

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Automatic Switching Between Video and Audio According to User's Context

Paulo Jorge Silva Ferreira



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: João Manuel Paiva Cardoso

Co-Supervisor: João Pedro Mendes Moreira

March 7, 2019

Automatic Switching Between Video and Audio According to User's Context

Paulo Jorge Silva Ferreira

Mestrado Integrado em Engenharia Informática e Computação

March 7, 2019

Abstract

Smartphones are increasingly present in human's life. These devices can be used for the most diverse tasks, such as communications, work or entertainment. For example, in the entertainment area many people use their smartphone to watch videos or listen to music. Many users stream or play videos with the intention to only listen to the audio track. These videos might be stored on smartphone memory or on a video streaming platform such as YouTube. In this way, important resources of the smartphone, such as battery power, bandwidth, and data traffic that are limited to most users, are unnecessarily consumed. Recognizing human activities with sensors allows the development of applications for many diverse areas, from medical health supervision to entertainment. Initially, these systems involved the use of various sensors scattered throughout the user's body, but with the increased processing power of smartphones and built-in sensors, it was possible using only the smartphone to recognize activities. The sensors embedded in most smartphones, such as accelerometer, gyroscope, among others, have allowed the emergence of multiple possibilities for a new type of applications based on the context acquired from the data they provide. With this dissertation we intend to develop a method that, based on the user context, can automatically switch between video and audio, reducing energy consumption. This would increase the time of use of the smartphone between battery recharges, and would solve the problem of unnecessary consumption of smartphone resources. This project aims to investigate the accuracy with which the context of the user is recognized, the method that obtains the best accuracy and the overhead that this system can have on the smartphone, to recognize if the user is watching a video or simply listening to the audio, analyze energy consumption and finally switches automatically between video and audio. In addition to these goals, we also intend to develop an Android application to test and evaluate the approach. A Supervised Learning approach is used along with the classifiers K-Nearest Neighbors, Hoeffding Trees and Naive Bayes, individually and combined to create an ensemble classifier. The dataset used was collected by the author and an overall accuracy of 93.54% was obtained for the ensemble classifier. Individually, the kNN obtained the best results when compared with the other classifiers used, obtaining an overall accuracy of 92.53%.

Keywords: Human activity recognition (HAR), mobile sensors, machine learning, mobile application, Supervised Learning, smartphone, kNN, Hoeffding Trees, Naive Bayes

Resumo

Os smartphones estão cada vez mais presentes no dia-a-dia das pessoas. Estes dispositivos podem ser usados para as mais diversas tarefas, tais como comunicações, trabalho ou entretenimento. Por exemplo, na área do entretenimento muitas pessoas utilizam o smartphone para ver vídeos ou ouvir música. Muitos utilizadores fazem stream ou reproduzem vídeos apenas com a intenção de ouvir a faixa de áudio. Estes vídeos poder podem estar guardados na memória do smartphone ou então numa plataforma de streaming de vídeo, como por exemplo o YouTube. Desta forma são consumidos, de forma desnecessária, recursos importantes do smartphone, tais como a energia da bateria, a largura de banda e o tráfego de dados que é limitado na maior parte dos utilizadores. Reconhecer atividades humanas com sensores permitiu o desenvolvimento de aplicações para as mais diversas áreas, desde de supervisão médica até ao entretenimento. Inicialmente estes sistemas implicam a utilização de vários sensores espalhados pelo corpo do utilizador, mas com o aumento do poder de processamento dos smartphones e dos sensores embutidos, foi possível passar a utilizar apenas o smartphone para reconhecimento de atividades. Os sensores embutidos na maior parte dos smartphones, tais como acelerómetro, giroscópio, entre outros, permitiram múltiplas possibilidades para um novo tipo de aplicações baseadas no contexto adquirido a partir dos dados que fornecem. Com esta dissertação pretende-se desenvolver um método, que com base no contexto do utilizador, possa alternar automaticamente entre o vídeo e o áudio, reduzindo assim o consumo de energia. Deste modo aumentar-se-ia o tempo de utilização do smartphone entre carregamentos da bateria. Desta forma resolver-se-ia o problema do consumo desnecessário dos recursos do smartphone. Este projeto tem como objetivos pesquisar a precisão com que é reconhecido o contexto do utilizador, o método que obtém melhor precisão e a sobrecarga que este sistema pode ter no smartphone, reconhecer se o utilizador está a ver um vídeo ou simplesmente a ouvir o áudio, analisar o consumo de energia e finalmente trocar automaticamente entre vídeo e áudio. Para além destes objetivos foi também desenvolvida uma aplicação Android para testar e avaliar a abordagem. Foi utilizada uma abordagem de Supervised Learning, juntamente com os classificadores K-Nearest Neighbors, Hoeffding Trees e Naïves Bayes, individualmente e combinados para criar um classificador ensemble. O dataset utilizado foi recolhido pelo autor e foi obtida uma exatidão global de 93.54% para o classificador ensemble. Individualmente o kNN obteve os melhores resultados quando comparado com os restantes classificadores utilizados, obtendo um exatidão global de 92.53%.

Palavras-Chave: Reconhecimento de atividades humanas, sensores móveis, machine learning, aplicação móvel, Supervised Learning, smartphone, kNN, Hoeffding Trees, Naive Bayes

Acknowledgements

My sincere thanks to the professors João Manuel Paiva Cardoso and João Pedro Mendes Moreira for all the support, guidance and supervision during the development and writing of this dissertation. I also thank Ricardo Magalhães, with whom I worked in an initial phase of this dissertation.

I would also like to thank the Professors of the Faculty of Engineering of the University of Porto with whom I contacted, for everything they taught me and advised during my academic course.

This work has been partially funded by the European Regional Development Fund (ERDF) through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT) within project POCI-01-0145-FEDER-016883. I would also like to thank the CONTEXTWA project for providing all the necessary tools to complete this dissertation, namely code and the Samsung Galaxy J5 smartphone.

Finally, I would like to express my sincere gratitude to my mother, Rosa Madalena Sousa Silva, my father, João Paulo Rocha Ferreira, and my brother, Hugo Miguel Silva Ferreira, who have always supported me throughout my life and provided resources and tools during my academic career.

Paulo Jorge Silva Ferreira

Agradecimentos

O meu sincero agradecimento aos professores João Manuel Paiva Cardoso e João Pedro Mendes Moreira por todo o apoio, orientação e supervisão durante a realização e escrita desta dissertação. Agradeço também ao Ricardo Magalhães, com quem trabalhei numa fase inicial desta dissertação.

Gostava também de agradecer aos Professores da Faculdade de Engenharia da Universidade do Porto com quem contactei, por tudo aquilo que me ensinaram e aconselharam durante o meu percurso académico.

Este trabalho foi realizado no contexto do projecto CONTEXTWA. Por iss queria também deixar o meu agradecimento ao projeto CONTEXTWA por disponibilizar todas as ferramentas necessárias para concluir esta dissertação, nomeadamente o código e o smartphone Samsung Galaxy J5.

Por fim, gostava de exprimir a minha mais sincera gratidão para com a minha mãe, Rosa Madalena Sousa Silva, o meu pai, João Paulo Rocha Ferreira, e ao meu irmão, Hugo Miguel Silva Ferreira que sempre me apoiaram durante toda a minha vida e proporcionaram todos os recursos e ferramentas durante o meu percurso académico.

Paulo Jorge Silva Ferreira

*“I see now that the circumstances of one’s birth are irrelevant.
It is what you do with the gift of life that determines who you are.”*

Takeshi Shudo

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation	2
1.3	Problem Definition	2
1.4	Approach	3
1.5	Goals	3
1.6	Document Structure	4
2	Background and Related Work	5
2.1	Human Activity Recognition	5
2.1.1	Data Collection	6
2.1.2	Preprocessing	7
2.1.3	Sliding Windows	7
2.1.4	Feature Extraction	8
2.1.5	Training	9
2.1.6	Classification	9
2.2	Semi Supervised Learning	9
2.2.1	Self-Training	10
2.2.2	Co-Training	10
2.3	Classifiers	11
2.3.1	K-Nearest Neighbors (KNN)	11
2.3.2	Decision Trees	12
2.3.3	Naïves Bayes	13
2.3.4	Ensemble Learning	14
2.4	Software	15
2.4.1	Massive Online Analysis (MOA)	15
2.4.2	Android	15
2.4.3	Plotly	16
2.5	Related Work	16
2.5.1	Online Human Activity Recognition on Smartphones	16
2.5.2	Location and Activity Recognition Using eWatch: A Wearable Sensor Platform	17
2.5.3	Activity Recognition Using Smartphone Sensors	18
2.5.4	COSAR: hybrid reasoning for context-aware activity recognition	19
2.5.5	Improving Human Activity Classification through Online Semi-Supervised Learning	20
2.5.6	A Preliminary Study on Hyperparameter Configuration for Human Activity Recognition	21

CONTENTS

2.6	Summary	22
3	Scenarios and Dataset	23
3.1	Scenarios	23
3.1.1	Walking	23
3.1.2	Running	24
3.1.3	Sitting on the Couch	24
3.1.4	Cooking in the Kitchen	25
3.2	Dataset	26
3.3	Summary	30
4	Implementation	33
4.1	Working Environment	33
4.2	Core Architecture	33
4.2.1	Data Acquisition Application	34
4.2.2	Desktop Program	34
4.3	Prototype	35
4.3.1	Architecture	36
4.3.2	The Prototype App	37
4.3.3	Overhead	40
4.4	Summary	42
5	Experiments and Results	43
5.1	Data Visualization	43
5.2	Battery Consumption	44
5.2.1	Video vs Audio (Local)	45
5.2.2	Local Video vs YouTube Video	45
5.2.3	YouTube Video vs YouTube Video with Static Image (Black)	46
5.2.4	Local Video vs Local Video with Classifiers	47
5.2.5	Local Audio vs Local Audio with Classifiers	48
5.2.6	Local Video with Classifiers vs Local Audio with Classifiers	49
5.2.7	Local Video and Audio with and without Classifiers	50
5.2.8	Battery Consumption per Classifier	51
5.3	Experiments	52
5.3.1	Global Accuracy	54
5.3.2	Accuracy by Scenario	55
5.3.3	Impact of Frequency Reduction	57
5.3.4	Impact of the use of Headsets	62
5.4	Summary	63
6	Conclusion	65
6.1	Final Conclusions	65
6.2	Future Work	66
	References	69
A		73
A.1	Conversion of Labels	73

List of Figures

2.1	Activity Recognition Steps (Source: [SBI+15]).	6
2.2	Classification of techniques applied to sensor signals for feature extraction. (Source: [FDfC10])	8
2.3	How the value of k can affect the classification of a new object (Source: [Car16]). In this example, the star represents the new object and with a k=3, 2 instances of class B and 1 of class A are the nearest neighbors.	12
2.4	Decisions at each node to get a final classification (Source: [dOL12]).	13
2.5	Bayes rule in Naïve Bayes Algorithm (Source: [Car16]).	13
2.6	A diagram of the ensemble learning [KMG+17].	15
4.1	Architecture of the desktop program.	35
4.2	Architecture of the prototype.	37
4.3	Screen to set the parameters of the prototype.	38
4.4	Screen where the behavior of the prototype is tested. In this case video is being played.	39
4.5	Screen where the behavior of the prototype is emulated. In this case audio is being played.	40
4.6	Average execution time for each classifier and for each possible combination of classifiers. In all measurements where the KNN was used, the value of k used was 3 (i.e., k = 3).	42
5.1	Plot of the accelerometer data which represents the activity "Running" (top) and the activity "Walking" (bottom). Each unit represents a single sensor reading.	44
5.2	Battery consumption during playback of a video and audio file. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.	45
5.3	Battery consumption during playback of a local video and a video on YouTube. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.	46
5.4	Battery consumption during playback of a YouTube video and a YouTube video with a static image. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.	47

LIST OF FIGURES

5.5 Battery consumption during playback of a local video and a local video with real time classification. In this measurement was used only the ensemble classifier. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.	48
5.6 Battery consumption during playback of a local audio file and a local audio file with real time classification. In this measurement was used only the ensemble classifier. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.	49
5.7 Battery consumption during playback of a local audio file with real time classification and a local video file with real time classification. In this measurement was used only the ensemble classifier. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.	50
5.8 Battery consumption for local video and audio with and without real-time classification. For measurements with classifiers the ensemble was used. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.	51
5.9 Battery consumption for the ensemble classifier and for each of the individual classifiers while playing a video. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.	52
5.10 Accuracies of the ensemble and of each of the individual classifiers for the different window sizes studied. The window sizes studied range from 50 to 300. A 0% overlap was used. For the kNN the value of k used was 3 (k=3).	53
5.11 Correct Classifications by Combining Classifiers.	55
5.12 Accuracies of the ensemble and each individual classifier for each scenario studied. These accuracies were obtained for a frequency reduction from 5 Hz to 2.5 Hz. A window of size 100 and with an overlap of 0% was used.	58
5.13 Accuracies of the ensemble and each individual classifier for each scenario studied. These accuracies were obtained for a frequency reduction from 5 Hz to 1.66 Hz. A window of size 100 and with an overlap of 0% was used.	59
5.14 All sampling frequencies studied: the frequency at which data was collected (5 Hz) and the two frequency reductions performed (2.5 Hz and 1.66 Hz). Each frequency has 5 bars, 4 for each scenario studied and 1 for all data. The accuracies presented concern only the ensemble classifier. A window of size 100 and with an overlap of 0% was used.	60
5.15 Frequency reduction from 5 Hz to 2.5 Hz. For each scenario, the accuracies for the ensemble classifier are presented for the two window sizes used. The size 100 window contains 40 seconds of data and the size 50 window has 20 seconds of data. A 0% overlap was used for both windows.	61
5.16 Frequency reduction from 5 Hz to 1.66 Hz. For each scenario, the accuracies for the ensemble classifier are presented for the two window sizes used. The size 100 window contains 60 seconds of data and the size 30 window has 20 seconds of data. A 0% overlap was used for both windows.	61

