Patrícia Justo Bota

Bachelor Degree in Biomedical Engineering Sciences

Human Activity Annotation based on Active Learning

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Biomedical Engineering

Adviser: Prof. Dr. Hugo Filipe Silveira Gamboa, Professor Auxiliar, Faculdade de Ciências e Tecnologias, Universidade Nova de Lisboa

Examination Committee

Chairperson: Prof. Dr. Carla Maria Quintão Pereira
Raporteur: Prof. Dr. Alexandre da Rocha Freire de Andrade
Member: Prof. Dr. Hugo Filipe Silveira Gamboa

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

December, 2018

Human Activity Annotation based on Active Learning

This document was created using the (pdf)LaTeX processor, based in the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Acknowledgements

First and foremost, I would like to express my utmost gratitude to my academic supervisor Prof. Hugo Gamboa, for giving me the opportunity to enter the world of machine learning and data science. Thank you for all the insights and excellent advice during the development of this dissertation.

I would like to thank *Associação Fraunhofer Portugal Research* for the opportunity to work on my thesis in such a good environment, where I was encouraged to always, think, learn and dare. Especially, I would like to thank Duarte Folgado and Joana Silva, for the guidance, support and promptly help, that allowed me to learn everyday and successfully complete this research with great enthusiasm.

Thank you to all my university friends and my colleagues at Fraunhofer, for being a great influence in my life, and for all the laughs and silly moments that made all the study and stress times easier.

Last but not least, I would like to thank my family, for believing in me and giving me the opportunity to continue to grow, both personally and professionally.

Abstract

Human activity recognition algorithms have been increasingly sought due to their broad application, in areas such as healthcare, safety and sports. Current works focusing on human activity recognition are based majorly on Supervised Learning algorithms and have achieved promising results. However, high performance is achieved at the cost of a large amount of labelled data required to train and learn the model parameters, where a high volume of data will increase the algorithm's performance and the classifier's ability to generalise correctly into new, and previously unseen data. Commonly, the labelling process of ground truth data, which is required for supervised algorithms, must be done manually by the user, being tedious, time-consuming and difficult.

On this account, we propose a Semi-Supervised Active Learning technique able to partly automate the labelling process and reduce considerably the labelling cost and the labelled data volume required to obtain a highly performing classifier. This is achieved through the selection of the most relevant samples for annotation and propagation of their label to similar samples. In order to accomplish this task, several sample selection strategies were tested in order to find the most valuable sample for labelling to be included in the classifier's training set and create a representative set of the entire dataset. Followed by a semi-supervised stage, labelling with high confidence unlabelled samples, and augmenting the training set without any extra labelling effort from the user. Lastly, five stopping criteria were tested, optimising the trade-off between the classifier's performance and the percentage of labelled data in its training set.

Experimental results were performed on two different datasets with real data, allowing to validate the proposed method and compare it to literature methods, which were replicated. The developed model was able to reach similar accuracy values as supervised learning, with a reduction in the required labelled data of more than 89% for the two datasets, respectively.

Keywords: Human Activity Recognition, Machine Learning, Feature Selection, Active Learning, Semi-Supervised Learning, Time Series.

Resumo

Os algoritmos de reconhecimento de atividade humana têm sido cada vez mais procurados devido à sua grande aplicabilidade, em áreas como a saúde, segurança e desporto. As soluções existentes são baseadas, principalmente, em algoritmos de Aprendizagem Supervisionada atingindo resultados promissores. Porém, requerendo uma grande quantidade de dados anotados, necessários para treinar e aprender os parâmetros do modelo. Esta anotação, dado que deve ser realizada manualmente pelo utilizador, é um processo fastidioso, demorado e difícil.

Assim, esta tese propõe uma técnica envolvendo métodos de Aprendizagem Ativa Semi-Supervisionada, capaz de automatizar parcialmente o processo de anotação através da seleção da amostra considerada como mais relevante para a classificação e propagando a sua etiqueta a amostras semelhantes. Deste modo, reduzindo significativamente o esforço manual por parte do utilizador na anotação. Para isto, foram testadas várias estratégias para a seleção da amostra para anotar e incorporar no conjunto de treino do classificador. Adicionalmente, foi introduzido um estado Semi-Supervisionado, permitindo uma anotação automática de amostras não anotadas e, assim, aumentar o conjunto de treino do classificador sem qualquer esforço por parte do utilizador. Por último, cinco critérios de paragem para o algoritmo realizado foram testados, tendo como objetivo otimizar o balanço entre o desempenho do classificador e a percentagem de dados anotados existentes no seu conjunto de treino.

Testes experimentais foram realizados em dois conjuntos de dados reais, permitindo validar o método proposto e compará-lo a métodos já existentes, que foram replicados. Nos resultados obtidos foi obtida uma precisão semelhante a alcançada pela técnica supervisionada, com uma redução na quantidade de dados anotados de 89%, para ambos os conjuntos de dados.

Palavras-chave: Reconhecimento de Atividade Humana, Aprendizagem Automática, Seleção de Características, Aprendizagem Ativa, Aprendizagem Semi-Supervisionada, Séries Temporais.

Contents

List of Figures xiii

List of Tables xv

Acronyms xix

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Objectives . . . 4
- 1.3 State of the Art . . . 5
- 1.4 Thesis Overview . . . 8

2 Theoretical Background 9
- 2.1 Wearable and Smartphone Sensors . . . 9
  - 2.1.1 Accelerometer . . . 9
  - 2.1.2 Gyroscope . . . 11
  - 2.1.3 Barometer . . . 12
- 2.2 Machine Learning . . . 13
  - 2.2.1 Signal Processing . . . 13
  - 2.2.2 Feature Engineering . . . 14
  - 2.2.3 Classification . . . 15
  - 2.2.4 Validation . . . 25

3 Framework for Human Activity annotation based on Active Learning 27
- 3.1 Signal Acquisition and Processing . . . 27
- 3.2 Feature Engineering . . . 28
- 3.3 General Active Learning Strategy . . . 30
  - 3.3.1 Initial Train Set . . . 31
  - 3.3.2 Sample Selection Strategy . . . 31
  - 3.3.3 Stopping Criterion . . . 36
- 3.4 Semi-Supervised Active Learning Framework . . . 39
  - 3.4.1 Similarity Measures . . . 41
- 3.5 Evaluation . . . 45

4 Results **47**
4.1 Datasets 47
4.2 Feature Extraction and Selection 48
4.3 Model Selection 49
4.4 Query Strategy Analysis 49
4.5 Active Learning Semi-Supervised Analysis 55
4.6 Stopping Criterion Analysis 59

5 Conclusion **65**
5.1 Overall Achievements 65
5.2 Future Work 68

Bibliography **69**

A Acquisition Devices **75**

B Algorithms **77**

C Horizon Plot **81**

List of Figures

1.1 Demonstration of the working principles of AL and SSAL. 4
1.2 Thesis overview structure. 8

2.1 Smartphone's sensors standard coordinate system and accelerometer sensor output signal with the device positioned into different orientations. 10
2.2 Smartphone's sensors standard coordinate system and gyroscope's output signal with the device positioned into different orientations. 11
2.3 Barometer's output signal from a user climbing up and downstairs. 12
2.4 Accelerometer's signal magnitude of the total and body accelerations. 13
2.5 Pipeline description of *Supervised Learning*. 15
2.6 Pool-based *Active Learning* cycle. 21
2.7 *Active Learning* training set samples two-dimensional principal component analysis. 22
2.8 *Active Learning* UCI test set samples two-dimensional principal component analysis. 22
2.9 *Self-Training* training set samples two-dimensional principal component analysis. 24
2.10 *Self-Training* UCI test set samples two-dimensional principal component analysis. 24

3.1 Schematic representation of the framework developed in this thesis. 27
3.2 Accelerometer's signal before and after the application of a band-pass filter. 29
3.3 Computation of the uncertainty score according to *Least Confident Sampling*. 32
3.4 Computation of the uncertainty score according to *Margin Sampling*. 33
3.5 Computation of the uncertainty score according to *Entropy Sampling*. 34
3.6 UCI samples' information density on a two-dimensional principal component analysis. 35
3.7 *Active Learning* performance curve throughout 300 iterations. 37
3.8 Classifier Least Confidence score and Classifier Overall Uncertainty score throughout 300 iterations. 37
3.9 Example of 1-*Nearest Neighbour* label propagation step. 40
3.10 Example of 1-*reverse-Nearest Neighbour* label propagation step. 41

3.11 Two-dimensional illustration of the *Euclidean distance* and the *Cosine similarity* between two samples. 42
3.12 Comparison between the *Euclidean Distance* and *Dynamic Time Warping Distance* between two signals. 44

4.1 Classifier's accuracy score during and before *Forward Feature Selection*. 49
4.2 *Active Learning* classifier's accuracy for the different developed query strategies. 52
4.3 Principal component analysis of the *Active Learning* training set and *Local Density * Margin Sampling* uncertainty score heatmap. 52
4.4 Principal component analysis of the *Active Learning* training set using the *Least Confident Sampling* and the *Local Density * Least Confident Sampling*. 53
4.5 *Local Density * Least Confident Sampling* and *Margin Sampling* uncertainty score heatmaps. 54
4.6 Classifier's accuracy for the developed *Semi-Supervised Active Learning* methods throughout the cycle iterations. 56
4.7 Percentage of the validation set unlabelled samples for the developed *Semi-Supervised Active Learning* methods throughout the *Active Learning* iterations. 57
4.8 Percentage of correctly automatically annotated samples for the developed *Semi-Supervised Active Learning* methods throughout the *Active Learning* iterations. 57
4.9 Horizon Plot for the misclassified activities 60
4.10 Confusion matrix for the *ST-SSAL* method using the *Overall Uncertainty SC*. . 61

A.1 CADL dataset acquisition devices. 75
A.2 Percentage of samples belonging to each performed activity for the UCI and CADL datasets. 76

C.1 Horizon Plot for the UCI dataset, allowing to visualise the behaviour of the best features set along the protocol activities. 82
C.2 Horizon Plot for the CADL dataset, allowing to visualise the behaviour of the best features set along the protocol activities. 83

List of Tables

1.1 Sources of labelled data with a brief description, main advantages and disadvantages. . . . 3

2.1 Statistical, Temporal and Spectral Domain Features used in the present work. 16
2.2 Common distance metrics and their math formula. . . . 17
2.3 Illustration of a Confusion Matrix. . . . 26
2.4 Evaluation metrics for a binary classification. . . . 26

4.1 Summary information on UCI and CADL dataset. . . . 48
4.2 SL and UL methods classification's performance shown in accuracy and ARI score, respectively. . . . 50
4.3 *Active Learning* experimental results for the developed query strategies. . . . 50
4.4 *Active Learning* experimental results for the developed query strategies. . . . 51
4.5 Experimental results for the SSAL methods. . . . 62
4.6 Experimental results for the developed stopping criteria methods. . . . 63

List of Algorithms

1 Filtering 28
2 Feature Extraction 29
3 Forward Feature Selection 30
4 General Active Learning 31
5 General Semi-Supervised Active Learning 39
6 K-fold Cross Validation 45
7 Active Learning Applying The Max-Confidence SC 77
8 Active Learning Applying The Overall Uncertainty SC 78
9 Active Learning Applying The Classification-Change SC 78
10 Active Learning Applying The Combination Strategy SC 78
11 Self-Training 79
12 5-Nearest Neighbour Semi-Supervised Active Learning 79
13 1-reverse-Nearest Neighbour Semi-Supervised Active Learning 80

Acronyms

AICOS	Assistive Information and Communication Solutions.
AL	Active Learning.
Ann Acc	Automated Annotation Accuracy.
ARI	Adjusted Rand Index.
Aut Ann	Automated Annotation Percentage.
CADL	Continuous Activities of Daily Living.
CC	Classification-Change.
CV	Cross Validation.
DBSCAN	Density-Based Spatial Clustering of Applications with Noise.
DTW	Dynamic Time Warping.
FN	False Negative.
FP	False Positive.
HAR	Human Activity Recognition.
Max-CC	Max-Confidence Uncertainty and Classification-Change.
Max-Conf	Max-Confidence.
MEMS	Micro-Electro-Mechanical-Systems.
NN	Nearest Neighbour.
Over-CC	Overall Uncertainty and Classification-Change.
Over-Unc	Overall Uncertainty.
PCA	Principal Component Analysis.
PL	Passive Learning.

QDA	Quadratic Discriminant Analysis.
QS	Query Strategy.
QSs	Query Strategies.
rNN	Reverse-Nearest Neighbour.
SC	Stopping Criterion.
SCs	Stopping Criteria.
SL	Supervised Learning.
SP	Stopping Point.
SSAL	Semi-Supervised Active Learning.
SSL	Semi-Supervised Learning.
ST	Self-Training.
SVM	Support Vector Machine.
TAM	Time Alignment Metric.
TN	True Negative.
TP	True Positive.
UCI	University of California, Irvine.
UL	Unsupervised Learning.

Chapter 1

Introduction

1.1 Motivation

Over the last years, the technological advances on ubiquitous sensing mechanisms allow the proliferation of available data, which often is unlabelled. Modern machine learning approaches require large amounts of labelled data to achieve adequate performance. This duality raises a relevant question: How can we simultaneously optimise the process of data annotation and still learn an accurate machine learning model?

Particularly, the HAR field has been a source of large quantity of available data, mostly due to its myriad of applications on real-life scenarios such as healthcare, indoor location, user-adaptive recommendations and transportation [1, 2]. According to the World Health Organisation [3], insufficient physical activity has been identified as the fourth leading risk factor for global mortality. Indeed, physical inactivity is one of the main causes of several health diseases, being correlated with overweight and obesity. On the other hand, the practice of physical exercise is correlated to an increase of cardio-respiratory and muscular fitness, functional health, cognitive functions and improvement of bones and joint health. As a result, human physical activities recognition has been increasingly sought in a personal fitness tracking context, giving the necessary motivation to physical activities practice.

Alongside with **healthcare**, the monitoring of the human movement can be paramount as a preventive and diagnosis tool, contributing to its user **safety** by the identification of psychiatric disorders signs, and warnings of unusual activity, such as falls, movement degeneration or cardiac abnormalities.

Likewise, in a **physiotherapy** or **sports** setting, monitoring the body movements can make it possible to gather long-term data, oversee detailed movements and obtain a statistical analysis of the body movements' range evolution. Thus, becoming a source of

data information and motivation for improvement.

Besides these concerns, Human Activity Recognition (HAR) can be a powerful tool in **smart homes**, allowing to interpret the current user state so the house system can better reply to the user's needs. As well for **indoor location** systems, where the GPS is often not available. Moreover, since many activities are location dependent, the recognition of human movement activities can be helpful to infer the user's position.

For this reason, human physical activity monitoring has been increasingly sought by physicians, athletes, physiotherapists, researchers and even healthy individuals, wanting to maintain and improve their healthy, active lifestyle [4].

Under those circumstances, HAR has been the subject for numerous studies, with the major part based on Supervised Learning (SL) machine learning techniques [4–6]. These have achieved great results, however, at the cost of involving the collection of large amounts of labelled data. Where a high volume of data will increase the algorithm's performance and the classifier's ability to generalise correctly into new, unseen data.

Moreover, data does not get labelled automatically but through a labelling process, also referred to as **annotation**, where each unlabelled sample is mapped into a label. Thus, becoming what is denoted as **labelled data**. Furthermore, most of the times, the annotation must be performed manually by the user, being time-consuming, difficult, or even impossible to obtain, as it there can be no means to know the samples' labels. Therefore, this fact highly limits a real-life application of today's approaches and their scalability. In fact, generally, behind every classifier training set, there are timeless, arduous hours of annotation usually performed manually by a human worker. Per example, ImageNet, an image database, had for 9 years, contributors manually annotating more than 14 million images [7]. A process that will be the core of any classifier using the database, where any mistake, per example, annotating a cat as a dog, will affect greatly the classifier's quality and its results. Therefore, most of the time, the developer's time is not spent on building the classifier, but on acquiring data and annotating it, in order to be able to create a highly confident classifier. Indeed, annotating a dataset is not only time costly and difficult, but also extremely expensive for a company. For instance, annotating the Cityscapes dataset [1], containing stereo video sequences with a total of 5000 high-quality pixel level annotated frames and, considering that annotating a single image can take around at least 1.5h. Annotating the entire dataset would require a total of 5000 * 1.5 = 7500h. Thus, taking into account that 1 hour approximately costs 4€ (approximately the minimum wage by the hour in Portugal [8]) that would make a total cost of 30 000€.

Table 1.1 enumerates some of the most common sources for training data. As it can be observed, there is not one to stand out as ideal [7]. Therefore, under those circumstances, the development of a new method able to partly automate the annotation process, and reduce considerably its expensive cost is challenging and particularly interesting.

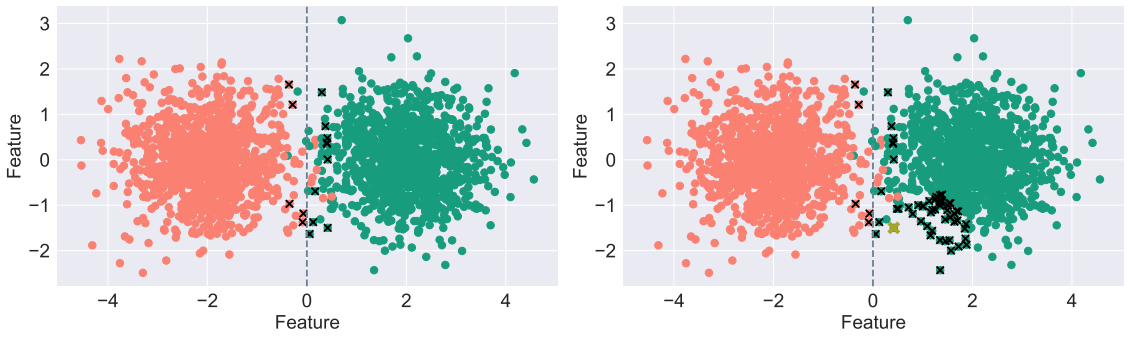
[1] https://www.cityscapes-dataset.com

Table 1.1: Sources of labelled data with a brief description, main advantages and disadvantages.

Approach	Description	Advantages	Disadvantages
Internal Labelling	Hand on labelling by the in-house team	High quality, control over the process	Time consuming, expensive, highly depends on the team diversity and scalability
Outsourcing	Recruitment of temporary employees or an external service provider	Speed, scalability diversity	Expensive
Crowdsourcing	Annotation by freelancers	Speed, diversity, costs	Inconsistent quality, need for annotation interface
Synthetic Labelling	Fake labelled data is synthetically generated	No privacy constrains, time and cost effective	High computational power, inconsistent quality
Data Programming	Automatic annotation via a programming script	Automatic, fast	Low quality, low diversity

In datasets with significant size, not all samples are equally informative to the classification process and an arbitrary unlabelled example may even be redundant. Active Learning (AL) provides methods to automatically identify the most relevant samples, which are posteriorly queued for expert annotation, that we denote as Oracle, without compromising the model performance. Figure 1.1a, illustrates the behaviour of AL where samples near the decision boundary are denoted as more informative, therefore, are selected for annotation. Additionally, Figure 1.1b displays the Semi-Supervised Active Learning automatic annotation behaviour where after the most informative sample selected by AL (in yellow) is annotated by the oracle, its nearest samples are automatically annotated (as shown by the $\times$ in black).

In this dissertation, we apply a SSAL algorithm for HAR, where we establish criteria to select the most relevant samples for annotation and propagate their label to similar samples. Moreover, the model is created with a minimal initial training set and a comprehensive study regarding state of art AL Query Strategies (QSs), Stopping Criteria (SCs) and distance functions was performed. The SSAL method based on Self-Training (ST) applied to HAR data, starting with near zero annotated data, achieved high results on algorithm's performance while reducing considerably the annotation effort and automatically annotation a substantial amount of the data. Thus, within an exploratory Machine Learning context, the automatic labelling of a large amount of unlabelled data, eliminating most of HAR existent SL methods' disadvantages, it is the main goal of this dissertation.



(a) Active Learning.

(b) Semi-Supervised Active Learning.

Figure 1.1: A dataset of 3000 samples illustrating the working principles of AL and Semi-Supervised Active Learning (SSAL). The samples are illustrated with colours identifying their respective class. The samples selected by the AL for expert annotation are depicted by the ×'s. The grey vertical line denotes the decision boundary between the two classes.

1.2 Objectives

This dissertation was developed in collaboration with Fraunhofer Assistive Information and Communication Solutions (AICOS) Portugal. This research centre focuses on creating 'Remarkable Technology, Easy to Use', developing tomorrow's technologies contributing to economic growth, social well-being and an improved quality of life of its end users [9].

Moreover, this thesis focuses on the development of a framework for HAR with minor annotation cost for the user, using the smartphone and wearable devices' sensors. Furthermore, this dissertation proposes to infer how much labelled data it is required to train and create a model able to reach a highly confident classification performance as the obtained by SL techniques.

In order to perform this task, two major approaches will be followed: the first based on **AL**, exploring several Query Strategy allowing to select highly informative samples to annotate and integrate into the classifier's training set. Followed by a second framework, combining both **AL** and a **Semi-Supervised Learning** step, aiming to, through the later, to automatically label additional unlabelled data with no annotation effort for the user. Thus, to develop the above-mentioned frameworks, the following steps, describing the core components of any HAR system, must be implemented:

1. **Signal Acquisition and Extraction:** acquisition and extraction of the smartphone and the wearable devices sensors signals. Analysis of the required number of devices, their position and orientation, and the number and type of sensors to use.

2. **Signal Processing:** signal segmentation and filtering. Analysis of the type of sliding window, dimension, type of filters, and cutoff frequencies.

3. **Feature Engineering:** feature extraction and selection on statistical, temporal and spectral domains. Features' normalisation.

4. **Active Learning and Semi-Supervised Active Learning Framework:** development of a framework to recognise human motion activities requiring minimal annotation cost.

5. **Frameworks' Optimisation:** study of the selective sampling QSs and the AL stopping criteria.

6. **Performance Evaluation:** evaluation of the developed algorithms.

1.3 State of the Art

In the literature, the discrimination of human activities is often covered by external or wearable sensors. The former include intelligent homes, where sensors are placed in critical devices and cameras. However, these raise numerous issues, such as privacy, pervasiveness and complexity concerns [2, 6]. Thus, motivating the use of wearable sensors, since their small size, low cost and non-obtrusiveness allow to integrate them easily into the users' daily living activities.

On this account, many wearable sensor-based Machine Learning classification techniques have surged, namely, SL techniques. As in the work of Silva [4], where the author used the smartphone accelerometer data, obtained using the acquisition device located on the user's waist, to train and classify a decision tree algorithm. Moreover, feature extraction and selection methods were applied to the signal, reducing significantly the algorithm time and computational complexity. In the final result, the classifier obtained a total accuracy of 86%. Thus, it was possible to conclude that, with a smartphone, it is possible to retrieve the users' activity data with good results and great comfort for the user.

Additionally, in order to overcome the implications of a pre-defined fixed device position, Figueira [5] developed a framework to perform HAR independently of the user and the device position in relation to the user's body. To perform this task a decision tree algorithm was implemented and obtained an accuracy of 94.6% using accelerometer and barometer data. The addition of the barometer data, measuring the ambient air pressure, shown to be especially important in the recognition of vertical movements, such as climbing and descending stairs.

However, these works used SL techniques, and therefore, require the entire classifier's training set to be previously annotated: a costly, difficult and time-consuming task as described in Section 1.1. On this account, in order to minimise the classifier's training set high annotation cost, we propose the use of an **AL** based methodology to recognise simple daily living activities. Furthermore, an AL System is composed by two main parts: the **Query System** that selects the relevant samples to be annotated and incorporated into the classifier's training set, and the **Oracle**, that labels the selected samples.

In the literature, several techniques for the query system have been proposed. However, regarding HAR, they majorly follow a *Query by Committee* or *Pool Sampling* strategy [10, 11].

Henceforth, in a **Query by Committee Sampling**, a committee of classifiers is created as a voting system for the label of each sample, so that in every iteration, the sample with the highest label disagreement between the different classifiers is selected as the most informative for the oracle to annotate.

On the other hand, in **Pool-based Sampling**, the unlabelled samples create a pool from which the learner selects the most informative sample for the oracle to annotate according to a pre-defined score [11–14].

With this in mind, Shahmohammadi et al. [10] applied both the *Query by Committee* and the *Pool-based Sampling* query systems in a smartwatch-based approach dedicated to HAR. The two approaches surpassed *Random Sampling* (where a sample is randomly selected from the unlabelled samples dataset), with *Pool-based Sampling* obtaining the best results. Moreover, generally speaking, the proposed method was able to achieve an accuracy of 92%, with a reduction of 46% in the amount of annotated data in comparison to SL. Therefore, proving the application of AL in the context of HAR and its ability to significantly reduce the required amount of labelled data.

Similarly, Rong Liu et al. [11] applied an AL novel method in the context of HAR using body-worn accelerometer sensor data. In the developed algorithm two QSs were implemented and in both the accuracy was higher than in *Random Sampling*. The first, entitled a *Pool-based Sampling* Query Strategy (QS) based on the classifier's prediction confidence. While the second followed a *Query by Committee Sampling* QS using two classifiers trained on the hip and wrist devices' data, respectively. The proposed method was evaluated with initial train sets of approximately 20% and 30% labelled data. As expected, with an initial higher quantity of labelled data, and therefore, a larger initial train set, both the AL and the SL technique achieve higher performance results. Overall, the best performance was obtained by the *Least Confident Sampling* QS, obtaining an accuracy of 75.96% for 50% of labelled data with an initial data set of 30% labelled data, using data from the hip and wrist sensors. For last, the presented method was able to outperform the SL technique (C4.5 Decision tree) when trained on the same amount of randomly labelled data. Thus, it was possible to conclude that the samples selected by both of the AL query systems were more informative than random selection.

Furthermore, one of the core points in the development of a highly confident AL system is the applied QS, which determines the sample selected for the oracle to annotate and to be integrated into the classifier's training set.

In the work of Hande Alemdar et al. [14], a HAR system was developed using Hidden Markov Models. Three functions were created to measure the classifier's prediction confidence (i.e uncertainty) namely: *Least Confident*, *Margin* and *Entropy-based Sampling*. The proposed QSs outperformed *Random Sampling* and allowed a reduction of the required amount of labelled data in the classifier training set from 80% to 66%.

Notwithstanding, uncertainty-based QSs usually choose samples near the decision boundaries. Thus, yielding good results for specific classifiers such as the Support Vector Machine (SVM), whose goal is to maximise the hyperplane margin between the decision boundaries. However, for other classifiers, *Uncertainty Sampling* can result in the neglect of the prior feature distribution space and the introduction of sampling bias to the QS system. On this account, the authors in [15] developed a cluster-based AL framework, that in order to select samples across the entire space distribution, created clusters of unlabelled data from which the most informative sample was selected through a function conjugating both entropy and similarity coefficients.

Additionally, experimental results show that, in some cases, uncertainty-based QSs may tend to select outliers rather than boundary samples [16]. Outliers are often noisy samples, which do not constitute representative data. Thus, introduce bias to the classifier. In order to overcome this issue, in [17–19], the authors used a sampling strategy combining the samples' uncertainty and the local data density. This strategy allowed to select a sample informative both in terms of uncertainty and representativeness, since it was inserted in a region of notable local density, hence, avoiding the selection of outliers. The previously-mentioned works were applied to text classification and multivariate time series classification, having yet, to the knowledge of this work author, have been applied to HAR using time series data.

