pays special attention to specific sensors depending on the target activity: the classifier for recognizing locomotion activities gives special importance to sensors located in the shoes, upper part of the arms, and the back, while the classifier for recognizing hand gestures focuses on the sensors in the upper and lower parts of the arm, and the back.

The performance of the implemented ARCs surpassed the baselines of the OCD. User-dependent experiments showed that this methodology achieves state-of-the-art results when recognizing locomotion activities and the user-independent scenario exhibited an overall decrease in the system's performance, enforcing the challenge of predicting activities from a subject, based only on knowledge gain from other subjects. This limitation can be addressed by including more subjects to the data collection experiments.

Finally, a specific use case was addressed: the recognition of drinking events. Drinking activities can be decomposed into three sequential actions, and this differentiation is also evident in the time series **(Figure 5.8)**. An average drinking event in the OCD lasts 6 seconds, and a segmentation of the time series into windows of 0.5 seconds cannot capture a complete activity, leading to fragmentation errors. A *drinking-focused* ARC was implemented with a time window of 6 seconds, using only the two lower arm IMUs, with the aim of resembling a real-life scenario, where a user cannot wear a highly-multimodal sensor setting as it was used for collecting the OCD. The system improved the event metrics passing from a 32.1% of correct classified events to 91.7%. This improvement is mostly due to the reduction of the fragmentation rate from 42.9% to 0% in the event metrics, and from 11.6 % to 0% in the frame metrics. Nonetheless, any variation in the way the activity is performed, and the length of

execution, might provoke that the system confuses the activity with any other. This behavior is present in the *drinking-focused* ARC, as the deletion rate increases from 8% to 35%. A workaround to this limitation would be to use a rule-based classifier on top of the ARC: composite activities such as drinking have an ordered sequence and these actions can be recognized directly with the ARC. A rule-based classifier would take the prediction of these actions as an input and would output the prediction of the drinking event.

Appendix A

Description of the considered publicly available datasets

This appendix gives an extended description of the considered datasets for this work. The database structure and the data collection protocol were reviewed in order to choose the final dataset. This review was done by the author of this thesis and is also part of one of the deliverables for the REACH project **(Konietzny et al., 2018)**.

A.1 Wearable Action Recognition Database

Database Structure

Yang et al. (2009) recorded data from 20 subjects (13 male and 7 female) by placing five wireless sensors in different body locations: two wrists, the waist, and two ankles. Each custom-built sensor carried a triaxial accelerometer and a biaxial gyroscope, where each sensor axis has a 12 bit resolution and a range of $\pm 2g$ and $\pm 500\frac{^\circ}{s}$ for the accelerometer and gyroscope, respectively[1].

The WARD dataset includes a set of 13 action categories: stand, sit, lie down, walk forward, walk left-circle, walk right-circle, turn left, turn right, go upstairs, go downstairs, jog, jump, and push wheelchair. Each subject performed the actions based on his/her own interpretation and style and every action was performed for more than 10 seconds and repeated at least ten times.

Data Collection Remarks

The data were collected over a period of two weeks. The wearable sensor network consisted of sensor nodes placed at multiple body locations, which communicate with a base station attached to a base-station computer via a USB port. The sensor nodes and base station were built using Tmote Sky boards[2]. Tmote Sky runs TinyOS on an 8MHz microcontroller with 10K RAM and communicates using the IEEE 802.15.4

[1]The WARD Dataset is publicly available here:
http://www.eecs.berkeley.edu/~yang/software/WAR/

[2]Tmote Sky boards datasheet available here:
http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf

wireless protocol. Each custom-built sensor board has a triaxial accelerometer and a biaxial gyroscope, which is attached to Tmote Sky. **(Yang et al., 2009)**

The authors implemented a TDMA protocol to avoid packet collision, that allocates each node a specific time slot during which to transmit data, achieving a sampling rate of 20Hz with minimal packet loss. To avoid drift in the network, the base station periodically broadcasted a packet to resynchronize the nodes' individual timers. **(Yang et al., 2009)**

Some amount of measurement error was introduced by the hardware itself: the accelerometers required some calibration in the form of a linear correction, as sensor output under $1g$ may be shifted up to 15% in some sensors. The authors also reported that the gyroscopes produced an indication of rotation under straight line motions and that those systematic errors were consistent across experiments for a given sensor board. Without calibration to correct these issues, the errors might affect the action recognition if different sets of sensors were used interchangeably in the experiment. **(Yang et al., 2009)**

A.2 The Opportunity activity recognition dataset (OPPORTUNITY)

Database Structure

Chavarriaga et al. (2013) designed an activity recognition environment to generate several activities in a realistic manner. They made publicly available the data from 4 subjects that participated in the experiment[3], where each one wore 7 inertial measurement units (IMUs), 12 3D acceleration sensors, and 4 3D coordinates from a location system[4]. Additionally, the experiment possessed 12 object sensors (12 objects that were instrumented with wireless sensor measuring 3D acceleration and 2D rate of turn), and 21 ambient sensors (13 switches and 8 3D acceleration sensors in kitchen appliances and furniture).

The OPPORTUNITY dataset has a collection of activities that are divided into four categories:

- Modes of locomotion, *i.e.,* stand, walk, sit, and lie.
- Low level activities, *i.e.,* activities made with one hand (unlock, stir, lock, close, reach, open, sip, clean, bite, cut, spread, release, and move) and objects held by one hand (bottle, salami, bread, sugar, dishwasher, switch milk, lower drawer, spoon, knife cheese, middle drawer, table, glass, cheese, chair, door 1, door 2, plate, top drawer, fridge, cup, knife salami, and lazychair.).

[3]The OPPORTUNITY Challenge dataset is publicly available here:
https://archive.ics.uci.edu/ml/datasets/opportunity+activity+recognition

[4]Four tags for an ultra-wideband localization system were placed on the left/right front/back side of the shoulder. These deliver 3D coordinates of the tag in a room coordinate system.

- Middle level activities, *i.e.,* activities made with both hands: open door, close door, open fridge, close fridge, open dishwasher, close dishwasher, open drawer, close drawer, clean table, drink from cup, and toggle switch.
- High level activities, *i.e.,* relaxing, coffee time, early morning, cleanup, and sandwich time).

The latter activities were made within two different protocols: ADL run and drill run. The former consists of temporally unfolding situations (*e.g.,* groom, relax, prepare coffee, drink coffee, etc.) and the latter consists of 20 repetitions of some sequence of activities (*e.g.,* open and close the fridge, open and close the dishwasher, clean the table, etc.) to generate a large number of activity instances. Each subject performed the ADL run five times and the drill run only once **(Roggen et al., 2010)**.

Data Collection Remarks

Seven computers acquired the data from specific sensors systems, and on-body sensors were managed by a dedicated laptop in a backpack **(Roggen et al., 2010)**. Data loss were most present when sensing body motion using streaming sensors, since this stressed the wireless infrastructure and led to packet losses. Wireless sensors communicated via Bluetooth and when one sensor lost connection, the system automatically attempted to reconnect to that sensor, minimizing the amount of data loss.

The multiple independent devices recording data were synchronized in post-processing.

The runs were recorded and the annotations were made based on the recordings. The authors reported that a 30-minutes video footage required about 7-10 hours to be annotated **(Roggen et al., 2010)**.