LIST OF FIGURES

- 5.17 Accuracy of the ensemble classifier for the various frequencies studied. The window size for the frequency of 5 Hz is 100, for the frequency of 2.5 Hz is 50 and for the frequency of 1.66 Hz is 30. A 0% overlap was used for all frequencies windows. [62](#)

LIST OF FIGURES

List of Tables

3.1	Summarizes the data collected for the Running scenario.	26
3.2	Summarizes the data collected for the Walking scenario.	27
3.3	Summarizes the data collected for the Sitting scenario.	29
3.4	Summarizes the data collected for the Kitchen scenario.	30
3.5	Collected Samples	30
3.6	Samples Collected per Activity	30
4.1	Average Execution Time for feature extraction and classification phases.	41
5.1	Time, in hours, that the playback of a video along with each of the classifier would take to completely discharge the battery.	52
5.2	Number of samples and windows per activity for the train set. The training set corresponds to 80% of the dataset. The window size is set to 100.	53
5.3	Number of samples and windows per activity for the test set. The test set corresponds to 20% of the dataset. The window size is set to 100.	54
5.4	Accuracy for each of the classifiers. For the kNN the value of k used was 3 (k = 3).	54
5.5	Accuracy for the Walking Scenario. For the kNN the value of k used was 3 (k = 3).	56
5.6	Accuracy for the Running Scenario. For the kNN the value of k used was 3 (k = 3).	56
5.7	Accuracy for the Sitting Scenario. For the kNN the value of k used was 3 (k = 3).	56
5.8	Accuracy for the Kitchen Scenario. For the kNN the value of k used was 3 (k = 3).	57
5.9	Accuracy (%) with and without headsets difference between them.	63
A.1	Conversion of Labels between Files	73

LIST OF TABLES

Abbreviations

HAR	Human Activity Recognition
HMM	Hidden Markov Models
KNN	K-Nearest Neighbors
SSL	Semi Supervised Learning
SL	Supervised Learning
SVM	Support Vector Machine
NB	Naive Bayes
HT	Hoeffding Trees
CSV	Comma-separated Values
ARFF	Attribute-Relation File Format

Chapter 1

Introduction

This chapter introduces the topic of the dissertation. It introduces the context, the problem, the approach that was followed, the motivation, the goals of this project and presents the overall structure of this document.

1.1 Context

Smartphones are increasingly present in people's lives. These devices can be used for many different tasks such as communication, work and entertainment. In entertainment, smartphones are used, for instance to watch videos. These videos may be available online, on streaming platforms, or on the device's own memory.

However, many users are only interested in listening the audio track associated to the video. In this case smartphone's resources such as battery power are unnecessary used.

One way to solve this problem would be for the smartphone to recognize what the user is doing in order to switch between video and audio. Human Activity Recognition (HAR) [ZWN⁺17] aims to recognize the activities performed by humans through the analyses of a series of observations on the actions of the users. This approach can be used in several areas such as computer vision, medicine, security, entertainment and sports.

Initially, mobile phones were considered as resource-limited devices due to the fact that they did not possess enough battery and computational resources to run these systems. However, modern smartphones have become more powerful in terms of available resources, such as CPU, memory and battery, allowing online activity recognition [SBI⁺15].

Smartphones have a set of built-in hardware sensors such as accelerometers, gyroscopes, GPS, proximity and illumination sensors, microphone and magnetometer. These sensors provide data that allow the smartphone to recognize the activities performed by users. This allow HAR to be performed directly on device with data gathered from the sensors.

HAR systems need to take into account various aspects such as power consumption, processing needs, accuracy and overall system overhead.

Main HAR tasks include data acquisition (sensing), preprocessing, feature extraction and either training or classification. At the end results, a model capable of detecting meaningful patterns. Many machine learning techniques [STJ14], such as Hidden Markov Models, Support Vector Machine or K-Nearest Neighbors, have been used to match patterns to the desired activities.

1.2 Motivation

The last decade witnessed an explosion of mobile communication infrastructure and services. The current smartphones are not just communication devices instead they are personal computer packed into a small gadget.

The spread of more and more complex sensores and increased processing power on smartphones, allow the development of increasingly complex and accurate HAR systems [SBI⁺15]. These developments have attracted many researchers to develop more and more HAR systems. Currently everybody carries at least one sensor on everyday life without knowing it, since many people do not know that the smartphones they own contain multiple sensors. HAR systems enable the development of applications for the most diverse areas of human activity, such as health supervision.

In health supervision, Sebestyen et al. [SSH16] proposed a method to infer a most probable sequence of activities performed by elderly people using various sensors placed in house and on mobile devices. This method allows monitoring and supervision of single old people in their homes.

In this dissertation a method is proposed that makes the switch between video and audio according to the activities performed by the user. The proposed method may increase the time of use of the smartphone by reducing battery consumption.

HAR systems use machine learning techniques to classify activities based on data collected by the sensors. Machine learning is a field of computer science increasingly used to solve classification problems. It is very promising and has been the subject of many researchers. In this dissertation, several machine learning techniques will be used in order to recognize what activities the user is performing.

With this thesis it is expected contributions to the HAR field with the development of an application capable of switching from video to audio automatically through recognition of user's context.

1.3 Problem Definition

Despite the significant advances in mobile computing, the smartphone continues to have limited resources. With the amount of applications and functions that smartphones have today, these resources run out very quickly.

As already mentioned, many users only want to hear the audio track associated with a video. This task consumes the important resources of the smartphone, such as the battery.

Introduction

On a smartphone, this distinction between listening to the audio or watching the video can be automatically identified, knowing the context of the user and the device. By identifying the use of the video just to listen to the audio, it will allow to change to the playback of the audio track instead of the video.

This switch will save energy, bandwidth and reduce the download size (if it is an online streaming platform), since audio files may be smaller and easier to decode when compared to video files.

Identifying the context of the user will be treated as a classification problem. A real-time activity classification system brings a huge computational weight to the smartphone. In this case, the battery lifetime has to be taken into account.

The application to be developed cannot have much impact on the user's smartphone performance and, at the same time, it has to collect and classify the data accurately.

1.4 Approach

The main goal of this project is to develop an Android application that, based on the context of the user, can switch between video and audio automatically, in order to save the essential features of the smartphone. A Supervised Learning approach will be used.

But before starting the project itself, a great deal of research is needed on Supervised Learning algorithms, the various classifiers used to recognize human activities, the tools to use, and some techniques that can help improve the final result.

After the search is finished, it is necessary to identify the scenarios where the user may be listening only to the audio track of a video. For example, if the user is running, he/she will hardly be watching the video, so chances are he/she is only listening to the audio. Other likely scenarios are studying, cooking, lying down or sitting on a sofa.

Once we have identified the possible scenarios, we need to build the dataset. This uses data from the various sensors present on a smartphone. Users use their smartphone while practicing the identified activities. The resulting data are labeled with the name of the activity practiced. The data are pre-processed to begin with the empirical research, where the algorithms are put into practice to generate classification models.

Then the most relevant features are extracted and later these features are used to train and test the classifiers. Later, we also developed an Android application that was able to show the operation and effectiveness of our approach.

The effectiveness of this approach is evaluated through the Accuracy of the classifications, energy consumption and the overhead associated with real-time recognition. Continuous testing are performed with the goal of keeping track of the project's evolution and progress.

At the end, conclusions are drawn based on the results obtained.

1.5 Goals

The main goals regarding the scope of this dissertation can be elaborated as follows:

- Research what kind of methods can achieve better results and the overhead associated with real-time recognition;
- Recognize if the user is watching the video or simply listening to the audio track, using data from the smartphone's sensors;
- Analyze energy consumption and download size when replacing video with audio;
- Automatically change between audio and video based on the user's context;
- Develop an app, for Android, to test and evaluate the approach.

1.6 Document Structure

The remainder of this document is structured as follows:

- Chapter 2, "Background and Related Work". The objective of this chapter is to provide an overview of the work done in this area to deal with this specific problem. Concepts like supervised, unsupervised and semi-supervised learning are explained. In addition, all steps involved in the recognition of human activities are described. Sliding Windows, classifiers, model evaluation, useful techniques and tools are also described. In the end, related work with the same goal are explored and exposed in terms of successes and failures.
- Chapter 3, "Scenarios and Dataset". In this chapter the scenarios considered in this dissertation are described, as well as the limitations of these scenarios. It also describes how the dataset was built and what data was collected.
- Chapter 4, "Implementation". In this chapter we describe the working environment and the core architecture of the project. It also describes, in more detail, the prototype developed to test the switch between video and audio according to the results of the real-time classification. The architecture, application structure and overhead associated with real-time classification are discussed.
- Chapter 5, "Experiments and Results". In this chapter all the experiments carried out in this dissertation, as well as their objectives, methodologies, results and conclusions are described. The experiments carried out aim to study the battery consumption during the reproduction of video or audio files with and without classifiers and the accuracy of the classifiers for the different scenarios under test.
- Chapter 6, "Conclusion". This chapter concludes the project with an overview of the results of the dissertation and the respective conclusions and a discussion on suggestions and ideas for future work.

Chapter 2

Background and Related Work

This chapter presents the basic concepts, methods and tools for the problem of Human Activity Recognition (HAR). It also reviews the state of art in the HAR field. The main technology and resources to be used in this dissertation are also explained.

2.1 Human Activity Recognition

The main goal of Human Activity Recognition (HAR) [CM17] is to develop systems capable of recognizing the actions of a human by automatically analyzing the ongoing events and extract their context from the captured data.

HAR has enabled novel applications in different areas, such as, healthcare, sports, security and context-aware services and applications [SBI⁺15]. In the initial phase of HAR systems, dedicated wearable motion sensors were used to recognize different physical activities [ZWR⁺17]. During data acquisition, Santos et al. [SCF⁺10] used a smartphone and a series of sensors connected via infrared, bluetooth and GPRS. Recently, thanks to the advances in mobile computing, such as the increase in processing power and the large number of sensors that a smartphone possesses, there has been a shift towards mobile phones [ZWR⁺17].

The smartphone based HAR, can be divided into two types. The first one is online activity recognition. In this type data collection, data processing and classification are carried out locally on the smartphones. The other type is offline activity recognition. In offline type the classification is carried out in non-real-time or offline [ZWR⁺17].

HAR systems consist of five phases:

- sensing or data acquisition (1);
- preprocessing (2);
- feature extraction (3);
- training (4);
- classification (5) [SBI⁺15].

As seen in Figure 2.1 there are two different stages: Preparation and Use.

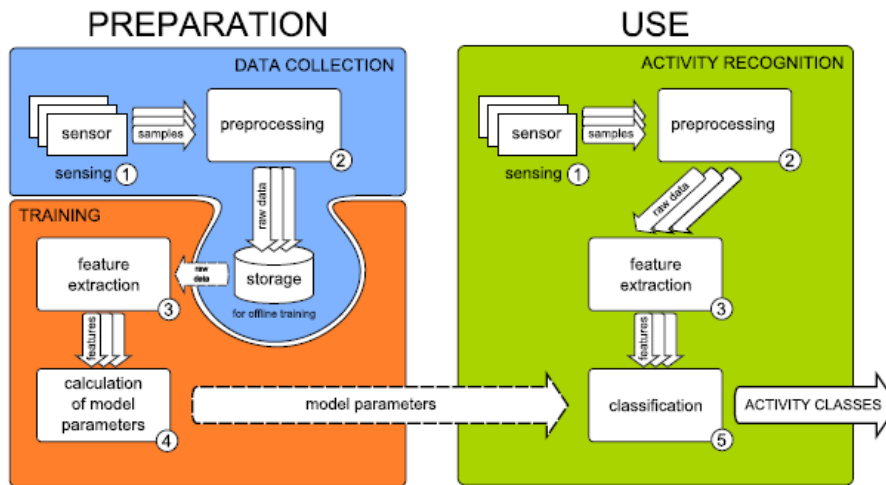


Figure 2.1: Activity Recognition Steps (Source: [SBI⁺15]).

The Preparation stage can either be offline on a desktop machine or online on the smartphone itself. The Preparation stage is used to build the model parameters to later be used in the classification. This stage includes the following HAR phases: data acquisition, preprocessing, feature extraction and training [SBI⁺15].

In the Use stage, the trained classifiers are used to classify different activities. This step can be done either offline in a machine learning tool, such as MOA, or online on the mobile smartphone itself. This stage includes the following HAR phases: data acquisition, preprocessing, feature extraction and classification [SBI⁺15].

2.1.1 Data Collection

The data needed to build the dataset is obtained through smartphone sensors. All the data are handled as streams. The sensor data are acquired at a fixed rate. The data is collected by test subjects when practicing the different pre-selected activities during a certain period of time. This period of time should be sufficient for the activities to be detected. Sebestyen et al. [SSH16] collected data lasting 20-25 seconds.