Furthermore, in order to automatically label more data and additionally expand the classifier's training set, in [19] the authors incorporated a *Semi-Supervised Learning* step to the AL framework. This task was performed using 1-Nearest Neighbour (NN) and a 1-Reverse-Nearest Neighbour (rNN) *Semi-Supervised Learning* technique, automatically labelling close neighbours of the newly annotated sample. In the end, for the same amount of initially labelled data, 1-rNN method outperformed the 1-NN method, obtaining a higher accuracy, F1-Score and percentage of automatically annotated samples.

Regarding *Semi-Supervised Learning* techniques applied in the context of HAR, Maja Stikic et al. [13] explored two different frameworks. The first based on *Co-Training* and ST, and the second based on AL. Both used the accelerometer and the infra-red data and were able to significantly decrease the required amount of labelled data to create a model obtaining similar performance results as SL. Moreover, regarding the first framework, using only the accelerometer data, *Co-training* outperformed ST, however not the SL technique when using the entire training set. Regarding the AL framework, two QSs functions were used, one based on the classifier's uncertainty in the sample's label prediction, and the other based on prediction conflicts between different classifiers. The first obtained the best results using only 12.5% of the dataset data and outperforming the SL technique when trained on the same amount of annotated data.

Thereupon, in order to obtain a HAR system obtaining competitive results to SL, but with a significant decrease in its classifier's training set annotation cost, in this dissertation, it is developed two frameworks based on a Pool-based AL system. The first consisting of an AL system, testing several of the QS found in literature, which the results

are compared to *Random Sampling*. Along with a SSAL approach, complementing the AL process with a *Semi-Supervised Learning* step, introduced with the goal of partly automate the annotation process and increase the available amount of labelled data.

Lastly, in order to verify the veracity of the algorithms proposed in this dissertation, aforementioned techniques such as such SL, Unsupervised Learning (UL) and Passive Learning (PL) are replicated, and its results compared.

1.4 Thesis Overview

This dissertation is divided into five chapters, organised as follows:

The present chapter, Chapter 1, starts by providing the major motivation leading to the development of this thesis and its major goals. Then, it is provided a brief literature survey on previous works focused on HAR, based on SL, AL and *Semi-Supervised Learning* techniques.

In the following chapter, Chapter 2, it is clarified the theoretical concepts needed to contextualise the reader with essential principles applied in this work. Then, in Chapter 3, it is described the methodology of the proposed framework for HAR, namely the different developed approaches based on AL and *Semi-Supervised Learning* techniques.

In Chapter 4, it is introduced the two real-world datasets used to validate and evaluate the proposed framework. Final experimental results are presented, with highlight in the obtained paramount results, reported with a brief discussion.

Lastly, in Chapter 5, the main conclusions and contributions from this dissertation are described, along with recommendations for a further work.

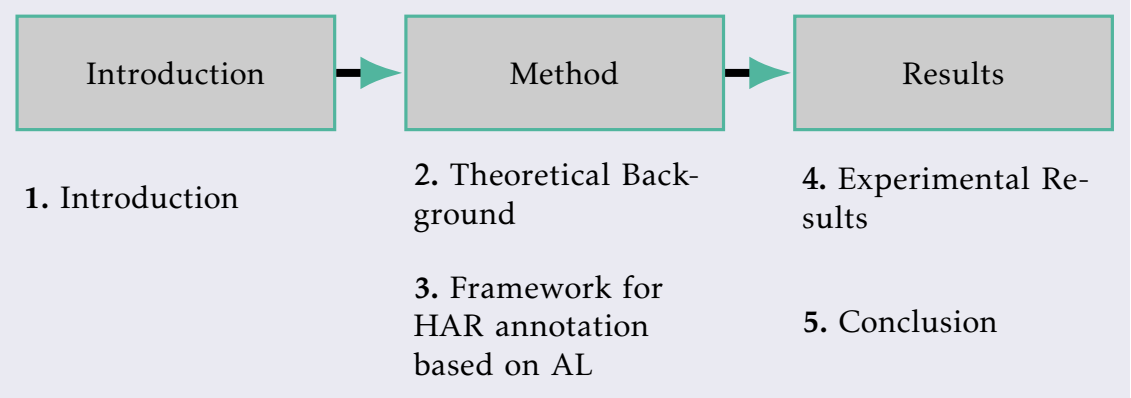


Figure 1.2: Thesis overview structure.

Chapter 2

Theoretical Background

This chapter is divided into two main sections. In the first section, Section 2.1, the accelerometer, gyroscope and barometer sensors integrated into the smartphone and wearable devices used for the acquisition of motion data are introduced. Then, in Section 2.2, it is provided essential theoretical background for the concepts applied throughout the framework developed in this dissertation.

2.1 Wearable and Smartphone Sensors

Smartphones and wearable devices are now, more than ever, present in everyone lives. Moreover, they are equipped with Micro-Electro-Mechanical-Systems (MEMS) chips composed by many sensors such as the accelerometer, barometer, gyroscope and others which are constantly acquiring data that can be used to improve the quality of life of many people, non-obtrusively, quickly and easily [20]. Effectively, MEMS technology has allowed the creation of small, practical, non-intrusive and high computational power sensors with low energy consumption. Thus, making them suitable devices to acquire human motion data in a daily living context with comfort for the user [21–23].

2.1.1 Accelerometer

The accelerometer is the most predominant sensor used for HAR [5, 22–24], highly due to its stable reliable results, especially for simple daily living activities, as the ones explored in this dissertation.

Theoretically, the acceleration is obtained by the application of Newton's second law, being F the force applied to the device and m the device mass. In addiction, the sensor can be represented as a mass-spring system, which by the application of F it is displaced a distance x, enabling the application of Hooke's law. Thus, knowing the system mass, m, and

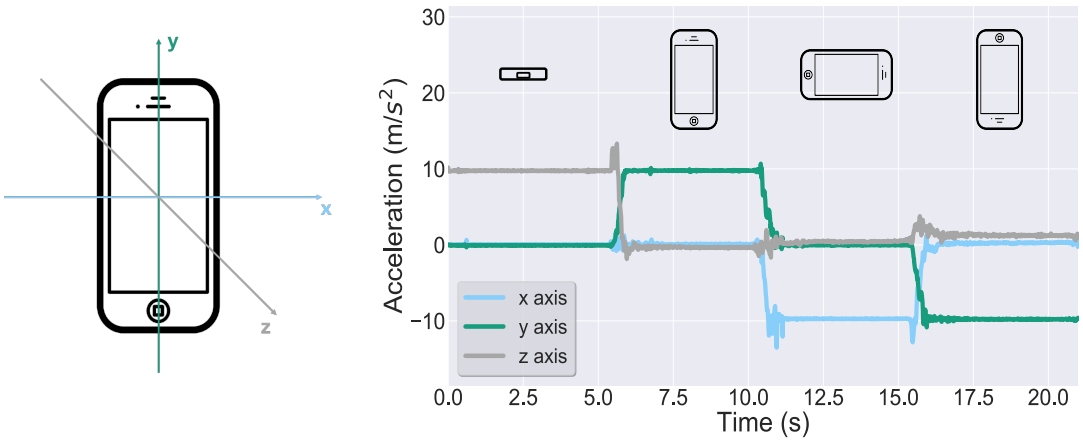
the string constant, k, the acceleration can be obtained by the respective displacement:

$$F = ma = kx \Leftrightarrow a = \frac{kx}{m} \tag{2.1}$$

Therefore, the accelerometer is a motion sensor measuring the acceleration force applied to the device from the user's movement, the gravitational force and its physical environment status, i.e in static or in movement [20].

$$a = -g - (\frac{1}{m}) \sum F \tag{2.2}$$

In order to understand the signal from the accelerometer sensor and its coordinate system, in Figure 2.1, it is shown in 2.1a the standard Android coordinate system and in 2.1b, the associated axis signal output with the device placed in different positions.



(a) Standard Android coordinate system.

(b) Sensor's signal response to the smartphone movement.

Figure 2.1: Smartphone's sensors standard coordinate system in Figure 2.1a, and accelerometer's output signal with the device positioned into different orientations in Figure 2.1b.

As expected, when the sensor is laid horizontally up on a platform, as in the first 5s, the accelerometer's signal on the y and x axis reads a magnitude of approximately 0m/s^2 since the device is at rest, while the z axis signal reads approximately 9.81m/s^2, corresponding to the device having a null acceleration minus the force of gravity, which is approximately 9.81m/s^2. Likewise, with the smartphone upright, the accelerometer's signal on the y axis reads a magnitude of approximately 9.81m/s^2 due to the force of gravity, while the x and z signal axis read a value close to 0. Leading to the conclusion, that, through the comparison of the accelerometer's signal values on the different axis, it is possible to infer the device orientation, and thus, the subject's posture. Thereupon, in the development of the dataset acquisition protocol, a fixed device position and orientation in relation to the user's body must be predefined and chosen according to the studied activity. Thus, ensuring the user's comfort, the reduction of motion artefacts and the obtainment of comparable results through equal acquisition conditions.

2.1.2 Gyroscope

The gyroscope is a motion sensor that measures the rate of rotation, i.e. the angular velocity of the device in radians per second (rad/s) around three axis: yaw, pitch and roll, shown in Figure 2.2a [25].

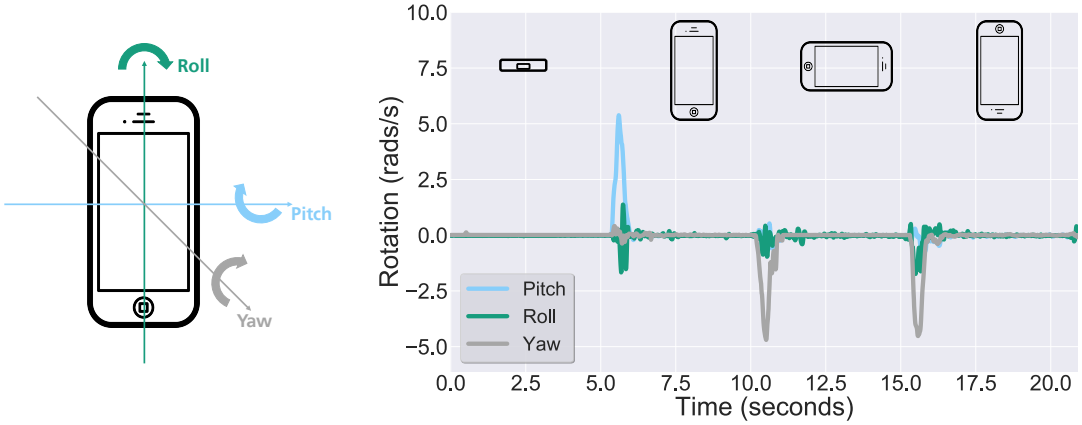
Furthermore, the angular position is obtained through the integration of the device's changes around the orientation axis over time, according to Equation 2.3 [5]:

$$\theta_p(t) = \int_{t0}^{t} \dot{\theta}_p(t)dt + \theta_{p0} \tag{2.3}$$

Where $p = \{Yaw, pitch, roll\}$ and θ_{p0} is the initial angle compared to the earth's axis coordinates in *radians* [5].

Regarding HAR, the gyroscope is majorly used to detect transitional postural activities, such as laying to standing and, along with the accelerometer sensor, the device's position and orientation.

Similarly as for the accelerometer sensor, a study was performed in order to understand the gyroscope's output signal and its coordinate system. Hence, in Figure 2.2, it is shown the sensor's signal while the device is placed in different positions. As expected,



(a) Standard Android coordinate system.

(b) Sensor's signal response to the smartphone movement.

Figure 2.2: Smartphone's sensors standard Gyroscope coordinate system in Figure 2.2a, and gyroscope's output signal with the device positioned into different orientations in Figure 2.2b.

in contrast to the accelerometer, the gyroscope's signal maintains stable values while in the different positions, exchanging its output value during the positions' transitions. Thus, confirming its utility in the recognition of the transitional stages between direction changes.

2.1.3 Barometer

The barometer is an environmental sensor that measures the ambient air pressure. Atmospheric pressure can be defined as the force per unit area exerted by the overhead atmosphere molecules, varying exponentially roughly by 1 hPa (1mbar) for every 10m increase in altitude, according to Equation 2.4 [26].

$$P = P_0 e^{-\frac{mgh}{kT}} \tag{2.4}$$

Where p_0 is the pressure at the referential point, g the gravitational field strength, h the height above the referential point, m the average mass of an air molecule, k the Boltzmann constant (1.380650310^{23})J/K and T the temperature in K [26].

Hence, air pressure will show significant variations across a building height, allowing the discrimination of vertical movements with alterations in altitude, such as climbing up or down the stairs, as seen in Figure 2.3.

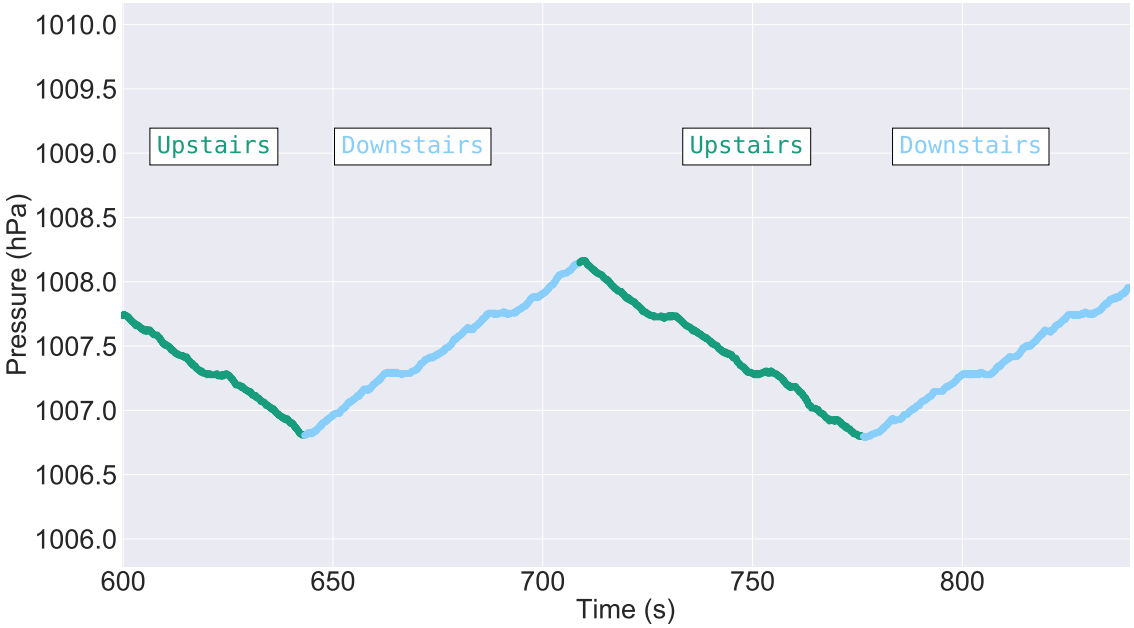


Figure 2.3: Barometer's output signal from a user climbing up and downstairs.

Indeed, as observed, as the user goes up a few flairs of stairs in the first approximately 45s, the air pressure significantly decreases. Followed by the user going downstairs, where an increase in air pressure is observed. Thus, in contrast to the prior sensors, the barometer is independent of the smartphone's position and orientation in relation to the user's body [27].

Additionally, it is important to notice that atmospheric pressure is relative since it can change even for a constant altitude in function of the user's location and the local weather. Indeed, depending on the building infrastructure, the air pressure can reach higher values in wide rooms or lower values, otherwise, while clouds or stormy conditions due to rising air masses may result in low air pressure, and cloud-free weather the opposite.

2.2 Machine Learning

In 1959, during the primordial times of Machine Learning, Arthur Samuel defined Machine Learning as "the field of study that gives computers the ability to learn without being explicitly programmed" [28]. In other words, Machine Learning has the ability to make predictions from patterns and, in a changing environment, the ability to learn, i.e continuously increasing its performance with experience.

2.2.1 Signal Processing

The accelerometer signal is composed by the body's movement acceleration, the gravitational acceleration, intrinsic noise arising from the electronic system and movement artefacts [22]. Therefore, after the sensor's data is extracted, the raw signal must be processed so redundancies and noise are removed. Thus, minimising the algorithm's error, time and computational complexity. To accomplish this task, a low-pass filter is often applied to the signal. Additionally, in some cases, it may be of interest to separate the gravitational and the body acceleration component, performed by the application of a high-pass filter to the signal. Figure 2.4 displays the total body acceleration magnitude and the body movement acceleration magnitude in green and blue, respectively.

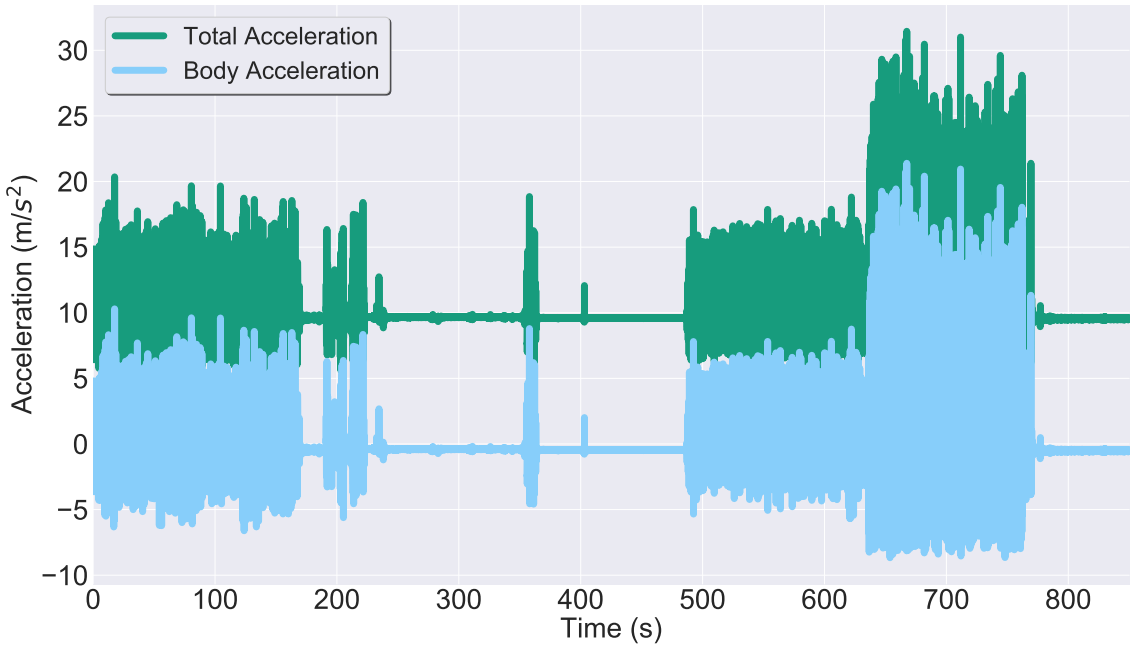


Figure 2.4: Accelerometer signal magnitude of the total acceleration in green and the body acceleration in blue.

However, not always it is recommended to remove the gravitational acceleration since it can provide information about the orientation of the sensor regarding the gravitational axis. Hence, in this dissertation, both the total acceleration and the body acceleration signals are kept and used to extract the features described in the next subsection.

Moreover, since one data sample does not correspond to one activity, in order to retrieve information from a time series, it is necessary to segment the signal into several windows. Only then, from each window, metrics are obtained and used as input to Machine Learning techniques.

There are two main types of window segmentation: **static** and **sliding windows**. In both, the signal is divided into equally sized windows [23], however, in the first, windows are consecutive with no gaps in between, while in the latter, a percentage of samples are overlapped between consecutive windows so one sample can belong to two different windows. Thus, sliding windows can be important to prevent the cut of the signal in inconvenient samples, such as during ongoing continuous cycles or transitional activities [25].

For last, in the segmentation process specifications must be predefined, such as the number of samples per windows, and, in the case of a sliding window, its overlapping samples percentage. Furthermore, the values of those parameters require a trade-off between the studied activity, the classifier's final performance, its resolution, and its computational complexity. Since for simple activities a smaller window will increase the algorithm's temporal resolution, the size of its training set and its performance, at the cost of higher computational complexity due to the higher number of windows [5]. However, for complex activities, each window may lack informative samples. Thus, requiring larger windows, where the total number of windows decreases and consequently, so does the algorithm's computational time [5, 25, 29].

2.2.2 Feature Engineering

As stated above, from each window, the signal is not evaluated sample by sample but rather by the properties extracted from the signal in each window. These properties can be classified according to their domain as **time**, **frequency** and **statistical domain features** and enable the characterisation of the signal in a compact way [23, 25], enhancing its characteristics. Therefore, the chosen set of features can strongly influence the outcome of a Machine Learning algorithm.

To characterise the signal, high dimensional data is translated into a feature vector, whose size equals the number of windows and contains all the information needed to infer each window corresponding activity. However, different features derived from different activities may have a very different range of values. Thus, causing smaller values to be ignored by a Machine Learning classifier while higher values are given more importance to. This tends to happen especially when the classifier involves distance measures. Therefore, it is important to normalise the feature vector values so that each feature contributes proportionately to the classifier. With this intention, in this work, each feature was scaled between 0 and 1 according to Equation 2.5 [30]:

$$X_{scalled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.5}$$

Where X is the feature's value, and X_{min} and X_{max} represent the minimum and maximum values of the feature vector values range, respectively.

In addition, to optimise the algorithm's computational and time complexity, highly correlated features with redundant information can be removed without any loss of information. Only then, through feature selection methods, the best set of features are selected, reducing the feature vector to a lower dimension and once again, the algorithm's computational and time complexity.

Table 2.1, summarises the set of features considered in this work.

2.2.3 Classification

There are essentially three major types of Machine Learning algorithms: SL, UL and SSAL:

- In **Supervised Learning (SL)** is given a training set of L samples mapped to their respective labels, $\{(x_i, y_i)\}$, $i = \{1, ..., L\}$, $x_i \in \mathbb{R}^m$. Where x_i denotes the input feature vector of dimension m, and y_i its class. Moreover, the goal is to create a hypothesis function $h : X \text{ -> } Y$, so that h, given a new unlabelled sample (x) is able to predict its class (y), as depicted in Figure 2.5 [31, 32].

  When y takes the form of discrete values, this process is entitled a **classification** problem, as in the case of HAR, where, for each sample corresponding to an activity, for example walking, it will be attributed a class label value y.

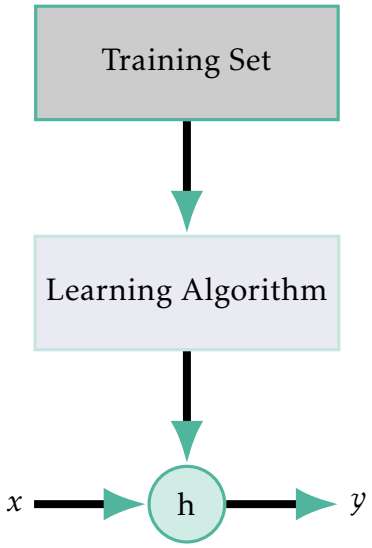


Figure 2.5: Pipeline description of *Supervised Learning*, where a function h given a new unlabelled sample (x) is able to predict its class (y) [31].

  Moreover, the process of mapping each sample to their labels in the ground truth data is denominated **labelling** or **annotation**. A process that, in most of the cases

Table 2.1: Statistical, Temporal and Spectral Domain Features extracted from the sensors' signals to use in the present work.

Statistical	**Skewness** measures the asymmetry of the signal distribution
	Kurtosis measures the distribution shape around a normal distribution
	Histogram plots the number of members for each bin against a total given number of bins
	Mean measures the signal arithmetic mean
	Variance measures the dispersion of the signal value
	Standard Deviation measures the dispersion of the signal values
	Interquartile Range computes the difference between the upper and lower quartile
Temporal	**Max** computes the signal maximum value
	Min computes the signal minimum value
	Centroid computes the arithmetic mean position of all the signal points
	Root Mean Square computes the square root of the arithmetic mean of the square of the signal values
	Median Absolute Deviation computes the median distance between each data value and the signal median value
	Zero Crossing Rate computes the rate at which the signal sign changes
	Autocorrelation computes the correlation of a signal with a lagged version of itself
	Linear Regression computes the linear regression of the signal
Spectral	**Max Frequency** computes where the FFT reaches its 95% of distribution
	Median Frequency computes where the FFT reaches its 50% of distribution
	Fundamental Frequency computes the frequency corresponding to the greatest common divisor of all the frequency components in the signal
	Max Power Spectrum computes the maximum value of the power spectrum density along a given axis.
	Total Energy computes the total energy of the signal given by the squared sum of the spectral coefficients normalised by the length of the sample windows
	Spectral Centroid computes the weighted mean position of the signal frequencies distribution and their probability
	Spectral Spread measures the variance of the signal frequencies asymmetry distribution
	Spectral Skewness measures the asymmetry of the signal frequencies distribution
	Spectral Kurtosis measures the signal frequencies distribution shape around a normal distribution
	Spectral Slope measures how quickly the spectrum decreases towards the high frequencies
	Spectral Decrease measures the decrease on the spectral amplitude though its linear regression computation
	Spectral Roll On computes the frequency so that 5% of the signal energy is contained below of this value
	Spectral Roll Off computes the frequency so that 95% of the signal energy is contained below of this value
	Curve Distance computes the euclidean distance of the signal's cumulative sum of the FFT elements to the linear regression
	Spectral Variation computes the spectral variance from the cross-correlation of two consecutive amplitude spectra

must be performed manually by the user, being a difficult, time-consuming, error-prone activity and sometimes impossible, as the ground truth information may be impossible to infer.

Additionally, the collection of a large amount of labelled data, necessary to increase the algorithm's performance and its ability to generalise correctly into new, and previously unseen data, involves a high volume of data that consequently increases the algorithm's computational and time complexity.

On this account, this dissertation focuses on the development of a framework with reduced manual annotation cost for the user and labelled data scalability. Moreover, we introduce next some of the most common SL techniques applied in HAR, replicated in this thesis in order to incorporate the best performing model into the proposed framework and compare its results:

- k-**NN** classifies each sample according to the most frequent class among its nearest k, $k \in \mathbb{N}$, neighbours. Moreover, generally speaking, larger values for k result in a more robust classifier, less susceptible to noise. However, at the cost of a less fitted decision boundary to the training samples [5, 29]. Furthermore, the overall performance of the classifier is highly influenced by the applied distance metric, which must be defined to better fit the dataset [33]. In the table 2.2 it is exhibited some of the most commonly used distance metrics and their respective math formula.

Table 2.2: Commonly used distance metrics to obtain the distance between two samples (x and y) and their respective math formula.

Distance Metric	Formula
Euclidean	$\mathrm{D(x,y)} = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$
Manhattan	$\mathrm{D(x,y)} = \sum_{i=1}^{m}\lvert x_i - y_i\rvert$
Chebyshev	$\mathrm{D(x,y)} = \max_{i=1}^{m}\lvert x_i - y_i\rvert$
Minkowski$_{order\,r}$	$\mathrm{D(x,y)} = (\sum_{i=1}^{m}\lvert x_i - y_i\rvert^r)^{1/r}$

- **Decision Tree** creates a three like sets of if-then-else rules at each node, from which a sample is carried on following the branches yielding the more information until it reaches a terminal node indicating its predicted output [5, 29, 33]. Moreover, the maximum depth of the tree must be defined in order to avoid overfitting to the training set [29].
- **Random Forest** generates multiple *Decision Trees* from different subsets of the dataset. Then, at each node, a different subset of features contributes

to the split into different branches. Each tree outputs a different prediction, contributing as a vote for the final averaged output [33].