A.3 Physical Activity Monitoring for Aging People (PAMAP)

Database Structure

Reiss and Stricker (2012) created the database from 9 human subjects (8 male and 1 female, aged 27.22 ± 3.31 years) who wore three inertial measurement units (IMUs) and a heart rate monitor[5]. The heart rate monitor and one IMU were placed on the chest of the subject; a second IMU was placed over the wrist on the dominant arm, and the third IMU on the dominant side's ankle. Their IMUs contained two 3-axis MEMS[6] accelerometers (scale: $\pm 16g // \pm 6g$, resolution: 13 bit), a 3-axis MEMS gyroscope (scale: $\pm 1500\frac{^\circ}{s}$, resolution: 13 bit), and a 3-axis magneto-resistive magnetic sensor (scale: $\pm 400\mu T$,resolution: 12 bit), all sampled at 100 Hz. The authors collected heart rate data since physiological sensing is useful for intensity estimation: inertial sensing alone can not reliably distinguish activities with similar movement characteristic but

[5]The PAMAP Dataset is publicly available here: http://www.pamap.org/demo.html

[6]Micro-Electro-Mechanical Systems, or MEMS, is a technology that in its most general form can be defined as miniaturized mechanical and electro-mechanical elements (i.e., devices and structures) that are made using the techniques of microfabrication.

different energy expenditure, e.g. walking and ascending stairs, or even more difficult: walking and walking with some load **(Reiss and Stricker, 2012b)**. To obtain heart rate information, the BM-CS5SR HR-monitor from BM innovations GmbH was used, measuring heart rate with a frequency of approximately 9Hz.

Each of the subjects followed a protocol of 12 activities (lie, sit, stand, walk, run, cycle, Nordic walk, iron, vacuum clean, rope jump, ascend and descend stairs), and optionally performed a few other activities (watch TV, computer work, drive car, fold laundry, clean house, play soccer) as well.

Data Collection Remarks

Over 10 hours of data were collected altogether from the 18 different activities. A Viliv S5 UMPC (Intel Atom Z520 1.33GHz CPU and 1GB of RAM) was used as data collection companion unit. Labeling of the currently performed activities was done via a GUI specifically developed for this purpose on the UMPC. The collection of all raw sensory data and the labeling were running in separate threads in an application running on the companion unit, to lighten the synchronization of all collected data **(Reiss and Stricker, 2012b)**.

There were two main reasons for data loss in the dataset. The first reason is data dropping caused by wireless data transfer. The second, more severe reason was the fragile system setup due to the additionally required hardware components: 2 USB-dongles, a USB-hub and a USB extension cable were added to the companion unit in the custom bag. Especially during activities like running or rope jumping, the system was exposed to a lot of mechanical stress. This sometimes caused loosing connection to the sensors, or even a system crash, when the data recording had to be restarted and in a few cases the data collection could not be recovered even this way **(Reiss and Stricker, 2012b)**.

A.4 Human Activity Recognition Dataset (HAR)

Database Structure

This database is a collection of measurements from group of 30 volunteers with ages ranging from 19 to 48 years. Each person was instructed to follow a protocol of activities while wearing a waist-mounted Samsung Galaxy S II smartphone[7]. **Anguita et al. (2013)** collected triaxial linear acceleration and angular velocity signals using the phone accelerometer and gyroscope at a sampling rate of 50Hz. These signals were preprocessed for noise reduction with a median filter and a 3rd order low-pass Butter-worth filter with a 20 Hz cutoff frequency. The authors affirm that this rate is sufficient for capturing human body motion since 99% of its energy is contained below 15Hz **(Karantonis et al., 2006)**.

The six selected activities were: standing, sitting, laying down, walking, walking downstairs and upstairs. Each subject performed the protocol twice: on the first trial

[7]The HAR dataset is publicly available here:
http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

the smartphone was fixed on the left side of the belt and on the second it was placed by the user himself as preferred.

A.5 Daily Life Activities (DaLiAc)

Database Structure

This database consists of data from 19 subjects (8 female and 11 male, age 26 ± 8 years, height 177 ± 11 cm, weight 75.2 ± 14.2 kg, mean ± standard deviation (SD)) that performed 13 daily life activities[8]. Four SHIMMER (Shimmer Research, Dublin, Ireland) sensors were used for data acquisition **(Burns et al., 2010)**. The sensor nodes were placed on the left ankle, the right hip, the chest, and the right ankle. Each sensor node was equipped with a triaxial accelerometer and a triaxial gyroscope. Data were sampled with a sampling frequency of 200 Hz and were stored on SD card. The resolution of the acquired data were of 12 bit. The range for the accelerometers was $\pm 6g$. The range of the gyroscopes was $\pm 500\frac{^\circ}{s}$ for the sensor nodes wrist, chest, and hip and $\pm 2000\frac{^\circ}{s}$ for the sensor node on the ankle, since larger angular velocities were expected in the lower extremities.

The 13 performed activities were: sitting, lying, standing, washing dishes, vacuuming, sweeping, walking outside, ascending stairs, descending stairs, treadmill running (8.3 km/h), bicycling (50 watt), bicycling (100 watt), rope jumping

Data Collection Remarks

A mobile phone (Samsung Galaxy S2) was used as labeling device. An Android-based labeling application (running on the mobile phone) was used to label start time and end time of single activities concurrently to data collection. A researcher that labeled the start and end of each activity accompanied the subject during the whole data acquisition **(Leutheuser et al., 2013)**.

A.6 Realistic Sensor Displacement (REALDISP)

Database structure

Baños et al. (2012) created a database where they tried to simulate some of the variability that may occur in the day to day usage of an activity recognition system, involving wearable or self- attached sensors. In total, 9 on-body XSens XM-B sensors[9] were used with 17 study participants performing 33 fitness activities[10].

[8]The DaLiAc dataset is publicly available here:
https://www.mad.tf.fau.de/research/activitynet/daliac-daily-life-activities/

[9]XM-B technical information is available here:
https://www.xsens.com/wp-content/uploads/2013/11/XM-B_User_Manual.pdf

[10]The REALDISP dataset is publicly available here:
https://archive.ics.uci.edu/ml/datasets/REALDISP+Activity+Recognition+Dataset

Ideal-placement or default scenario (*i.e.,* the sensors are positioned by the instructor to predefined locations within each body part), self-placement (*i.e.,* the user is asked to position a subset of the sensors themselves on the body part specified by the instructor), and mutual-displacement (*i.e.,* an intentional de-positioning of sensors using rotations and translations with respect to the ideal placement is introduced by the instructor). **(Baños et al., 2012)**

The performed activities were: walking, jogging, running, jump up, jump front & back, jump sideways, jump leg/arms open/closed, jump rope, trunk twist (arms outstretched), trunk twist (elbows bended), waist bends forward, waist rotation, waist bends (reach foot with opposite hand), reach heels backwards, lateral bend, lateral bend arm up, repetitive forwards stretching, upper trunk and lower body opposite twist, arms lateral elevation, arms frontal elevation, frontal hand claps, arms frontal crossing, shoulders high amplitude rotation, shoulders low amplitude rotation, arms inner rotation, knees (alternatively) to the breast, heels (alternatively) to the backside, knees bending (crouching), knees (alternatively) bend forward, rotation on the knees, rowing, elliptic bike, and cycling.