During data collection phase, it is important to pay attention to the type of sensors and the sampling rates. High sampling rates can avoid the information loss of signals and can lead to high accuracy of recognition [ZWR⁺17]. Liang et al. [LZY⁺12] used a sampling rate of 2 Hz for a tri-accelerometer and obtained an average accuracy of 89,1% for 11 activities, namely: standing, sitting, lying, driving, walking, running, ascending stairs, descending stairs, cycling and jumping.

However, the higher the sampling rates, the more energy is consumed. Due to this fact, it is important to consider the trade-off between the sampling rate and the energy consumption [ZWR⁺17].

2.1.2 Preprocessing

Sensor data acquired from the sensors are fed to the preprocessing stage. The preprocessing phase prepares the raw data to be transformed into a set of finite value features [SCF⁺10]. Normally the raw data collected by the sensors are noise-corrupted, which can lead to measurement inaccuracies and makes it hard to reflect the motion changing of smartphones as accurately as possible [CS17].

After obtaining the data from the previous phase, it is necessary to apply a filtering process before classification, in order to remove the noise associated with the data. In most cases, the noise associated with sensor data is neglected. However there is a type of noise that can seriously affect the classification of activities. This noise is caused by the vibrations that occur on the smartphone when it is carried by the user [MGARDIC15].

Morillo et al. [MGARDIC15] used a Butterworth low pass filter to reduce the noise generated. In order to remove the noise, Sebestyen et al. [SSH16] used a combination of filters, a Median Filter and a Moving Average Filter applied 4 times.

2.1.3 Sliding Windows

An activity, by itself, can not be represented by a single read from smartphone sensors, but it is a pattern on a sequence of the sensor readings [Car16]. Since the sensor data are streams, it is necessary to find methods of analyzing these data streams. One way to analyze data streams is to use sliding windows. The sliding window method is the most used to segment data for activity recognition. The sliding window method accumulates sensor data over a fixed time window. In this method, features are computed over one time window, and then are used as an instance for a training and testing set [KLG11]. A key factor is the selection of sliding window length, because the computational complexity depends on the number of samples [LaL13]. According to Zheng et al. [ZWR⁺17] the accuracy of the activity recognition systems increases with increasing of sliding window length. However, as the length of the sliding window increases, the delay in recognition of activities increases [ZWR⁺17].

The sliding window method uses two approaches: overlapping and non-overlapping sliding windows. In the non-overlapping approach, consecutive time windows do not share common data samples. The overlapping approach shares common data samples between time intervals, for example two consecutive time windows may have 50% of data samples in common [KLG11]. Cardoso et al. [CM17] tested the accuracy of activity recognition using non-overlapping and a 70% overlapping window. In the end they concluded that Overlapping Windows are significant and consistently better than Non-Overlapping Windows.

2.1.4 Feature Extraction

Human activities are performed over a longer period of time when compared to the sensor sampling rate. This step is important for filtering relevant information and obtaining quantitative measures that allow signals to be compared [LaL13]. Extracting the right features is very important to the final recognition performance [STJ14].

According to Figo et al. [FDFC10] there are three types of features extraction domains: Time Domain, Frequency Domain and Discrete Domain. Figure 2.2 present all the domains of feature extraction, as well as some features associated to each domain.

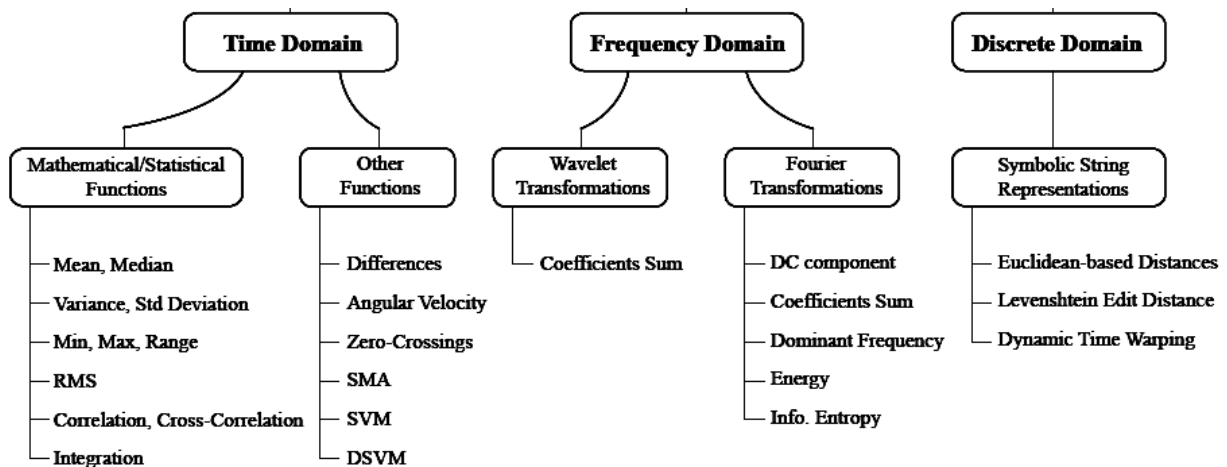


Figure 2.2: Classification of techniques applied to sensor signals for feature extraction. (Source: [FDFC10])

Time Domain: Time-domain features contain the basic statistics of each data segment [STJ14]. These features are simple mathematical and statistical metrics and are used to extract basic signal information from raw sensor data. Furthermore, these metrics are used as preprocessing steps for metrics in other domains and a way to select key signal characteristics [FDFC10].

Frequency Domain: Frequency-domain features describe the periodicity of the sensor signals [STJ14]. These repetitions usually correlates to the periodic nature of a specific activity such as walking or running. An often used signal transformation technique is the Fourier transform [FDFC10]. This transformation allows to represent in the frequency domain important characteristics of a time-based signal, such as average and dominant frequency components [FDFC10].

Discrete Domain: The interest in transforming sensor data into strings of discrete symbols has been proposed [FDFC10]. A fundamental aspect of this transformation has been the process of discretization. Although a limited symbol alphabet can lead to substantial compression in the signal representation, there is a risk of information loss. As soon as the signals are mapped to strings, exact or approximate matches and edit distances are fundamental techniques used to evaluate the similarity of strings and consequently to discover know patterns or classify the user activity [FDFC10].

2.1.5 Training

Before a HAR system can be used it is necessary to train the classifiers. The training phase is the preparation step to obtain the model parameters for later use in classification [SBI⁺15]. In this phase, the extracted features, from the previous phase, are used as input to train the classifier [BBS14].

To train the classifiers it is necessary to separate the data into training and testing sets. The training set is the actual dataset that is used to train the model. The model learns from this data. The test set provides the gold standard used to evaluate the model. It is only used once a model is completely trained [LCB06].

The training can be offline, on a desktop, or online on the smartphone itself. If training is offline, raw data from activities is first collected and stored. Later, these data are used for obtaining the model parameters. On the other hand, if training is online, the raw data are not stored but instead directly processed for training. This step is performed infrequently due to the high processing effort, and the resulting models are stored for future use in the online activity recognition. Due to the fact that it is computationally expensive, many researchers choose to use offline training [SBI⁺15]. Kose et al. [KIE12] used the online training for activity recognition. Liang et al. [LZY⁺12] used offline training on a desktop machine.

2.1.6 Classification

In this phase, the trained classifiers are used to classify different activities [SBI⁺15]. The features extracted from the raw sensor data are used as inputs of the classification algorithms [AMD⁺15].

Classification is considered a form of prediction, i.e., classification is the problem of identifying to which activity a new observation belongs. In case of Unsupervised Learning the procedure is known as clustering, and involves grouping data into categories based on some measure of inherent similarity [Car16].

Supervised Learning requires the use of labeled data to train the classifiers. On the other hand, Unsupervised Learning, tries to directly construct recognition models from unlabeled data [CN09]. Semi-Supervised Learning uses labeled and unlabeled data for training. Usually uses small amounts of labeled data and large amounts of unlabelled data [CM17].

2.2 Semi Supervised Learning

Semi-supervised learning is a special form of classification. As already mentioned, traditional supervised classifiers use only labeled data to train. However, labeled instances are often difficult, expensive, or time consuming to obtain, as they require the efforts of experienced human annotators. Meanwhile, unlabeled data may be relatively easy to collect [Zhu08]. Typically Semi Supervised Learning uses a small amount of labeled data versus a large amount of unlabeled one [CM17].

Semi-supervised learning addresses this problem by using large amount of unlabeled data, together with the labeled data, to build better classifiers. This learning technique requires less human effort and gives higher accuracy [Zhu08].

2.2.1 Self-Training

Self-training [Zhu08] is a technique used for semi-supervised learning. This technique consists of training the classifier with a small set of labeled data and then use the classifier to classify the unlabeled data, adding the most confident classifications and their predicted labels, to the training set. The classifier is re-trained and the procedure repeated. Summing up the classifier uses its own predictions to teach itself [Zhu08].

Self-training has been applied to several natural language processing tasks and to object detection systems from images [Zhu08].

2.2.2 Co-Training

Co-Training [Zhu08] is a technique used for semi-supervised learning. This technique assumes that:

- features can be split into two sets;
- each sub-feature set is sufficient to train a classifier;
- the two sets are conditionally independent given the class.

Co-Training follows the following steps:

- First two separate classifiers are trained with the labeled data, on the two sub-feature sets respectively;
- Then each classifier assigns a label to the unlabeled data, and ‘teaches’ the other classifier with the few unlabeled examples they feel most confident;
- Finally each classifier is retrained with the additional training examples given by the other classifier, and the process repeats [Zhu08].

In co-training, unlabeled data helps reducing the version space size, i.e., the two classifiers must agree on the much larger unlabeled data as well as the labeled data. We need the assumption that sub-features are sufficiently good, so that we can trust the labels by each learner. The sub-features need to be conditionally independent so that one classifier’s high confident data points are independent and identically distributed samples for the other classifier [Zhu08].

This technique only works well if the sub-feature sets are independent, i.e., one of the classifiers correctly labels a piece of data that the other classifier previously misclassified. If both classifiers agree on all the unlabeled data, in other words they are not independent, labeling the data does not create new information [Car16].

Zhou and Li [ZL05] propose ‘tri-training’ which uses three learners. If two of them agree on the classification of an unlabeled point, the classification is used to teach the third classifier. This approach thus avoids the need of explicitly measuring label confidence of any learner [ZL05].

2.3 Classifiers

Classifiers are algorithms responsible for categorizing new observations, after training on known instances [Car16].

2.3.1 K-Nearest Neighbors (KNN)

In general, KNN [STJ14] is one of the most popular algorithms for pattern recognition and perhaps the simplest classification algorithm.

This algorithm is based on lazy learn, which means that it does not have an explicit learning phase. Instead, this algorithm memorizes the training objects, keeping them in a memory. The value of K defines how many neighbor objects have their class label consulted ¹.

It works by searching for the K closest training examples in the feature space. A new object is therefore classified by a majority vote of its neighbors, i.e., assigned to the class most common among its K nearest neighbors [KS12]. KNN uses a local-learning approach since it only uses the class information of those objects most similar to the new object ¹.

Before using the KNN is necessary to set the K, i.e., the number of nearest neighbors that are considered. The best choice of K depends upon the data [KS12]. A very large value may include neighbors that are very different to the object to be classified. But on the other hand if the value of K is too small, only objects very similar to the object to have its class predicted will be considered. Larger values of K may reduce the effect of noise on the classification, but make boundaries between classes less distinct. Also, if the goal is binary classification, it is helpful to choose an odd number to avoid ties [KS12].

¹ João Moreira. 2017. "Classification". Presentation, FEUP, 2017

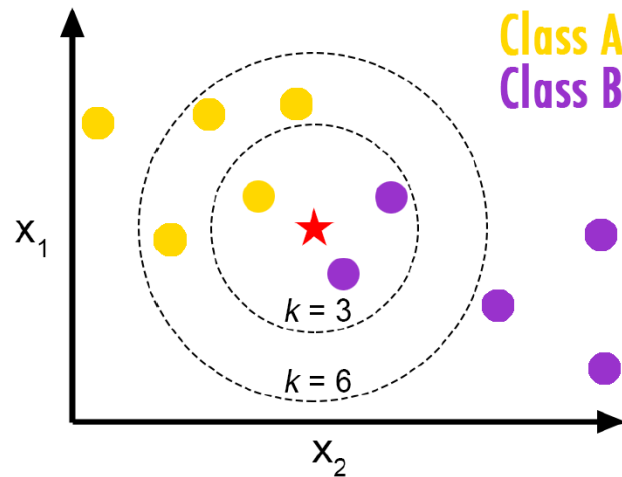


Figure 2.3: How the value of k can affect the classification of a new object (Source: [Car16]). In this example, the star represents the new object and with a $k=3$, 2 instances of class B and 1 of class A are the nearest neighbors.

Kose et al. [KIE12] developed a Clustered KNN for their online activity recognition system. They evaluated the impact of K value on the classification performance of clustered KNN and concluded that increasing the K value affected accuracy rates positively. They observed on average 87% accuracy with $K=10$ whereas it increased to 91% when $K=50$ for the classification of four different activities: running, walking, standing and sitting.

Siirtola and Rönig [SR12] used kNN in activity recognition process. The kNN was used to classify five activities: walking, running, cycling, driving a car and idling (sitting or standing). The average classification accuracy using knn was 94.5%.

2.3.2 Decision Trees

Decision trees build a hierarchical model. In this model attributes are mapped to nodes and edges represent the possible attribute values. Each branch from the root to a leaf node is called a classification rule [LaL13].