- **SVM** finds the hyperplane that maximises the distance between two different classes in the feature space. The samples having the closest distance to the hyperplane, define the classes decision boundary and are named support vectors. In this sense, SVM is a binary classifier, that in order to be extended to a multi-class problem, a set of multiple binary classifiers must be created, consequently augmenting the time and complexity cost of the algorithm [5, 33].
- **AdaBoost** creates an ensemble of classifiers to test on weighted randomly selected samples from its training set. Moreover, each consecutive classifier is fitted with the adjusted weights enhancing the incorrectly predicted samples so each consecutive learner is trained in error-prone instances [30].
- **Gaussian Naive Bayes** is based on the application of the Bayes theorem to create the decision boundary function [33]:

$$P(y|x_1, x_2, ..., x_m) = \frac{P(y)P(x_1, x_2, ..., x_m|y)}{P(x_1, x_2, ..., x_m)} \tag{2.6}$$

Where y is the predicted output class and (x_1, x_2, ..., x_m) a dependent sample feature vector with m features.

Moreover, under the assumption that every pair of features are independent, the Gaussian Naive Bayes classifier assigns each sample x_i to a class y according to the following equation [30]:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{m} P(x_i|y) \tag{2.7}$$

Where $p(y)$ is estimated using *Maximum A Posteriori* estimation and the likelihood of the features is assumed to be Gaussian so:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{2.8}$$

Where σ_y and μ_y are estimated using maximum likelihood.

- **Quadratic Discriminant Analysis (QDA)** fits a Gaussian density to each class generated by by fitting class conditional densities to the data and using Bayes' rule to return a quadratic decision boundary. Assumes the covariance matrix to be different for each class, hence, it will estimate the covariance matrix separately for each class, fitting the data better than LDA, however, at the cost of more parameters [30].

- **Unsupervised Learning (UL)** includes Clustering, Dimensionality Reduction, Anomaly Detection and Quantile Estimation. This work will focus on the former.

  In a clustering problem, we are given a training set of U samples $X_U = \{x_1, x_2, ..., x_U\}$, $i = \{1, ..., U\}$. Where $x_i \in \mathbb{R}^m$ denotes the input feature vector of dimension m, however, in contrast to SL, in UL it is not given any labels y_i [32].

  In essence, a clustering algorithm finds structure on the feature space where no prior information is given and divides it into different clusters. Moreover, an effective clustering algorithm should maximise intra-cluster similarities and minimise inter-cluster similarities so homogeneous and well-separated groups are created [22].

  Likewise as for SL, we introduce next some of the most common clustering techniques applied in HAR, replicated in this thesis in order to compare its results to the obtained by the algorithms developed in this dissertation:

  - In **K-Means** the unlabelled dataset is divided into k clusters. Initially, k points are chosen randomly as cluster centres, named centroids. Then, each sample is assigned to the closest centroid cluster, re-calculating the cluster's centroid as the centre of all samples in each cluster. This process is repeated as the samples are assigned to a different cluster and the centroids adjusted. Some of this algorithm's main characteristics are its low time complexity, the fact that it changes significantly with the initial cluster partition, the creation of spherical and equally sized clusters, and the requirement of the number of clusters as an input parameter [22].
  - **Mini Batch K-Means** is very similar to *K-Means*, distinguishing from *K-Means* by the use of subsets of the dataset with a fixed size, mini-batches, instead of using the entire dataset. Thus, minimising the algorithm's time and computational complexity, however, at the cost of a loss in the classifier's performance [22].
  - **Spectral Clustering** applies *K-Means* clustering to the projection of normalised eigenvectors obtained from the Laplacian of the samples' similarity graph. This method is especially useful to create clusters with non-convex boundaries.
  - In **Density-Based Spatial Clustering of Applications with Noise (DBSCAN)** high-density samples, named core data points, are grouped together into a cluster whose size keeps growing until lower-density regions are reached. The density is calculated through the number of data points in a fixed radius. Moreover, this method is used, especially, to create clusters with similar density and in contrast to the previous clustering algorithms, DBSCAN does not require the number of clusters as an input parameter [22].

  - **Gaussian Mixture** consists of a probabilistic model generating clusters from a mixture of Gaussians distributions with unknown mean and covariance, fitted to the feature space data through the implementation of the expectation-maximisation algorithm. Moreover, this method allows to easily describe unusual distributions and, along with DBSCAN, does not require the number of clusters as an input parameter.

In the final analysis, UL techniques' biggest advantage is having no annotation cost, as they do not require the input samples to be labelled. However, this fact comes at the cost of not knowing the samples' ground truth labels, necessary to ascertain the classifier's results. Thus, since HAR models should return a label indicating the performed activity, the HAR systems tend to be supervised or semi-supervised [6].

Therefore, this thesis focuses on a hybrid setting between SL and UL, *Semi-Supervised Learning*, with the goal of enhancing the positive aspects of both. That is, to obtain the SL good results and obtain the respective samples' class labels, with UL small scalability and annotation cost.

- In **Semi-Supervised Learning (SSL)** in addition to the labelled data, the classifier incorporates additional information incorporated in new unlabelled samples (easily available at a large scale) to enhance the its performance and reach a more accurate prediction. Thus, the dataset X is partitioned into the labelled samples $X_L = \{x_1, x_2, ..., x_L\}$ mapped to the labels $Y_L = \{y_1, y_2, ..., y_L\}$ and the unlabelled samples $X_U = \{x_{L+1}, x_{L+2}, ..., x_{L+U}\}$ for which the labels are unknown [32].

  Hence, *Semi-Supervised Learning* allows to achieve similar results as implementing SL, but with smaller annotation cost, as there is a significantly less amount of labelled data. With this in mind, since the goal of this dissertation is to decrease the annotation cost of the SL techniques, *Semi-Supervised Learning* was the Machine Learning type of learning methodology chosen to be implemented in this dissertation.

  Moreover, this work will implement three types of *Semi-Supervised Learning*: ST, AL and PL.

  - In **Active Learning (AL)** a selective sampling function selects from the large unlabelled dataset (also referred as pool set) the samples which are more informative to be labelled and added to the classifier's labelled training set. Furthermore, the samples considered more informative, are usually the samples with the highest gain for the classification process, so that, with a lower amount of labelled data and therefore, lower data volume and manual annotation effort from the user, it is possible to reach a classification performance similar to SL.

    Thus, as presented in Figure 2.6, firstly, in order to learn the model's parameters (θ), a model is initialised with the labelled train set (L). Next, from the

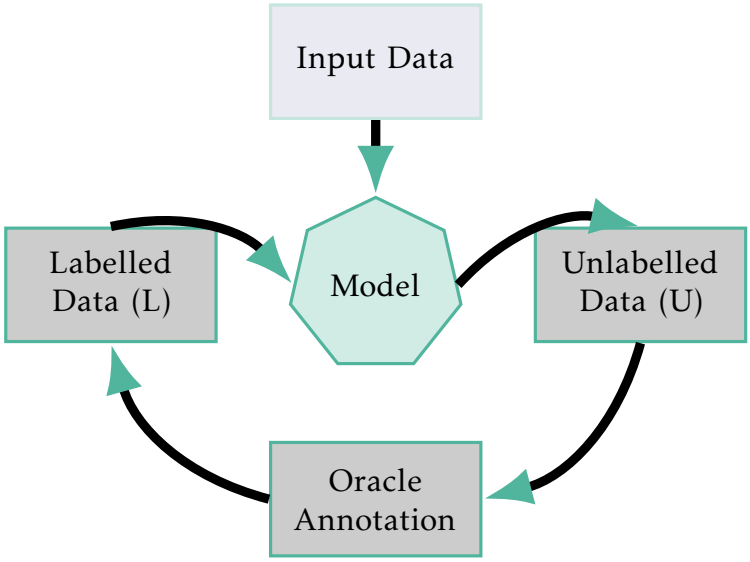


Figure 2.6: Pool-based AL cycle. Where in every iteration the oracle annotates a new sample that is integrated into the model's training set. Followed by an update of the model with its new, augmented training set.

large unlabelled dataset (U), through a QS (Q), it is selected for the oracle to annotate, the most informative sample (x^*). This process is then repeated iteratively until a Stopping Criterion (SC) is met. Hence, initially $L << U$, however in every iteration, the newly annotated sample x^* is removed from U and added to L. Incrementing the labelled train set and consequently, reducing U [14, 34].

Figure 2.7 shows through a two-dimensional Principal Component Analysis (PCA), the growth of the labelled training set throughout the iterations as the oracle annotates one sample per iteration. Moreover, one dot represents one sample whose colour indicates the class that it belongs to so that samples with the same colour belong to the same class.

Figure 2.8 shows the classifier's prediction on the test set samples through a two-dimensional PCA, where each dot embodies a test data sample whose colour represents its predicted class.

Thus, through the comparison of Figures 2.7 and 2.8, it is possible to note the impact of the learner's training set size on the classifier's prediction performance. As it can be observed, initially, in iteration 0, where the classifier's training set is small, it is possible to infer that the classifier's performance is poor since all the samples are predicted as belonging to the same class. However, as its training set grows in every iteration, becoming more representative of the entire dataset, its performance improves. As it is verified, in iterations 150 and 300, where the classifier's prediction start to match more and more the ground truth data, shown in the right lower plot in Figure 2.8.

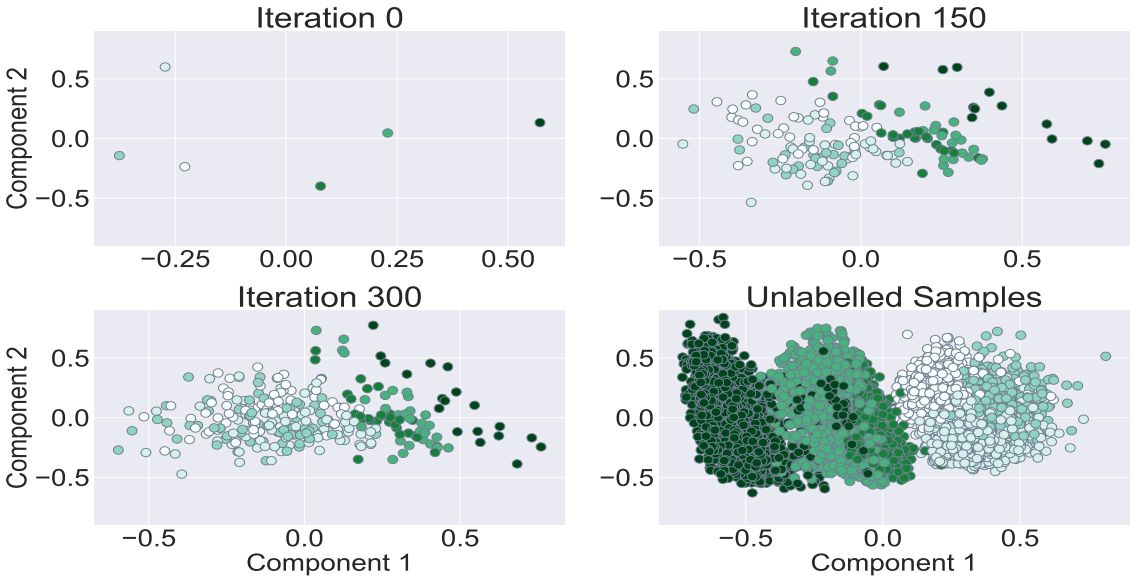


Figure 2.7: Growth of the AL training set from iteration 0 till iteration 300 and the remaining unlabelled samples on the pool set, on a two-dimensional PCA. Each dot embodies a data sample whose colour represents its class. Example performed using the University of California, Irvine (UCI) dataset [35].

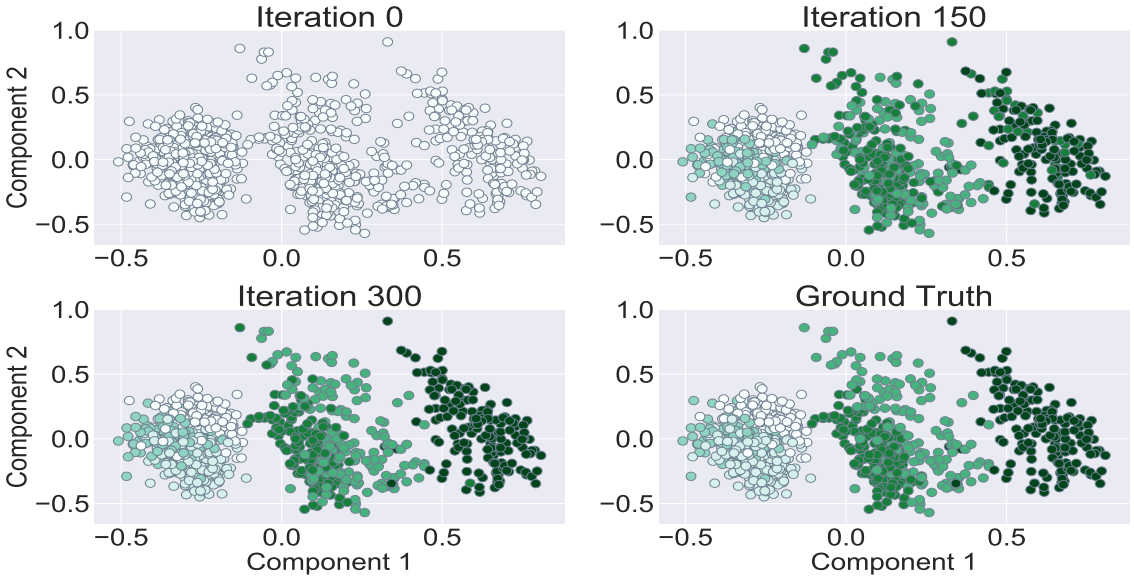


Figure 2.8: UCI test set samples through a two-dimensional PCA. Each dot embodies a data sample whose colour represents the class predicted by the AL classifier, so that, samples having the same colour are predicted by the classifier as belonging to the same class.

– **Passive Learning (PL)** follows the same methodology as AL, but there is no selective sampling function. Instead, the sample is randomly selected from the unlabelled dataset.

– In **Self-Training (ST)** a classifier is trained on the available labelled dataset and posteriorly tested on the unlabelled dataset. Then, test samples having the higher prediction confidence score are added to the classifier's training set and removed from the unlabelled dataset. This process is repeated iteratively as the classifier is re-trained on an increasingly larger and larger training set, increasing its performance.

  Therefore, under the assumption that highly confident predicted labels are correct, the learner uses its own predictions to iteratively teach himself, consequently improving its performance till the unlabelled data set is exhausted, and all the samples become labelled [36], as shown in Figures 2.9 and 2.10. Moreover, comparing both figures similarly as performed for the AL process, the learner starts at iteration 0 with a very poor prediction due to its small non-representative training set, verified by the fact that all samples are predicted as belonging to the same class (since they present the same colour). Nevertheless, as the classifier's training set grows (from iteration 0 to iteration 300), through the addition of highly confident predicted samples to the learner's training set, its performance increases until its predictions match the ground truth data, as observed in the plot in the lower right corner in Figure 2.10.

  Additionally, due to the ST ability to grow and teach himself with the available unlabelled data, the learner's training set size in the iterations 150 and 300 in Figure 2.10 present a significant growth of the train set data volume in comparison to the obtained in the AL process shown in Figure 2.8, iterations 150 and 300. This fact was expected since the later only increments its training set 1 sample per iteration (the sample that the user annotated) while ST adds to its training set all the samples that it is able to predict with high confidence.

  To conclude, as has been noted, both *Semi-Supervised Learning*, ST and AL address the issue that labelled data is scarce and difficult to obtain. Therefore, they focus on the information that can be derived from the unlabelled data while preserving a reduced annotation cost and volume data scalability.

  Moreover, while AL explores the unknown (choosing unlabelled data to be annotated), *Semi-Supervised Learning*, uses what has known to exploit the unknown (unlabelled data to add to its training set and self-improve reaching a higher performance [34]), increasing further the classifier's training set towards a more representative training set. Henceforth, it is only natural to combine both to perform the recognition of human motion activities, as it is employed in this dissertation.

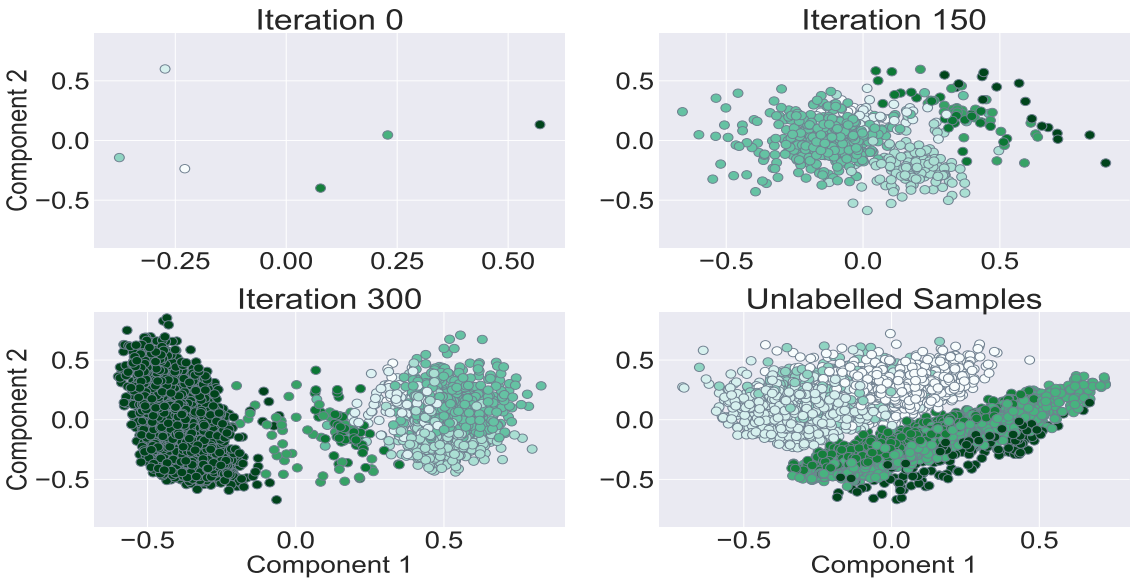


Figure 2.9: Growth of the ST training set from iteration 0 till iteration 300 and the remaining unlabelled samples on the pool set, on a two-dimensional space PCA. Each dot embodies a data sample whose colour represents its class. Example performed using the UCI dataset.

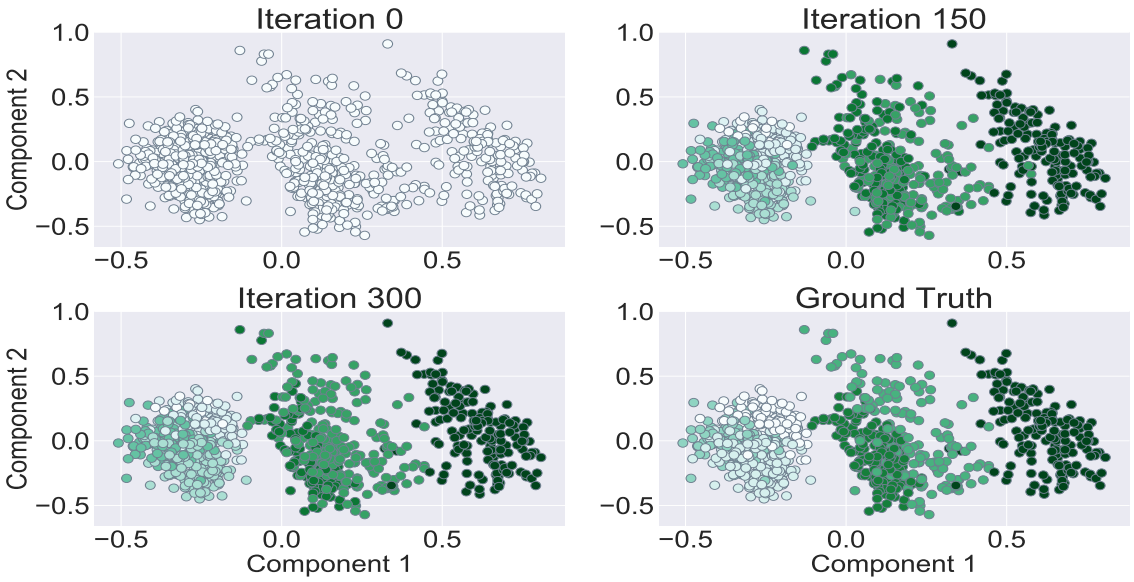


Figure 2.10: UCI dataset test set samples through a two-dimensional PCA. Each dot embodies a data sample whose colour represents the class predicted by the ST classifier, so that, samples having the same colour are predicted by the classifier as belonging to the same class.

2.2.4 Validation

One of the key characteristics of a Machine Learning model is its ability to **generalise** to new unseen data and avoid **overfitting** in its training set samples. Therefore, in order to achieve this, the total dataset must be divided into a train set, to train the model and an independent test set, to test the model. In addition, in some cases, the dataset can be partitioned into three subsets: train set, validation set and test set. The validation set allows to adjust and improve the model on further unseen data before the final results are obtained with the test set. Moreover, in order to yield statistically meaningful results, the test set should take a considerable size, should be representative of the entire dataset and, for last, should not be repeated in order to avoid overfitting [37].

On this account, in this work, we perform K-**fold Cross Validation (CV)**, where the dataset is partitioned into equally sized *K-folds*. Thus, k iterations are performed where in each iteration, $k-1$ folds are used for training and the remaining for testing, so a fold is used for testing only once. The results of the k iterations are averaged and its standard deviation calculated to produce a final overall performance output.

Moreover, to visualise the relation between the classifier's prediction and the ground truth labels, it is commonly used a **Confusing Matrix** (Figure 2.3). Where, considering that there are n classes, the confusion matrix will be a $n \times n$ matrix with a row and column for every class label. Each cell C_{ij} is filled with the total number of predicted samples belonging to the class label i and predicted with the label j. Thus, correctly predicted class labels will occupy the diagonal cells while misclassifications occupy the remaining cells.

Furthermore, from the confusion matrix values, it is possible to obtain statistical metrics, presented in Figure 2.4 for a binary classification. These metrics allow doing an empirical evaluation of the classifier's overall performance and efficiency, through which, it is possible to withdrawn meaningful conclusions.

However, the present work consists in a multiclass classification problem. Therefore, the **accuracy metric** shown in Equation 2.9 [37] is used instead, to evaluate the model's classification performance:

$$Accuracy = \frac{\sum_{i=1}^{n} TP_i}{\#Samples} \tag{2.9}$$

Where n is the total number of classes.

On the other hand, regarding UL, even though they do not require labelled data to perform a classification. They still need to have available the ground truth data in order to asses the model's performance and verify the predicted results.

In the present work, the evaluation of the clustering algorithms is performed using the Adjusted Rand Index (ARI) score. Moreover, the ARI score is a corrected-for-chance version of the **Rand Index** given by Equation 2.10. The *Rand Index* obtains the similarity between clusters through the count of the samples assigned to the same or different clusters in the predicted and ground truth clusters [30]. In addition, since that, by random

change some samples can be assigned to the ground truth cluster, the *Rand Index* is re-scaled into the ARI score, given by Equation 2.11, where the expected RI of random labellings ($E[RI]$) is removed. Therefore, the ARI ensures a value of 1 for equal clusters (allowing permutation) and 0 for random labelling. Thus, ARI $\in [-1, 1]$, being a symmetric measure and ignoring permutations so ARI(a, b) == ARI(b, a) where a and b are considered equal clusters [30].

$$RI = \frac{a+b}{C_2^{\#Samples}} \tag{2.10}$$

$$ARI = \frac{RI - E[RI]}{max(RI) - E[RI]} \tag{2.11}$$

Where a is the total number of pairs of elements in the same cluster in the classifier's prediction and the groundtruth clusters, and b the total number of pairs assigned to different clusters in both the ground truth and the classifier's prediction. Lastly, $C_2^{\#Samples}$ represents the total number of possible pairs combinations (order is not taken into consideration) in the dataset.

Table 2.3: Illustration of a Confusion Matrix.

		Predicted	
		Positive	**Negative**
Real	**Positive**	TP	FN
	Negative	FP	TN

Table 2.4: Evaluation metrics for a binary classification.

Metrics	Formula
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision (P)	$\frac{TP}{TP+FP}$
Recall (R)	
Sensitivity	$\frac{TP}{TP+FN}$
TPR	
Specificity	
TNR	$\frac{TN}{TN+FP}$
F-Score	$\frac{2\times(P\times R)}{P+R}$

CHAPTER 3

Framework for Human Activity annotation based on Active Learning

The following section describes the methodology of the main steps applied in the development of the framework introduced in this dissertation. Moreover, Figure 3.1 illustrates the framework's overall system architecture, developed in Python, using an adaption of *modAL*, a modular active learning framework for Python 3 [1].

3.1 Signal Acquisition and Processing

Hence, the first step in the development of the proposed framework consisted of the acquisition of the sensors' data. Namely, from the **accelerometer**, **gyroscope** and **barometer** sensors, selected to be used in this thesis based on the literature review presented in Section 1.3. The accelerometer and gyroscope sensors since they are the most commonly applied sensors in HAR due to their good classification performance, reliability and cost. Together with the barometer sensor, included in order to improve the discrimination between vertical activities, such as climbing up and downstairs.

Moreover, the embedded device's tri-axis accelerometer and gyroscope sensors, obtain the signal from the respective three axis: x, y and z. From which the magnitude of the

[1] https://cosmic-cortex.github.io/modAL/index

Figure 3.1: Schematic representation of the framework developed for HAR in this thesis.

signal can be computed, measuring the instantaneous intensity of the user's movement at time t, according to Equation 3.1.

$$mag(t) = \sqrt{acc_x^2(t) + acc_y^2(t) + acc_z^2(t)} \tag{3.1}$$

Furthermore, the signal from each axis is a linear sum of the body acceleration component, the gravitational acceleration component and noise from the measurement system (Equation 3.2).

$$acc(t) = acc_{body}(t) + acc_{gravity}(t) + noise(t) \tag{3.2}$$

According to literature, the body acceleration is contained mostly below 15Hz, while the gravitational component goes up to 3Hz [22, 24]. Thus, in order to retrieve the noise from the signal, a band-pass filter [2], with cutoff frequencies of 0.3Hz and 15Hz was applied to the signal. Consequently, retrieving high-frequency noise and the acceleration gravitational component. However, since as stated in Subsection 2.2.1, the gravitational acceleration component can provide additional information, namely, regarding the orientation of the sensor's device, the filtered acceleration signal was kept as well. As presented in Algorithm 1, where the original accelerometer's signal, the filters cut-off frequencies (lower f_L, and upper f_U) and the order of the filter are given as input and the filtered total acceleration signal and the filtered body component signal are returned as output.

Algorithm 1 Filtering

Inputs: $total_{acc}$ signal, lower cutoff frequency f_L, upper cutoff frequency f_U, $order$
Output: filtered $body_{acc}$, filtered $total_{acc}$

1: $fs \leftarrow len(t)/t[-1]$ ▷ Obtains the signal's frequency sampling
2: $total_{acc} \leftarrow lowpass(total_{acc}, f_U, order, fs)$ ▷ Attenuates the frequencies higher than f_U
3: $body_{acc} \leftarrow bandpass(total_{acc}, f_L, f_U, order, fs)$ ▷ Attenuates the frequencies outside f_L and f_U
4: Return $total_{acc}$, $body_{acc}$

Figure 3.2 illustrates the accelerometer's signal where a zoom in was performed before (middle plot) and after the application of the band-pass filter (lower plot), where the signal's noise and gravitational acceleration component were removed.