Data Collection Remarks

The sensors and the Xsens master are wired together in a serial connection. The Master device was interfaced over Bluetooth to a laptop which continuously stores the information delivered by the nodes. The laptop was also used for labeling purposes. Both the data storage and the labeling process are performed using the CRN Toolbox **(Bannach and Lukowicz, 2008)**. The sampling rate was established to 50Hz **(Baños et al., 2012)**.

All sessions were recoded using a video camera. The video recording was used to check anomalous or unexpected patterns in the data and correct labeling mistakes.

A.7 Heterogeneity Human Activity Recognition Data Set (HHAR)

Database structure

Stisen et al. (2015) gathered data from nine users (age range 25 - 30 years). All users followed a scripted set of activities while carrying eight smartphones (2 instances of LG Nexus 4, Samsung Galaxy S+ and Samsung Galaxy S3 and S3 mini) and four smart watches (2 instances of LG G and Samsung Galaxy Gear)[11]. All eight smartphones were kept in a tight pouch and carried by the users around their waist, whereas two smartwatches were worn on each arm.

The activity protocol included the following activities: biking, sitting, standing, walking, stair up, and stair down. Each participant conducted five minutes of each activity, which ensured a near equal data distribution among activity classes (for each user and device).

[11]The HHAR dataset is publicly available here:
https://archive.ics.uci.edu/ml/datasets/Heterogeneity+Activity+Recognition

Data Collection Remarks

These smartphone models yielded different maximum sampling frequencies: 200 Hz for LG Nexus 4, 150 Hz for the Samsung Galaxy S3, 100 Hz for Samsung Galaxy S3 mini and 50 Hz for Samsung Galaxy S plus, approximately. The smart watches also varied in the supported maximum sampling rate, e.g., 200 Hz for LG G and 100 Hz for Samsung Galaxy Gear. Furthermore, the devices exhibited different accelerometer biases and gains **(Stisen et al., 2015)**.

A.8 Activity Recognition system based on Multisensor data fusion

Database structure

This dataset is a collection of 15 repetitions of seven activities. **Palumbo et al. (2016)** used wearable and environmental sensors and based the recognition of the user activity both on accelerometers embedded on the wearable sensors and on the Received Signal Strength (RSS) of the beacon packets exchanged between all the sensors (both wearable and environmental). RSS data was collected using IRIS nodes embedding a Chipcon AT86RF230 radio subsystem that implements the IEEE 802.15.4 standard and programmed with a TinyOS firmware. Three of them were placed on the user's chest and ankles, another one was placed on a furniture in the environment representing a meaningful place for a particular activity (*i.e.* a stationary bike for cycling activity recognition)[12].

The user performed the following activities: lying, sitting, standing, walking, bending (both keeping the legs straight and keeping the legs folded), falling, and cycling (using a stationary bike).

Data Collection Remarks

Each sequence in the dataset has the duration of 2 minutes (480 time steps) and corresponds to the actor performing a specific activity. The dataset contains a total number of 88 sequences, pertaining to the different activities. From the raw data they extracted time-domain features to compress the time series and slightly remove noise and correlations **(Palumbo et al., 2016)**.

The authors chose an epoch time of 250 milliseconds. In such a time slot they elaborated 5 samples of RSS (sampled at 20 Hz) for each of the three couples of sensor nodes (*i.e.* Chest-Right Ankle, Chest-Left Ankle, Right Ankle-Left Ankle). The features include the mean value and standard deviation for each reciprocal RSS reading from worn sensors **(Palumbo et al., 2016)**.

[12]The AReM dataset is publicly available here: https://archive.ics.uci.edu/ml/datasets/Activity+Recognition+system+based+on+Multisensor+data+fusion+(AReM)

Appendix B

OD: Exploratory Data Analysis

The first step when working with a dataset is to make an exploratory data analysis (EDA), which is an analysis that pursues getting acquainted with the data, uncovering underlying structures, and detect outliers and abnormalities **(Andrienko and Gennady, 2006)**. It is of interest within this EDA of the OPPORTUNITY dataset (OD) to calculate the length of the experimental runs to estimate the proportion of data that is available per subject with respect to the whole dataset. Additionally, by counting the number of instances that an activity took place is possible to estimate how difficult would represent to a model to recognize an activity due to *class imbalance*.
`NA` distribution through the sensors is inspected in order to assure the quality of the data that will be used as input in the Activity Recognition Chain (ARC). Finally, a visualization of sensor data and correlations between sensors was done in order to understand the relevance of processes involved in the ARC, such as sensor and feature selection.

B.1 Length of experimental runs

The ADL and Drill protocols for data collection **(Section 4.1.1)** in the OD did not establish a time in which the subjects should perform the activities. Instead, subjects were allowed to perform activities as naturally as possible and the time to perform the ADL and Drill protocols could have variations.

Each observation in the OD was taken every 33 milliseconds. An inspection of the number of observations that were taken for each subject and run was done in order to know if it was possible to treat the subjects indistinctly when splitting the runs between training and testing set. This inspection is shown in **Figure B.1** and shows that the distribution of observations between subjects is almost homogeneous. The first ADL run is systematically longer than the others ADL runs and might indicate that the subjects took more time in this run since they were getting used to the system and the data collection experiment. The homogeneity among the subjects' number of observations indicates that an approximately same proportion of data between the training and testing sets is always obtained when experimenting with a *user independent* setting by splitting the runs with the strategy of 'leave-one-run-out' (*i.e.,* selecting 3 subjects as training, one as testing).

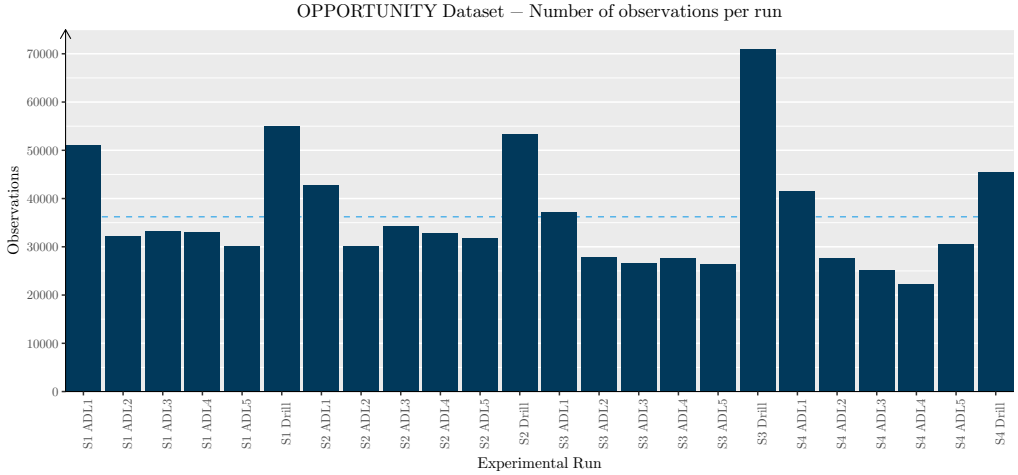


FIGURE B.1: Length of experimental runs. The number of observations recorded for each run is displayed. Each observation was recorded each 33 milliseconds. The blue light line highlights the dataset's mean number of observations.