A decision tree uses a divide-and-conquer strategy [Utg89]. In this strategy, a problem is solved by dividing it into simpler and smaller problems, and then applying the same strategy to the smaller problem. This division continues until the problem is simple enough to be directly solved². Solutions of these sub-problems can then be combined into a solution of the complex problem [Utg89].

The decision tree's structure consists of a tree in which each node is a decision node, usually containing some condition based on attribute values, and two or more successors nodes. It can be also a leaf node, which is usually labeled with a class [Utg89].

² João Moreira. 2017. "Advanced Classification Methods". Presentation, FEUP, 2017

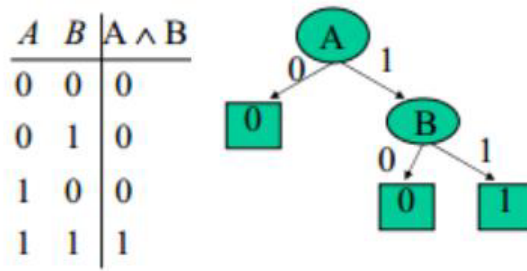


Figure 2.4: Decisions at each node to get a final classification (Source: [dOL12]).

For being an algorithm with low complexity in implementation, decision trees are used as the main classifier in many activity recognition systems [STJ14].

The disadvantage of decision tree lies on updating the model. After the decision tree model is built, it might be costly to update the model to integrate the new training examples [STJ14].

Walse et al. [WDT16] use several machine learning algorithms based on decision trees to classify six activities: walking, jogging, walking upstairs, walking downstairs, sitting, and standing. The algorithms used are: Decision Stump, Hoeffding Tree, Random Tree, J48, Random Forest and REP Tree. Results were obtained from 57.31% for Decision Stump, 87.84% for Hoeffding Tree, 95.69% for Random Tree, 97.33% for J48, 97.83% for Random Forest and 94.44% for REP Tree.

2.3.3 Naïves Bayes

The Naïve Bayes belongs to the family of probabilistic classifiers¹. Naïve Bayes is a generative Machine Learning algorithm. This type of algorithm induce classification models based on joint probabilities, the probability that a given object belongs to a particular class¹.

The Naïve Bayes classifier greatly simplify learning process by assuming that features are independent given class [Ris01].

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

↓
↓

Posterior Probability
Predictor Prior Probability

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Figure 2.5: Bayes rule in Naïve Bayes Algorithm (Source: [Car16]).

Background and Related Work

For a new example, the probability of a class given that example can be calculated by applying Bayes theorem:

- $P(c|x)$ is the posterior probability of class (target) given predictor (attribute);
- $P(c)$ is the prior probability of class. Can be obtained by simply verifying how frequent has the class been outputted in the seen data;
- $P(x|c)$ is the likelihood which is the probability of predictor given class. The likelihood of a class can be obtained by multiplying the frequency of the example attribute values that result in that class, on the seen data;
- $P(x)$ is the prior probability of predictor. This is a constant and can be ignored [Car16].

By multiplying the likelihood by prior probability of class, we obtain the relative probabilities of each class, given the new example. The class with more probability is more likely to be the true class of the example. In this approach the input features are assumed to be independent [AMD⁺15].

It only requires a small amount of training data to estimate the parameters necessary for classification. This is important when dealing with an application in mobile environment, where the memory space is limited as well the processor power [Car16].

The Naive Bayes classifier is very popular due to its simplicity and ease of implementation [AMD⁺15]. Naive Bayes has proven effective in many practical applications, including text classification, medical diagnosis, and systems performance management [Ris01]. The Naive Bayes classifier is also quite effective in various data mining tasks [KHRM06].

2.3.4 Ensemble Learning

Ensemble Learning [LaL13] instead of using a single learning algorithm, this method strategically generates and combines the output of several classifiers to improve classification accuracy. This way solves the same problem and gets better predictive performance than of the algorithms individually [LaL13].

Ensemble Learning has been shown to be an efficient way of improving predictive accuracy and decomposing a complex problem into easier sub-problems. The selection of classifiers is an important factor. An ideal ensemble includes individual classifiers which are characterized by high diversity and accuracy. Classifiers must be selected to obtain positive results from their combination. It is also important to propose a combination rule, responsible for the final decision of the ensemble, which should explore the strengths of each individual classifier [KMG⁺17].

Ensemble Classifiers are clearly more expensive, computationally speaking, as they require several models to be trained and evaluated [LaL13].

There are several ensemble learning algorithms [Oza05]. The best known are Boosting and Bagging [Oza05]. In bagging each classifier has the same vote while boosting assigns different voting strengths to each classifier on the basis of their accuracy [Qui96].

Previous work in HAR uses Ensemble Learning, for instance, [Car16].

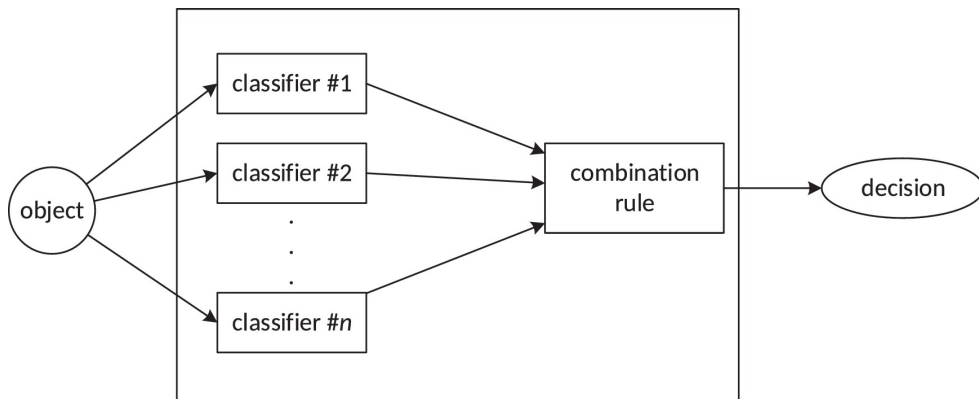


Figure 2.6: A diagram of the ensemble learning [KMG⁺17].

2.4 Software

The software expected to be used in the development of this dissertation are described below. These software will enable the development of the application that will allow to test and evaluate the proposed solution.

2.4.1 Massive Online Analysis (MOA)

MOA [BHKP10] is a software environment written in JAVA, for implementing algorithms and running experiments for online learning from evolving data streams. It includes a collection of machine learning algorithms (classification, regression, clustering, outlier detection, concept drift detection and recommender systems) and tools for evaluation.

It is related to WEKA (Waikato Environment for Knowledge Analysis) [BHKP10], which is an award-winning open-source workbench containing implementations of several batch machine learning methods.

2.4.2 Android

Android [Goo18] is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets.

The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language. It includes a debugger, libraries, emulators, documentation, sample code, and several tutorials. Developers have full access to the same framework API's used by the core applications.

Since both Android and MOA have Java as their programming language, it becomes easier to integrate both together [Goo18].

2.4.3 Plotly

Plotly [Plo18] is a Python open source graphing library. Plotly is an interactive, open-source, and browser-based graphing library for Python. Also ships with over 30 chart types, including scientific charts, 3D graphs, statistical charts, SVG maps, financial charts, and more.

Plotly's Python graphing library makes interactive, publication-quality graphs online. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts that can be seen in [Plo18].

2.5 Related Work

It was previously explained the main steps that a HAR system should follow, the classifiers most used in these systems and the main concepts behind machine learning applied to data streams.

In this section we present similar work done by other researchers. For each similar project is presented the main idea, the dataset collected, the features extracted from the raw data, the main experiments performed, the recognized activities, the obtained results and some disadvantages of these systems.

The analysis of the related work allows to elaborate the best strategy to follow in this dissertation. The papers were chosen because they relate to this dissertation in terms of the classifiers used, recognized human activities and the use of supervised and semi-supervised learning.

2.5.1 Online Human Activity Recognition on Smartphones

Kose et al. [KIE12] developed an app for online activity recognition on smartphones using the built-in accelerometers. They use a single tri-axial accelerometer to recognize four main activities: walking, running, standing and sitting.

For data collection, they developed an application for Android called Activity Logger. In this application, user selects the activity to be performed, places the smartphone into the pocket and starts to perform the selected activity. For each activity, this application creates different training data files in which raw data from the 3-axes of the accelerometer is being logged. In the test phase, the application allows the user to select the system parameters, such as the sensor sampling rate and window size. Tests and data collection were performed with five volunteer subjects whose average age is 27 (1 female and 4 male). Both training and classification stages are performed on the smartphone. Supervised learning was used for classification.

All subjects carried the smartphone in the front pocket of their pants during both test and training phase. Each experiment lasted 4 minutes where each activity was performed for 60 seconds. The same test scenario was repeated 9 times with different system parameters based on window size and sampling rate. They have used several types of Android smartphones, Samsung SII Galaxy, HTC Desire Z, Samsung Galaxy Gio and T-Mobile G2 Touch (HTC Hero).

Background and Related Work

After data acquisition, the raw data is preprocessed using a low pass filter for noise removal. Different features were chosen depending on the classifier used. Clustered KNN and Naïve Bayes were used as classifiers. For clustered KNN four features, which are average, minimum, maximum, and standard deviation, were extracted.

In addition to the accuracy, the impact that this system had on the smartphone's resources, namely CPU utilization and memory usage, was also analyzed. Naïve Bayes achieved a 48% average accuracy rate for all subjects with different sampling rates and window sizes. Compared to Naïve Bayes, on average, clustered KNN achieved a much better classification performance, around 92% accuracy. They also evaluated the impact of K value on the classification performance of clustered KNN and as expected, increasing the K value affected accuracy rates positively. They observed on average 87% accuracy with K=10 whereas it increased to 91% when K is 50. However, accuracy rates were not affected from further increase of K values. They further analyzed the effect of sampling rate and window size on accuracies. In general, worst results are observed in cases when window size is selected as 2 seconds whereas best results are obtained with window size of 1 second. For the window sizes studied, sampling interval does not have a significant impact on the accuracy results. They obtained best results in the case where K is selected as 50, window size is selected as 1 second and sampling interval is selected as 50 msec. According to the tests performed with five different subjects, they obtained average 92% accuracy rate for this case.

To analyze the impact of the application on the smartphone all measurements were taken from Samsung Galaxy Gio. According to the results, CPU and memory usage never exceeded 42% and 22 MB, respectively. Applications using clustered KNN consume nearly the same amount of resources. On the other hand, Naïve Bayes has considerably higher CPU usage.

2.5.2 Location and Activity Recognition Using eWatch: A Wearable Sensor Platform

eWatch [MRSS06] is an online HAR system which that was built into a wrist watch form. This device was built with 4 sensors, namely an accelerometer, a light sensor, a thermometer and a microphone. In the study they focussed on six primary activities: sitting, standing, walking, ascending stairs, descending stairs and running.

To build the dataset they placed the sensor hardware on the left wrist, belt, necklace, in the right trouser pocket, shirt pocket, and bag. The subjects wore six eWatch devices located at these body positions during the study. The eWatch recorded sensor data from the accelerometer and light sensor into the flash memory. The annotations were done using an application running on an extra eWatch worn by the lead experimenter. Six subjects participated in the study; each subject performed the given tasks in 45 to 50 minutes. In total, it was collected over 290 minutes of sensor data. eWatch recorded both axes of the accelerometer and the light sensor. All sensors values were recorded with a frequency of 50 Hz and with 8 bit resolution. The data was collected under

Background and Related Work

controlled conditions, with a lead experimenter supervising and giving specific guidelines to the subjects on how to perform the activities.

The raw data collected from the accelerometers and the light sensor were split into short time windows. These windows are then transformed into the feature space by calculating several feature functions over the individual windows. Only time domain features were considered to avoid the costly computation that is required to transform the signal into the frequency domain.

They evaluated and compared several classification methods, namely Decision Trees (C4.5 algorithm), k-Nearest Neighbor, Naive Bayes and the Bayes Net classifier. Decision Trees and Naive-Bayes were found to achieve high recognition accuracy with acceptable computational complexity. Due to this fact the C4.5 classifier was chosen since it provides a good balance between accuracy and computational complexity.

The overall accuracy was up to 92.5% for the six ambulation activities. The execution time for feature extraction and classification is less than 0.3 ms, making the system very responsive. However, it achieved less than 70% for activities such as descending and ascending stairs.

To test the device on a daily basis a subject wore the eWatch with the built-in activity classifier on the wrist during the day. The system classified the activity in real time and recorded the classification results to flash memory. In this test the device store 100 minutes of activity classification, as the user walked to a restaurant, sat down, ate lunch, went back to the office and sat down to continue working. The classification results match well with the actual activities. However eating lunch was partially interpreted as walking or running activity due to arm movements.

2.5.3 Activity Recognition Using Smartphone Sensors

Anjum et al. [AI13] developed a smartphone app that performs activity recognition that does not require any user intervention. This app is called Activity Diary and was developed for Android. It is also capable of recognizing human activities in real time. The app is capable to recognize seven main activities: inactive, walking, running, climbing stairs, descending stairs, cycling and driving.

To build the dataset, an app was developed to collect data from smartphone sensors. Data collection was performed on a Samsung Galaxy Y phone with Android Gingerbread version 2.3.3. Smartphone placements during data collection were varied and included placement in hand, pants pocket, shirt pocket and handbag. They collected data from ten different people ranging in age from 12 to 25. A group of four people were used for training the data and the remaining users provided test data for the activity classification. In total were obtained 510 activity traces. Each activity trace contains data consisting of time series of 3 accelerometers and 3 gyroscopes. It also contains data from the phones GPS unit that includes longitude, latitude and speed.