3.2 Feature Engineering

In order to map the processed data into a suitable Machine Learning input: **time**, **statistical** and **frequency domain features** (described in Subsection 2.2.2) were extracted from the sensors' data, obtaining a high dimensional feature vector (Algorithm 2, adapted from the **Feature Extraction Library and Classification library** [5]). Moreover, in the present work two datasets were used: UCI public dataset [35], and a dataset obtained at

[2] https://github.com/hgamboa/novainstrumentation

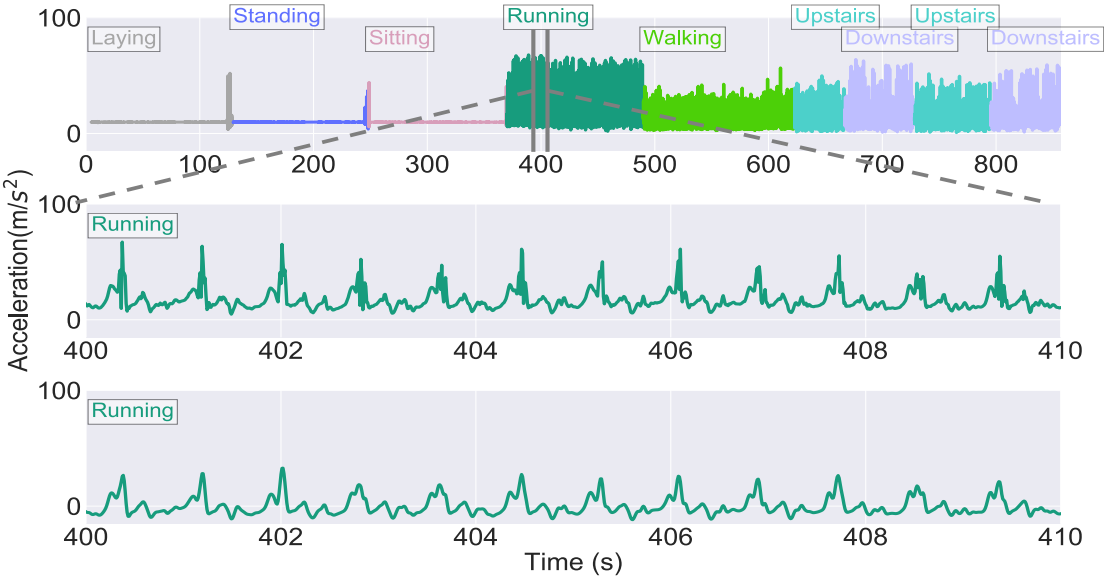


Figure 3.2: Accelerometer's signal during the UCI protocol activities (upper plot) where a zoom in was performed before (middle plot), and after (lower plot), a band-pass filter was applied, removing the signal's noise and gravitational acceleration component.

Fraunhofer, Portugal. Regarding the former, for the application of Algorithm 2, a sliding windows of 2.5*s* with 50% overlap was applied, while a 5*s* windows was stipulated for the Fraunhofer dataset.

Algorithm 2 Feature Extraction

Input: path, ACC file name, BAR file name, GYR file name, windows size.
Output: features file, features names file, labels file, subjects names file, directories file

Furthermore, from the feature vector obtained by Algorithm 2, it is necessary to clean the data and select the most relevant features. Thus, removing redundant information, reducing the algorithm's time and computational complexity and incrementing its classification performance. This task was performed, first, through the **Python Data Analysis Library profiling method** [3]. Where features having more than 90% of correlation with each other are identified and removed. Along with sklearn's **Recursive Feature Elimination with CV** [30], which selects the optimal features over a 10-fold CV, by iteratively considering smaller and smaller sets of features, where, at every iteration, the least important are removed until the classifier's accuracy decreases.

However, due to the high amount of features remaining after a backwards feature elimination, there was a need for a different method. Thus, a *Forward Feature Selection* method was implemented, following Algorithm 3. In the **Forward Feature Selection** method, a set composed by the best features is obtained by the iterative selection and incorporation into the best features set, of to the feature achieving the highest accuracy

[3] https://github.com/pandas-profiling/pandas-profiling

score. Hence, the set with the best features is augmented iteratively one feature per iteration until the addition of a new feature no longer results in an increase of the classifier's accuracy. Moreover, in Algorithm 3, BS_f denotes the set with the optimal features, k its length, and *clf.accuracy_score*, the 10-fold CV classifier's accuracy score function.

Algorithm 3 Forward Feature Selection

Input: all features
Output: best set of features

$BS_f \leftarrow []$ ▷ Initialises variables
$k \leftarrow 0$
$acc_k \leftarrow 0$
Selects feature f^* achieving the highest accuracy score
$BS_f \leftarrow BS_f \cup f^*$ ▷ Adds f^* to best features set
$k \leftarrow k+1$ ▷ Updates current number of optimal features
$acc_k \leftarrow clf.accuracy\_score(BS_f)$ ▷ Sets best accuracy constant to the classifier's score with f^*
while $clf.accuracy\_score(BS_f + f^*) < acc_k$ **do**
    $BS_f \leftarrow BS_f \cup f^*$ ▷ Augments the best feature set with f^*
    $k \leftarrow k+1$ ▷ Updates current number of optimal features
    $acc_k \leftarrow clf.accuracy\_score(BS_f)$ ▷ Updates best accuracy constant
end while
Returns BS_f ▷ Return set with best features

To finish, for the reasons presented in Subsection 2.2.2, each feature of the feature vector was scaled to a range between 0 and 1, according to Equation 2.5.

3.3 General Active Learning Strategy

As mentioned in the literature review in Section 1.3, the discrimination of human motion activities is often covered by sensor-based Machine Learning classification techniques. Furthermore, according to the motivation provided in Section 1.1, in the present work two main frameworks are implemented, allowing to perform the recognition of human activities with minor annotation cost for the user. The first focusing on AL and the second, incorporating into the AL process a semi-supervised step. The current section focuses on the former and 3.4 on the latter.

Hence, in the AL process, previously described in more detail in Subsection 2.2.3 and in short in Algorithm 4, after a learner is initialised on the initial training set, a QS (Q) selects the most informative sample (x^*) from the unlabelled data (U) for the oracle to annotate and add to the learner's training set. This process is then repeated iteratively until a stopping criterion is met, with, in every iteration the learner's training set expanding with informative data and its performance improving.

Moreover, the AL process is independent of the classifier used. Therefore, in Section 4.3, a study was performed in which several of the most commonly applied SL and UL classifiers were tested, where *Random Forest* obtained the best results. On this account,

Algorithm 4 General Active Learning

Input: initial train set L, unlabelled validation set U, independent test set T
Output: predicted labels for the test set

$\theta \leftarrow clf.fit(L)$ ▷ Learns model on initial training set
while SC not met **do**
    Selection by Q, of most informative sample: x^*
    Ask Oracle for x^*'s label
    $L \leftarrow L \cup x^*$ ▷ Augments the model's training set with x^*
    $U \leftarrow U \setminus x^*$ ▷ Removes x^* from unlabelled samples U
    $\Theta \leftarrow clf.fit(L)$ ▷ Updates model
    Return $clf.predict(T)$ ▷ Returns predicted labels for the test set
end while

Random Forest was the classifier used in the AL process and the classifier implied when the *learner*, *model* or *classifier* is referred to from this point forward.

Given this points, in order to obtain a highly confident AL system, there are three main issues that will be addressed in the forthcoming subsections: the initial train set L, the QS Q and the SC.

3.3.1 Initial Train Set

Being the major goal of this dissertation to develop a framework requiring the minimum annotation effort from the user. The initial train set (L) was created with merely one sample per class. Thus, resulting in an initial train set with six randomly chosen samples for the UCI dataset, and seven samples for the Fraunhofer dataset. Therefore, it is made the assumption that through randomly selecting one sample per class, it is possible to initialise an AL learner capable of achieving a labelled train set representative of the entire feature space.

3.3.2 Sample Selection Strategy

The second core element of an AL system is the development of a QS (Q), able to select from the unlabelled dataset the sample considered as the most informative that will cause an improvement of the classification performance when added to its training set. Thus, through the augmentation of the classifier's training set with the samples most contributing to a good performance, the AL system optimises the trade-off between the classifier's performance and the number of labelled samples. Since a smaller amount of labelled data will be able to achieve a better performance and the cost in annotating each sample is compensated by its high value for the classification.

Henceforth, the ability of the AL system to create a representative labelled training set, reaching a highly accurate classification with less labelled data is denoted as **Selective Sampling**. In contrast to PL, where the samples are chosen randomly from the entire

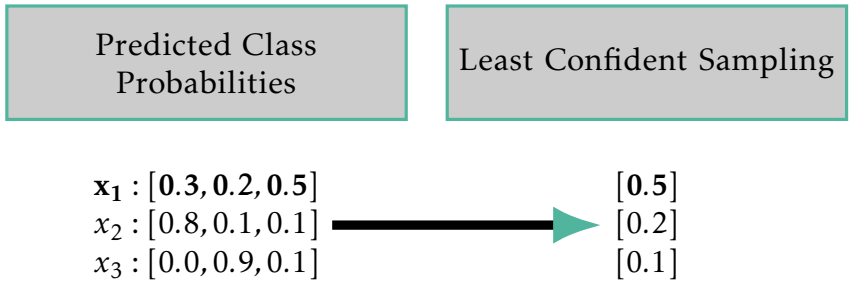


Figure 3.3: Computation of the uncertainty score according to *Least Confident Sampling* QS. In this figure, it is presented the predicted class probabilities for the samples x_1, x_2 and x_3 on the left, and the respective uncertainty scores on the right. The most informative sample is shown in bold.

dataset. Possibly, leading to a classifier requiring extra annotation effort that may not generalise well due to its poor non-representative training data.

A common metric to evaluate the sample's usefulness for the classification is to access the classifier's prediction confidence in that sample's label [11, 13, 14], which is given by the classifier's uncertainty in the prediction of the sample's label. Moreover, in the present work, it is studied three metrics to evaluate the classifier uncertainty in the labels' prediction, corresponding to three different selective sampling functions.

Hence, considering a probabilistic model, such as *Random Forest*, the classifier prediction output is a $U \times n$ matrix, where U represents the total number of the unlabelled validation set samples ($X_U = \{x_1, x_2, ..., x_U\}$, $x_i \in \mathbb{R}^m$, $i = \{1, ..., U\}$), and n the total number of classes existent in the validation set. Where each row is a $1 \times n$ vector with the sample's predicted class probabilities with each cell value given by the prediction posterior probability ($P_\theta(y_k|x_i)$, $k \in \{0, ..., n\}$). With this in mind, it is presented next the developed QSs used in this dissertation:

- **Least Confident Sampling:** selects the sample whose the label the classifier is least certain about, according to Equation 3.3 [12, 34, 38].

$$\begin{aligned} x^*_{LC} &= \arg\max_x (1 - P_\theta(\hat{y}|x)) \\ \hat{y} &= \arg\max_y (P_\theta(y|x)) \end{aligned} \tag{3.3}$$

  Where $\hat{y}$ is the class label which the predictor considers most probable for the sample x.

  Figure 3.3 demonstrates the computation of the uncertainty score according to *Least Confident Sampling*, in a dataset composed by three samples (x_1, x_2, x_3) and three classes (0, 1, 2). Thus, per example, for the first sample (x_1), with the classification posterior probabilities of (0.3, 0.2, 0.5) for the three labels, respectively. The most likely class label for x_1 according to the classifier, with a confidence of 0.5 and uncertainty of 0.5, is the class label 2, since it is the label with the highest prediction class

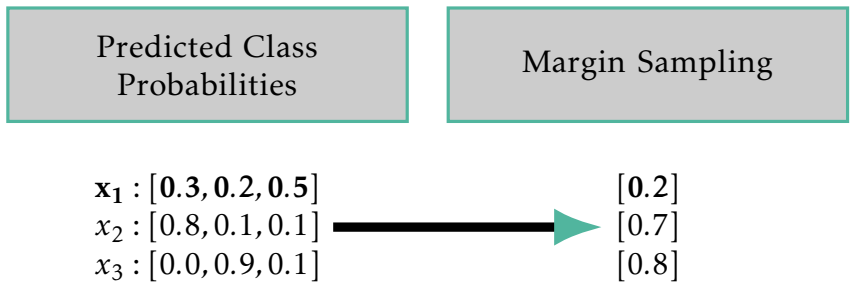


Figure 3.4: Computation of the uncertainty score according to *Margin Sampling* QS. In this figure, it is presented the predicted class probabilities for the samples x_1, x_2 and x_3 on the left, and the respective uncertainty scores on the right. The most informative sample is shown in bold.

probability and least uncertainty. Performing the same thought for the remaining samples (x_2 and x_3), the sample considered as most informative, hence, the sample selected by the *Least Confident Sampling* QS is the sample x_1, since it has the highest uncertainty score, 0.5, in comparison to the remaining samples, with uncertainty score values of 0.2 and 0.1 respectively.

- **Margin Sampling:** selects the sample with the minimum difference (margin) between the classification probabilities of the first and second most likely class, according to Equation 3.4.

$$x_M^* = \arg\min_x (P_\theta(\hat{y_1}|x) - P_\theta(\hat{y_2}|x))) \tag{3.4}$$

  Where $\hat{y_1}$ and $\hat{y_2}$ represent the first and second class labels which the classifier considers as most probable for the sample x. Thus, the *Margin Sampling* QS allows to incorporate into the uncertainty calculation score, the classification's probability distribution of one more class label in comparison to *Least Confident sampling*.

  The *Margin Sampling* strategy yields good results especially with SVM classifiers (described in Section 1.3), whose goal is to maximise the hyperplane margin between the decision boundaries. Moreover, a sample with a small margin in its classification probabilities is a sample in which the classifier is torn and unsure in its prediction. Since these samples are usually located near decision boundaries, through their selection, *Margin Sampling* contributes to a better discrimination between classes and the redefinition of the classifier's decision boundaries [12, 34, 38].

  Figure 3.4 demonstrates the computation of the uncertainty score according to *Margin Sampling*, in a dataset composed by three samples (x_1, x_2, x_3) and three classes (0, 1, 2). Thus, per example, for the first sample (x_1), with the prediction class probabilities of (0.3, 0.2, 0.5) for the three labels, respectively. The first and second most likely classes for x_1 according to the classifier's prediction are the classes 2 and 3 with a confidence of 0.5 and 0.3 and uncertainty values of 0.5 and 0.7, respectively. Therefore, the sample x_1 will present a Margin score of 0.2. Performing the same

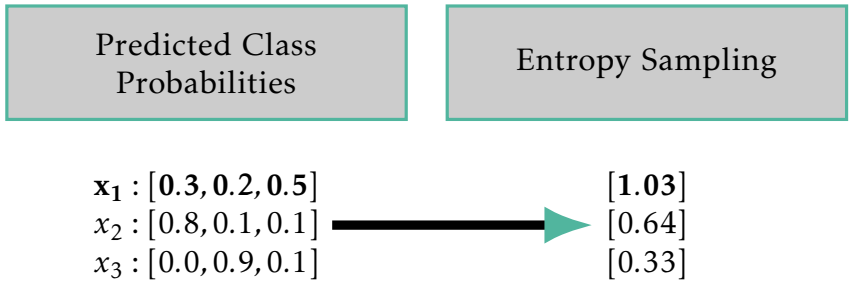


Figure 3.5: Computation of the uncertainty score according to *Entropy Sampling* QS. In this figure, it is presented the predicted class probabilities for the samples x_1, x_2 and x_3 on the left, and the respective uncertainty scores on the right. The most informative sample is shown in bold.

thought for the remaining samples (x_2 and x_3), the sample considered as most informative, hence, the sample selected by the *Margin Sampling* QS is the sample x_1, since it has the minimum margin score, 0.2, in comparison to the remaining samples, with margin score values of 0.7 and 0.8, respectively.

- **Entropy Sampling:** selects the sample with the greatest entropy value, according to Equation 3.5.

$$x_E^* = \arg\max_x \left(-\sum_k^n (P_\theta(\hat{y}_k|x_i)) \log P_\theta(\hat{y}_k|x_i) \right) \tag{3.5}$$

Where $\hat{y}_i$ represents the posteriori probability for the sample x_i ($x_i \in \{x_1, x_2, ..., x_U\}$, $i = \{1, ..., U\}$), belonging to the class y_k, $k \in \{0, ..., n\}$. Thus, this function has the advantage of considering the classification's prediction probability for all the class labels, in contrast to the previously-mentioned QSs [12, 34, 38].

Figure 3.5 demonstrates the computation of the uncertainty score according to *Entropy Sampling*, in a dataset composed by three samples (x_1, x_2, x_3) and three classes (0, 1, 2). Thus, per example, for the first sample (x_1), with the prediction class probabilities of (0.3, 0.2, 0.5) for the three labels, respectively. Applying Equation 3.5, the sample x_1 will present an entropy value of -(0.3log(0.3) + 0.2log(0.2) + 0.5log(0.5)) = 1.03. Performing the same calculation for the remaining samples, the sample x_2 will shown an entropy value of 0.64 and the sample x_3 an entropy value of 0.33. Therefore, the sample considered as most informative, hence, the sample selected by the *Entropy Sampling* QS is the sample x_1, since it has the highest entropy score (1.03) in comparison to the remaining samples.

Furthermore, in order to create an homogeneous initial training set, a weight $W = (1 - p_l)$ was introduced to the QS while the training set was less than 1% of the validation set, according to the following equation [39].

$$x^* = (1 - p_l) * f \tag{3.6}$$

Where f = {Least Confident Sampling, Entropy Sampling, Margin Sampling} and p_l constitutes the percentage of each label in the training set.

Additionally, according to the literature presented in Section 1.3, a sample with high uncertainty will most likely be an outlier. Thus, to overcome this issue, we introduce the *Local Density Sampling*.

- **Local Density Sampling:** adapted from [39], selects the sample with higher representation on the feature space, i.e located in a high-density region, which is measured by the amount of NNs surrounding the sample, according to Equation 3.7.

$$x^*_{LD} = \arg\max_x \left(\sum_i^U \left(\sum_j^k \frac{1}{1 + dist(NN(x_i, x_j))} \right) \right) \tag{3.7}$$

Where x_i and x_j are two samples belonging to the unlabelled samples' dataset and *dist* was obtained using the *Euclidean Distance*. For last, *NN* represents the *5-NNs* of every sample in the unlabelled dataset U.

(a) UCI dataset. (b) Fraunhofer dataset.

Figure 3.6: UCI dataset samples information density on a two-dimensional space obtained through a PCA. Samples in high-density regions are represented in darker colours, while low-density region samples exhibit lighter colours.

Figure 3.6 shows the UCI dataset samples information density on a two-dimensional space obtained through a PCA. Moreover, each sample is displayed as a point where, according to the vertical colour bar shown on the right, samples in high-density regions are represented in darker colours, while low-density region samples exhibit lighter colours. Therefore, *Local Density Sampling* would select and regard the greener samples as having the higher utility to be annotated. Thus, tending to select samples in the centre of the displayed clusters.

- **Uncertainty and Local Density Sampling:** obtained through the linear combination between the previously mentioned QSs, according to Equation 3.8. Where *f1*= {*Least Confident Sampling*, *Margin Sampling*, *Entropy Sampling*}, f_2={*Local Density Sampling*} and α was empirically set to 0.5.

$$x^*_{UD} = \arg\max_x(\alpha f_1 + (1-\alpha) f_2) \tag{3.8}$$

Uncertainty and Density Sampling is introduced with the goal of selecting a sample with both high uncertainty and density on the feature space.

To conclude, according to literature, there is not an optimal QS, depending rather, on the application context [34]. However, *Margin* and *Least Confident Sampling* tend to select samples that redefine the decision boundaries of the classifier, reducing its error rate, while *Entropy Sampling* tends to select samples that minimise its log-loss [38]. Apart from these, *Local Density Sampling* selects samples with prominent representation localised in high-density regions, and for last but not least, *Uncertainty and Density Sampling* selects samples whose label the classifier is highly uncertain about, however, avoiding the selection of outliers, due to the introduction of the density weight.

3.3.3 Stopping Criterion

The last core point to be defined on an AL process is its SC. Since, as it is shown in Figure 3.7, there is an instant during the AL cycle, in which the classification's performance stabilises and annotating more samples will not improve the classifier's performance, but rather add additional samples, which result in an unnecessary annotation effort. Hence, in this instant, the AL process should be ended, optimising the trade-off between the classifier's performance and the oracle annotation effort.

Moreover, the AL system must not stop too early, at the cost of resulting in a limited labelled training set and underperforming classification. As well as it must not stop too late either, at the cost of exceeding annotation work.

Therefore, ideally, we would like to stop when the accuracy of the learner stabilises around its maximum value. However, in a real-life application, we expect to work with unlabelled data, so its ground truth is not available and the accuracy of the classification cannot be obtained.

On this account, in this dissertation with the goal of obtaining a SC general enough to be applicable to all developed SSAL methods, QSs and datasets, it is presented the following stopping criteria:

- **Max-Confidence (Max-Conf) SC** [40]: in which, as described above in Subsection 3.3.2, in the *Least Confident Sampling*, from the unlabelled data, it is selected for the oracle to annotate the sample with the highest uncertainty, i.e the sample that the classifier is least confident in its classification. Moreover, if the selected sample has a low uncertainty score, it is possible to presume that the classifier is able to confidently classify that sample, as well as the remaining samples. Hence, the AL process can be stopped.

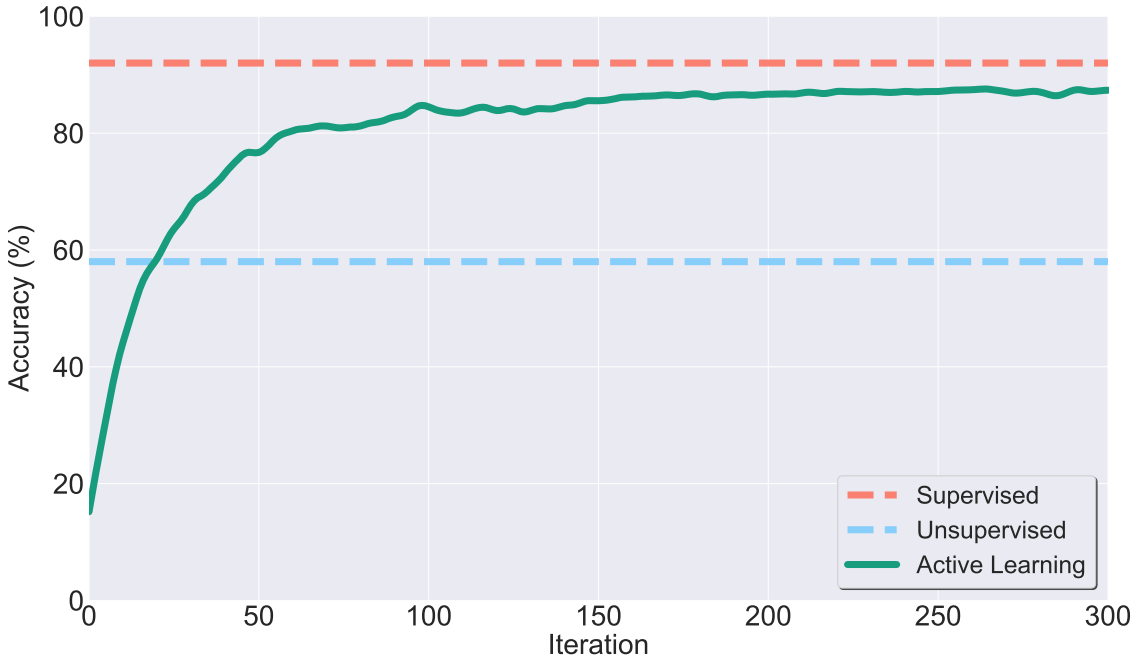
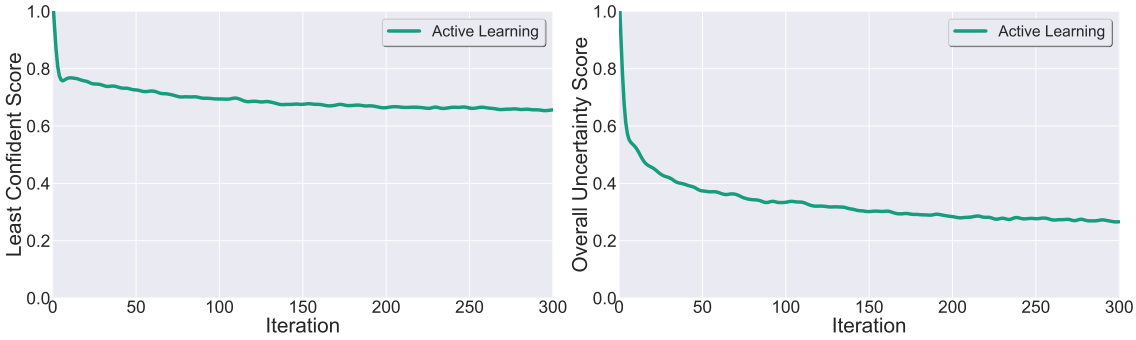


Figure 3.7: AL performance, starting at iteration 0 till iteration 300, in which the classifier increases its accuracy score since in every iteration one more sample is annotated and integrated into its training set. The horizontal red and blue lines denote the accuracy average score of SL and UL, respectively.

- **Overall Uncertainty (Over-Unc) SC** [40]: similar to be above-mentioned SC but instead of stopping the AL system if the least confident score is low, it is used the average of the least confident score computed on the remaining unlabelled samples. That is, if this value, denominated overall uncertainty score, shows insignificant low values, we can assume that the classifier has sufficient confidence in the classification of the remaining unlabelled samples and therefore, the AL cycle can stop.



(a) Least Confidence Uncertainty Score.

(b) Overall Uncertainty Score.

Figure 3.8: Classifier Least Confidence score and Classifier Overall Uncertainty score throughout 300 iterations.

Furthermore, comparing Figure 3.8 and Figure 3.7, it is possible to verify that the stabilisation of the AL performance, overlaps the stabilisation of both the least confident score and the overall uncertainty score. Hence, in the developed stopping criteria, the AL process is terminated once the above-mentioned uncertainty scores stabilise. To perform this task, windows of 5 iterations were created and from each, the *mean (μ)* and *standard deviation (σ)* of the least confident score and overall uncertainty score were extracted. Hence, the AL process stops when both the two following conditions are verified: $|\mu_k - \mu_{k-s}| \leq \delta\ \mu_{SC}$ and $|\sigma_k - \sigma_{k-s}| \leq \delta\ \sigma_{SC}$; $k \in \{0, 2S, ..., N\}$, $S = 5$ and $N =$ number of iterations.

The pipelines of the aforesaid algorithms are shown in Algorithm 7 and Algorithm 8, respectively, in Appendix B. Where A and B are two consecutive windows of 10 iterations each.