B.2 Density of activities through the dataset

Chavarriaga et al. (2013) annotated different 'levels' of activities, that is, each sensor measurement had attached to it several labels that corresponded to locomotion activities, hand gestures, and high level activities.

Class imbalance is a challenge in HAR **(Section 2.4)** and it is necessary to analyze the density of labels per activity within the dataset in order to estimate how challenging it is to recognize the target activities by training a model with a given dataset.

The distribution of activity labels for modes of locomotion is shown in **Figure B.2** and exhibits that the NULL class does not dominate in terms of quantity among this activity level. This is expected, since most activities performed in the OD involve human motion, which can be classified in one of the four target categories of the modes of locomotion of the dataset. Standing is the most frequent label, indicating that most activities within the run took place while the subject was standing up. On the contrary, the '*Lie*' activity is the less frequent activity in the dataset, which is expected as well since this activity was only performed at the beginning and the end of the ADL runs only.

Nonetheless, the labels distribution for hand gestures **(Figure B.3)** is completely different. The NULL class represents more than 70% of the total samples among this activity level. The most sampled hand gesture corresponds to the activity *'Drink from Cup'*, which represents the 6% of the total samples. The rest of the hand gestures approximately evenly distributed and each possesses less than 2% of the sample data. This unbalance might affect the generalization capabilities for each hand gesture and could lead to models where one activity might be confused as another one (or NULL class), due to the lack of enough training data.

As a note, **Figure B.3** highlights the relevance of the *Drill* runs in a data collection experiment: take for example the '*Drink from Cup*' activity in **Figure B.3b**: *drinking* is an activity that does not happen frequently and, in order to capture more instances

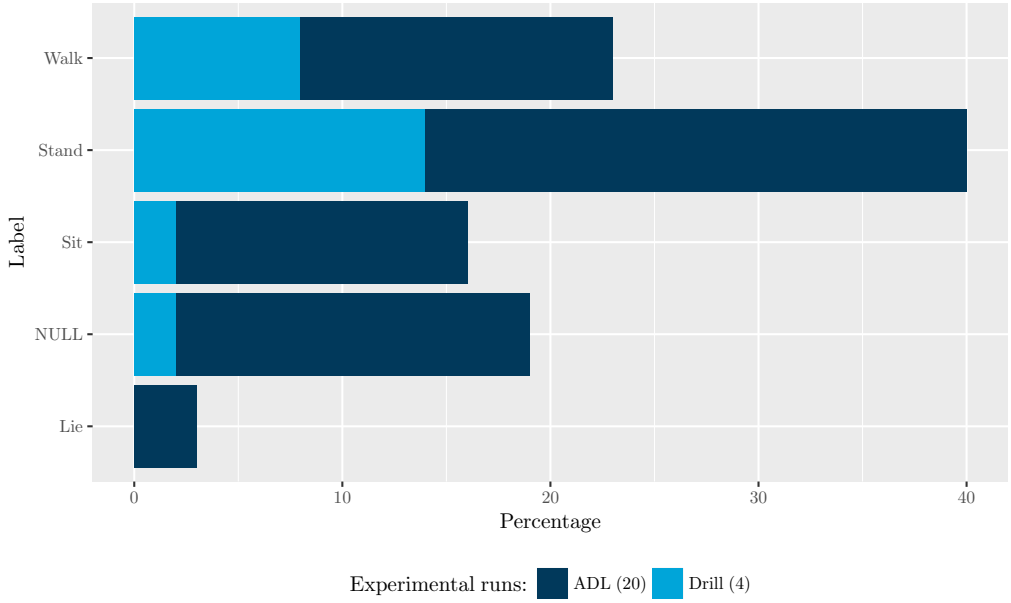


FIGURE B.2: Labels' distribution of modes of locomotion. The proportion of labels with respect to the total labels of the dataset is displayed.

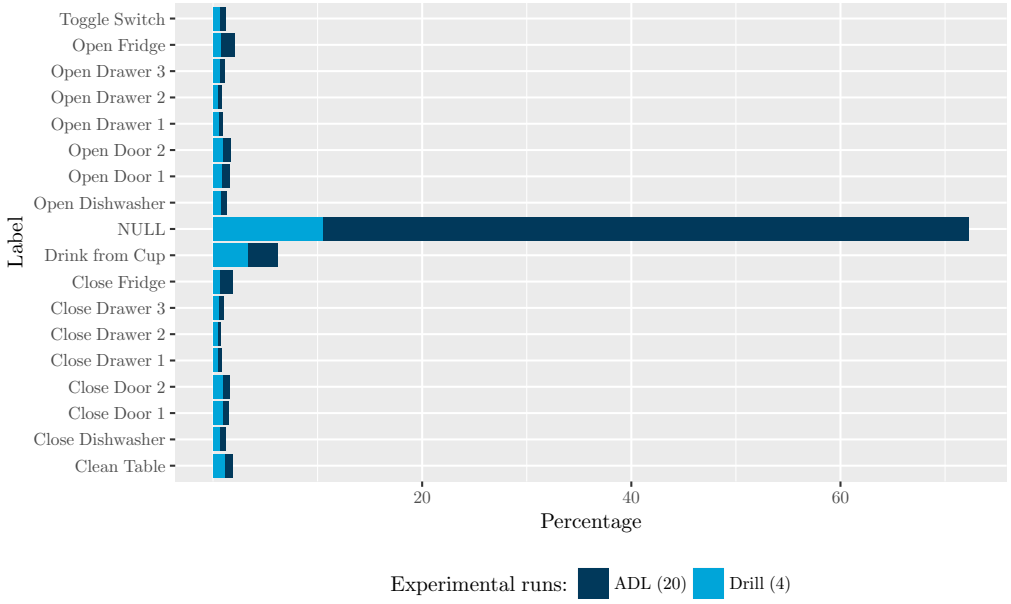
of this activity, the *Drill* run simulates repetitions of this activity to generate more instances that can be used for training. In the figure, the relevance of these runs is evident: 20 ADL runs generated less samples of this activity in comparison with the 4 *Drill* runs.

B.3 Density of NA per sensors

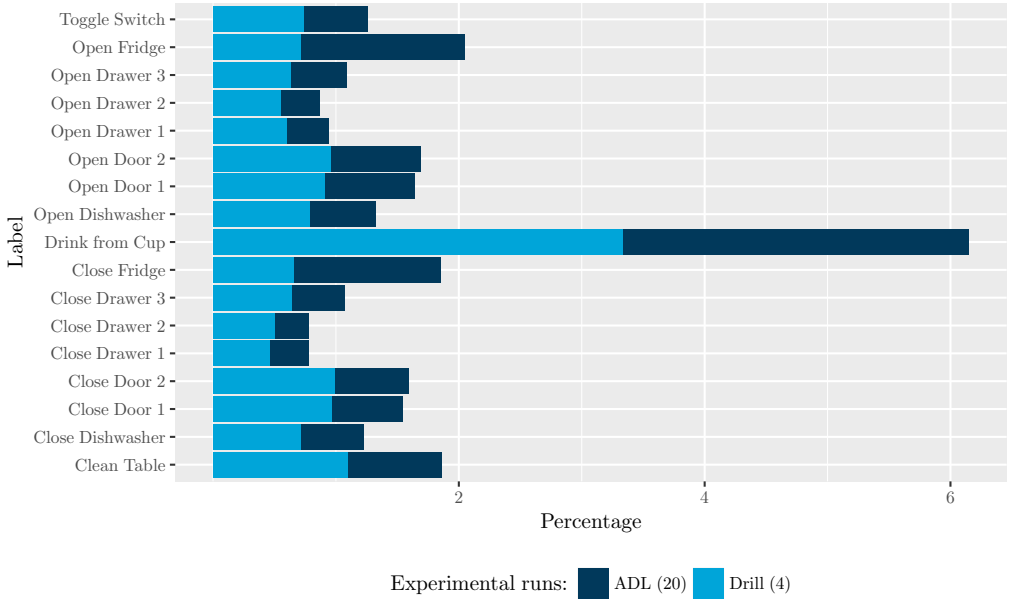
Missing values (NA) occur when there is no measurement stored for an observation. In the OPPORTUNITY dataset, several sensors present missing values because of connectivity issues (*e.g.,* stress in the wireless network causing failure in the communication between the sensor and the data collectors) and mechanical failures (**e.g.,** the wired sensors presented sporadic disconnections when a subject performed a movement) (Roggen et al., 2010).