In the pre-processing phase, using Eigen-decomposition of the covariance matrix, they corrected the varying orientation of the smartphone. The varying orientation of a smartphone does not allow a meaningful comparison of measurements of a particular axis accelerometer with measurements of the same axis accelerometer from a different activity trace.

Background and Related Work

Only time domain features were considered to avoid the costly computation that is required to transform the signal into the frequency domain. The time domain features used were the mean, variance, period, and Goodness of Linear Fit.

They trained and evaluated four different classification algorithms: Naive Bayes, C4.5 Decision Tree, K-Nearest Neighbor (KNN) and Support Vector Machine (SVM). For performance evaluation they used 10-fold cross validation. This means, activity traces in the data set were divided into 10 sets. One set was selected to serve as test data, while the remaining 9 were used for training the classifier. Naive Bayes obtained a precision of 83%. The C4.5 obtained a precision of 94%. The KNN, with K=1, obtained a precision of 80%. The SVM obtained a precision of 79%.

2.5.4 COSAR: hybrid reasoning for context-aware activity recognition

Riboni and Bettini [RB11] developed a mobile context-aware activity recognition system, called COSAR (Combined Ontological/Statistical Activity Recognition) supporting hybrid statistical and ontological reasoning. They propose a new variant of multiclass logistic regression as the statistical recognition method, and design an algorithm to integrate this method with ontological reasoning.

The system is capable to recognize the following activities and contexts: brushing teeth, hiking up, hiking down, riding bicycle, jogging, standing still, strolling, walking downstairs, walking upstairs and writing on blackboard.

To build the dataset the different activities were performed both indoor (hospital building, kitchen, laboratory, living room and rest room) and outdoor (garden, meadow, urban area and wood) by 6 volunteers (3 men and 3 women, ages ranging from 30 to 60). Each activity was performed by 4 different volunteers for 450 seconds each. While performing the activities, volunteers wore one sensor on their left pocket and one sensor on their right wrist to collect accelerometer data, plus a GPS receiver to track their current physical location. Overall, each activity was performed for 30 min; hence, the dataset is composed of 5 hours of activity data. Samples from accelerometers were taken at 16 Hz. For each activity instance, accelerometer readings were merged to build a feature vector composed of 148 features, including means, variances, correlations, kurtosis, and other statistical measures.

To make the system more resistant to misclassification, COSAR uses a concept of potential activity matrix to filter activities based on the user's location. For instance, if the user is walking downstairs, he is probably not in the woods.

They also developed the COSAR-hist (historical) which takes into account the classifications of previous windows. For example, if the predictions for the last five time windows were "jogging-jogging-walking-jogging-jogging", the third window ("walking") was likely a misclassification. In this case "walking", will be ignored. The COSAR technique obtained an accuracy of 89%. The COSAR-hist technique obtained an accuracy of 93%. In some cases, standing still was confused with writing on a blackboard, as well as hiking up with hiking down. This is due to the fact that they are very similar activities in the movements performed.

2.5.5 Improving Human Activity Classification through Online Semi-Supervised Learning

Cardoso and Moreira [CM17] developed an online human activity recognition system through the use of Semi-Supervised Learning. They also compare the accuracy of algorithms with supervised learning and with semi-supervised learning.

The data set was collected with 30 volunteers, aged between 19 and 48 years. Each volunteer performed a protocol composed of 12 basic activities and their transitions: standing, sitting, laying, walking, walking downstairs, walking upstairs, stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie and lie-to-stand. The recording was performed at a constant rate of 50Hz by an accelerometer and a gyroscope of a Samsung Galaxy SII. Each person repeated the activity routine twice, totaling around 15 minutes of data per person.

The following experiments were carried out: Perfect Segmentation Online Supervised Learning, Perfect Segmentation Online Semi-Supervised Learning, Fixed-Length Sliding Window Online Supervised Learning, Dynamic Data Segmentation, Fixed-Length Sliding Window Online Semi-Supervised Learning and Online Semi-Supervised Learning Using The Author's Data. From these experiences the ones that will be analyzed are Fixed-Length Sliding Window Online Supervised Learning and Fixed-Length Sliding Window Online Semi-Supervised Learning, because they are the closest ones with the objectives of this thesis.

The used classifiers were: Naive Bayes, Very Fast Decision Trees and KNN. They also used Democratic Ensemble and Confidence Ensemble.

For the Democratic Ensemble the votes of each classifier in the ensemble were collected, equally scaled, and added. In the end, the final classification is the most voted activity. On the other hand the Confidence Ensemble uses the confidence of classifiers to classify an instance. For example: if a classifier is 99% certain of an activity label, but the other two agree on a different activity with just 30% certainty, then first classifier's opinion should be taken into account.

In the experience Fixed-Length Sliding Window Online Supervised Learning both overlapping and non-overlapping sliding windows were tested, with the overlapping having an overlap value of 70%. They obtained an average accuracy of 81% for Non-Overlapping and 83% for Overlapping. From the analysis of the obtained results they concluded that Overlapping Windows are significant and consistently better than Non-Overlapping Windows.

In the experience Fixed-Length Sliding Window Online Semi-Supervised Learning, Semi-Supervised Learning and Supervised Learning were compared with a sliding window with 70% overlapping. For Supervised Learning an accuracy of 80% was obtained and for Semi-Supervised Learning an accuracy of 79% was obtained, for individual classifiers. Supervised Learning got an accuracy of 82.69% for Democratic and 84.55% for Confidence. Semi-Supervised Learning got an accuracy of 83.03% for Democratic and 84.63% for Confidence. In average, the Semi-Supervised Learning approach reduced the models performance. However, when working as an ensemble, their view of the data was beneficial at achieving consistently positive results.

2.5.6 A Preliminary Study on Hyperparameter Configuration for Human Activity Recognition

Garcia et al. [GCM⁺18] conducted a study on the impact of two hyperparameters, window size and overlapping between windows, on activity classification accuracy, for each user, using a leave-one-user-out evaluation approach. In addition, they also performed some experiments in the ODROID-XU+E board to evaluate the performance of the hyperparameters in terms of energy consumption and execution time.

In the study, the PAMAP2 public dataset was used. The data from this dataset was collected from three devices positioned in different body areas: wrist, chest and ankle. Each device contributes with data from three sensors: a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer. The PAMAP2 dataset contains 18 different activities: walking, running, nordic walking, cycling, lying, sitting, standing, ascending and descending stairs, ironing, vacuum cleaning, rope jumping, watching TV, computer work, car driving, folding laundry, house cleaning and playing soccer. PAMAP2 contains 1,926,896 samples of raw sensor data from 9 different users. Data were collected at a frequency of 50 Hz.

A sliding window of raw sensor data from PAMAP2 dataset, is processed and converted into a single instance containing the features calculated based on the raw data. instance containing features to be used for the classification. These features include time-domain features, specifically mean and standard deviation for each sensor signal and Pearson Correlation between axes for the 3D sensors.

The hyperparameters explored were: the sampling window size (from 100 to 1000 with increments of 100) and the overlapping between windows (from 0.0 to 0.9 with increments of 0.1). They conducted four experiments with a HAR system featuring an ensemble classifier consisting of three classifiers: kNN, Nave Bayes, and Hoeffding Tree (VFDT). They also intend to analyze the impact of using a semi supervised approach for activity recognition with a leave-one-user-out implementation. Results are shown by: window size, overlapping size, user and activity.

The results showed that Semi-supervised can improve accuracy, but not significantly. The Semi-supervised approach reduces variance for most of overlapping window and has average accuracy close to 90%. Overlapping has more influence on accuracy for values higher than 0.7, but semi-supervised is less susceptible to that influence than the supervised approach. Semi supervised also improves accuracy for window size hyperparameter. Windows with small sizes has worse results, especially for sizes equal to 100 and 200. For user 5, either in supervised approach or semi supervised, the variance is higher than users 2 and 1. The semi-supervised ensemble reduces accuracy variance in users 4 and 8.

For window size 500 and overlapping of 0.8, the accuracy of each activity has notable differences depending on the user. For example, for user 1 the highest accuracy is for the activities Running, Cycling, Vacuum Cleaning and Ironing, but is not the same for user 7, where activities, such as: Lying, Standing, Walking and Nordic Walking, have better accuracy values.

With overlapping of 0.2, activity Cycling is less affected to window size than with overlapping

0.8. However, the results show higher accuracy results. This is noticeable for the other activities too. Especially for window size 100, accuracy values are higher for most of the activities with overlapping 0.2.

The results show that window size and window overlapping do influence the accuracy.

For the time and energy consumption, they conducted experiments in ODROID-XU+E. The experiments focus on a single user, user 6 and the execution time and energy required to process the 250,096 raw samples.

Since the feature extraction depends on the window size, the time to calculate all instances increases as the window size also increases, despite the decreasing number of calculated features. This means that the feature extraction phase is sensitive to the number of raw instances to process. Furthermore, as was increased the overlap factor, due to the increased number of instances that are calculated, the time also increases. The classification time is rather small and slightly increases as the number of calculated features augments.

As for energy consumption, smaller windows result in less energy consumption than bigger windows. It is also perceivable that increasing the overlap factor also increases the energy spent, essentially due to the increased number of feature calculations and classifications to be carried out.

Relating the energy consumption with the accuracy achieved for a given configuration, it is observable that the best accuracy values were obtained when energy consumption was higher. Lower window sizes present less accuracy while higher window sizes provide more accuracy.

The results confirm the need of adapting the classification model to the current user. Due to the impact of window size and overlapping, each activity requires a specific configuration of these hyperparameters to improve classification accuracy. Furthermore, the results also motivate the development of a system that can adapt the application at runtime when trade-offs between performance accuracy and energy consumption need to be considered.

2.6 Summary

In this chapter all the phases associated with the Human Activity Recognition were explained. The classifiers to be used in this dissertation as well as the additional software that will be used have also been described in this chapter. Also discussed were techniques that improve the recognition of activities, namely Ensemble Learning. Using Ensemble Learning allows to improve the classification when compared to the best individual classifiers [CM17] [GCM⁺18].

Chapter 3

Scenarios and Dataset

This chapter discusses the scenarios used in this dissertation, the situations considered for each scenario and the limitations of each one. In addition, it explains the construction of the dataset.

3.1 Scenarios

Before proceeding to the implementation of the solution, it was necessary to identify usage scenarios, where the user could be watching a video, listening to audio or none of the above. There are several usage scenarios that could be explored within the scope of the dissertation, but due to time constraints, only the most common were considered.

The scenarios considered in this dissertation were: Running, Walking, Sitting in the Couch and Cooking in the Kitchen. Each scenario is described in more detail later. Due to the location of the headset input, in the Running and Walking scenarios, for the situations where the smartphone is in the user's pocket, it was necessary to consider two positions: right side up and up side down.

In these scenarios were not considered certain possible situations to happen when the user uses or not the smartphone to play audio or video. These situations were not considered due to time constraints and also due to complexity, which would lead to adopt a different approach than the one used for most of the occurrences covered by this dissertation. The not considered situations will be specified in each scenario.

The identified scenarios served as a basis for the planning and collection of data needed for this dissertation. Data collection is discussed later.

3.1.1 Walking

In this scenario, three possible situations are considered: the user is walking and watching a video; the user is walking and listening to audio; the user is just walking. For the situation "Walking and watching a video" the following was considered: the user is walking and watching a video, with

headset, with the smartphone in portrait mode; the user is walking and watching a video, without headset, with the smartphone in portrait mode.

On the other hand, in the "Walking and listening to audio" situation the following possibilities were taken into account: the user is walking and listening audio with smartphone in left pocket with the screen facing outwards and up side down; the user is walking and listening audio with smartphone in right pocket with the screen facing outwards and up side down. In this case all in all possibilities mentioned, the earphones were connected to the smartphone.

Finally, in the "Walking" situation the following possibilities were considered: the user is walking with smartphone in right pocket with the screen facing outwards and upside down; the user is walking with smartphone in left pocket with the screen facing outwards and upside down; the user is walking with smartphone in left pocket with the screen facing outwards and right side up.

This scenario does not consider the situations where: the user changes the file that is being played; the user continues walking even after the media file is finished playing; the user views videos in landscape mode. Listening to audio, without headphones, while walking was not considered due to the fact that it is a potential source of annoyance for people who are close to the user.

3.1.2 Running

In this scenario two possible situations are considered: the user is running and listening to audio; the user is just running. It was not considered the situation of watching video, because due to the various movements performed by the human body during the performance of the activity it becomes quite difficult to view a video while the user is running.

In what concerns to "Running and Listening Audio", it was considered: the user is running and listening to audio with smartphone in left pocket with the screen facing outwards and upside down.

As for "Running" it was considered: the user is running with smartphone in the left pocket with the screen facing outwards and right side up; the user is running with smartphone in the left pocket with the screen facing outwards and upside down.

As already mentioned, this scenario does not include all the possible activities that the user can perform. The following activities were not included: running and listening to audio without headphones; running with the headset, but not listening, because the audio playback has ended.

3.1.3 Sitting on the Couch

For this scenario, three possible situations were considered: the user is sitting on the couch while watching a video; the user is sitting on the couch while listening to audio; the user is just sitting on the couch.

For the activity "Sitting on the Couch and Watching a Video", it was considered: the user is sitting on the couch and watching a video with the smartphone in portrait mode with headset; the

user is sitting on the couch and watching a video with the smartphone in landscape mode with headset; the user is sitting on the couch and watching a video with the smartphone in portrait mode without headset; the user is sitting on the couch and watching a video with the smartphone in landscape mode without headset.