- **Classification-Change (CC) SC** [40, 41]: As stated in the literature review in Section 1.3, uncertainty-based QSs tend to select as the most informative samples, the ones located near decision boundaries. Thus, dictating the class to which each sample is allocated to, and therefore, significantly changing the classifier's performance and its prediction output. With this in mind we introduce the CC SC, where the AL is stopped once decision boundaries samples have been annotated and added to the classifier's training set. Moreover, this SC works under the assumption that the *Least Confident Sampling* tends to select samples near decision boundaries and that those significantly change the classifier's prediction output. Under these circumstances, alterations in the classifier's prediction of the unlabelled data labels can be used to infer if the decision boundaries have been changed. Thus, if in two consecutive iterations the classifier's labels prediction has been constant, then, we can assume that the newly annotated samples are not near a decision boundary but rather inside it. Assuming then that main boundary samples have been annotated, and therefore, that the annotation of a new sample will not introduce any new information to the classification process.

  Moreover, to infer the similarity of the classifier's labels prediction over two consecutive iterations, it was used the $F1$-*score* metric. Where the classifier's prediction output is a vector with dimension $U \times 1$ where U is the size of the unlabelled dataset and each element takes the form of the most likely class label for that sample. Algorithm 9, in Appendix B, presents the integration of *Classification-Change SC* in the general AL system.

- **Combination Strategy SC:** a multi-createria-based strategy that combines the prior stopping criteria, namely *Overall Uncertainty SC* and *Classification-Change SC* as the **Overall-CC SC** and *Max-Confidence Uncertainty SC* and *Classification-Change SC* as the **Max-CC SC**.

This method is justified in the cases where the uncertainty score quickly drops to insignificant low values, however, there are still some inconsistencies in the classifier's predictions. Thus, the annotation of new samples may result in changes on the decisions boundaries and therefore, on an improvement of the classifier's performance. Algorithm 10, in Appendix B, presents the algorithm pipeline of the *Combination Strategy SC* where $f_1 = \{$*Overall Uncertainty SC*, *Max-Confidence Uncertainty SC*$\}$ and $f_2 = \{$*Classification-Change SC*$\}$.

3.4 Semi-Supervised Active Learning Framework

As stated in Section 1.1, there is a soaring need for a new annotation technique, able to partly automate the annotation process and reduce considerably the annotation cost of constructing a representative labelled dataset. Thus, with the goal of augmenting notably the amount of available labelled data, in this section, it is introduced the SSAL framework, whose algorithm pipeline is presented in Algorithm 5. The SSAL model is equal to the standard AL framework presented above in Section 3.3, except the lines presented in bold, where the system performs an automatic annotation of the available unlabelled data, without any human involvement. Those samples are then added to the classifier's training set and removed from the unlabelled data. Therefore, this method allows optimising the trade-off between the classifier's training set oracle annotation effort and the number of labelled samples in its training set.

Algorithm 5 General Semi-Supervised Active Learning

Input: initial train set L, unlabelled validation set U, independent test set T
Output: predicted labels for the test set

$\Theta \leftarrow clf.fit(L)$ ▷ Learns model on initial training set
while SC not met **do**
  Selection by Q, of most informative sample: x^*
  Ask Oracle for x^*'s label
  $L \leftarrow L \cup x^*$ ▷ Augments the model's training set with x^*
  $U \leftarrow U \setminus x^*$ ▷ Removes x^* from unlabelled samples
  $\Theta \leftarrow clf.fit(L)$ ▷ Updates model
  Automatically label confident samples C in U
  $L \leftarrow L \cup C$ ▷ Augments the model's training set with C
  $U \leftarrow U \setminus C$ ▷ Removes C from unlabelled samples
  $\Theta \leftarrow clf.fit(L)$ ▷ Updates model
  Return $clf.predict(T)$ ▷ Returns predicted labels for the test set
end while

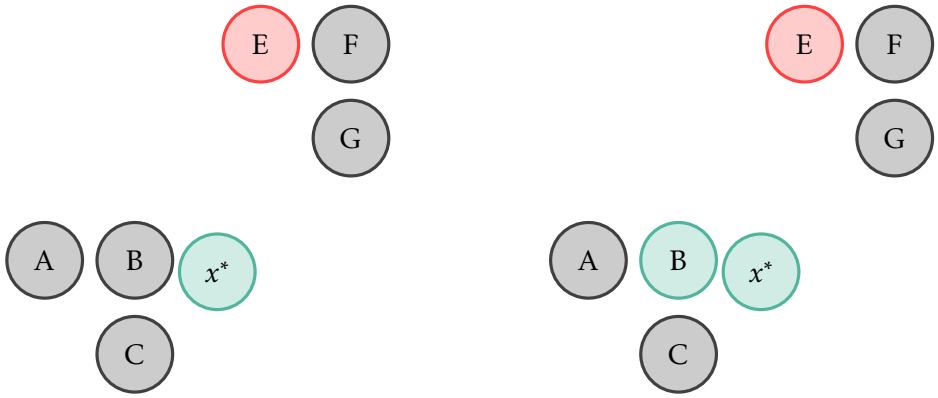
With this in mind, to complete this task with great confidence, three semi-supervised techniques are presented:

- **Self-Training (ST-SSAL)**: ST was described in more detail in Subsection 2.2.3, in which samples which the classifier has a 100% certainty in their class label, are

labelled and added to the classifier's training set. Algorithm 11, in Appendix B, presents the ST-SSAL algorithm pipeline. Where C is the set of confidently annotated samples and $P_\theta(y|x)$ the classifier's prediction posterior probability. Hence, a sample will get annotated if $P_\theta(\hat{y}|x) >= \delta_{ST}$ with $\hat{y}$. δ_{ST} will influence the amount of propagation and its accuracy. A larger δ_{ST} will increase the automatic annotation but decrease its accuracy, since the model is less certain in the annotated label. On the other hand, a smaller δ_{ST} will decrease the amount of annotation but the increase its accuracy, as the few annotations are performed with high certainty. In the present work, δ_{ST} was empirically set to 0.98 in order to obtain a significant automatic annotation while maintaining a good certainty in the annotation.

- **k-Nearest Neighbour label propagation (k-NN-SSAL)**: whose algorithm pipeline is described in Algorithm 12, in appendix B. Where, as observed in bold, after the AL process, the sample selected by the QS (x^*) propagates its label to its k-NNs, creating the set of confidently annotated samples C.

  Figure 3.9 depicts the k-NN label propagation step for $k = 1$. Where each circle represents a sample whose colours (green and red) represent two different classes, while samples in grey denote unlabelled samples. Thus, in this example, the sample x^* propagates its label to its 1-NN, the sample B.



(a) Step 0. (b) Step 1.

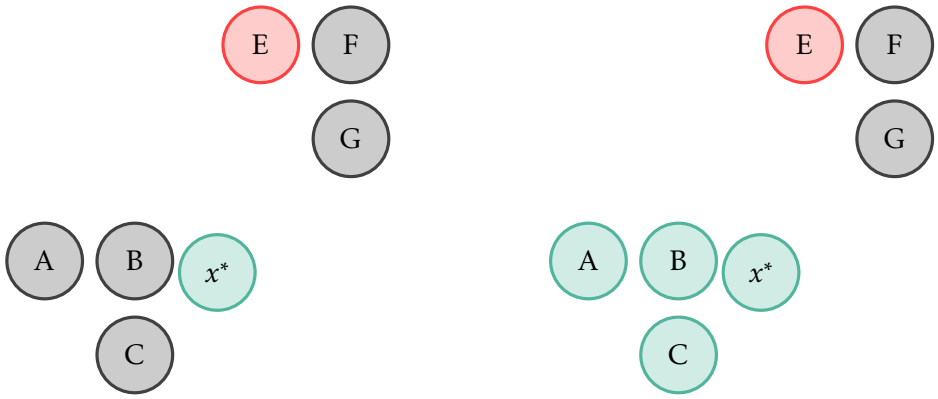
Figure 3.9: Example of 1-NN label propagation, in which the sample x^* propagates its label (in this example represented by the colour green) to its 1-NN, the sample B. Thus, each circle represents a sample whose colours, green and red, represent two different classes, while samples in grey denote unlabelled samples.

Defining k, the number of NNs to propagate x^*'s label to, requires a trade-off between the amount of automatic annotation and the addition of error to the system. Since, with a small k, few samples are automatically annotated, but, the ones annotated are done so with a good confidence, as they are close in the feature space. On the other hand, with a higher k, more samples are annotated, however, at the cost of possibly adding error to the classifier, as x^* is giving its label to samples at a further distance and therefore, may be wrongly annotated.

Under these circumstances, in the present work, k was set to 5, in order to obtain a significant amount of automatically labelled data while maintaining the confidence in the label propagation.

- **rNN label propagation (k-rNN-SSAL)**: whose algorithm pipeline is described in Algorithm 13, in appendix B. Where, as observed in bold, after the AL process, the sample selected by the QS (x^*) propagates its label to all the samples to which, regarding the labelled samples, it is their NN, creating the set of confidently annotated samples C. For the rNN method, k was empirically set to 1 to enhance the propagation performance.

  Figure 3.10 illustrates the rNN label propagation step. Where each circle represents a sample whose colours (green and red) represent two different classes, while samples in grey denote unlabelled samples. Thus, in this example, the sample x^* propagates its label to the samples A, B and C.



(a) Step 0. (b) Step 1.

Figure 3.10: Example of 1-rNN label propagation, in which the sample x^* propagates its label (in this example represented by the colour green) to the samples to which, regarding the labelled samples, it is their NN, the samples A, B and C. Thus, each circle represents a sample whose colours (green and red) represent two different classes, while samples in grey denote unlabelled samples.

Moreover, comparing the above-named k-NN and rNN label propagation methods, namely the Figure 3.9 and Figure 3.10. As observed, the latter generally allows to significantly increase the number of automatically annotated samples. Thus, resulting in a larger labelled set.

3.4.1 Similarity Measures

When performing the label propagation step in the NN-SSAL and rNN-SSAL methods, there is a need for a measurement function able to obtain the distance between the different instances, so each sample's NN can be obtained.

Henceforth, in this section, it is provided four distance measurements. The first two, measuring the distance between the feature vector samples, and the latter between the sensors' data time series:

- **Euclidean Distance**: measures the length of the straight line distance between two samples, x_1, x_2, of m dimensions. Where m is the number of features describing each sample in the feature vector.

$$ED(x_1, x_2) = \sqrt{\sum_{i=1}^{m} (x_1 - x_2)^2} \tag{3.9}$$

  Furthermore, as previously mentioned, in order to use the *Euclidean distance* to obtain the samples' NNs, it is important to first normalise the vectors, as performed in Section 3.2, so both samples equally contribute to the distance value.

  Moreover, considering d: $X \times X$ –> $\mathbb{R}$, being x_1, x_2 and x_3, three samples in X. For d to be considered a distance metric, some conditions must be hold:

  - Non negativity: $d(x_1, x_2) \geq 0$;
  - Identity of indiscernibles: $d(x_1, x_2) = 0$, if and only if, $x_1 = x_2$;
  - Symmetry: $\mathrm{d}(x_1, x_2) = d(x_2, x_1)$;
  - Triangle inequality: $d(x_1, x_2) \leq \mathrm{d}(x_1, x_3) + d(x_3, x_2)$.

  Euclidean distance satisfies all the above properties, thus, it is the most commonly applied distance metric for similarity search [42].

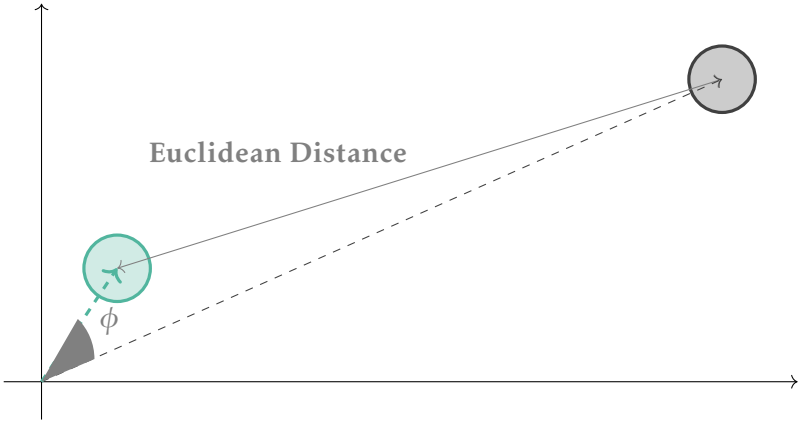


Figure 3.11: Two-dimensional illustration of the *Euclidean distance* and the *Cosine similarity* between two samples. The former is represented by the straight line uniting both samples, while the latter is given by the cosine of the angle ϕ.

- **Cosine Similarity**: measures the cosine of the angle between two samples, x_1, x_2.

$$CSD(x_1, x_2) = 1 - Cos\ Sim(x_1, x_2)$$
$$CSD(x_1, x_2) = \cos(\phi) = \frac{x_1 \cdot x_2}{\|x_1\| \cdot \|x_2\|} = \frac{\sum_{i=1}^{m} A_i B_i}{\sqrt{\sum_{i=1}^{m} A_i^2}\sqrt{\sum_{i=1}^{m} B_i^2}} \tag{3.10}$$

  Furthermore, $\cos(\phi) \in [-1:1]$, where -1 denotes opposite samples and $+1$ coincident samples. Thus, considering that the distance is given by $1 - Cosine\ similarity$, the distance value will become larger as the samples become less similar.

  Figure 3.11 displays a two-dimensional illustration of the *Euclidean distance* and the *Cosine similarity* between two samples. The former is represented by the straight line uniting both samples, while the latter is given by the cosine of the angle between both samples (ϕ). As observed, although the samples are at a significant distance between each other, the angle ϕ is small. Thus, resulting in a large *Euclidean distance* and high *Cosine similarity* (small distance, according to $1 - \cos(\phi)$).

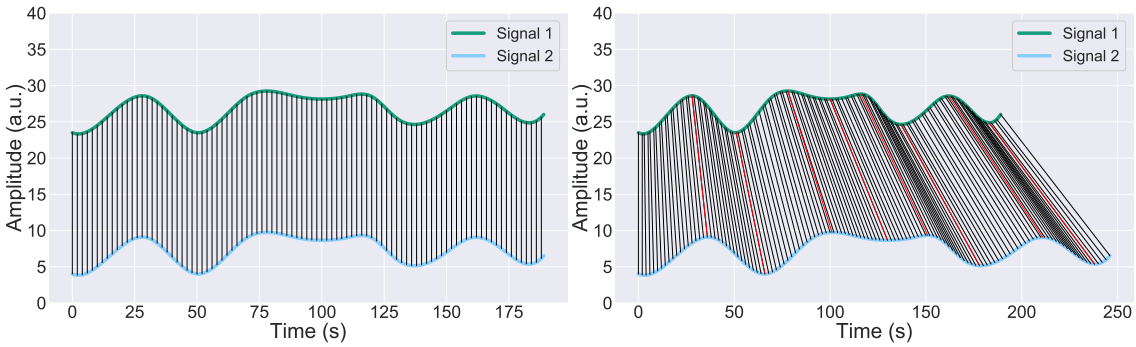
  Moreover, since in *Cosine similarity* the compared samples are normalised to unit length, this metric is usually used in the context of patterns with different or variable sizes, when the magnitude of the vectors do not matter and we are looking for directional similarity, rather than magnitude differences [43].

  Lastly, so far the precedent similarity metrics do not take into account the sensor's signal, but rather, the features derived from it. On this account, we introduce the Dynamic Time Warping (DTW) and Time Alignment Metric (TAM) metrics, using the sensor's data signal to obtain the sample's NNs.

- **Dynamic Time Warping (DTW)**: measures the similarity between two time-dependent sequences through a non-linear alignment minimising the distance between both [43]. Moreover, the minimal distance is obtained through the computation of a local cost measure $C(S1, S2)$, where $S1 := \{s1_1, s1_2, ..., s1_N\}$, $S2 := \{s2_1, s2_2, ..., s2_M\}$ are two time series of length N and M; $N, M \in \mathbb{N}$, respectively, producing a $N \times M$ cost matrix. Where each element corresponds to the euclidean distance, between each pair of elements in the both sequences. Thus, *C(S1, S2)*, will hold a small value (low cost) if $S1$ and $S2$ are similar, or a larger value (high cost) otherwise [42]. Hence, the DTW finds the *warping path*, W, yielding the minimum total cost amount all possible warping paths, by going through the low cost values in the local cost matrix. Lastly, the optimal alignment path, $W := \{w_1, w_2, ..., w_K\}$ with $W_k = \{n_k, m_k\} \in [1:N] \times [1:M]$ for $k \in [1:K]$ must preserve the following conditions [42]:

  1. Boundary condition: $w_1 = C(1,1)$ and $w_k = C(n,m)$;
  2. Monotonicity condition: $n_1 \leq n_2 \leq ... \leq n_K$ and $m_1 \leq m_2 \leq ... \leq m_K$;
  3. Step size condition: W_{k+1} - $W_K \in (1,0), (0,1), (1,1)$ for $k \in [1:K-1]$.

Figure 3.12 presents visually, the difference between calculating the distance between two signals using the *Euclidean distance* and *DTW*. As observed in Figure 3.12a, in the *Euclidean distance* metric the signal points are assumed to be aligned in time. Thus, for similar signals but delayed in time, as in the case of Figure 3.12b, the *Euclidean distance* will fail to see the similarity between the signals, outputting higher distance values in comparison to *DTW*. Additionally, signals can be very



(a) Euclidean Distance. (b) Dynamic Time Warping.

Figure 3.12: Comparison between the *Euclidean Distance* and *DTW* between two signals.

similar in amplitude but show significant differences in the temporal domain. In those cases, the DTW will fail to output a proper distance measure. Hence, we present the TAM distance metric.

- **Time Alignment Metric (TAM)** [42]: uses the optimal time alignment obtained by the DTW to infer the intervals when two time series are in phase, advance or in delay in relation to each other. Thus, returning a distance metric benefiting series in phase, and penalising when signals are in advance or delay with each other. Therefore, the TAM output value will increase as the dissimilarity between the two signals increases and decrease otherwise.

  Considering again, two time sequences $S1 := \{s1_1, s1_2, ..., s1_N\}$ and $S2 := \{s2_1, s2_2, ..., s2_M\}$ of length N and M; $N, M \in \mathbb{N}$. Moreover, assuming $S2$ is delayed in relation to $S1$, by a total time $\overleftarrow{\theta}$, advanced a total time $\overrightarrow{\theta}$ and in time by a time $\overleftrightarrow{\theta}$. The TAM ($\Gamma$) of both time series is given by:

$$\Gamma = \psi_{advance} + \psi_{delay} + (1 - \psi_{phase}), \quad \Gamma \in \{R_0^+ | \Gamma \in [0:3]\} \tag{3.11}$$

  Where, $\psi_{advance}$, ψ_{delay} and ψ_{phase} are given by:

$$\psi_{advance} = \frac{\overrightarrow{\theta}}{N}, \quad \psi_{delay} = \frac{\overleftarrow{\theta}}{M}, \quad \psi_{phase} = \frac{\overleftrightarrow{\theta}}{min(N,M)} \tag{3.12}$$

  Hence, if both signals are constantly in phase, TAM will hold its minimum value of $\Gamma = 0$, with $\overleftrightarrow{\theta} = 1$, $\overrightarrow{\theta} = \psi_{advance} = 0$ and $\overleftarrow{\theta} = \psi_{delay} = 0$.

On the other hand, if both signals are completely out of phase, TAM will yield a value of $\Gamma = 3$, with $\overleftrightarrow{\theta} = 0$, $\overrightarrow{\theta} = N \Longrightarrow \psi_{advance} = 1$ and $\overleftarrow{\theta} = M \Longrightarrow \psi_{delay} = 1$.

3.5 Evaluation

For last, in order to validate the proposed frameworks, it was used a *k-fold CV*, with k set to 10. This value was chosen so the application of a CV allowed to ensure a user independent classifier, avoid overfitting into the training data users, and obtain a diverse test set, large enough to yield meaningful results.

Thus, the entire dataset was divided into the train and test set. Furthermore, from the train set, one sample per class was chosen to integrate the classifier's initial training. The rest of the train set was used to create the validation set, used to improve the learner in the AL process.

Algorithm 6 describes in more detail the developed CV algorithm, in which k iterations were performed, so each user was used exactly once in the test set and $k-1$ times on the validation set. After each iteration, the accuracy score was obtained using the test set and the last 5 iteration's score values were averaged and appended to the iterations' accuracy list. Where all the accuracy values were kept so a final average and standard deviation of the k accuracy values could be computed.

Algorithm 6 K-fold Cross Validation

Input: features by subject, labels by users ▷ Features and labels divided by users
Output: accuracy average, accuracy standard deviation

1: **for** fold in k **do**
2: Divide data into train and test sets
3: $accuracy\_value \leftarrow mean(clf.accuracy([-5:]))$ ▷ Obtains the average accuracy of the framework's last 5 iterations
4: $accuracy\_list \leftarrow append(accuracy\_value))$ ▷ Appends accuracy value to accuracy list
5: **end for**
6: $accuracy_{avg} \leftarrow mean(accuracy\_list)$ ▷ Calculate average of the k-fold accuracies
7: $accuracy_{std} \leftarrow std(accuracy\_list)$ ▷ Calculate standard deviation of the k-fold accuracies
8: Return $accuracy_{avg}$, $accuracy_{std}$

Chapter 4

Results

In this chapter, we start by introducing the datasets used during this dissertation. Followed by an analysis of the performances of the aforementioned methods over several evaluation criteria.

4.1 Datasets

The performances of the proposed frameworks were evaluated using two real-world datasets: the public **Human Activity Recognition Using Smartphones Dataset** (UCI) [35] and the **Continuous Activities of Daily Living (CADL) dataset**, obtained in the context of this dissertation, whose information is summarised in Table 4.1. Moreover, in Appendix A, it is shown the acquisition devices used in the construction of the CADL dataset, along with a brief description of the dataset activities. Thereupon, according to [35] the data from the UCI dataset was previously pre-processed through the application of a low-pass filter and then segmented using sliding windows of 2.56s and 50% overlap, resulting in 128 readings per window. Additionally, a signal having only the body acceleration component was obtained through the application of a Butterworth low-pass filter with a 0.3Hz cutoff frequency.

On the other hand, the data from the CADL dataset was submitted to the signal processing steps denoted in Section 3.2.

As observed in Table 4.1, the UCI dataset is significantly larger than the CADL dataset, with a total of 10 299 samples against 2 047 samples, respectively. This fact is reflected in the annotation cost of each dataset, here presented in an estimation of the time required for the samples' manual annotation. Moreover, the UCI dataset was video recorded so it could be later manually annotated. Hence, considering as an empirical value that every minute of video requires 2mins of annotation and that the dataset has a total of

Table 4.1: UCI and CADL dataset information on: number of users, activities performed, sensors, acquisition device position, dataset size and estimated annotation effort.

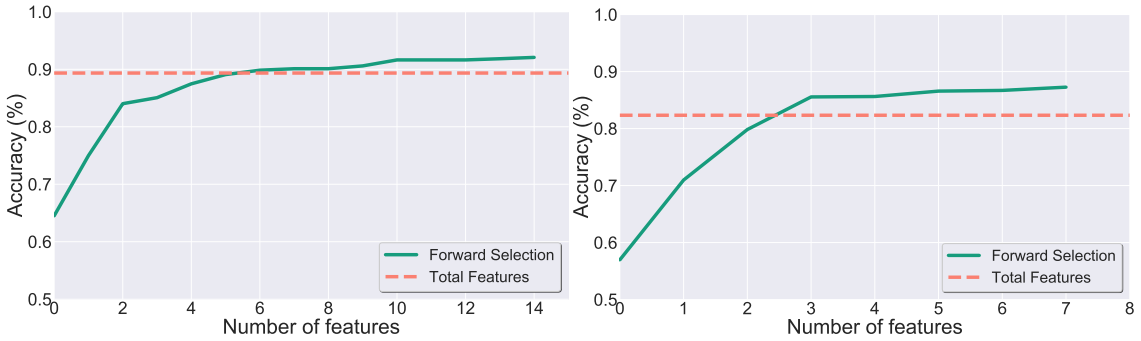
	Datasets	
	UCI HAR Using Smartphones	**Continuous Activities of Daily Living**
Number Of Users	30	12
Activities	Laying, sitting, standing, walking, walking upstairs and walking downstairs	Laying, sitting, standing, running, walking, walking upstairs and walking downstairs
Sensors	Accelerometer (50Hz), gyroscope (50Hz)	Accelerometer (100Hz), gyroscope (100Hz), barometer (30Hz)
Acquisition Device	Samsung Galaxy S II	Samsung S5, IoTip (wearable sensor)
Device Positions	Waist	Right hand, left wrist, right ankle and right side of the waist
Number Of Samples	10 299	2 047
Annotation Effort	~14h	~3h

10 299 windows of 2.56s each, the dataset is estimated to demand an annotation effort of approximately 14h. On the other hand, the CADL dataset was annotated in real time by the user performing the activities. Thus, taking into account that the dataset is composed of 12 volunteers that performed an activity protocol for 14mins each, the dataset annotation cost was approximated to 3h.

4.2 Feature Extraction and Selection

From the previously-mentioned datasets, a set with the best features was obtained through the implementation of the feature extraction and selection methods described in Section 3.2. As depicted in Figure 4.1, the accuracy of the classifier increases with the addition into the best features set, of every feature selected by the *Forward Feature Selection* method until it reaches stabilisation. Henceforth, the algorithm's final feature vector was composed by a total of 15 features for the UCI dataset, and 8 for the CADL dataset, shown in the Horizon plot in Figure C.1 and Figure C.2, respectively, in Appendix C. Furthermore, in the displayed Horizon plots it is possible to visualise the behaviour of the features' values composing the best features sets along the respective protocol activities.

To conclude, it should be noted that, with the developed *Forward Feature Selection* method, it was possible to increase the classifier's performance with a significant decrease in the number of features. Thus, relieving the algorithm's time and computational complexity with minimal information loss.



(a) UCI dataset.

(b) CADL dataset.

Figure 4.1: Illustration in green, for both datasets, of the classifier's accuracy score increasing, as each feature f^* of the best feature set is added to the learner's training set. In red, it is shown the total accuracy score obtained for the initial total set of features before *Forward Feature Selection*.

4.3 Model Selection

The proposed method is independent of the applied SL or UL methods. Therefore, an analysis with some of the most common SL and UL techniques, described in Subsection 2.2.3, was performed with the purpose of finding the optimal technique to incorporate as the learner into the AL process. The respective performance results are shown in Table 4.2, for the SL and UL methods, respectively. Where each value corresponds to the 10-CV percentage average and standard deviation, the latter between parenthesis. As observed, Random Forest achieved the highest accuracy in both datasets, 91.4 (2.4)% and 89.1 (4.0)%, for the UCI and CADL dataset, respectively. For the UL methods, Spectral Clustering attained the highest ARI values for both datasets with a score of 57.8 (3.5)% and 61.9 (8.9)%, respectively. It was compared the performance results, for the public UCI dataset, to state of art researches [44], namely, [23, 35, 45] who have achieved accuracies of 86%, 96% and 96%, respectively. When training and evaluating the SL method on the same train and test set, it obtained an average accuracy of 89.1 (0.6)%.

4.4 Query Strategy Analysis

In the current section, the developed QSs are analysed using the AL framework described in Section 3.3. This is performed in order to find the optimal QS, able to obtain the most representative labelled set and consequently attain the highest performance so it could be incorporated into the SSAL frameworks.