Analyzing the amount of NA values per sensor is essential before performing any NA imputation method since highly dense NA time series will not produce meaningful results after the imputation is done. For example, if a sensor has 90% of its data missing, then the imputation method will reconstruct this amount of data based only on 10% of the information, leading to a misleading predictor.

Remember that **Chavarriaga et al. (2013)** gathered 24 runs from 4 subjects. Therefore, there might be unique runs where one sensor's performance might not be suitable, while in other runs the amount of NA values is not critical. However, when building a model that is trained on all runs, it is necessary to establish which predictors (sensors) will be used for modeling, meaning that if at least one run presents one sensor that failed, then such sensor cannot be considered for modeling at all.



(A) NULL class displayed



(B) No NULL class displayed

FIGURE B.3: Labels' distribution of hand gestures. The proportion of labels with respect to the total labels of the dataset is displayed. *NULL* class represents more than 70% of the labels and for visualization purposes this class is removed in one plot.

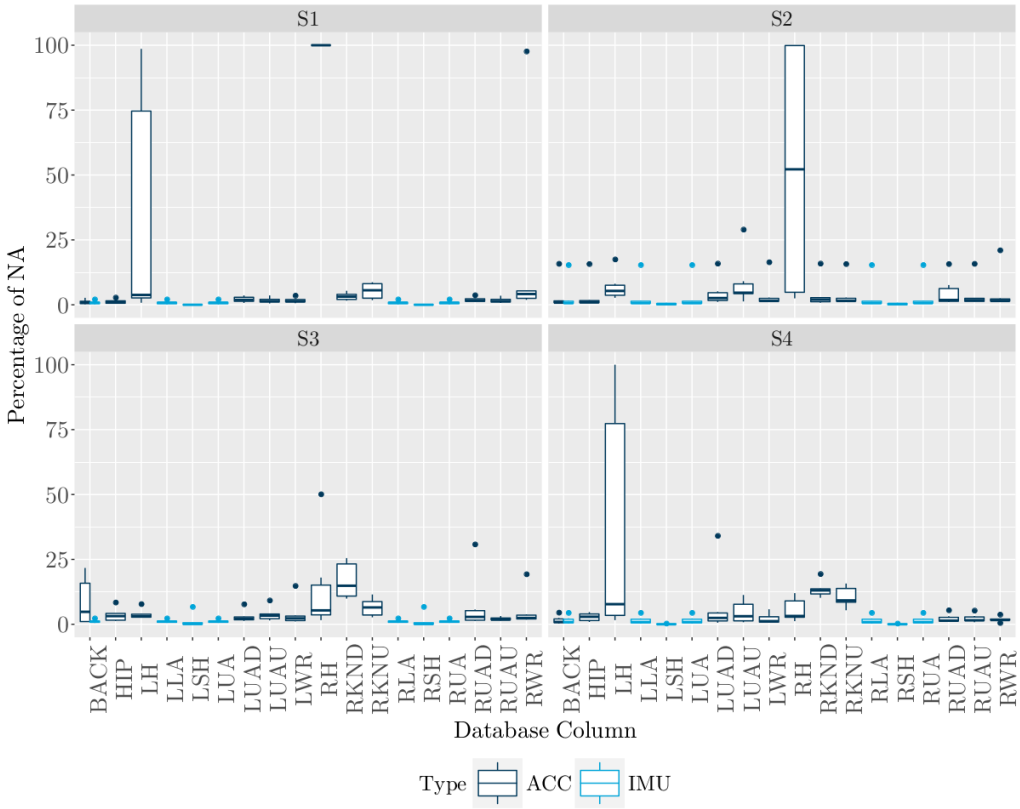
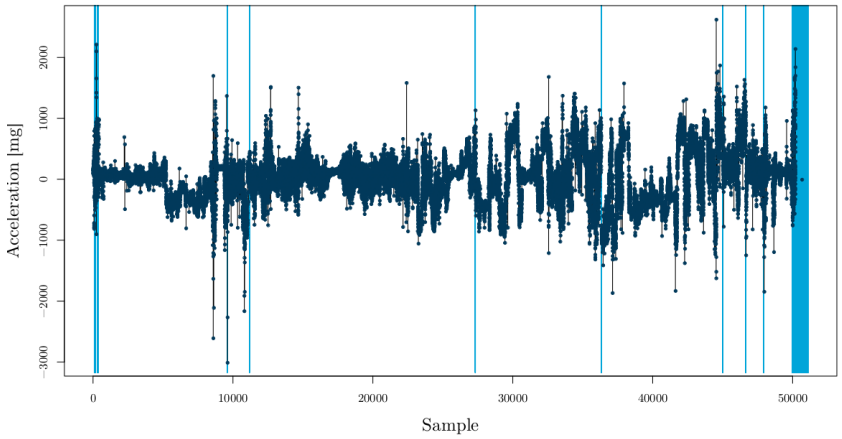


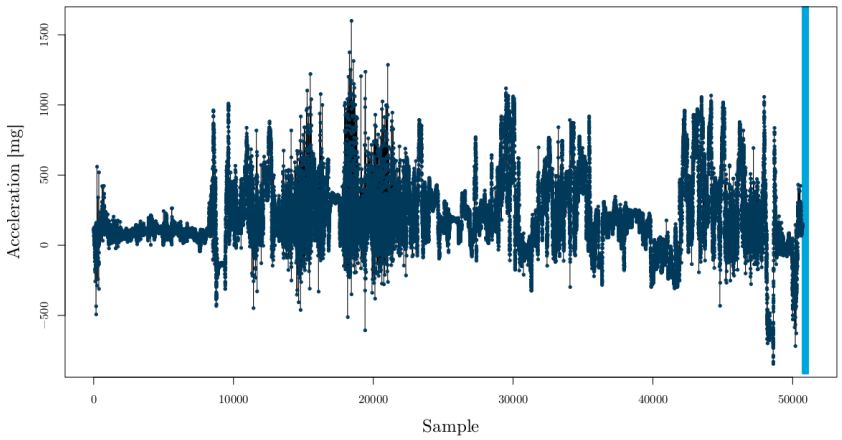
FIGURE B.4: Percentage of NA values per sensor *(x-axis)* and per subject *(S)*. Most sensors have under 5% of NA values. Moreover, accelerometer data have more missing values than IMUs.