As for “Sitting on the Couch and Listening to Audio”, it was considered: the user is sitting on the couch and listening to audio with smartphone on the couch next to him and with the screen facing up; the user is sitting on the couch and listening to audio with smartphone in left pocket with screen facing up and upside down; the user is sitting on the couch and listening to audio with smartphone in right pocket with screen facing up and upside down; the user is sitting on the couch and listening to audio with the smartphone in portrait mode while interacting with the device.

Finally for the activity “Sitting on the Couch”, it was considered: the user is sitting on the couch with smartphone in left pocket with screen facing up and upside down; the user is sitting on the couch with smartphone in right pocket with screen facing up and upside down.

For the scenario “Sitting on the Couch” the following activities were not considered: the user is sitting on the couch interacting with the smartphone without listening to audio or watching videos; the user is sitting on the couch listening to audio with the smartphone in landscape mode while interacting with the device; the user is listening to audio without headphones; the user chooses another file to play.

3.1.4 Cooking in the Kitchen

For this scenario, three possible situations were considered: the user is cooking in the kitchen and watching a video; the user is cooking in the kitchen and listening to audio; the user is cooking in the kitchen. In this scenario, no headphones were used.

For the activity “Cooking in the Kitchen and Watching a Video”, it was considered: the user cooking in the kitchen and watching a video with the smartphone on the table with the screen facing up; the user cooking in the kitchen and watching a video with the smartphone leaning against an object in portrait mode; the user cooking in the kitchen and watching a video with the smartphone leaning against an object in landscape mode.

As for “Cooking in the Kitchen and Listening to Audio”, it was considered: the user is cooking in the kitchen and listening to audio with the smartphone on the table with the screen facing up; the user is cooking in the kitchen and listening to audio with the smartphone on the table with the screen facing down. Due to the fact that the smartphone has the loudspeaker in the back, it was considered the situation where the user places the smartphone with the screen face down because it is likely that the user will position the device so as not to obstruct the sound output.

Finally for the activity “Cooking in the Kitchen”, it was considered: the user is cooking in the kitchen with the smartphone on the table with the screen facing up.

As already mentioned, this scenario does not include all the possible activities that the user can perform. It was not considered situations in which the user interacts with the smartphone to change the file that is being played. We have not taken into account situations where the user can connect some sound amplification device, such as Bluetooth speakers, to the smartphone.

3.2 Dataset

One of the most influential aspects about a successful machine learning project is a good dataset. Unfortunately there were no available datasets that met the requirements of the intended activities for this project. Due to this reason it was decided to create a specific dataset that would meet the needs of this project.

The recording was performed at a constant rate of 5Hz by a Samsung Galaxy J5's sensors. It was decided to use a frequency of 5 Hz, because according to Zheng et al. [ZWR⁺17] at this frequency it is possible to save energy when compared with higher frequencies and yet maintain a high accuracy in the recognition of activities. Seven sensors of three dimensions and two of a single dimension were used. The three-dimensional sensors used to collect the data were: accelerometer, game rotation vector, gravity, gyroscope, linear acceleration, magnetic field and rotation vector. The one-dimensional sensors used to collect the data were: light and proximity. The three-dimensional sensors have three axes: x, y and z. In addition, values from the Android API, which indicate whether or not the headset was connected to the smartphone, were also recorded at the same rate as the other sensors. The data about the status of the headset is treated as if it were a one-dimensional sensor. The data were collected by the author in laboratory context.

For the "Running" scenario, the following data were collected:

- running and listening to audio with a headset for 50 minutes with the smartphone in left users pocket with the screen facing outwards and upside down;
- running for 50 minutes with the smartphone in left users pocket with the screen facing outwards and upside down;
- running for 50 minutes with the smartphone in left users pocket with the screen facing outwards and right side up.

Table 3.1 summarizes the data that was collected for the Running scenario. Table 3.1 summarizes for each activity the position and orientation of the smartphone, the use or not of the headsets and the duration of the collection.

Table 3.1: Summarizes the data collected for the Running scenario.

Activity	Smartphone Position	Smartphone Orientation	Duration	Headsets
Audio	left pocket	screen facing outwards and upside down	50 minutes	With
Just Running	left pocket	screen facing outwards and upside down; screen facing outwards and right side up	100 minutes	Without

In the "Walking" scenario the following data were collected:

Scenarios and Dataset

- walking and watching a video with the smartphone in the right hand of the user for 50 minutes, with headsets, in portrait mode and with the inclination of the device varying between 20° and 80°;
- walking and watching a video with the smartphone in the right hand of the user for 50 minutes, without headsets, in portrait mode and with the inclination of the device varying between 20° and 80°;
- walking and listening to audio for 50 minutes, with headsets and with the smartphone in left users pocket with the screen facing outwards and upside down;
- walking and listening to audio for 50 minutes, with headsets and with the smartphone in right users pocket with the screen facing outwards and upside down;
- walking for 25 minutes, without headsets and with smartphone in right user's pocket with the screen facing outwards and upside down;
- walking for 25 minutes, without headsets and with smartphone in left user's pocket with the screen facing outwards and upside down;
- walking for 50 minutes, without headsets and with smartphone in left user's pocket with the screen facing outwards and right side up.

Table 3.2 summarizes the data that was collected for the Walking scenario. Table 3.2 summarizes for each activity the position and orientation of the smartphone, the use or not of the headsets and the duration of the collection.

Table 3.2: Summarizes the data collected for the Walking scenario.

Activity	Smartphone Position	Smartphone Orientation	Duration	Headsets
Video	right hand	portrait mode, with the inclination of the device varying between 20° and 80°	100 minutes	With and Without
Audio	left pocket and right pocket	screen facing outwards and upside down	100 minutes	With
Just Walking	left pocket and right pocket	screen facing outwards and upside down; screen facing outwards and right side up	100 minutes	Without

The data collected for the "Sitting on the Couch" scenario were as follows:

- sitting on the couch and watching a video with the smartphone in the right hand of the user for 25 minutes, with headsets, in portrait mode and with the inclination of the device varying between 20° and 80°;

Scenarios and Dataset

- sitting on the couch and watching a video with the smartphone in the right hand of the user for 25 minutes, with headsets, in landscape mode and with the inclination of the device varying between 20° and 80°;
- sitting on the couch and watching a video with the smartphone in the right hand of the user for 25 minutes, without headsets, in portrait mode and with the inclination of the device varying between 20° and 80°;
- sitting on the couch and watching a video with the smartphone in the right hand of the user for 25 minutes, without headsets, in landscape mode and with the inclination of the device varying between 20° and 80°;
- sitting on the couch and listening to audio with the smartphone on the couch next to the user (horizontal) and with the screen facing up, with headsets, for 20 minutes;
- sitting on the couch and listening to audio for 20 minutes, with headsets and with the smartphone in left users pocket with the screen facing outwards and upside down;
- sitting on the couch and listening to audio for 10 minutes, with headsets and with the smartphone in right users pocket with the screen facing outwards and upside down;
- sitting on the couch and listening to audio with the smartphone in the right hand of the user for 25 minutes, with headsets, in portrait mode and with the inclination of the device varying between 20° and 80°;
- sitting on the couch for 25 minutes, without headsets and with the smartphone in left users pocket with the screen facing outwards and upside down;
- sitting on the couch for 25 minutes, without headsets and with the smartphone in right users pocket with the screen facing outwards and upside down.

Table 3.3 summarizes the data that was collected for the Sitting scenario. Table 3.3 summarizes for each activity the position and orientation of the smartphone, the use or not of the headsets and the duration of the collection.

Scenarios and Dataset

Table 3.3: Summarizes the data collected for the Sitting scenario.

Activity	Smartphone Position	Smartphone Orientation	Duration	Headsets
Video	right hand	portrait and landscape mode, with the inclination of the device varying between 20° and 80°	100 minutes	With and Without
Audio	on the couch next to user; left and right pocket; right hand	screen facing up; screen facing outwards and upside down; portrait mode, with the inclination of the device varying between 20° and 80°	75 minutes	With
Just Sitting	left and right pocket	screen facing outwards and upside down	50 minutes	Without

Finally for the scenario "Cooking in the Kitchen" the following data were collected:

- cooking in the kitchen and listening to audio, without headsets and with the smartphone on the table with the screen facing up, for 25 minutes;
- cooking in the kitchen and listening to audio, without headsets and with the smartphone on the table with the screen facing down, for 25 minutes;
- cooking in the kitchen and watching a video, without headsets and with the smartphone on the table with the screen facing up, for 25 minutes;
- cooking in the kitchen and watching a video, without headsets and with the smartphone in portrait mode leaning against something and with the inclination of the device varying between 45° and 90°, for 25 minutes;
- cooking in the kitchen and watching a video, without headsets and with the smartphone in landscape mode leaning against something and with the inclination of the device varying between 45° and 90°, for 25 minutes;
- cooking in the kitchen, without headsets and with the smartphone on the table with the screen facing up, for 25 minutes.

Table 3.4 summarizes the data that was collected for the Kitchen scenario. Table 3.4 summarizes for each activity the position and orientation of the smartphone, the use or not of the headsets and the duration of the collection.

Scenarios and Dataset

Table 3.4: Summarizes the data collected for the Kitchen scenario.

Activity	Smartphone Position	Smartphone Orientation	Duration	Headsets
Video	over the table	screen facing up; portrait and landscape mode, with the inclination of the device varying between 45° and 90°	75 minutes	Without
Audio	over the table	screen facing up; screen facing down	50 minutes	Without
Just Cooking	over the table	screen facing up	25 minutes	Without

Table 3.5 lists the number of samples collected for each scenario as well as the total number of samples collected for the construction of the dataset.

Table 3.5: Collected Samples

Scenario	Running	Walking	Sitting	Cooking in the Kitchen	Total
Number of Samples	44966	89886	67377	44910	247139

Table 3.6 lists the number of samples for each activity (Audio, Video and Other). These numbers are independent of the scenario and only concern the activity, i.e. for the Video activity there are samples of the Walking, Sitting and Kitchen scenarios.

Table 3.6: Samples Collected per Activity

Activity	Audio	Video	Other
Number of Samples	82361	82394	82384

Attempts have been made to have approximately the same number of samples for each activity in order to make the dataset more balanced. An unbalanced dataset may cause problems in training, because the majority activity can dominate the minority activities and then the classifiers may only learn one activity instead of three distinct activity.

3.3 Summary

In this chapter the scenarios identified for this dissertation were specified. In addition, all the situations that each scenario contemplates as well as its limitations, that is, possible scenarios, but which were not taken into account for this dissertation, were described. Also explained is the process of collecting the data for the construction of the dataset. In each scenario all the activities that were considered in the data collection are mentioned, as well as the position and orientation

Scenarios and Dataset

of the smartphone, the duration of the collection and the use or not of the earphones during the collection of the data for the activity in question.

Scenarios and Dataset

Chapter 4

Implementation

This chapter describes implementation details. The various applications and programs used in the development of the project are described. In addition, the project architecture is described. Lastly, and in more detail, the chapter presents the prototype developed, its architecture and the overhead associated with real-time recognition on the smartphone.

4.1 Working Environment

Before starting the project development, it is necessary to consider the technologies to be used and the experiments to be carried out.

The project was built using the Java language, since it is the dominant development tool for the Android platform. The chosen IDE was Android Studio [And18a], since it allows developing applications for Android and also for Desktop.

It was decided early on that the MOA [BHKP10] would be included in this project, since it allows dealing with data streams and offers a collection of incremental machine learning algorithms. In addition the MOA framework is also written in Java which facilitated integration with the Android platform. The MOA was imported as a project dependency in the gradle file.

Aside from the main project, Python was also used. Several Python scripts that allowed for data plotting and analysis were written, which helped to visualize and understanding the dataset and the results. The plots were generated with the help of the library Plotly [Plo18].

This contributed to a very capable working environment that allowed the use of incremental machine learning algorithms with MOA.

4.2 Core Architecture

The project was divided into three distinct parts: an application responsible for collecting sensor data; a Desktop program responsible for classification; a prototype to test the switch between

video and audio according to the context of the user.

The application for data acquisition and the desktop program had already been developed under the CONTEXTWA project [CMV18]. However some changes had to be made to the existing code in order to adjust them to the needs of this dissertation.

4.2.1 Data Acquisition Application

The application for collecting sensor data collects the data from the sensors while the user performs the various activities described in the previous section. This application collected the data used to construct the dataset used in this project. This application saves the data in CSV files, creating a file for each sensor available. The application also saves data from the Android API, which records whether the headset is connected to the smartphone or not.

4.2.2 Desktop Program

The desktop program is responsible for pre-processing the CSV files and also for data classification. It is in this program that the MOA and its classifiers are used. The desktop program receives the CSV files generated by the data collection app and pre-processes them in ARFF files, which is MOA's format of choice. Although during the data collection a label was assigned to each activity performed, these labels were not used in the ARFF files.

These labels only served to discriminate the activity associated with the data. The new labels used in the ARFF files were obtained from the original CSV file labels. For example if the label of an activity in the CSV files is `Running_audio`, the corresponding label in the ARFF file will be `Audio`. The process of transforming the labels of the CSV files into the labels of the ARFF files can be found in Appendix A.1.

The labels obtained especially for the ARFF files were:

- **Audio** - which indicates that the user is listening to audio;
- **Video** - indicating that the user is watching a video;
- **Other** - which indicates that the user is not listening to audio or watching a video.