On this account, in Table 4.4, the developed QSs are presented against PL (in which the sample is selected randomly from the unlabelled dataset), SL and UL. Moreover, the following results were obtained averaging the last five values of 250 iterations. Furthermore, for the first iteration, a *Random Forest* model was initialised using an initial training set of one sample per class, consisting of six samples in total (0.06 (0.00)% of the

Table 4.2: SL and UL methods classification's performance shown in accuracy and ARI score, respectively. For all listed values it is shown its 10-CV average and standard deviation, the latter between parenthesis. All listed values are in percentage. The highest performance is shown in bold for each dataset.

(a) Supervised Learning

	Dataset	
Supervised Learning Method	**UCI**	**CADL**
Nearest Neighbours	91.0 (1.9)	83.6 (3.4)
Decision Tree	87.4 (3.5)	83.6 (4.3)
Random Forest	**91.4 (2.4)**	**89.1 (4.0)**
SVM	90.7 (2.6)	77.6 (3.3)
AdaBoost	40.8 (6.8)	54.1 (4.6)
Naive Bayes	88.9 (2.9)	75.9 (3.1)
QDA	90.8 (2.7)	79.0 (3.7)

(b) Unsupervised Learning

	Dataset	
Unsupervised Learning Method	**UCI**	**CADL**
K-Means	52.1 (4.3)	50.9 (6.1)
Mini Batch K-Means	50.7 (5.5)	50.5 (5.3)
Spectral Clustering	**57.8 (3.5)**	**61.9 (8.9)**
Gaussian Mixture	49.8 (2.7)	58.9 (6.6)
DBSCAN	16.4 (7.2)	13.9 (6.5)

Table 4.3: Experimental results for the QSs in terms of: classifier's accuracy and the QS algorithm's computational expense. For all listed values it is shown its 10-CV average and standard deviation, the latter between parenthesis. The best performing algorithm is shown in bold for each dataset.

	UCI Dataset		CADL Dataset	
Query Strategy	**Accuracy in %**	**QS Time in s**	**Accuracy in %**	**QS Time in s**
Local Density * Least Confident	87.6 (4.0)	27.2 (5.1)	83.5 (6.7)	0.9 (0.1)
Least Confident	87.9 (3.7)	0.1 (0.1)	72.0 (5.0)	0.1 (0.1)
Local Density * Entropy	85.7 (3.7)	22.5 (0.4)	80.3 (7.9)	0.9 (0.1)
Entropy	87.1 (2.5)	0.1 (0.1)	70.9 (6.0)	0.1 (0.1)
Local Density * Margin	52.5 (8.6)	22.6 (0.5)	32.8 (8.0)	0.9 (0.1)
Margin	**88.4 (2.8)**	**0.1 (0.1)**	**84.8 (7.0)**	**0.1 (0.1)**
Local Density	63.6 (5.9)	22.5 (0.4)	68.9 (8.5)	0.9 (0.1)
Passive Learning	88.0 (2.8)	0.1 (0.1)	82.8 (6.7)	0.1 (0.1)
Supervised Learning	91.4 (2.4)		89.1 (4.0)	
Unsupervised Learning	57.8 (3.5)		61.9 (8.9)	

Table 4.4: Experimental results for the developed QSs in terms of: classifier's accuracy and the QS algorithm's time expense. For all listed values it is shown its 10-fold average and standard deviation, the latter between parenthesis. The best performing algorithm is shown in bold for each dataset.

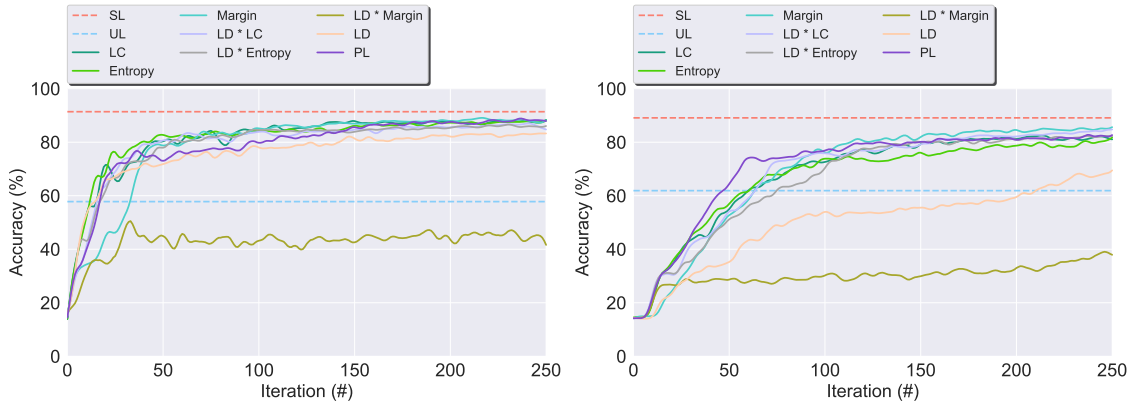
Query Strategy	UCI Dataset		CADL Dataset	
	Accuracy in %	QS Time in s	Accuracy in %	QS Time in s
Local Density * Least Confident	89 (3.1)	29 (7.2)	83 (7.2)	2.0 (0.2)
Least Confident	87 (4.5)	0.059 (0.001)	80 (9.2)	0.0254 (0.0004)
Local Density * Entropy	87 (3.6)	26 (4.3)	80 (10.5)	1.3 (0.2)
Entropy	86 (3.8)	0.059 (0.002)	77 (7.4)	0.0260 (0.0004)
Local Density * Margin	33 (9.4)	25 (3.2)	24 (5.7)	1.3 (0.3)
Margin	**90 (3.2)**	**0.0582 (0.0006)**	**86 (7.2)**	**0.0254 (0.0002)**
Local Density	64 (5.5)	24 (2.3)	60 (7.4)	2 (0.2)
Passive Learning	86 (2.7)	0.0482 (0.0006)	80 (6.1)	0.0172 (0.0005)
Supervised Learning	92 (2.4)		89 (5.0)	
Unsupervised Learning	58 (3.5)		62 (8.9)	

validation dataset) for the UCI dataset, and seven samples (0.38 (0.01)% of the validation dataset) for the CADL dataset. The initial dataset was created through a random selection of samples from the validation set, being removed afterwards.

The comparison between the different QSs techniques is performed based on the following criteria:

1. **Accuracy:** generally speaking, the obtained accuracy values from the QSs are very similar and tend to the value obtained by the SL algorithm. This fact is supported by Figure 4.2, where it is presented the increase of the classifier's average accuracy for the developed QSs throughout the AL iterations. Moreover, the horizontal lines denote the 10-CV average accuracy for SL (in red), and UL (in blue). As expected, the learner becomes more reliable as its training set size increases. Thus, explaining the continuous increase in the classifier's accuracy value throughout the iterations.

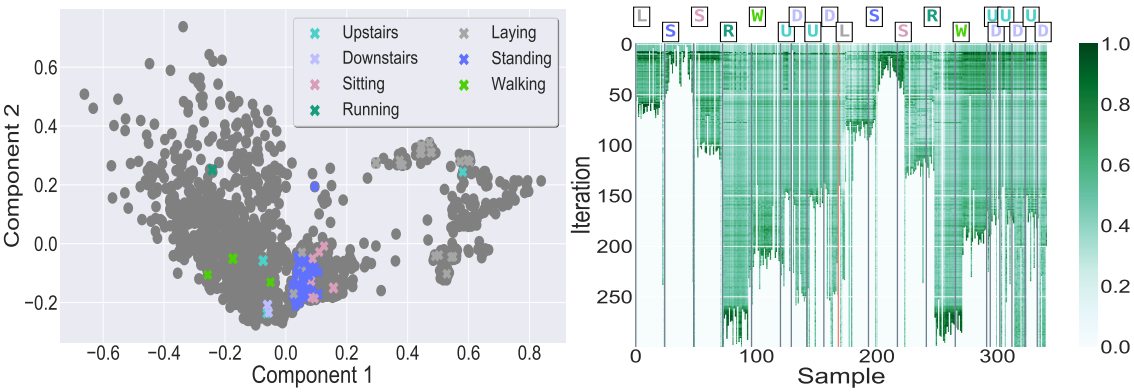
   Furthermore, *Margin Sampling* and *Local Density * Least Confident Sampling* achieve the highest classification's performances, outperforming PL. In contrast to *Local Density* and *Local Density * Margin Sampling*, attaining the lowest score values, not achieving a seasonable performance and being outperformed by UL. These QSs' low performances are explained by the biased training set, non-representative of the entire dataset distribution under which the classifier operates. Since, as supported by Figure 4.3, the density weight in both QSs causes the preferential selection of activities located in high-density regions, for the deterioration of the remaining as they become unknown for the classifier. Under these circumstances, the classifier does not have a homogeneous training set with sufficient amount of samples from



(a) UCI dataset.

(b) CADL dataset.

Figure 4.2: Average increase of the AL classifier's accuracy for the developed QSs throughout the cycle iterations. The horizontal lines denote the average accuracy for SL (in red), and UL (in blue). LD denotes the *Local Density Sampling*, LC the *Least Confident Sampling*.



(a) *Local Density * Margin Sampling.*

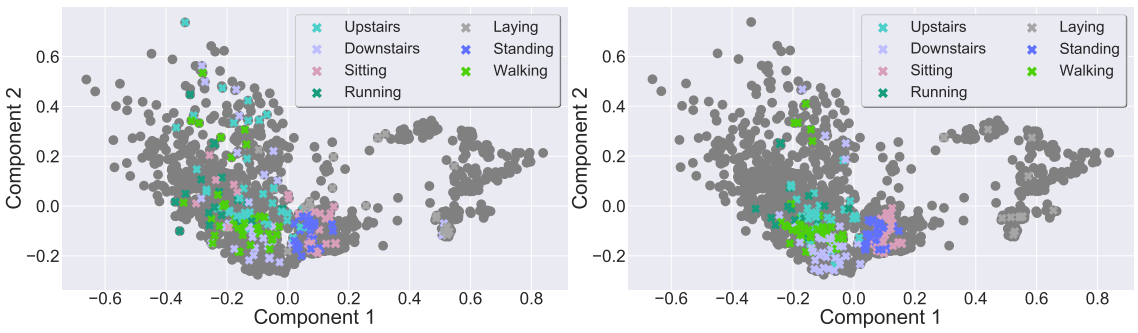
(b) *Local Density * Margin Sampling.*

Figure 4.3: PCA of the AL training set and *Local Density* Margin Sampling* uncertainty score heatmap. Both figures were obtained using the CADL dataset and 200 queries. In Figure 4.3a, the learner's training set samples are depicted by the ×'s, whose colours identify their respective label. The darker grey dots represent the unselected samples existent in the validation set. Figure 4.3b was obtained using 2 users from the CADL dataset whose performed activities are shown by the initial letters and colours according with Figure 4.3a. The users' samples are separated by the red vertical line so user 0 samples are located before the red line and user 1 samples after the red line.

all the class labels from which it can learn to be able to correctly predict all the samples' labels. Figure 4.3a, shows a two-dimensional PCA illustration of the AL training set using the *Local Density * Margin Sampling* QS. Moreover, the classifier's training set samples are depicted by the ×'s, whose colours identify their respective label. While the darker grey dots represent the unselected samples existent in the validation set.

Figure 4.3b, presents a heatmap of the *Local Density * Margin Sampling* uncertainty score throughout 200 queries for a training set formed by two users, whose samples are separated by the red vertical line. Furthermore, in the heatmap darker colours denote samples with high *Local Density * Margin* score values, while lighter colours, denote otherwise.

Comparing both figures, it is possible to observe that the preferential annotation of the high-density region's activities, such as Standing and Sitting, results in the reduction of the classifier's uncertainty for those activities' samples. While unselected activities' samples, such as Upstairs and Downstairs, maintain high uncertainty values across both users since the classifier does not have enough information in its training set to correctly predict those samples' labels. Therefore, it is possible to conclude that not always a large training set is equivalent to the creation of a classifier achieving a good performance. Still, regarding the uncertainty density weighted

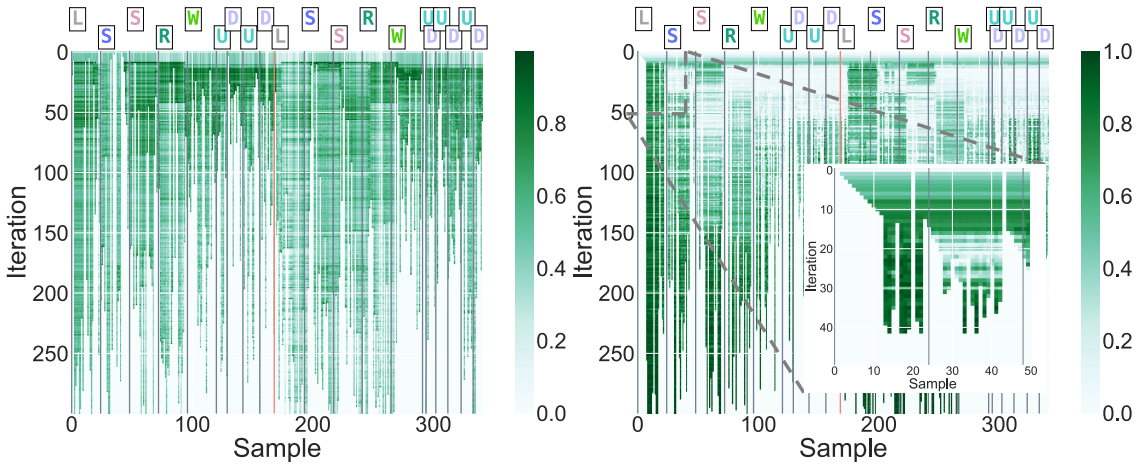


(a) *Least Confident Sampling*. (b) *Local Density * Least Confident Sampling*.

Figure 4.4: PCA of the classifier's training set performing AL using the *Least Confident Sampling QS* and the *Local Density * Least Confident Sampling QS*. Both figures were obtained using the CADL dataset and 200 queries. The learner's training set samples are depicted by the ×'s, whose colours identify their respective label. The darker grey dots represent the unselected samples existent in the validation set.

QSs, with the exception of *Local Density * Margin Sampling*, the remaining obtained results that are in accordance with the literature review [17, 19]. Where the introduction of a density weight to the uncertainty sampling functions resulted in an increase of the classifier's performance.

Moreover, as observed in Figure 4.4, the introduction of the density weight to the *Least Confident Sampling QS* allowed to avoid the selection of outliers through the



(a) *Local Density * Least Confident Sampling*.

(b) *Margin Sampling*.

Figure 4.5: *Local Density * Least Confident Sampling* and *Margin Sampling* uncertainty score heatmaps. Both figures were obtained using 2 users from the CADL dataset whose performed activities are shown by the initial letters and colours according to the antecedent figures, (Figure 4.3 and Figure 4.4). The users' samples are separated by the red vertical line. Figure 4.5b shows a zoom on the heatmap for the first dozens iterations.

selection of samples of high uncertainty in high-density regions across all the feature space. Reinforced by Figure 4.5a, displaying the selection of samples belonging to all activities and how it impacts the uncertainty score across the entire feature space. Furthermore, the arising white vertical lines denote the oracle annotation of the respective sample, resulting in the null uncertainty score value. Moreover, as the samples across the entire feature space are annotated, the uncertainty score decreases homogeneously, thus, presenting the samples in latter iterations lighter colours, correlated to lower uncertainty score values. Hence, in the *Least Confident Sampling* a classifier able to generalise well into new unseen samples with high confidence is obtained since it has a representative training set, featuring samples covering the entire data space.

For last, all things considered, the presented QSs were able to surpass PL, but not very significantly as observed in Figure 4.2. This fact can be explained by the initial deficient classifier's training set (composed by one sample per class), that creates an initial poor probabilistic model, incapable of properly calculating the uncertainty of the unlabelled samples' predictions during the first iterations. Which is verified in Figure 4.5b where the uncertainty score value is constant for the entire validation's set samples during the first iterations. Therefore, as a result, the initial eleven samples selected by the QSs were sequential and consequently, all belonging to the same class, thus, introducing bias to the classifier.

2. **QS Time:** with the exception of the Density weighted QSs, on the whole, the selective sampling functions hold a small execution time. However, the execution time value increases significantly for the aforementioned Density weighted QSs. Especially emphasised for the UCI dataset, due to its larger size. This increase is explained by the calculation of the density weight, that, as shown in subsection 3.3.2, requires the calculation of each sample's NNs. Thus, augmenting notably the algorithm's time and computational complexity. The execution times were obtained using a E3-1285 v6 @ 4.10GHz CPU and 16 GiB of RAM.

Due to the coherent high accuracy performance, surpassing PL, and its low time and computational complexity, *Margin Sampling* was selected as the most suitable QS to be included in both the AL and SSAL frameworks. Hence, forthcoming result presentations on this section were achieved using Margin Sampling. Besides the algorithm's performance analysis, it is also worth to mention a comparison between the amount of labelled data for SL and AL. From 100% of the validation set annotated in SL, to, approximately 2.8 (0.1)% and 13.9 (0.5)%, for the UCI and CADL dataset, corresponding to the annotation of 250 samples and a reduction of 97.2 (0.1)% and 86.0 (0.5)% in the validation set annotation cost, respectively. These results confirm the applicability of AL in the context of HAR and its efficiency in reducing the annotation effort required to construct a highly confident classifier.

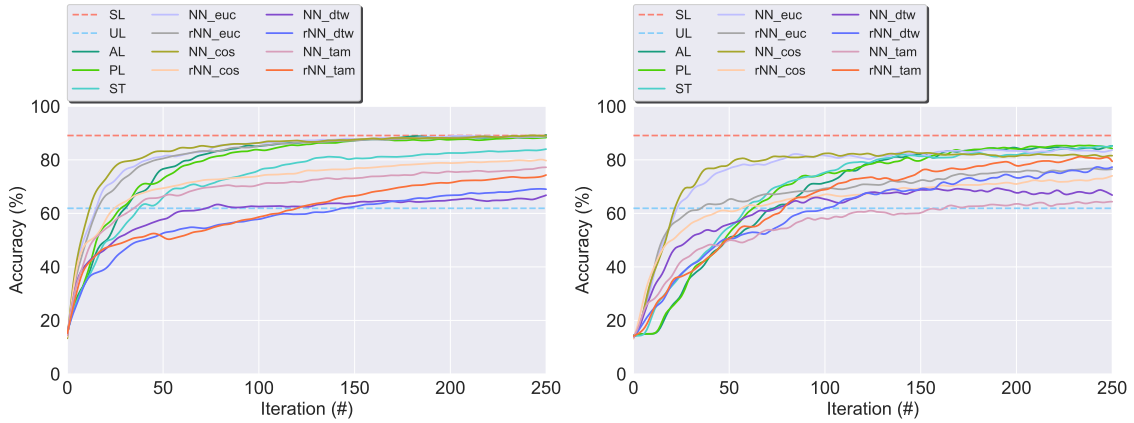
4.5 Active Learning Semi-Supervised Analysis

The current section, applies *Margin Sampling* in the proposed SSAL frameworks, described in Section 3.4. This strategy aims to compare and select the optimal automatic annotation method.

Table 4.5 presents the developed frameworks compared against techniques previously applied in the context of HAR, described in the literature review in Section 1.3, such as AL, SL and UL, replicated in order to verify the model competitiveness.

Henceforth, the following results were obtained averaging the last five values for 250 iterations using the UCI and CADL datasets, respectively, using an initial training set of one sample per class. With the comparison between the different techniques performed based on the following criteria:

1. **Accuracy:** Experimental results demonstrated that with the exception of the SSAL methods using the DTW or TAM distance, the accuracy of the proposed methods converge to the results of the SL technique. Figure 4.6 presents the classifier's accuracy for the developed SSAL methods throughout the AL iterations. For each method, in every iteration the model training set grows, resulting in the increase of the classification's accuracy.

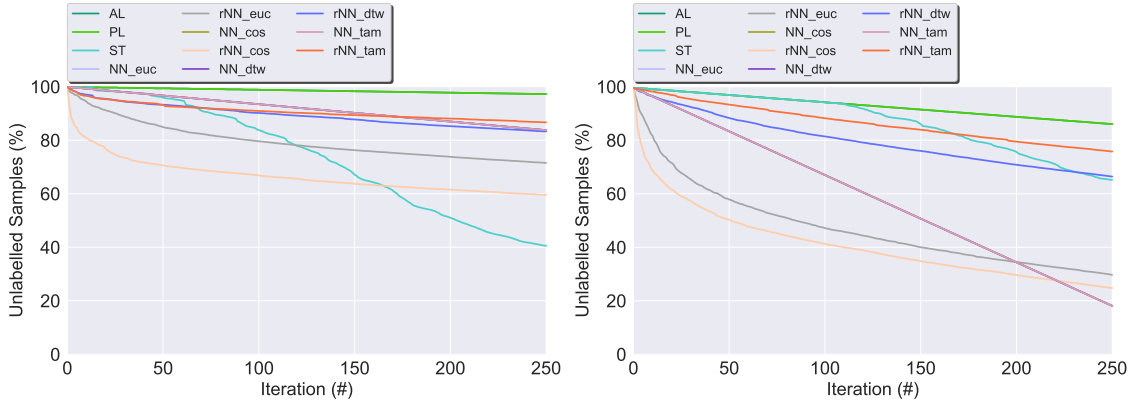


(a) UCI dataset.

(b) CADL dataset.

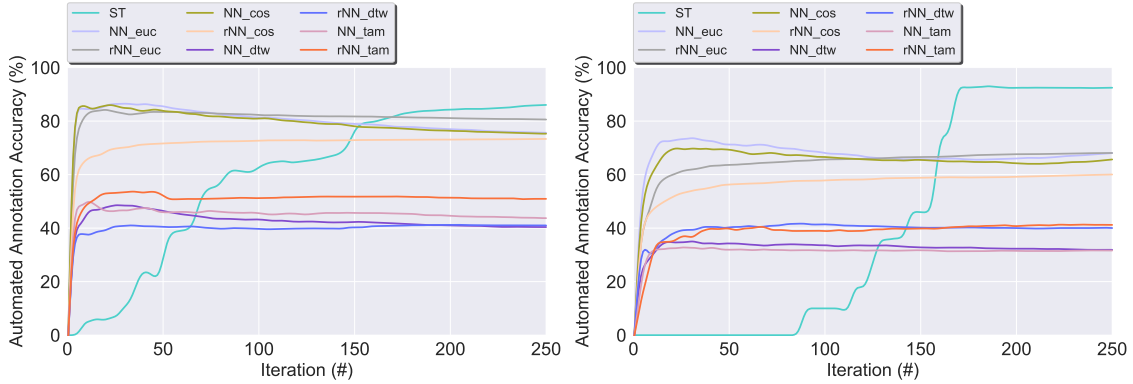
Figure 4.6: Classifier's accuracy for the developed SSAL methods throughout the AL iterations. The horizontal lines denote the 10-CV average accuracy for SL (in red), and UL ARI score (in blue). Following the underscore in the NN and rNN methods: Euc, Cos, DTW and TAM, denote the distances used.

2. **Automated Annotation Percentage (Aut Ann):** Consists of the percentage of samples automatically annotated in relation to the total validation set size. Figure 4.7 displays the evolution on the percentage of the validation set unlabelled samples for the developed SSAL methods throughout the AL cycle iterations. In the AL and PL, the oracle annotates one sample per iteration, therefore, in Figure 4.7, both present an overlapping linear decline in the amount of unlabelled samples. The NN-SSAL methods annotate six samples per iteration, one by the oracle and five by the automatic annotator, therefore, these show in Figure 4.7 an overlapping linear decline with higher slope than AL and PL. On the other hand, rNN-SSAL presents a curved decline in the amount of unlabelled samples, outperforming the remaining during the first iterations. ST-SSAL displays during the initial iterations an amount of annotation similar to AL and PL, with only the expert annotator labelling new samples. ST-SSAL increase in the automatic annotation on latter iterations is explained by a 0.98 prediction confidence threshold required to automatically annotate unlabelled samples, only reached after the classifier's training set is significant and representative. On the whole, ST-SSAL attains the highest performance for the UCI dataset, and NN-SSAL for the CADL dataset, the latter closely followed by rNN-SSAL. Leading to the conclusion that, in larger datasets, such as the UCI, NN-SSAL automatic annotation becomes negligible in comparison to ST.



(a) UCI dataset. (b) CADL dataset.

Figure 4.7: Evolution on the percentage of the validation set unlabelled samples for the developed SSAL methods throughout the AL cycle iterations. Following the underscore in the NN and rNN methods: Euc, Cos, DTW and TAM, denote the similarity distances used in the respective method. Both the AL and PL, and the NN_euc, NN_cos, NN_dtw and NN_tam appear overlapping since one and five samples are removed, respectively, in every iteration from their classifier's training set.



(a) UCI dataset. (b) CADL dataset.

Figure 4.8: Percentage of correctly automatically annotated samples by the semi-supervised step throughout the AL iterations, for the developed SSAL methods, using the UCI and the CADL datasets. Following the underscore in the NN and rNN methods: Euc, Cos, DTW and TAM, denote the similarity distances used in the respective method. Both the AL and PL appear overlapping at the constant value of 0 throughout the iterations since they do not posses the semi-supervised step to perform the automatic annotation of unlabelled samples.

3. **Automated Annotation Accuracy (Ann Acc):** Consists of the percentage of correctly automatically annotated samples. Moreover, Figure 4.8 presents the evolution throughout the AL process of the automated annotation accuracy for the developed SSAL methods. As observed, ST-SSAL outperforms the remaining, attaining high results, specially for the latter iterations. ST-SSAL high annotation accuracy on the latter iterations results from the δ_{ST} threshold required for automatic annotation to be performed. This threshold is only reached during the latter iterations when the model training set becomes representative of the dataset, and predictions can be performed with high certainty. In contrast to the remaining methods, where higher results are obtained during the first iterations. For the NN-SSAL methods, this is justified by the queried sample propagating its label to closer samples during the first iterations. Whereas in the latter iterations, its closest neighbours start to be already annotated so the sample's label is given to further away samples. The same is applied to rNN-SSAL, with the stabilisation of the propagation accuracy being accompanied by the stabilisation of the amount of automatic propagation (Figure 4.7). Additionally, this metric allows to better discriminate between the performance of the different similarity measures. As can been seen, the *Euclidean distance* and *Cosine similarity* obtained competitive, similar results. In contrast to DTW and TAM, presenting a poor percentage of correctly annotated samples, explaining their low classification performance.

4. **Execution time:** AL shows the fastest execution time. The algorithm execution time, allows to favour between the different similarity measures, since the DTW and TAM expensive time and computational complexity, render those algorithms non-applicable to a viable solution. Furthermore, comparing the presented four distance metrics, *Euclidean distance* presents the lowest time expense and, therefore, was chosen as the most suitable distance metric.

As noted, generally speaking, ST-SSAL outperformed the remaining SSAL methods reaching an higher classification accuracy due to its good performance in the automatic annotation, as well in the amount of automatic annotation. rNN-SSAL, although annotating a substantial percentage of the dataset, its lower automatic annotation accuracy performance resulted in the decay of its classification accuracy.