Figure B.4 shows the percentage of NA values per sensor and per subject. NA values are distributed through all axes when a sensor fails to measure. For example, each time that one body IMU fails to register a sample, then it produces 1 NA per measured component in the sensor (*i.e.,* 1 NA on each accelerometer, gyroscope, magnetometer, and quaternion axis). The boxplots show that most of the sensors present a low amount of NA values, with the exception of the left hand (LH) and right hand (RH) accelerometers which on several runs these sensors did not record any measurements. The right hand accelerometer (RWR) has one run that has complete NA values (which in the figure is shown as the outlier point). These three sensors were taken out from the ARC implementation because of the low quality of the measured data. Even though the RWR accelerometer has only one run that is missing, the position of this accelerometer is close to the RLA inertial measurement unit, which possess as well acceleration recordings of that section of the arm.

Additionally, **Figure B.4** shows that the quality of IMU data has a higher quality in comparison with the accelerometers, since the percentage of NA values is systematically higher in accelerometers than IMUs. Inspecting sensor data individually, it was found that most accelerometers failed several times through the experimental runs (*e.g.,* **Figure B.5a**), while most IMUs present only data loss at the end of the run (*e.g.,* **Figure B.5b**). Most all sensors have NA values at the end of the runs, which perhaps is related to the stage where sensors were being shut down after the data collection experiment and this was not done simultaneously. It was inspected that the last 2% of the data contains for most IMU sensors the 100% of NA. Furthermore, the



(A) Accelerometer: left hand (LH). Subject 1 (ADL 1). `NA` values are present along the experimental run and the biggest `NA` gap appears at the end of the run.



(B) IMU: back (BACK). Subject 1 (ADL 1). `NA` values are only present at the end of the experimental run.

FIGURE B.5: Missing data in sensor recordings. The `NA` values are highlighted in light blue and represent the moments the sensor failed to record a measurement.

last 2% of data contains only *NULL* class labels in all activity levels. Therefore, the last 2% of the data was removed as a pre-processing step.

Appendix C

Details about the implementation in R

C.1 General structure

The code consists of four main sections that resume the ARC process: reading data, preprocessing, feature extraction, and classification. The last three sections were wrapped into functions ('ARC-functions') to assure reproducibility of the method with future datasets, while the reading data process may have particularities according to the dataset (*e.g.,* type of data, number of files, definition of `NA` values, column names legend, labels legend).

C.2 'Control functions'

The ARC-functions do not take directly the parameters, but they require a *'control-object'* as an input. Control-objects are lists of parameters created by *control-functions*. These functions receive as input the parameters that will be used in the ARC-functions and output an organized list of the input parameters. This methodology (inspired in the `caret` package for the programming language `R`, developed by **Kuhn and Johnson (2013)**) serves to check errors in the parameters that are likely to cause errors in the ARC.

For example, the `preprocessing` function has its corresponding `control_preprocessing()` function that checks:

- The predictors' and ground truth labels' column names of the dataset.
- The threshold of maximum `NA` values allowed in a time series before imputation (optional).
- The method of `NA` imputation (optional).
- The parameters to perform the calculation of the magnitude vector (optional).

C.3 Packages used in the ARC

The following list mentions the packages that were used in the implementation of the ARC, and a brief description of them:

- `readr`: a package focused on methods for importing data in `R`.
- `tibble`: a modern approach of `R` data frames.
- `dplyr`: a collection of tools to manipulate tibbles (*e.g.,* subset data, modify values from data, merge and split tibbles).
- `imputeTS`: a collection of univariate time series imputation methods.
- `ggplot2`: a package dedicated to visualization methods of data.
- `ggrepel`: an extension of layers that can be applied to `ggplot2` graphics.
- `purrr`: a package that allows to apply functions to lists and tibbles.
- `stringr`: a collection of string manipulation tools.
- `tictoc`: a helper package to track the runtime of `R` functions.
- `caret`: a library of machine learning methods and analysis tools.
- `randomForest`: a package focused on random forest algorithms.

Bibliography

Andrienko, Natalia Andrienko and Gennady (2006). *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Springer, pp. 1–712. ISBN: 9783540259947. DOI: 10.1007/3-540-31190-4. URL: papers2://publication/uuid/94E92610-4D66-4CE5-8FD1-6C6E61DA889D.

Anguita, Davide, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz (2013). "A Public Domain Dataset for Human Activity Recognition Using Smartphones". In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* April, pp. 24–26. URL: http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Bannach, David and Paul Lukowicz (2008). "Rapid prototyping of activity recognition applications". In: *IEEE Pervasive Computing*. DOI: 10.1109/MPRV.2008.36.

Baños, Oresti, Miguel Damas, Héctor Pomares, Ignacio Rojas, Máté Attila Tóth, and Oliver Amft (2012). "A benchmark dataset to evaluate sensor displacement in activity recognition". In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, p. 1026. ISSN: 1617-4909. DOI: 10.1145/2370216.2370437. URL: http://dl.acm.org/citation.cfm?doid=2370216.2370437.

Banos, Oresti, Juan-Manuel Galvez, Miguel Damas, Hector Pomares, and Ignacio Rojas (2014). "Window Size Impact in Human Activity Recognition". In: *Sensors* 14.4, pp. 6474–6499. ISSN: 1424-8220. DOI: 10.3390/s140406474. URL: http://www.mdpi.com/1424-8220/14/4/6474/.

Bao, Ling and Stephen S. Intille (2004). "Activity Recognition from User-Annotated Acceleration Data". In: pp. 1–17. ISSN: 03029743. DOI: 10.1007/978-3-540-24646-6_1. arXiv: 9780201398298. URL: http://link.springer.com/10.1007/978-3-540-24646-6{_}1.

Box, G E P, G M Jenkins, and G C Reinsel (1994). *Time Series Analysis: Forecasting & Control*, p. 709. ISBN: 8131716333. DOI: 10.1016/j.ijforecast.2004.02.001. arXiv: arXiv:1011.1669v3.

Bulling, Andreas, U Blanke, and Bernt Schiele (2014). "A tutorial on human activity recognition using body-worn inertial sensors". In: *ACM Computing Surveys (CSUR)* 46.June, pp. 1–33. ISSN: 03600300. DOI: http://dx.doi.org/10.1145/2499621. URL: http://dl.acm.org/citation.cfm?id=2499621.

Burns, Adrian, Barry R Greene, Michael J Mcgrath, Benjamin Kuris, Steven M Ayer, Florin Stroiescu, and Victor Cionca (2010). "SHIMMER - A Wireless Sensor Platform for Noninvasive Biomedical Research". In: *IEEE SENSORS JOURNAL* 10.9, pp. 1527–1534.

Calatroni, Alberto, Daniel Roggen, and Gerhard Tröster (2011). "Collection and curation of a large reference dataset for activity recognition". In: *IEEE International Conference on Systems, Man and Cybernetics*, pp. 30–35. ISBN: 9781457706523. DOI: 10.1109/ICSMC.2011.6083638.