After converting CSV files to ARFF, data windows with a size that is specified at the beginning of program execution are extracted. For example: a window size of 100 means that 100 samples of data are extracted. From these windows are extracted the features and a vector is created with these features that are later the input of the classifiers, both in the training phase and in the test phase.

A total of 76 features are extracted from the ARFF files: 10 for each of the seven three-dimensional sensors and 2 for each of the three one-dimensional sensors.

The 10 features extracted from the 3D sensors are as follows:

- **Arithmetic Mean** of the X, Y and Z axis, individually and also together, resulting in 4 features for each of the sensors;

Implementation

- **Standard Deviation** of the X, Y and Z axis, individually, resulting in 3 features for each of the sensors;
- **Pearson Correlation** of axis X and Y, Y and Z, and X and Z, resulting in 3 features for each of the sensors.

The 2 features extracted from the 1D sensors are the following:

- **Arithmetic Mean** of the X axis, resulting in 1 feature for each of the sensors;
- **Standard Deviation** of the X, resulting in 1 feature for each of the sensors.

Figure 4.1 summarizes, in the form of a block diagram, the architecture of the program desktop.

The desktop program consists of two phases: the training phase and the test phase. During the training phase, training data is used to train the classifiers. This phase results in the classifiers already trained that will be used in the test phase. In the test phase the trained classifiers are used to classify the test data. This phase serves to test the accuracy of classifiers since the data used for testing are already labeled.

The three main incremental algorithms used were Naive Bayes, Hoeffding Trees and K-Nearest Neighbors, especially as an ensemble [GCM⁺18].

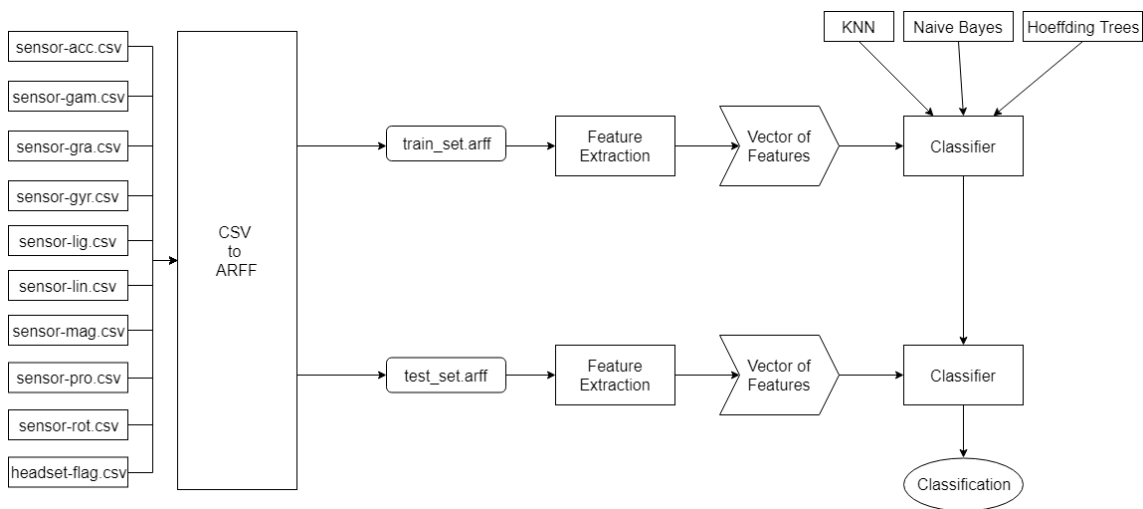


Figure 4.1: Architecture of the desktop program.

The prototype is discussed in more detail in Section 4.3.

4.3 Prototype

The prototype is an Android application that was developed as proof of concept to test the classification in real time and based on the result switch between video and audio. The prototype is

Implementation

composed of data acquisition and desktop program features. It also allows playback of video and audio files through the MediaPlayer class of the Android API [And18b]. In order to emulate the switch between video and audio, an mp4 file and another mp3 are stored in the internal memory of the smartphone. These files concern the same live music show and have the same duration.

4.3.1 Architecture

The prototype allows access to the sensor data in real time and data stream classification functionality of the desktop program.

Due to the high computational weight and time required to train the classifiers, the models are trained in the desktop program and then used in the prototype. The MOA classifiers used in the prototype to classify the real-time sensor data were: Naive Bayes, Hoeffding Trees and KNN.

The desktop program saves the trained classifiers to files that are then loaded by the prototype before starting the real-time classification process. The trained classifiers are stored in the internal memory of the smartphone. In this way the prototype only has to classify the data coming from the sensors, reducing the energy consumption and computational cost required during the training phase of the classifiers.

To train the models were used 197945 samples of sensors. The training was done with a window of 100 and with a 0% overlap, resulting in approximately 1979 training instances.

Loading the KNN already trained takes some time, since this classifier loads a large knowledge base and needs to build the feature space before it can be used in the classification process.

The prototype has 10 buffers, one for each sensor, which are filled with the data coming from the respective sensors. Once the classification process is started by the user, the buffers begin to be filled with data from the sensors. When the buffers reach the capacity defined by the user, the features are extracted. The extracted 76 features are the same as those used in the desktop program and are properly identified in the "Core Architecture" section. Once the features are extracted, an instance of the MOA is created, consisting of all extracted features and later used as input of the classifiers, which were previously loaded.

The instance is passed as an argument to the classifiers which, based on the training done, produce a prediction. Possible predictions are: Other, Audio and Video. Based on the prediction, the prototype can adopt one of the following behaviors:

- If the prediction is "Audio" and the video file is playing, the prototype switches to the audio;
- If the prediction is "Video" and the audio file is playing, the prototype switches to the video;
- If the prediction is "Audio" and the audio file is playing, the prototype keeps the audio file playing;
- If the prediction is "Video" and the video file is playing, the prototype keeps the video file playing;
- If the prediction is "Other", the prototype does not switch regardless of the file being played.

Implementation

Figure 4.2 summarizes, in the form of a block diagram, the architecture of the prototype.

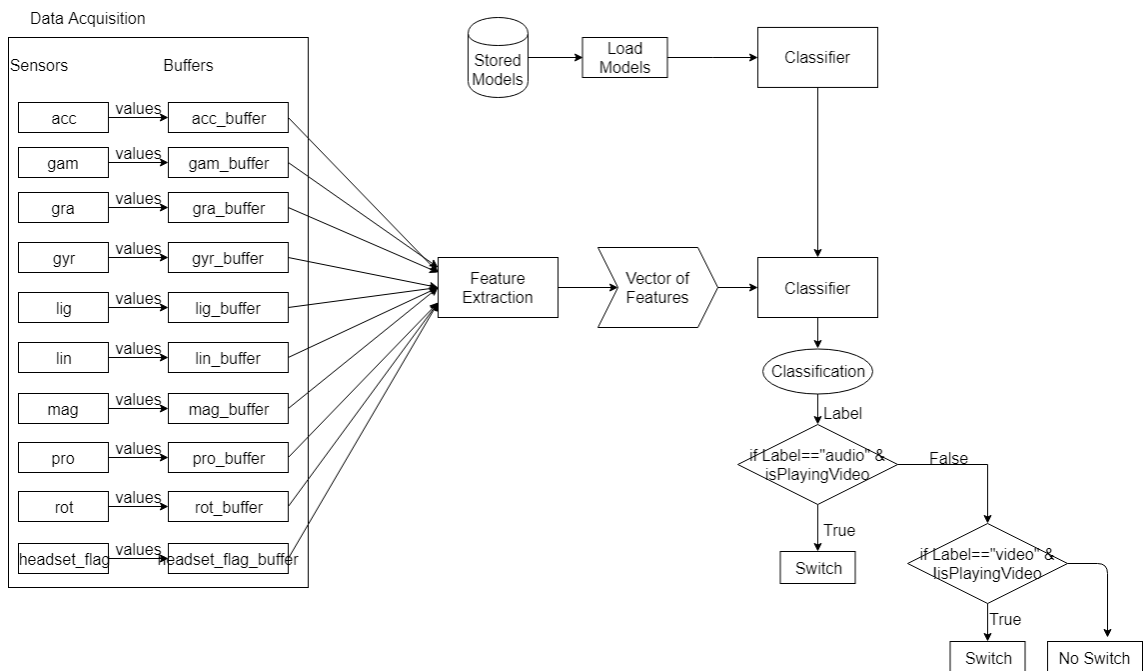


Figure 4.2: Architecture of the prototype.

When switching from Video to Audio or vice versa is carried out, the time already played is retained, that is, if the user watches 10 minutes of the video file and the exchange is made for audio when the exchange completes the audio playback continues from these 10 minutes.

4.3.2 The Prototype App

The application developed for the prototype contains two main screens. The first screen presents a set of parameters that the user can define and that influences the behavior of the prototype. The second screen presents the prototype itself.

4.3.2.1 First Screen

In this screen the user can choose which classifiers he/she wants to use in the classification process, the frequency of the sensors, the size of the data buffer and the overlap between two consecutive buffers. For example, if the user chooses an overlap of 70%, it means that the next buffer will hold 70% of the previous buffer data, i.e. only 30% will refer to new data. The buffer size refers to the number of sensor data that the prototype holds by sensor and which will be classified. In other words, a buffer size of 100 means that the application stores up to 100 sensor data each time and it is on this data that the features will be extracted.

Implementation

The user can also choose whether or not to make the switch between video and audio. If the switch is not activated the application only plays the mp4 file and displays the result of the real-time classification.

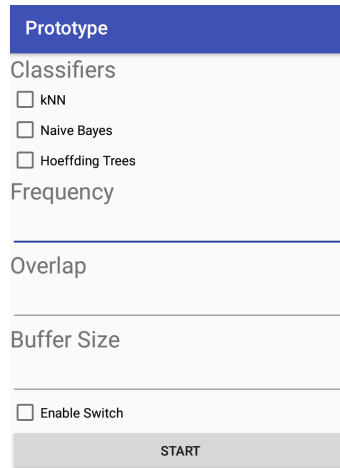


Figure 4.3: Screen to set the parameters of the prototype.

As previously mentioned the parameters specified by the user can influence the behavior of the prototype. The choice of classifiers influences system overhead, since, as shown later, there are combinations of classifiers that take longer to execute than others. Regarding the sampling frequency parameter: the higher the frequency, the more samples per second and consequently the shorter the time to fill the buffers with the sensor data. The overlap, as it allows preserving data from the previous buffer, the larger the overlap the smaller the time to fill the remainder of the buffer with new data. The larger the buffer size the longer it will take to fully populate the buffers with the data, for the same frequency. After specifying the desired parameters and pressing the "Start" button the user is taken to the second screen where he/she can see the running prototype according to the desired parameters.

4.3.2.2 Second Screen

In this screen (see Figure 4.4) the elements of the graphical interface may differ depending on the type of media being played.

Implementation

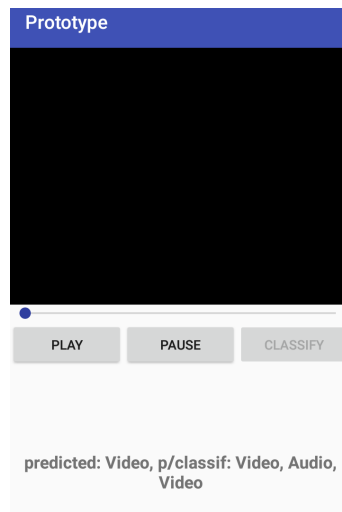


Figure 4.4: Screen where the behavior of the prototype is tested. In this case video is being played.

If a video file is playing on this screen, the graphic interface elements are as follows:

- Surface View where the video is played;
- Seek Bar where playback progress is updated and where the user can interact to advance or rewind playback of the file;
- Play Button to start file playback;
- Pause Button to pause playback of the file;
- Classify Button that starts the real-time classification process;
- Text View where the result of the ensemble classification and the individual classifiers that are active (predicted: ENSEMBLE, p / classif: KNN, NB, HT).

If an audio file is playing on this screen the graphic interface elements are practically the same when playing the video file, except Surface View which in this case is not present (see Figure 4.5).

Implementation

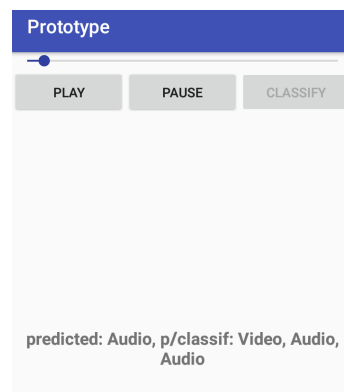


Figure 4.5: Screen where the behavior of the prototype is emulated. In this case audio is being played.

When the video-to-audio switch is made, mp4 file playback ends, Surface View is no longer visible, and mp3 file playback starts.

When the audio-to-video switch is made, mp3 file playback ends, Surface View becomes visible, and mp4 file playback starts.

4.3.3 Overhead

To study the overhead associated with real-time classification, an experiment was carried out. Overhead is studied from the point of view of the execution time of the functions associated to the real-time classification.

Execution time measurements were made for the following situations:

- Features Extraction;
- All Classifiers;
- Features Extraction and All Classifiers
- KNN;
- NB;
- HT;
- NB and HT;
- NB and KNN;
- KNN and HT.

In the phase of the extraction of features are extracted 76 features. During this phase, two feature extraction functions are performed for each one-dimensional sensor and three feature extraction functions for each of the three-dimensional sensors. In total, at this stage, 6 functions

Implementation

are performed for single-dimension sensors and 21 functions for three-dimensional sensors. The functions performed for the sensors of a dimension are the calculation of the mean and the standard deviation, resulting from these executions 6 features. The functions performed for the three-dimensional sensors are the calculation of the mean, standard deviation and Pearson correlation, resulting from these executions 70 features.