For last, an analysis of the introduction of the density weight to an uncertainty-based sampling function was performed to observe if it would result in an increase of the amount of automatically labelled samples im the rNN method. Thus, for the *Local Density* * *Least Confident Sampling* (one of the best performing QSs in the previously-mentioned Section 4.4) and 50 iterations (the sufficient to attain stabilisation in both accuracy and the amount of automatically annotated samples according to Figure 4.6 and Figure 4.7). The former QS attained for the UCI and CADL dataset an amount of automatically annotated samples of 40 (5.4)% and 51 (3.3)%, respectively, against 30 (5.8)% and 48 (2.0)% obtained for the *Margin Sampling* QS. Thus, confirming the augment of the amount of

automatically annotated samples with the introduction of the density weight of approximately 10% and 3%, for both the above-mentioned datasets, respectively.

4.6 Stopping Criterion Analysis

In this section, it is analysed the introduction of a SC to the SSAL process. Previously, the presented results were based on a pre-defined number of queries (250 queries). The introduction of a SC allows to terminate the AL system early, optimising the computational demands of the pipeline.

As depicted in Figure 4.6, methods such as the rNN-SSAL and NN-SSAL using the *Euclidean distance*, after a reduced value of iterations, quickly reach their highest accuracy score, stabilising around that value for the forthcoming iterations. In contrast, methods such as AL, PL and ST-SSAL which require a larger labelled training set in order to reach a stable performance. Thus, as explained in Subsection 3.3.3, in order to optimise the trade-off between the classifier's performance and the expensive training set annotation cost, the number of iterations should be minimised according to the respective algorithm and dataset.

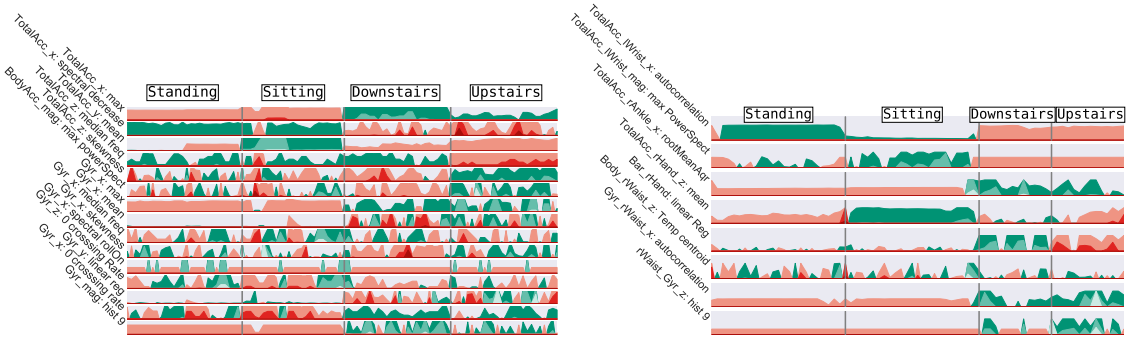
Hence, Table 4.6 presents for both datasets the experimental results for the developed SSAL methods using the proposed SCs methods in terms of: accuracy and total number of iterations. Moreover, in the columns SP, for both datasets it is shown the accuracy score for each method in stabilisation and below, the considered optimal stopping point. These values were selected in order to achieve a stable accuracy performance with the minimal annotation effort and higher coherency between different folds from the 10-CV (i.e. minimal standard deviation).

The most suitable SC is overall coherent between the different datasets and highly changes according to the SSAL algorithm. In Table 4.6, the number of iterations and, consequently, the required annotation cost was notably reduced. For the ST-SSAL, using the Over-CC SC, an accuracy of 84.5 (4.1)% and 84.7 (7.2)% was attained, with the annotation cost of 214.0 (46.5) and 182.0 (53.9) queries, for the UCI and CADL datasets, respectively, consisting of 2.4 (0.5)% and 10.2 (2.8)% of the validation set. Moreover, the automated annotation along with the manually annotated samples enabled to label 55.8 (11.8)% and 19.1 (13.4)% of the validation set with an accuracy on the automated annotation of 90.5 (4.6)% and 56.7 (46.3)%. Thereupon, the ST-SSAL method allowed to reduce the manual annotation cost on 97.6 (0.6)% and 89.8 (2.8)% for both datasets.

Additionally, it should be noticed that the obtained results are strongly influenced by the choice of the threshold in the formulation of the SC, which were selected in excess according to the 5 iterations windows, in order to allow the average of the last 5 values for evaluation purposes. Moreover, this value can be easily adapted according to the user's objectives.

For last, a confusion matrix for the ST-SSAL method using the Over-CC SC is presented in Figure 4.10, where it is possible to establish conclusions regarding the activities

correctly and incorrectly predicted by the classifier. For both datasets, the misclassification was higher between downstairs/upstairs and sitting/standing. The barometer's linear regression feature, as seen in Figure 4.9, presents high distinction between Downstairs/Upstairs, thus, allowed to improve the discrimination between these activities in the CADL dataset. Dynamic activities, due to its distinct motion characteristics and cyclic behaviour presented an overall clear discrimination against static activities.

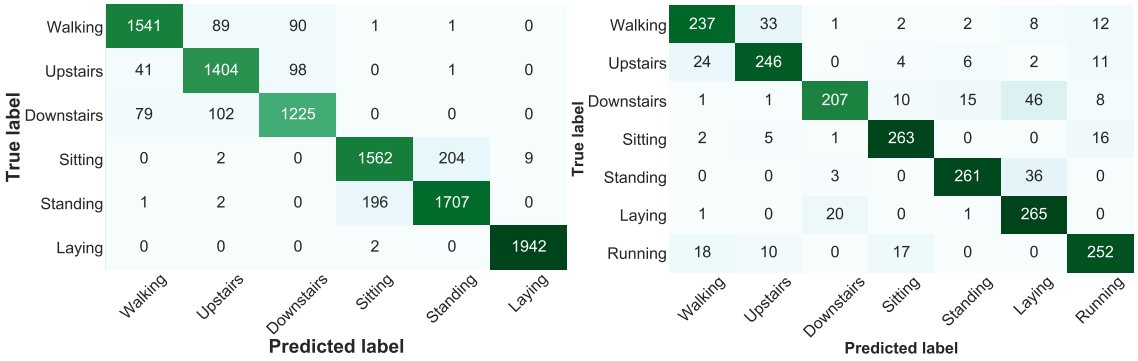


(a) UCI dataset.

(b) CADL dataset.

Figure 4.9: Horizon Plot showing the features' behaviour for the misclassified activities. In the y axis it is presented the information about the sensor (Acc: Accelerometer, Gyr: Gyroscope and Bar: Barometer), its signal axis (x, y, z and mag:vector magnitude) and the feature name. The green and red colours denote the signal's positive and negative values, respectively, with its intensity increasing with the feature's normalised absolute value and decreasing otherwise.

Lastly, comparing the performance results of the best performing method ST-SSAL method using the Over-CC SC, for the public UCI dataset, to state of art researches, namely, [23, 35, 45] who have achieved accuracies of 86%, 96% and 96%, respectively. When evaluating on the same test set, ST-SSAL obtained an accuracy of 83.2 (4.5)%, after 230.5 (21.9) queries. Therefore, although it did not outperform the aforementioned researches, satisfactory results were achieved, annotating 48.5 (18.1)% of the validation set with an accuracy of 88.0 (5.4)%, and a notable reduction of 96.8 (0.3)% in the training set annotation cost.



(a) UCI dataset. (b) CADL dataset.

Figure 4.10: Confusion matrix for the ST-SSAL method using the *Overall Uncertainty* SC.

Table 4.5: Experimental results for the SSAL methods: accuracy, automated annotation percentage, automated annotation accuracy and the algorithm's execution time. For all listed values it is shown its 10-CV average and standard deviation, the latter between parenthesis. Following the underscore in the NN and rNN methods: Euc, Cos, DTW and TAM, denote the similarity distances used in the respective method. The best performing algorithm is shown in bold for each dataset.

	UCI Dataset				CADL Dataset			
Method	**Accuracy in %**	**Aut Ann in %**	**Ann Acc in %**	**Time in s**	**Accuracy in %**	**Aut Ann in %**	**Ann Acc in %**	**Time in s**
NN_Euc	88.1 (2.7)	13.5 (0.1)	76.8 (1.0)	92.3 (9.4)	82.5 (5.7)	68.2 (2.7)	68.2 (1.0)	44.3 (4.8)
NN_Cos	89.4 (3.0)	13.5 (0.1)	75.3 (1.4)	860.9 (7.4)	82.8 (8.2)	68.2 (2.7)	64.6 (1.7)	79.1 (4.2)
NN_DTW	70.3 (6.0)	13.5 (0.1)	39.7 (1.7)	6772.0 (4.5)	68.4 (5.4)	68.2 (2.7)	30.3 (3.1)	6735.4 (1.3)
NN_TAM	75.2 (4.8)	13.5 (0.1)	44.1 (4.0)	6771.4 (5.4)	68.8 (7.3)	68.2 (2.7)	31.6 (1.2)	6735.4 (1.9)
rNN_Euc	85.4 (2.7)	33.3 (2.0)	77.3 (3.8)	437.9 (61.9)	74.9 (7.8)	56.6 (1.3)	66.5 (2.7)	60.0 (10.8)
rNN_Cos	82.5 (3.6)	37.7 (4.0)	74.7 (5.2)	1165.3 (86.4)	71.3 (8.1)	61.7 (2.5)	59.5 (5.6)	89.7 (11.2)
rNN_DTW	65.1 (4.9)	13.7 (1.6)	41.8 (2.9)	6995.4 (74.9)	77.3 (7.1)	20.8 (0.8)	39.9 (1.9)	6739.2 (7.6)
rNN_TAM	64.5 (8.4)	11.9 (3.9)	45.7 (4.9)	6968.3 (81.2)	81.8 (8.4)	11.2 (2.0)	41.3 (4.6)	6733.8 (7.3)
ST	**84.0 (6.3)**	**56.7 (11.6)**	**86.1 (10.5)**	**99.3 (17.5)**	**84.8 (7.0)**	**20.9 (6.9)**	**92.5 (2.7)**	**11.3 (1.0)**
AL	88.4 (2.8)			23.6 (5.0)	84.8 (7.0)			12.0 (1.0)
PL	88.0 (2.8)			23.2 (1.4)	82.8 (6.7)			10.3 (1.0)
SL	91.4 (2.4)			0.7 (0.1)	89.1 (4.0)			0.1 (0.1)
UL	57.8 (3.5)			0.3 (0.2)	61.9 (8.9)			0.1 (0.1)

Table 4.6: SC methods accuracy average, standard deviation, acc (std) in %, and average iterations over a 10-CV on the UCI and CADL datasets. Moreover, under the SP column, it is shown the accuracy results in stabilisation and below, the considered optimal stopping point. Most suitable method is shown in bold for each dataset.

		UCI						CADL					
Method		**SP**	**Max-Conf**	**Over-Unc**	**CC**	**Max-CC**	**Over-CC**	**SP**	**Max-Conf**	**Over-Unc**	**CC**	**Max-CC**	**Over-CC**
NN_Euc	Acc	85.5 (3.3)	69.2 (12.5)	78.7 (8.7)	73.6 (11.9)	**82.2 (8.2)**	81.5 (9.2)	81.6 (5.1)	68.2 (8.9)	76.7 (6.3)	57.3 (14.1)	77.9 (6.0)	**80.2 (6.4)**
	N.it	68.5 (26.1)	20.0 (11.4)	37.5 (9.5)	28.9 (12.4)	**68.0 (39.3)**	82.5 (44.1)	69.6 (38.4)	44.5 (23.4)	64.5 (14.3)	21.2 (8.2)	123.0 (97.5)	**103.0 (73.8)**
NN_Cos	Acc	79.6 (7.3)	70.3 (11.3)	79.0 (9.6)	68.8 (13.3)	81.5 (10.4)	**82.7 (10.8)**	74.6 (15.6)	62.8 (16.9)	69.8 (9.9)	44.2 (20.5)	74.9 (11.0)	**78.4 (10.5)**
	N.it	61.1 (29.7)	22.0 (13.2)	41.5 (11.9)	16.8 (7.6)	76.5 (47.5)	**67.5 (27.0)**	80.5 (31.4)	28.0 (10.0)	52.5 (15.8)	13.5 (6.8)	88.5 (65.1)	**88.5 (56.9)**
NN_DTW	Acc	67.0 (4.3)	54.3 (6.0)	47.8 (12.4)	57.6 (244.7)	**67.4 (7.4)**	58.1 (17.5)	58.7 (12.5)	43.9 (7.6)	33.8 (7.3)	52.0 (17.0)	**62.8 (4.0)**	64.5 (7.3)
	N.it	250.0 (0.0)	26.0 (7.4)	16.0 (2.2)	244.7 (155.0)	**297.7 (105.9)**	220.4 (158.8)	117.0 (95.1)	22.5 (4.5)	16.0 (5.5)	215.1 (134.5)	**278.3 (79.9)**	305.3 (11.2)
NN_TAM	Acc	77.0 (4.9)	59.5 (9.0)	**68.0 (7.9)**	63.8 (20.0)	73.1 (7.0)	67.3 (12.4)	62.0 (6.0)	49.5 (8.0)	**50.2 (8.4)**	60.9 (12.7)	63.9 (5.2)	65.8 (5.4)
	N.it	313.2 (78.3)	37.5 (17.0)	**53.5 (10.8)**	244.6 (153.2)	318.1 (95.7)	260.8 (136.7)	153.0 (86.3)	47.5 (23.9)	**47.5 (14.8)**	274.9 (90.0)	305.3 (11.2)	305.3 (11.2)
rNN_Euc	Acc	84.2 (2.3)	72.6 (12.4)	**83.7 (3.9)**	61.9 (13.4)	72.6 (13.9)	84.9 (2.7)	77.2 (5.9)	59.9 (14.5)	**76.9 (9.0)**	45.0 (10.8)	74.9 (7.0)	78.1 (5.4)
	N.it	92.5 (23.0)	37.5 (17.6)	**97.0 (37.6)**	15.9 (8.7)	45.5 (27.4)	129.4 (52.1)	198.3 (133.3)	32.5 (11.8)	**174.3 (97.2)**	14.0 (2.6)	209.6 (125.4)	319.1 (92.7)
rNN_Cos	Acc	65.5 (9.9)	**60.6 (11.6)**	66.0 (8.2)	52.7 (10.4)	65.9 (13.3)	72.0 (12.1)	52.7 (16.1)	53.4 (14.7)	**60.6 (11.6)**	34.2 (17.5)	53.7 (9.4)	60.5 (14.4)
	N.it	37.0 (10.4)	**31.0 (19.8)**	37.5 (15.2)	9.7 (3.5)	63.5 (71.9)	66.5 (37.6)	43.0 (21.5)	32.0 (11.6)	**72.5 (34.2)**	12.3 (5.6)	87.4 (101.3)	165.7 (136.7)
rNN_DTW	Acc	38.7 (15.4)	**44.6 (8.9)**	43.0 (6.8)	29.2 (9.7)	56.4 (13.9)	56.1 (14.6)	41.2 (14.8)	35.4 (6.9)	35.9 (6.3)	23.4 (5.8)	**56.9 (20.9)**	61.2 (18.2)
	N.it	39.0 (27.2)	**21.0 (5.9)**	31.5 (13.2)	7.4 (0.9)	156.6 (158.0)	133.2 (142.5)	51.0 (32.9)	25.0 (7.3)	30.0 (13.0)	9.3 (4.3)	**97.0 (94.2)**	132.0 (87.2)
rNN_TAM	Acc	68.6 (9.4)	43.1 (10.1)	48.2 (7.0)	40.5 (10.2)	**59.7 (16.7)**	50.1 (15.8)	56.9 (19.6)	29.8 (7.8)	27.7 (7.4)	33.1 (22.6)	**53.2 (19.0)**	47.9 (20.3)
	N.it	290.7 (118.6)	17.0 (3.7)	21.0 (5.5)	12.2 (7.9)	**124.3 (137.2)**	128.2 (146.3)	77.0 (34.4)	18.0 (4.6)	14.5 (5.5)	22.0 (24.7)	**54.0 (29.6)**	128.2 (146.2)
ST	Acc	85.2 (3.5)	48.8 (15.3)	66.3 (9.9)	49.1 (18.9)	75.3 (10.7)	**84.5 (4.1)**	82.8 (6.6)	33.0 (13.0)	61.9 (12.1)	15.0 (0.6)	76.2 (12.7)	**84.7 (7.2)**
	N.it	201.5 (83.8)	22.5 (9.5)	61.0 (29.2)	66.4 (107.1)	81.0 (41.6)	**214.0 (46.5)**	164.0 (70.1)	25.0 (10.2)	67.0 (21.1)	12.0 (0.0)	120.0 (28.4)	**182.0 (53.9)**
AL	Acc	86.1 (2.6)	60.3 (11.7)	65.7 (10.5)	37.4 (11.1)	**84.0 (4.7)**	86.0 (5.6)	84.6 (7.4)	51.2 (15.5)	64.0 (15.0)	15.0 (0.6)	**76.8 (9.7)**	76.6 (16.5)
	N.it	109.5 (24.4)	28.5 (10.1)	38.0 (16.4)	10.9 (3.9)	**98.0 (26.1)**	97.0 (40.8)	193.0 (51.0)	58.0 (17.9)	118.0 (59.5)	12.0 (0.0)	**116.0 (42.0)**	139.0 (43.1)
SL		91.4 (2.4)						89.1 (4.0)					
UL		57.8 (3.5)						61.9 (8.9)					

Chapter 5

Conclusion

The section starts by summarising the main achievements of this dissertation, presenting an overview of the proposed techniques and a summary of its most predominant results. Followed by the presentation of the next research steps for future work in Section 5.2.

5.1 Overall Achievements

Over the last years, the advances on smartphone and wearable technology allowed the proliferation of their use as unobstructive and pervasive sensors. The volume of the recorded data by these equipment's is significant and poses challenges on the development of traditional machine learning approaches that rely on annotated data. The process of annotating a large dataset requires a great effort by the manual annotation of an expert. Traditional HAR approaches rely on SL models which require a large amount of labelled data to guarantee accurate model performance. Based on the aforementioned challenges in the HAR context, this dissertation addressed a semi-automatic data annotation approach. Our method relies on two steps: (1) a QS criterion to select the most relevant samples to be labelled by an expert; (2) an automatic method to propagate the annotated sample's label over similar samples on the entire dataset.

Our main contribution consists of a comprehensive study of this approach in two HAR datasets using state-of-the-art QSs and SCs. These methods were evaluated over several SSAL strategies based on different distance functions to build an optimal SSAL system with applications for human movement.

In order to accomplish this task, two datasets were used and allowed to verify the feasibility of the proposed framework: the UCI HAR public dataset and a new dataset obtained in the context of this dissertation composed by 12 subjects.

Regarding the developed framework: a *Forward Feature Selection* method was implemented, able to search for the optimal features set that best distinguishes the different activities, to be given as input to the classifier. The developed algorithm was able to reduce the number of features from the UCI dataset from 480 to 15, and from 1960 to 8 in the Fraunhofer dataset. Thus, improving both accuracy and the algorithm's time and computational performance.

The resulting feature vector was used as input for the classification process. Where two frameworks were developed in which a learner is firstly initialised with a minimal labelled training set, consisting of one sample per label:

- **Active Learning framework:** that attained an accuracy of 84.0 (4.7)% and 76.8 (9.7)% for the UCI and Fraunhofer datasets, respectively. Thus, having into account that SL attained an accuracy of 91.4 (2.4)% and 89.1 (5.0)% for the aforementioned datasets, respectively, with the annotation of 100% of the validation set. It is possible to conclude that, the AL process was able to reach competitive performance results to SL with a decrease in the amount of labelled data of 98.8 (0.6)% and 92.7 (2.5)%. Thus, presenting a notable decrease in the annotation effort required to obtain a stable classifier performing HAR.

- **Semi-Supervised Active Learning framework:** appending a semi-supervised step to the aforementioned AL process in order to automate the annotation process, through an automatic annotation of unlabelled data. Thus, creating a larger labelled set, more representative of the entire dataset with no additional annotation effort from the user.

  For this step, three semi-supervised methods were implemented: NN-SSAL, RNN-SSAL and *Self-Training*. Moreover, regarding the first two, four different similarity measures were used to infer the distance between different samples: *Euclidean distance*, *Cosine similarity*, *DTW* and *TAM*.

  If we compare ST-SSAL and AL, both methods achieve similar classification performance. However, ST-SSAL was able to annotate a higher volume of data with similar annotation effort, without compromising the classification accuracy. This study extends the work conducted by [13] on HAR, since it applies ST on the labels previously selected by AL. The ST-SSAL using the Overall Uncertainty and Classification-Change (Over-CC) SC obtained an accuracy of 84.5 (4.1)% and 84.7 (7.2)% for the UCI and CADL datsets, respectively, with a reduction in the Oracle annotation effort on 97.6 (0.6)% and 89.8 (2.8)% of total number of samples for both datasets.

Moreover, in the construction of the AL process, two core considerations were analysed:

- **Query Strategy**: where eight QSs were tested: *Least Confident Sampling*, *Margin Sampling*, *Entropy Sampling*, *Local Density Sampling*, *Uncertainty and Local Density Sampling*, *Margin and Local Density Sampling*, *Entropy and Local Density Sampling* and *Random Sampling*. All things considered, *Margin Sampling* was selected as the most suitable QS, due to its low time, computational complexity, and accuracy results, surpassing PL (*Random Sampling*). Additionally, overall, the introduction of the density weight resulted in a slight increase in the classifier's accuracy and percentage of automatically annotated samples. However, at the cost of considerable time and computational cost, required for the calculation of each sample's density.

- **Stopping Criterion**: where four stropping criteria were presented: *Max-Confidence SC*, *Overall Uncertainty SC*, *Classification-Change SC* and two SC combining the precedent strategies.

  Globally, the most suitable SC was coherent between the different datasets and highly changed accordingly with the SSAL method. Notwithstanding, undoubtedly, with the introduction of the SC to the AL process, the number of iterations the AL performed, and consequently, the manual annotation effort required from the user was significantly reduced.

To conclude, comparing the performance results of the best performing method ST-SSAL method using the , for the public UCI dataset, to state of art researches, namely, [23, 35, 45] who have achieved accuracies of 86%, 96% and 96%, respectively. When evaluating on the same test set, ST-SSAL obtained an accuracy of 83.2 (4.5)%, after 230.5 (21.9) queries. Therefore, although it did not outperform the aforementioned researches, satisfactory results were achieved, annotating 48.5 (18.1)% of the validation set with an accuracy of 88.0 (5.4)%, and a notable reduction of 96.8 (0.3)% in the training set annotation cost, completing with success the main objective of this dissertation.

5.2 Future Work

The following paragraphs describe some topics that could be explored in a future research:

- Utilisation of a multi-oracle system with non-expert users and evaluation of the system response to the possible integration of bias in the AL annotation process.
- Development of the annotation cost estimation for different samples. Since, different samples may present different annotation costs according to the sample characteristics, the oracle expertise on the dataset and the mode of annotation.
- Enlargement of the Fraunhofer dataset, with more volunteers performing the activities for a greater timespan. Acquisition of a broader range of activities, with the incorporation of complex activities, allowing the recognition of a complete detailed daily monitoring. Where a larger dataset is linked to a more versatile classifier, able to generalise into different users and achieve more reliable results.
- Integration of a device position independent algorithm into the proposed method, enabling the user to fix the acquisition device in any desired location.
- Development of an annotation interface, with the adaptation of the proposed algorithm to an Android environment.
- Computational optimisation of DTW and TAM distance metrics, and density computation so both distance metrics and QSs integrating the local density can become competitive.

Bibliography

[1] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers. "A review of wearable sensors and systems with application in rehabilitation." In: *Journal of NeuroEngineering and Rehabilitation* 9.1 (2012), p. 21. ISSN: 1743-0003. DOI: 10.1186/1743-0003-9-21. URL: https://doi.org/10.1186/1743-0003-9-21.

[2] S. Ramasamy Ramamurthy and N. Roy. "Recent trends in machine learning for human activity recognition survey." In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4 (2018), pp. 1–11. ISSN: 19424795. DOI: 10.1002/widm.1254. URL: http://mpsc.umbc.edu/wp-content/uploads/2018/05/A-survey_Wiley18.pdf.

[3] W. H. Organisation. *Global Strategy on Diet, Physical Activity and Health*. 2018. URL: http://www.who.int/dietphysicalactivity/pa/en/ (visited on 08/10/2018).

[4] J. R. C. Silva. "Smartphone-Based Human Activity Recognition." Master's thesis. Faculdade de Engenharia, Universidade do Porto, 2014, p. 147. ISBN: 978-3-319-14273-9. DOI: 10.1007/978-3-319-14274-6.

[5] C. Figueira, R. Matias, and H. Gamboa. "Body Location Independent Activity Monitoring." Master's thesis. Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2016, pp. 190–197. ISBN: 978-989-758-170-0. DOI: 10.5220/0005699601900197. URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005699601900197.

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

[7] Altexsoft. *How to Organize Data Labeling for Machine Learning: Approaches and Tools*. 2018. URL: https://www.altexsoft.com/blog/datascience/how-to-organize-data-labeling-for-machine-learning-approaches-and-tools/ (visited on 08/26/2018).

[8] Eurostat. *Monthly minimum wages - bi-annual data*. Tech. rep. 2018. URL: http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=earn_mw_cur&lang=en.

[9] *Fraunhofer AICOS*. URL: https://www.fraunhofer.pt/en/fraunhofer_portugal/about_us.html (visited on 09/21/2018).

[10] F. Shahmohammadi, A. Hosseini, C. E. King, and M. Sarrafzadeh. "Smartwatch Based Activity Recognition Using Active Learning." In: *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. 2017, pp. 321–329. ISBN: 978-1-5090-4722-2. DOI: 10.1109/CHASE.2017.115. URL: http://ieeexplore.ieee.org/document/8010670/.

[11] R. Liu, T. Chen, and L. Huang. "Research on human activity recognition based on active learning." In: *Machine Learning and Cybernetics (ICMLC), 2010 International Conference on (Volume:1)*. July. 2010, pp. 285 –290. ISBN: 9781424465279. DOI: 10.1109/ICMLC.2010.5581050. URL: https://ieeexplore.ieee.org/document/5581050/.

[12] C. C. Aggarwal, X Kong, Q Gu, J Han, and P. S. Yu. *Active Learning : A Survey*. Tech. rep. 2014, pp. 571–605. URL: http://www.charuaggarwal.net/active-survey.pdf.

[13] M. Stikic, K. Van Laerhoven, and B. Schiele. *Exploring semi-supervised and active learning for activity recognition*. Tech. rep. 2008, pp. 81–88. DOI: 10.1109/ISWC.2008.4911590. URL: http://ieeexplore.ieee.org/document/4911590/.

[14] H. Alemdar, T. van Kasteren, and C. Ersoy. "Using Active Learning to Allow Activity Recognition on a Large Scale." In: *Ambient Intelligence* 7040 (2011), pp. 105–114. DOI: 10.1007/978-3-642-25167-2_12. URL: https://link.springer.com/chapter/10.1007/978-3-642-25167-2_12.

[15] H. M. Hossain, M. A.A. H. Khan, and N. Roy. "Active learning enabled activity recognition." In: *Pervasive and Mobile Computing*. Vol. 38. 2017, pp. 312–330. ISBN: 9781467387798. DOI: 10.1016/j.pmcj.2016.08.017. URL: http://mpsc.umbc.edu/wp-content/uploads/2015/04/PerCom16.pdf.

[16] J. Zhu, H. Wang, B. K. Tsou, and M. Ma. "Active learning with sampling by uncertainty and density for data annotations." In: *IEEE Transactions on Audio, Speech and Language Processing* 18.6 (2010), pp. 1323–1331. ISSN: 15587916. DOI: 10.1109/TASL.2009.2033421. URL: https://ieeexplore.ieee.org/document/5272205/.