Chavarriaga, Ricardo, Hesam Sagha, Alberto Calatroni, Sundara Tejaswi Digumarti, Gerhard Tröster, José Del R Millán, and Daniel Roggen (2013). "The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition". In: *Pattern Recognition Letters* 34.15, pp. 2033–2042. ISSN: 01678655. DOI: 10.1016/j.patrec.2012.12.014. URL: http://dx.doi.org/10.1016/j.patrec.2012.12.014.

Fawcett, Tom (2006). "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27.8, pp. 861–874. ISSN: 01678655. DOI: 10.1016/j.patrec.2005.10.010. arXiv: /dx.doi.org/10.1016/j.patrec.200 [http:].

Fu, Tak Chung (2011). "A review on time series data mining". In: *Engineering Applications of Artificial Intelligence* 24.1, pp. 164–181. ISSN: 09521976. DOI: 10.1016/j.engappai.2010.09.007. URL: http://dx.doi.org/10.1016/j.engappai.2010.09.007.

Goldstone, Jack A. (2010). *The New Population Bomb*. URL: https://www.foreignaffairs.com/articles/2010-01-01/new-population-bomb.

Gonz, Sergio, Min Chen, Senior Member, and Victor C M Leung (2017). "A Survey on Activity Detection and Classification Using Wearable Sensors". In: *IEEE SENSORS JOURNAL* 17.2, pp. 386–403. ISSN: 1530-437X. DOI: 10.1109/JSEN.2016.2628346.

Guo, Junqi, Xi Zhou, Yunchuan Sun, Gong Ping, Guoxing Zhao, and Zhuorong Li (2016). "Smartphone-Based Patients' Activity Recognition by Using a Self-Learning Scheme for Medical Monitoring". In: *Journal of Medical Systems* 40.6. ISSN: 1573689X. DOI: 10.1007/s10916-016-0497-2. URL: http://dx.doi.org/10.1007/s10916-016-0497-2.

Guyon, Isabelle and André Elisseeff (2006). *Feature Extraction: Foundations and Applications*. ISBN: 978-3-540-35487-1. DOI: 10.1007/978-3-540-35488-8_1. URL: http://link.springer.com/10.1007/978-3-540-35488-8{_}1.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2001). "The Elements of Statistical Learning". In: *The Mathematical Intelligencer* 27.2, pp. 83–85. ISSN: 03436993. DOI: 10.1198/jasa.2004.s339. arXiv: 1010.3003. URL: http://www.springerlink.com/index/D7X7KX6772HQ2135.pdf{\%}255Cnhttp://www-stat.stanford.edu/{~}tibs/book/preface.ps.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani (2013). *An Introduction to Statistical Learning*, p. 618. ISBN: 9780387781884. DOI: 10.1016/j.peva.2007.06.006. arXiv: arXiv:1011.1669v3. URL: http://books.google.com/books?id=9tv0taI8l6YC.

John, George H., Ron Kohavi, and Karl Pfleger (1994). "Irrelevant Features and the Subset Selection Problem". In: *Machine Learning Proceedings 1994*, pp. 121–129. ISSN: 00189340. DOI: 10.1016/B978-1-55860-335-6.50023-4.

Karantonis, Dean M, Michael R Narayanan, Merryn Mathie, Nigel H Lovell, and Branko G Celler (2006). "Implementation of a Real-Time Human Movement Classifier Using a Triaxial Accelerometer for Ambulatory Monitoring". In:

IEEE Transactions on information technology on biomedicine 10.1, pp. 156–167.

Kavitha, R. and N. J. Sijo Antony (2017). "Human Activity Recognition from Sensor data using Random Forest Algorithm". In: *International Journal of Advanced Research in Computer Science* 8.3, pp. 2015–2018. URL: http://www.ijarcs.info/index.php/Ijarcs/article/view/3009.

Kim, A. and M.F. Golnaraghi (2004). "A quaternion-based orientation estimation algorithm using an inertial measurement unit". In: *Position Location and Navigation ...* Pp. 268–272. DOI: 10.1109/PLANS.2004.1309003. URL: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=1309003.

Konietzny, Sebastian, Agustín Vargas Toro, Pearl Pu, Igor Kulev, Mirana Randriambelonoro, Caroline Perrin, Thomas Linner, Amir Kabouteh, and Henning Andersen (2018). *REACH - Deliverable D3.1: Data collection requirements, ethnographic studies, and a validated, significant set of patient data*. Tech. rep. URL: http://reach2020.eu/?page{_}id=1190.

Kotsiantis, S B, D Kanellopoulos, and P E Pintelas (2006). "Data preprocessing for supervised learning". In: *International Journal of Computer Science* 1.2, pp. 111–117. ISSN: 1306-4428. DOI: 10.1080/02331931003692557. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.132.5127{\&}rep=rep1{\&}type=pdf.

Kuhn, Max and Kjell Johnson (2013). *Applied Predictive Modeling*. Springer, p. 620. ISBN: 1461468485. DOI: 10.1007/978-1-4614-6849-3. URL: http://www.amazon.com/Applied-Predictive-Modeling-Max-Kuhn/dp/1461468485/ref=pd{_}bxgy{_}b{_}img{_}z.

Lara, Oscar D. and Miguel A. Labrador (2013). "A Survey on Human Activity Recognition using Wearable Sensors". In: *IEEE Communications Surveys & Tutorials* 15.3, pp. 1192–1209. ISSN: 1553-877X. DOI: 10.1109/SURV.2012.110112.00192. URL: http://ieeexplore.ieee.org/document/6365160/.

Lara, Óscar D., Alfredo J. Pérez, Miguel A. Labrador, and José D. Posada (2012). "Centinela: A human activity recognition system based on acceleration and vital sign data". In: *Pervasive and Mobile Computing* 8.5, pp. 717–729. ISSN: 15741192. DOI: 10.1016/j.pmcj.2011.06.004. URL: http://dx.doi.org/10.1016/j.pmcj.2011.06.004.

Lenz, J. and S. Edelstein (2006). "Magnetic sensors and their applications". In: *IEEE Sensors Journal* 6.3, pp. 631–649. ISSN: 1530-437X. DOI: 10.1109/JSEN.2006.874493. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1634415.

Leutheuser, Heike, Dominik Schuldhaus, and Bjoern M. Eskofier (2013). "Hierarchical, Multi-Sensor Based Classification of Daily Life Activities: Comparison with State-of-the-Art Algorithms Using a Benchmark Dataset". In: *PLoS ONE* 8.10. ISSN: 19326203. DOI: 10.1371/journal.pone.0075196.

Liu, Kai-chun, Chien-yi Yen, Li-han Chang, Chia-yeh Hsieh, and Chia-tai Chan (2017). "Wearable Sensor-Based Activity Recognition for Housekeeping Task". In: *IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. IEEE, pp. 67–70. ISBN: 9781509062447. URL: http://ieeexplore.ieee.org/document/7936009/.

Liu, Ye, Liqiang Nie, Li Liu, and David S. Rosenblum (2016). "From action to activity: Sensor-based activity recognition". In: *Neurocomputing* 181, pp. 108–115. ISSN: 18728286. DOI: 10.1016/j.neucom.2015.08.096. URL: http://dx.doi.org/10.1016/j.neucom.2015.08.096.

Louppe, Gilles, Louis Wehenkel, Antonio Sutera, and Pierre Geurts (2013). "Understanding variable importances in forests of randomized trees". In: *Advances in Neural Information Processing Systems 26*, pp. 431–439. ISSN: 1098-6596. DOI: NIPS2013_4928. arXiv: arXiv:1011.1669v3. URL: http://media.nips.cc/nipsbooks/nipspapers/paper{_}files/nips26/281.pdf.

Moritz, S. and T. Bartz-Beielstein (2017). "imputeTS: Time series missing value imputation in R". In: *R Journal* 9.1, pp. 1–12. ISSN: 20734859.

Ni, Qin, Ana Belén García Hernando, and Iván Pau de la Cruz (2015). *The elderly's independent living in smart homes: A characterization of activities and sensing infrastructure survey to facilitate services development*. Vol. 15. 5, pp. 11312–11362. ISBN: 3463414929. DOI: 10.3390/s150511312.

Palumbo, Filippo, Claudio Gallicchio, Rita Pucci, and Alessio Micheli (2016). "Human activity recognition using multisensor data fusion based on Reservoir Computing". In: *Journal of Ambient Intelligence and Smart Environments* 8.March, pp. 87–107. DOI: 10.3233/AIS-160372.

Plötz, Thomas, Nils Y Hammerla, and Patrick Olivier (2011). "Feature Learning for Activity Recognition in Ubiquitous Computing". In: *Proceeding IJCAI'11 Proceedings of the Twenty-Second international joint conference on Artificial Intelligence* Volume 2, pp. 1729–1734. ISSN: 10450823. DOI: 10.5591/978-1-57735-516-8/IJCAI11-290. URL: http://dl.acm.org/citation.cfm?id=2283516.2283683.

Rahman, Hala Abdul, Di Ge, Alexis Le Faucheur, Jacques Prioux, and Guy Carrault (2017). "Advanced classification of ambulatory activities using spectral density distances and heart rate". In: *Biomedical Signal Processing and Control* 34, pp. 9–15. DOI: 10.1016/j.bspc.2016.12.018.

Ranasinghe, Suneth, Fadi Al MacHot, and Heinrich C. Mayr (2016). "A review on applications of activity recognition systems with regard to performance and evaluation". In: *International Journal of Distributed Sensor Networks* 12.8. ISSN: 15501477. DOI: 10.1177/1550147716665520.

Rashidi, Parisa and Alex Mihailidis (2013). "A survey on ambient-assisted living tools for older adults". In: *IEEE Journal of Biomedical and Health Informatics* 17.3, pp. 579–590. ISSN: 21682194. DOI: 10.1109/JBHI.2012.2234129.

Reiss, Attila and Didier Stricker (2012a). "Creating and benchmarking a new dataset for physical activity monitoring". In: *Proceedings of the 5th International Conference on Pervasive Technologies Related to Assistive Environments* February, p. 1. ISSN: 15504816. DOI: 10.1145/2413097.2413148. URL: http://dl.acm.org/citation.cfm?doid=2413097.2413148.

– (2012b). "Introducing a new benchmarked dataset for activity monitoring". In: *Proceedings - International Symposium on Wearable Computers, ISWC*, pp. 108–109. ISSN: 15504816. DOI: 10.1109/ISWC.2012.13.

Roggen, Daniel and Gerhard Tröster (2010). "OPPORTUNITY - Deliverable D5.1". In: 12.

Roggen, Daniel, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Forster, Gerhard Troster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, Jakob Doppler, Clemens Holzmann, Marc Kurz, Gerald Holl, Ricardo Chavarriaga, Hesam Sagha, Hamidreza Bayati, Marco Creatura, and Jose Del R. Millàn (2010). "Collecting complex activity datasets in highly rich networked sensor environments". In: *INSS 2010 - 7th International Conference on Networked Sensing Systems* 00, pp. 233–240. ISSN: 03029743. DOI: 10.1109/INSS.2010.5573462. arXiv: 1206.5533.

Roggen, Daniel, Gerhard Tröster, Mangenat Stéphane, and Markus Waibel (2011). "Designing and sharing activity recognition systems across platforms: methods from wearable computing". In: *Stephane.Magnenat.Net*. DOI: 10.3929/ethz-a-010001702.

Sano, Akane and Rosalind W. Picard (2013). "Stress Recognition Using Wearable Sensors and Mobile Phones". In: *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pp. 671–676. ISSN: 2156-8103. DOI: 10.1109/ACII.2013.117. URL: http://ieeexplore.ieee.org/document/6681508/.

Shoaib, Muhammad, Hans Scholten, and P.J.M. Havinga (2013). "Towards Physical Activity Recognition Using Smartphone Sensors". In: *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, pp. 80–87. ISSN: 1007-0214. DOI: 10.1109/UIC-ATC.2013.43. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6726194.

Shokoohi-Yekta, Mohammad, Yanping Chen, Bilson Campana, Bing Hu, Jesin Zakaria, and Eamonn Keogh (2015). "Discovery of Meaningful Rules in Time Series". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1085–1094. ISSN: <null>. DOI: 10.1145/2783258.2783306. URL: http://dl.acm.org/citation.cfm?doid=2783258.2783306.

Stisen, Allan, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen (2015). "Smart Devices are Different: Assessing and MitigatingMobile Sensing Heterogeneities for Activity Recognition". In: *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pp. 127–140. DOI: 10.1145/2809695.2809718. URL: http://dl.acm.org/citation.cfm?doid=2809695.2809718.

Wang, Jindong, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu (2018). "Deep learning for sensor-based activity recognition: A Survey". In: *Pattern Recognition Letters* 0, pp. 1–9. ISSN: 01678655. DOI: 10.1016/j.patrec.2018.02.010. arXiv: 1707.03502. URL: https://doi.org/10.1016/j.patrec.2018.02.010.

Ward, Jamie A, Paul Lukowicz, and Hans-Werner Gellersen (2011). "Performance metrics for activity recognition". In: *Acm Tist* 2.1, pp. 111–132.

Wen, Jiahui and Zhiying Wang (2017). "Learning general model for activity recognition with limited labelled data". In: *Expert Systems with Applications*

74, pp. 19–28. ISSN: 09574174. DOI: 10.1016/j.eswa.2017.01.002. URL: http://dx.doi.org/10.1016/j.eswa.2017.01.002.

Yang, Allen Y, Philip Kuryloski, and Ruzena Bajcsy (2009). "WARD : A Wearable Action Recognition Database". In: *CHI*.

Yang, Hua Cong, Yi Chao Li, Zhi Yu Liu, and Jie Qiu (2014). "HARLIB: A human activity recognition library on Android". In: *2014 11th International Computer Conference on Wavelet Active Media Technology and Information Processing, ICCWAMTIP 2014*, pp. 313–315. DOI: 10.1109/ICCWAMTIP.2014.7073416.

Zhao, Zhongtang, Zhenyu Chen, Yiqiang Chen, Shuangquan Wang, and Hongan Wang (2014). "A Class Incremental Extreme Learning Machine for Activity Recognition". In: *Cognitive Computation* 6.3, pp. 423–431. ISSN: 18669964. DOI: 10.1007/s12559-014-9259-y.

Zhu, Jiadong, Ruben San-Segundo, and Jose M Pardo (2017). "Feature extraction for robust physical activity recognition". In: *Human-centric Computing and Information Sciences* 7, pp. 1–16. ISSN: 2192-1962. DOI: 10.1186/s13673-017-0097-2.