[17] J. Zhu, H. Wang, T. Yao, and B. K. Tsou. "Active Learning with Sampling by Uncertainty and Density for Word Sense Disambiguation and Text Classification." In: *Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08)* August (2008), pp. 1137–1144. ISSN: 1558-7916. DOI: 10.1109/TASL.2009.2033421. URL: http://dl.acm.org/citation.cfm?id=1599081.1599224.

[18] G. He, Y. Duan, Y. Li, T. Qian, J. He, and X. Jia. "Active Learning for Multivariate Time Series Classification with Positive Unlabeled Data." In: *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)* (2015), pp. 178–185. ISSN: 10823409. DOI: 10.1109/ICTAI.2015.38. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7372134.

[19] G. He, Y. Li, and W. Zhao. "An uncertainty and density based active semi-supervised learning scheme for positive unlabeled multivariate time series classification." In: *Knowledge-Based Systems* 124 (2017), pp. 80–92. ISSN: 09507051. DOI: 10.1016/j.knosys.2017.03.004. URL: https://www.sciencedirect.com/science/article/pii/S0950705117301235.

[20] Android. *Motion Sensors*. 2016. URL: http://developer.android.com/guide/topics/sensors/sensors_motion.html (visited on 07/26/2018).

[21] J. J. Kavanagh and H. B. Menz. "Accelerometry: A technique for quantifying movement patterns during walking." In: *Gait and Posture* 28.1 (2008), pp. 1–15. ISSN: 09666362. DOI: 10.1016/j.gaitpost.2007.10.010. URL: https://www.ncbi.nlm.nih.gov/pubmed/18178436.

[22] A. L. Gomes. "Human Activity Recognition with Accelerometry : Novel Time and Frequency Features." Master's thesis. Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2014.

[23] J. Silva, M. Monteiro, and F. Sousa. "Human activity classification with inertial sensors." In: *Studies in Health Technology and Informatics*. Vol. 200. 2014, pp. 101–104. ISBN: 9781614993926. DOI: 10.3233/978-1-61499-393-3-101. URL: https://www.ncbi.nlm.nih.gov/pubmed/24851971.

[24] I. P. Machado. "Human Activity Data Discovery based on Accelerometry." Master's thesis. Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2013.

[25] J. K. Aggarwal. *Human Activity Recognition*. 2005, pp. 39–39. ISBN: 9781466588288. DOI: 10.1007/11590316_6. URL: http://link.springer.com/10.1007/11590316_6.

[26] D. J. Jacob. *Chapter 2. Atmospheric Pressure*. 1999, pp. 12–20. ISBN: 978-0691001852. DOI: 10.1111/j.0954-6820.1949.tb11329.x. URL: http://acmg.seas.harvard.edu/publications/jacobbook/bookchap2.pdf.

[27] K. Sankaran, M. Zhu, X. F. Guo, A. L. Ananda, M. C. Chan, and L.-S. Peh. "Using mobile phone barometer for low-power transportation context detection." In: *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems - SenSys '14* (2014), pp. 191–205. DOI: 10.1145/2668332.2668343. URL: http://dl.acm.org/citation.cfm?doid=2668332.2668343.

[28] A. Munoz. "Machine Learning and Optimization." In: *Courant Institute of Mathematical Sciences* (2014), pp. 1–2. URL: https://cims.nyu.edu/~munoz/files/ml_optimization.pdf.

[29] R. Leonardo. "Contextual information based on pervasive sound analysis." Master's thesis. Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2017.

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

[31] A. Ng. *1. Supervised learning*. Tech. rep. 2012, pp. 1–30. DOI: 10.1111/j.1466-8238.2009.00506.x. arXiv: arXiv:1011.1669v3.

[32] C. Olivier, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. Vol. 1. 2. 2006, p. 524. ISBN: 9780262033589. DOI: 10.1007/s12539-009-0016-2. arXiv: arXiv:1011.1669v3.

[33] D. S. Gomes. "Eating and drinking recognition for triggering smart reminders." Master's thesis. Faculdade de Engenharia, Universidade do Porto, 2017.

[34] B. Settles. *Active Learning Literature Survey*. Tech. rep. 2. 2010, pp. 201–221. DOI: 10.1.1.167.4245. arXiv: 1206.5533. URL: http://burrsettles.com/pub/settles.activelearning.pdf.

[35] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. "A Public Domain Dataset for Human Activity Recognition Using Smartphones." In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. April. Bruges, Belgium, 2013, pp. 24–26. ISBN: 9782874190810. URL: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2013-84.pdf.

[36] X. Zhu and A. B. Goldberg. *Introduction to Semi-Supervised Learning*. Vol. 3. 1. 2009, pp. 1–130. ISBN: 9781598295474. DOI: 10.2200/S00196ED1V01Y200906AIM006. arXiv: 1412.6596. URL: http://www.morganclaypool.com/doi/abs/10.2200/S00196ED1V01Y200906AIM006.

[37] *Introduction to Machine Learning*. URL: https://developers.google.com/machine-learning/crash-course/ml-intro (visited on 08/01/2018).

[38] Y. Fu, X. Zhu, and B. Li. "A survey on instance selection for active learning." In: *Knowledge and Information Systems* 35.2 (2013), pp. 249–283. ISSN: 02191377. DOI: 10.1007/s10115-012-0507-8. URL: https://link.springer.com/content/pdf/10.1007%2Fs10115-012-0507-8.pdf.

[39] Y. Chen, S. Mani, and H. Xu. "Applying active learning to assertion classification of concepts in clinical text." In: *Journal of Biomedical Informatics* 45.2 (2012), pp. 265–272. ISSN: 15320464. DOI: 10.1016/j.jbi.2011.11.003. arXiv: NIHMS150003. URL: https://www.sciencedirect.com/science/article/pii/S1532046411001912?via%3Dihub.

[40] J. Zhu, H. Wang, and E. Hovy. "Multi-criteria-based strategy to stop active learning for data annotation." In: *Proceedings of the 22nd International Conference on Computational Linguistics* August (2008), pp. 1129–1136. DOI: 10.3115/1599081.1599223. URL: http://portal.acm.org/citation.cfm?doid=1599081.1599223.

[41] M. Bloodgood and K. Vijay-Shanker. "A Method for Stopping Active Learning Based on Stabilizing Predictions and the Need for User-Adjustable Stopping." In: June (2014), pp. 39–47. arXiv: 1409.5165. URL: http://arxiv.org/abs/1409.5165.

[42] D. Folgado. "Measuring Repetitive Tasks using Inertial Sensors." Master's thesis. Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2015.

[43] F. Iglesias and W. Kastner. "Analysis of Similarity Measures in Times Series Clustering for the Discovery of Building Energy Patterns." In: *Energies* 6.2 (2013), pp. 579–597. ISSN: 19961073. DOI: 10.3390/en6020579. URL: https://www.mdpi.com/1996-1073/6/2/579.

[44] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. L. Reyes-Ortiz, M. KOSE, O. D. INCEL, C. ERSOY, J. L. Reyes-Ortiz, A. Ghio, D. Anguita, X. Parra, J. Cabestany, A Catal, A. Stisen, H. Blunck, S. Bhattacharya, and T. S. Prentow. "Human activity and motion disorder recognition: Towards smarter interactive cognitive environments." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 19. April. 2013, pp. 24–26. ISBN: 9782874190810. DOI: 978-1-4503-1227-1. URL: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2013-11.pdf.

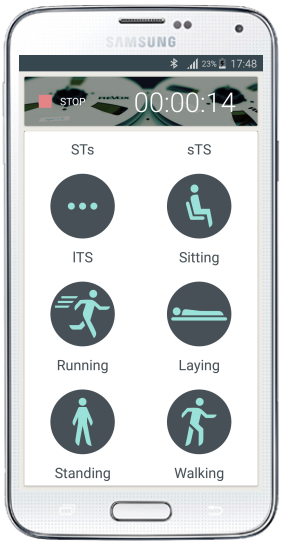
[45] Romera-Paredes, M. Aung, and N. Bianchi-Berthouze. "A One-vs-One Classifier Ensemble with Majority Voting for Activity Recognition." In: *Esann 2013*. April. 2013, pp. 24–26. ISBN: 9782874190810. URL: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2013-124.pdf.

Appendix A

Acquisition Devices

In this appendix are presented the acquisition devices used in the construction of the CADL dataset. Namely, in Figure A.1b, the IoTiop wearable device, developed by Fraunhofer, which the user wore on the left wrist, right ankle and right side of the waist and in Figure A.1a, the smartphone, a Samsung S5, showing the acquisition application (*IoTip Recorder*) which the user kept on the right hand to annotate its current activity.

(a) Fraunhofer Iotip device



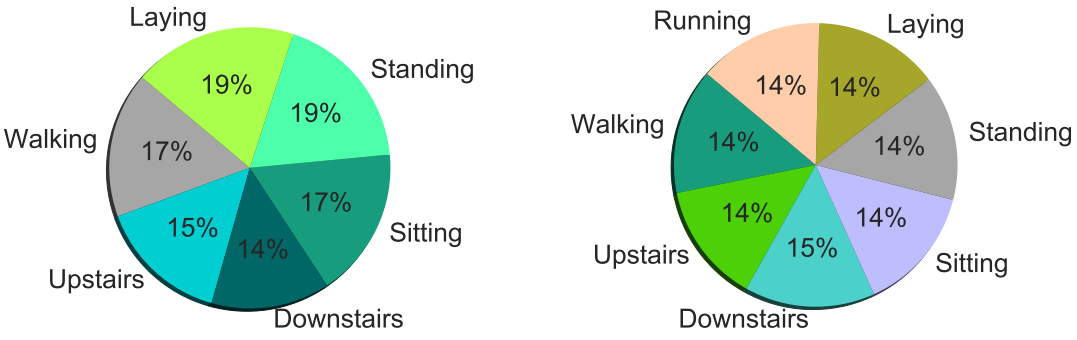
(b) Smartphone Samsung S5 showing the IoTip Recorder application

Figure A.1: IoTip device in Figure A.1a and the Smartphone Samsung S5 with *IoTip Recorder* app in Figure A.1b.

Moreover, the activities were performed in a controlled environment, with the volunteers performing the following acquisition protocol.

1. **Laying:** the subject should lay down up, still, without any interruption for a period of 2mins in any direction at his preferred speed.
2. **Standing:** the subject should stand still without any interruption for a period of 2mins.
3. **Sitting:** the subject should sit still without any interruption for a period of 2mins.
4. **Running:** the subject should run without any interruption for a period of 2mins in any direction at his preferred speed.
5. **Walking:** the subject should walk without any interruption for a period of 2mins in any direction at his preferred speed.
6. **Walking Upstairs:** the subject should walk upstairs without stopping between floors for a period of 2mins in any direction at his preferred speed.
7. **Walking Downstairs:** the subject should walk downstairs without stopping between floors for a period of 2mins in any direction at his preferred speed.

Each activity was set to be performed for 2mins since it was important to obtain a homogeneous dataset so the classifier would not be biased towards a certain activity. This fact is verified by the pie charts below, showing the percentage of samples in each activity in relation to the entire dataset.



(a) UCI dataset

(b) CADL dataset

Figure A.2: Percentage of samples belonging to each performed activity in relation to the entire dataset for the UCI dataset in Figure A.2a and for the CADL dataset in Figure A.2b.

Appendix B

Algorithms

In this appendix is presented additional algorithm's pipelines referred throughout the current dissertation. Starting by the algorithm's pipelines focusing in bold on the four developed stopping criteria, described in more detail in Subsection 3.3.3: *Max-Confidence SC*, *Overall Uncertainty SC*, *Classification-Change SC* and *Combination Strategy SC*. Together with the algorithm's pipelines for the *SSAL* framework, described in Section 3.4: *Self-Training*, *NN-SSAL* and *rNN-SSAL*.

Algorithm 7 Active Learning Applying The Max-Confidence SC

Input: initial train set L, unlabelled validation set U, independent test set T
Output: predicted labels for the test set

$\theta \leftarrow clf.fit(L)$ ▷ Learns model on initial training set
while $\mu_{LC}(A)$ - $\mu_{LC}(B) > \delta\mu_{MC}$ and $\sigma_{LC}(A)$ - $\sigma_{LC}(B) > \delta\sigma_{MC}$ **do**
    Selection by Q, of most informative sample: x^*
    Ask Oracle for x^*'s label
    $L \leftarrow L \cup x^*$ ▷ Augments model training set with x^*
    $U \leftarrow U \setminus x^*$ ▷ Removes x^* from unlabelled samples
    $\Theta \leftarrow clf.fit(L)$ ▷ Updates model
    Return $clf.predict(T)$ ▷ Returns predicted labels for the test set
end while

Algorithm 8 Active Learning Applying The Overall Uncertainty SC

Input: initial train set L, unlabelled validation set U, independent test set T
Output: predicted labels for the test set

$\theta \leftarrow clf.fit(L)$ ▷ Learns model on initial training set
while $\mu_{OU}(A)$ - $\mu_{OU}(B) > \delta\mu_{OU}$ and $\sigma_{OU}(A)$ - $\sigma_{OU}(B) > \delta\sigma_{OU}$ **do**
    Selection by Q, of most informative sample: x^*
    Ask Oracle for x^*'s label
    $L \leftarrow L \cup x^*$ ▷ Augments model training set with x^*
    $U \leftarrow U \setminus x^*$ ▷ Removes x^* from unlabelled samples
    $\Theta \leftarrow clf.fit(L)$ ▷ Updates model
    Return $clf.predict(T)$ ▷ Returns predicted labels for the test set
end while

Algorithm 9 Active Learning Applying The Classification-Change SC

Input: initial train set L, unlabelled validation set U, independent test set T
Output: predicted labels for the test set

$\theta \leftarrow clf.fit(L)$ ▷ Learns model on initial training set
while accuracy_score($clf.predict_k$, $clf.predict_{k+1}$) $< \delta_{F1\ score}$ **do**
    Selection by Q, of most informative sample: x^*
    Ask Oracle for x^*'s label
    $L \leftarrow L \cup x^*$ ▷ Augments model training set with x^*
    $U \leftarrow U \setminus x^*$ ▷ Removes x^* from unlabelled samples
    $\Theta \leftarrow clf.fit(L)$ ▷ Updates model
    Return $clf.predict(T)$ ▷ Returns predicted labels for the test set
end while

Algorithm 10 Active Learning Applying The Combination Strategy SC

Input: initial train set L, unlabelled validation set U, independent test set T
Output: predicted labels for the test set

$\theta \leftarrow clf.fit(L)$ ▷ Learns model on initial training set
while f_1 not True and f_2 not True **do**
    Selection by Q, of most informative sample: x^*
    Ask Oracle for x^*'s label
    $L \leftarrow L \cup x^*$ ▷ Augments model training set with x^*
    $U \leftarrow U \setminus x^*$ ▷ Removes x^* from unlabelled samples
    $\Theta \leftarrow clf.fit(L)$ ▷ Updates model
    Return $clf.predict(T)$ ▷ Returns predicted labels for the test set
end while

Algorithm 11 Self-Training

Input: initial train set L, unlabelled validation set U, independent test set T
Output: predicted labels for the test set

$\Theta \leftarrow clf.fit(L)$ ▷ Learns model on initial training set
while SC not met **do**
    Selection by Q, of most informative sample: x^*
    Ask Oracle for x^*'s label
    $L \leftarrow L \cup x^*$ ▷ Augments classifier training set with x^*
    $U \leftarrow U \setminus x^*$ ▷ Removes x^* from unlabelled samples
    $\Theta \leftarrow clf.fit(L)$ ▷ Updates model
    for x in U **do**
        if $P_\theta(\hat{y}|x)) >= 0.98$ **then** ▷ Finds confidently predicted samples C in U
            $C \leftarrow C \cup x$
        end if
    end for
        $L \leftarrow L \cup C$ ▷ Augments model training set with C
        $U \leftarrow U \setminus C$ ▷ Removes C from unlabelled samples
        $\Theta \leftarrow clf.fit(L)$ ▷ Updates model
        Return $clf.predict(T)$ ▷ Returns predicted labels for the test set
end while

Algorithm 12 5-Nearest Neighbour Semi-Supervised Active Learning

Input: initial train set L, unlabelled validation set U, independent test set T
Output: predicted labels for the test set

$\Theta \leftarrow clf.fit(L)$ ▷ Learns model on initial training set
while SC not met **do**
    Selection by Q, of most informative sample: x^*
    Ask Oracle for x^*'s label
    $L \leftarrow L \cup x^*$ ▷ Augments classifier training set with x^*
    $U \leftarrow U \setminus x^*$ ▷ Removes x^* from unlabelled samples
    $\Theta \leftarrow clf.fit(L)$ ▷ Updates model
    for x in $x^{*\prime}$ 5-NN **do** ▷ Adds $x^{*\prime}$ 5-NN to C
        $C \leftarrow C \cup x$
    end for
        $L \leftarrow L \cup C$ ▷ Augments model training set with C
        $U \leftarrow U \setminus C$ ▷ Removes C from unlabelled samples
        $\Theta \leftarrow clf.fit(L)$ ▷ Updates model
        Return $clf.predict(T)$ ▷ Returns predicted labels for the test set
end while

Algorithm 13 1-reverse-Nearest Neighbour Semi-Supervised Active Learning

Input: initial train set L, unlabelled validation set U, independent test set T
Output: predicted labels for the test set

$\Theta \leftarrow clf.fit(L)$ ▷ Learns model on initial training set
while SC not met **do**
    Selection by Q, of most informative sample: x^*
    Ask Oracle for x^*'s label
    $L \leftarrow L \cup x^*$ ▷ Augments classifier training set with x^*
    $U \leftarrow U \setminus x^*$ ▷ Removes x^* from unlabelled samples
    $\Theta \leftarrow clf.fit(L)$ ▷ Updates model
    for x in U **do** ▷ Iterate over unlabelled samples
        if x^* is x's labelled 1-NN **then** ▷ Adds samples to which x^* is its NN
            $C \leftarrow C \cup x$
        end if
    end for
        $L \leftarrow L \cup C$ ▷ Augments model training set with C
        $U \leftarrow U \setminus C$ ▷ Removes C from unlabelled samples
        $\Theta \leftarrow clf.fit(L)$ ▷ Updates model
        Return $clf.predict(T)$ ▷ Returns predicted labels for the test set
end while

Appendix C

Horizon Plot

This appendix shows the obtained Horizon Plots for the UCI and CADL dataset, respectively.

Furthermore, the Horizon Plots allow to visualise the behaviour of the best features sets along the protocol activities, obtained for one of the dataset users.

Moreover, on top, the name of the activities are shown, while on the left, the name of each feature is shown per row. Thus, each row depicts the performance of each feature along every activity. The x, y, z and mag in each feature name, denotes the axis to which it corresponds to: the x axis, y axis, z axis or to the signal magnitude.

The green and red values, denote positive and negative values, respectively, with the colour becoming darker with an increase of the feature's absolute value and lighter otherwise.

On this account, the following Horizon Plots confirm the utility of the presented features, as they change significantly along each activity, thus, allowing a confident discrimination between activities.

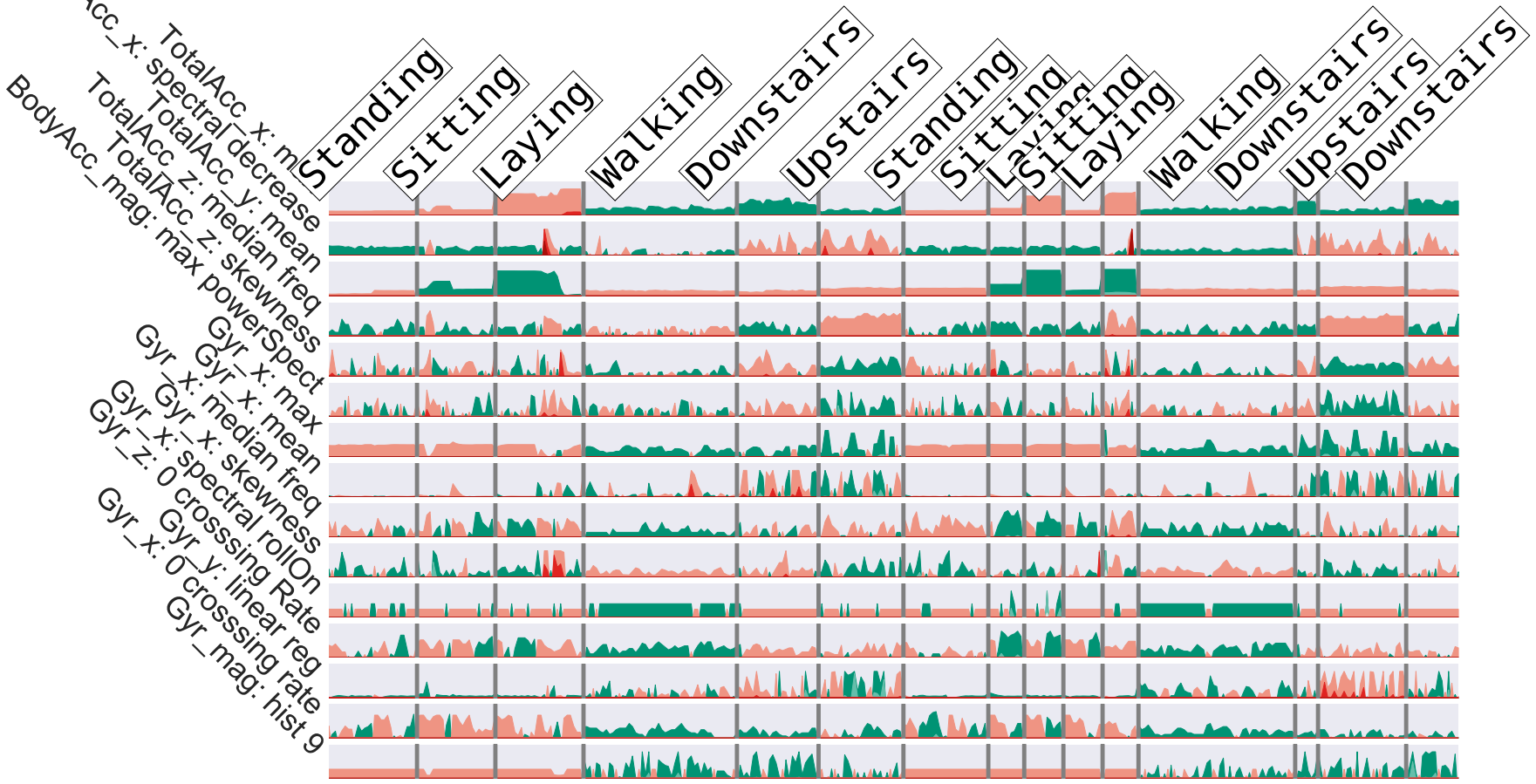


Figure C.1: Horizon Plot for the UCI dataset, allowing to visualise the behaviour of the best features set, shown in the y axis, along the protocol activities, on top. The x, y, z and *mag* in each feature name, denote the axis to which it corresponds to: the *x axis*, y axis, z axis or to the signal magnitude. The green and red values, denote positive and negative values, respectively, with the colour becoming darker with an increase of the feature's absolute value and lighter otherwise.

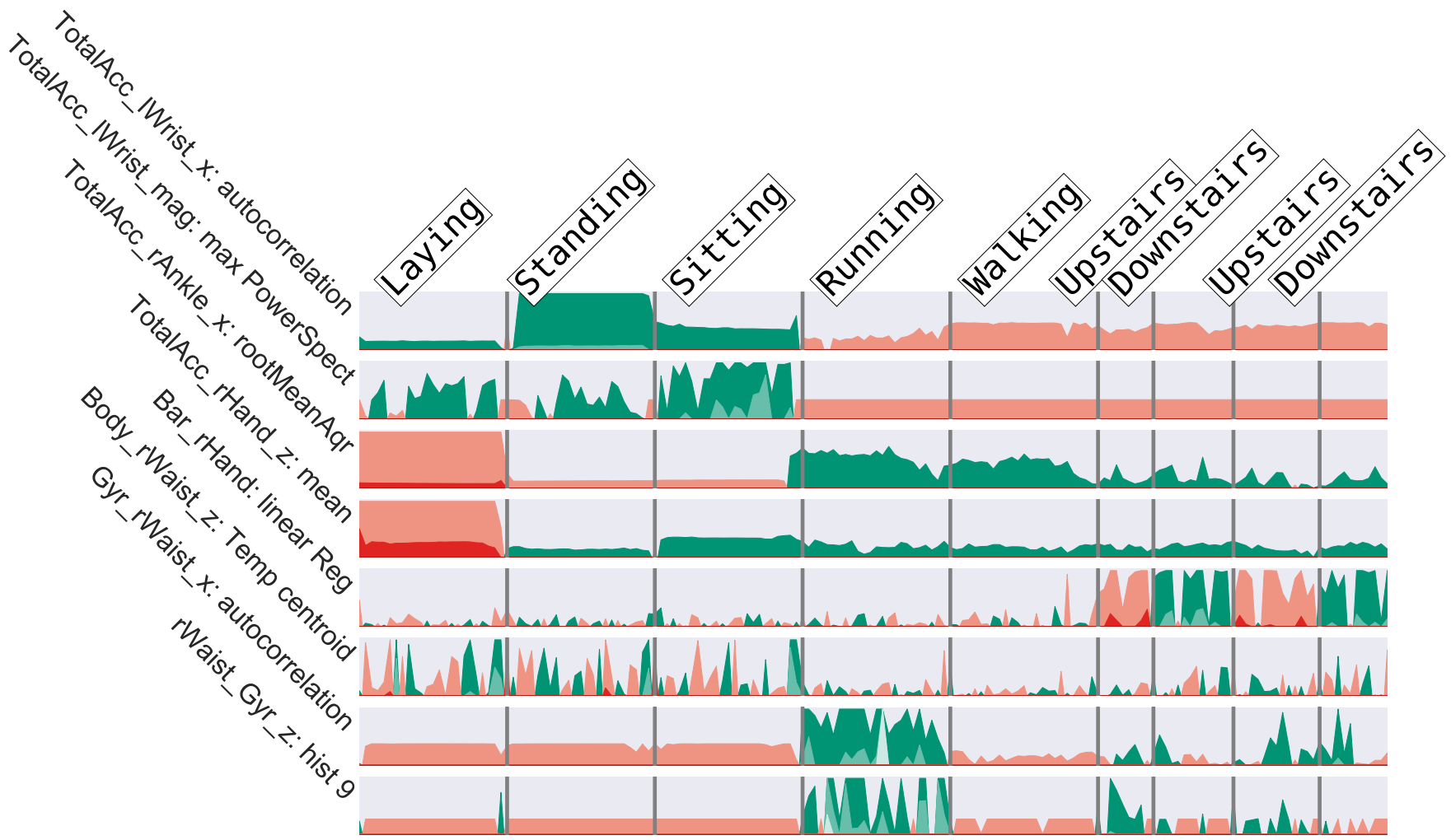


Figure C.2: Horizon Plot for the CADL dataset, allowing to visualise the behaviour of the best features set, shown in the y axis, along the protocol activities, on top. The x, y, z and *mag* in each feature name, denote the axis to which it corresponds to: the x *axis*, y axis, z axis or to the signal magnitude. The green and red values, denote positive and negative values, respectively, with the colour becoming darker with an increase of the feature's absolute value and lighter otherwise.