Feature extraction begins as soon as sensor buffers are fully populated. Features are extracted from these buffers. From this process results a vector with the 76 features extracted from the buffers. The execution time of the feature extraction refers to the time it takes to obtain the vector with the features extracted from the buffers. The execution time of the classification process refers to the time it takes to classify the vector of features obtained in the extraction phase of the features.

The parameters established for this experiment were: frequency of 5 Hz, 0% overlap and buffer size with 100 samples (corresponding to a window size of 100). Twenty measures were taken for each of the situations under analysis and then the average of the values obtained was calculated. The average run times are shown in the following table.

Table 4.1: Average Execution Time for feature extraction and classification phases.

	Average Execution Time (ms)
Feature Extraction and Ensemble Classification	98.95
Feature Extraction	15.15
Ensemble Classification	79.05

From the analysis of the results obtained, the entire classification process (which includes the Features Extraction and the classification with all the active classifiers) takes on average 98.95 milliseconds. Most of this time concerns the classification that, with all active classifiers, takes on average 79.05 ms to classify a new instance. Features Extraction takes on average 15.15 ms.

In order to evaluate which classifier or combination of classifiers has the least impact on the prototype, the results presented in Figure 4.6 were measured.

Implementation

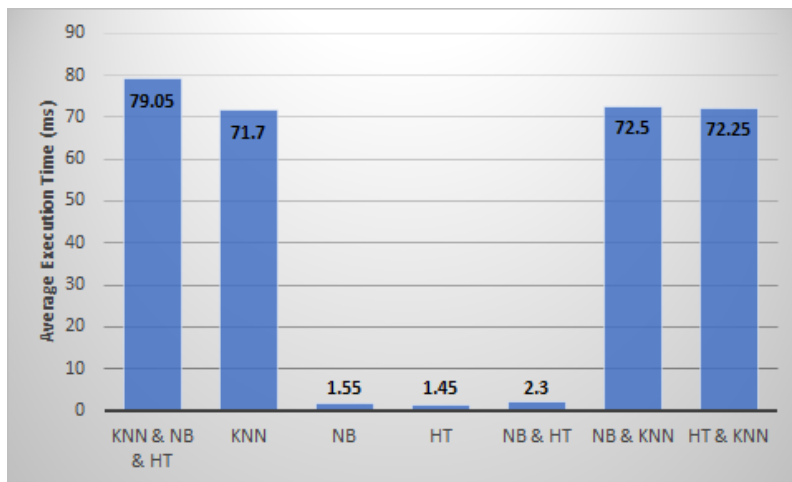


Figure 4.6: Average execution time for each classifier and for each possible combination of classifiers. In all measurements where the KNN was used, the value of k used was 3 (i.e., $k = 3$).

Figure 4.6 shows that when all the classifiers are active there is, as expected, a considerable impact on the prototype, that is, it takes on average more time to classify an instance. Classifying an instance with all active classifiers takes an average of 79.05 ms. KNN consumes significantly more time than the other classifiers, approximately 71.70 ms. KNN needs an average of 70 ms more time than NB and HT to classify an instance. For this reason all combinations of classifiers where KNN is active take longer to classify an instance. The combination NB with HT is the combination of classifiers that takes the least time to classify an instance, taking an average of 2.25 ms. With these values it is possible to conclude that the KNN is the classifier that takes the most time to classify an instance and thus that each combination where the KNN is active will take more time when compared to the combination where the KNN is inactive. NB and HT, individually, are the ones that take the least time to classify an instance, and the HT, although by a rather small difference, is the one that takes the least time.

4.4 Summary

This chapter presents an explanation of the investigation methodologies in order to solve our problem and the way the entire working environment was built. In addition, the architecture of the whole project developed for the final solution is presented. This chapter focuses especially on the prototype, its operation, its graphical interface and the overhead that the real-time classification has on the prototype.

Chapter 5

Experiments and Results

This chapter describe the main experiments carried out in this dissertation. Also shown are the results of these experiments as well as the conclusions drawn from each one.

5.1 Data Visualization

It is important to have a graphical perspective on the data that we have to work with so that we can fully understand them. Looking at the data from a graphical point of view makes it easier to read and consequently allows us to make analyzes and draw conclusions about them.

The figure 5.1 shows the accelerometer data charts for two different activities. The top chart refers to the "Running" activity and the bottom graph represents the "Walking" activity.

Analyzing the figure 5.1 it is possible to observe significant differences between the two activities, which is why the charts were useful for analyzing the data.

These plots were used throughout the experiment phase and proved to be very useful in explaining and solving doubts, problems and obstacles that arose naturally and which were related to the data used.

Experiments and Results

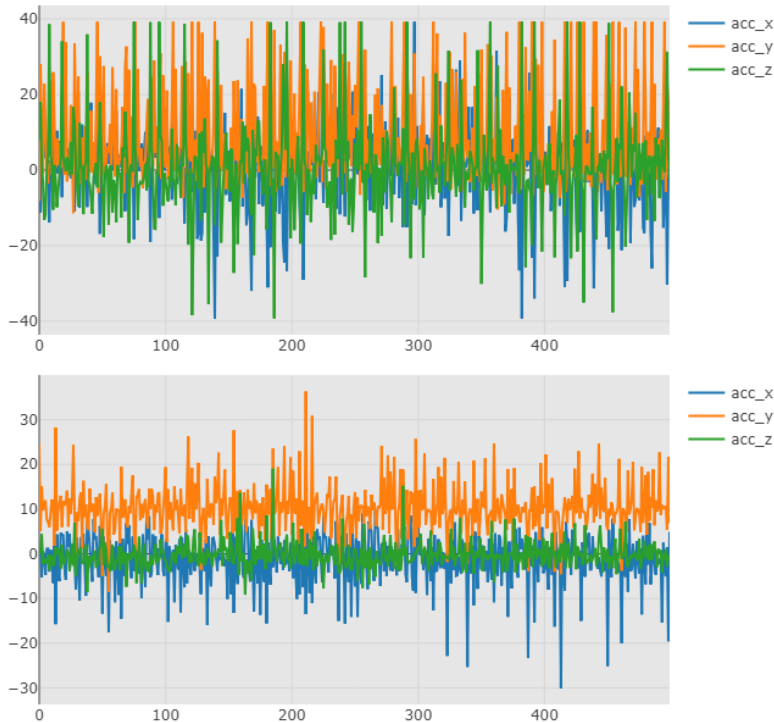


Figure 5.1: Plot of the accelerometer data which represents the activity "Running" (top) and the activity "Walking" (bottom). Each unit represents a single sensor reading.

5.2 Battery Consumption

The goal of this experiment is to measure the battery consumption while playing media files stored locally or available online on the YouTube platform.

To measure the consumption of the battery has been developed an application in Java, capable of playing video and audio files stored in the internal memory of the smartphone and also videos that are available on YouTube [You].

To play the files, the Android API MediaPlayer [And18b] class is used. The MediaPlayer [And18b] class can be used to control the playback of an audio / video files and streams. The YouTube API [You15] was used to play the videos of the platform in the application developed.

To measure battery consumption while playing multimedia files and with real-time classification, a feature has been added to the prototype that allows it to record the battery level throughout the file playback.

While playing the files, the application and the prototype registers the battery level every minute until the file is played back. When the playback of a file ends, the application and the prototype save in a csv file the battery level recorded every minute during the playback.

Experiments and Results

To carry out the measurements we used a Samsung Galaxy J5 smartphone, with a battery capacity of 3000 mAh. No other app was running in the background and all the experiments were conducted in the same conditions.

5.2.1 Video vs Audio (Local)

For the accomplishment of this experience, it was chosen a video of a show with the duration of 2:00:30 hours. Subsequently, the file was split into mp4 (590 MB) and mp3 (165 MB) formats. The files were stored in the internal memory of the smartphone. Figure 5.2 shows the measurements obtained. During this experience the Wi-Fi was off.

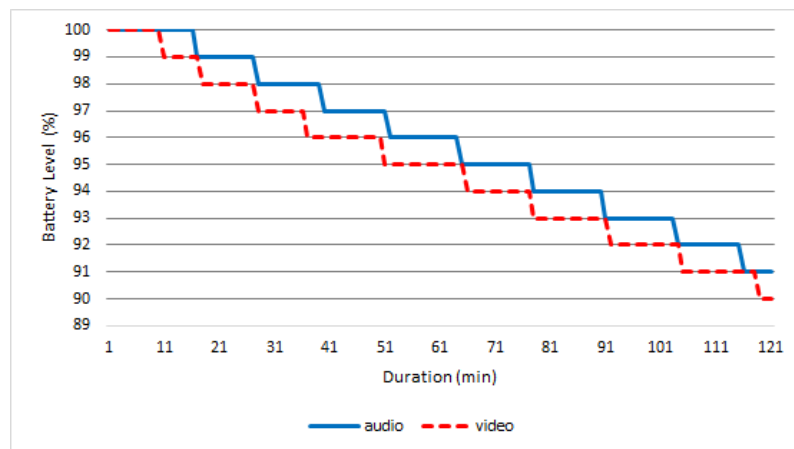


Figure 5.2: Battery consumption during playback of a video and audio file. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.

From the analysis of the results in Figure 5.2 it is possible to verify that the reproduction of the video file consumes more battery when compared to the respective audio file. Based on the analysis of the chart it is possible to conclude that switching from video to audio saves the battery resources of the smartphone. Assuming that the user starts playing the video when the battery level is at 100% and continues playing until the battery level reaches 0%, when switching to the audio file the user will get an additional 5 hours of battery life. This value was obtained assuming that the battery discharge is linear with the time and the battery level.

5.2.2 Local Video vs YouTube Video

For this experiment it was used the same mp4 file that was in the previous experiment and its respective version on YouTube. The mp4 file was stored in the internal memory of the smartphone. During the measurement of the local file, the Wi-Fi was disabled. Figure 5.3 shows the measurements obtained.

Analyzing the results in Figure 5.3 it is possible to verify that the reproduction of the YouTube video consumes more battery when compared to the reproduction of the respective local video file.

Experiments and Results

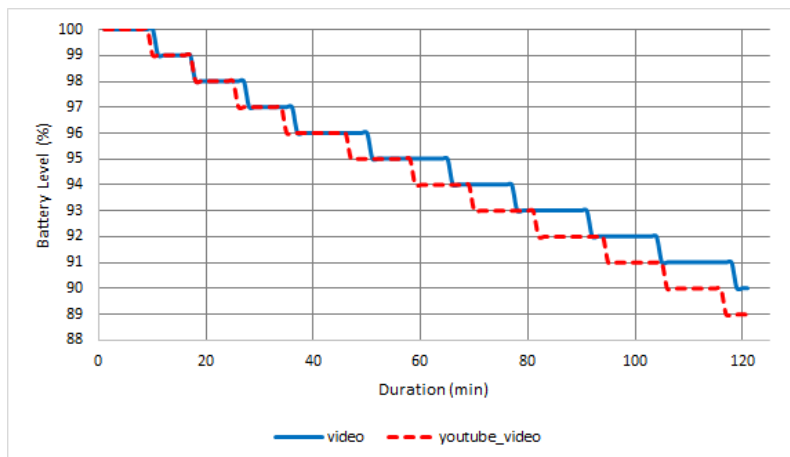


Figure 5.3: Battery consumption during playback of a local video and a video on YouTube. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.

This extra consumption is due to the fact that the smartphone is using the Wi-Fi network to play the video. Assuming that the user starts playing the YouTube video when the battery level is at 100% and continues playback until the battery level reaches 0%, switching to the local file the user will get an additional 3.3 hours of battery life. This value was obtained assuming that the battery discharge is linear with the time and the battery level.

5.2.3 YouTube Video vs YouTube Video with Static Image (Black)

For this experiment it was used a normal YouTube video and another one where throughout the video the image did not change, in this case a whole black background. The chosen video has approximately the same duration, in this case 2 hours. Figure 5.4 shows the measurements obtained.

Experiments and Results

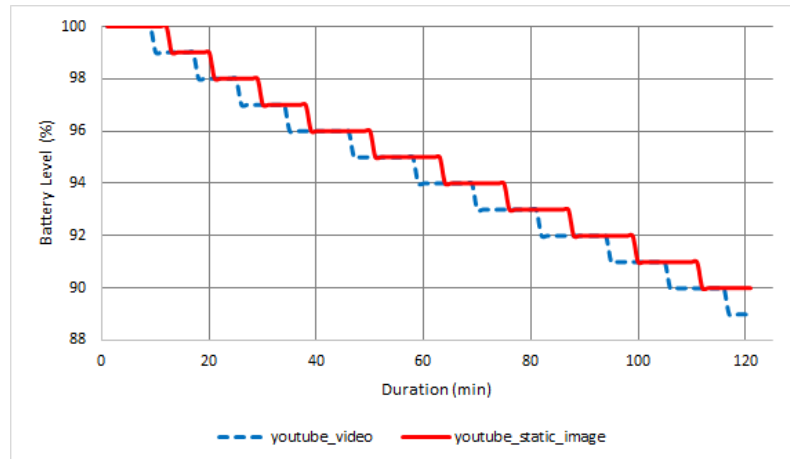


Figure 5.4: Battery consumption during playback of a YouTube video and a YouTube video with a static image. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.

Analysis of the results in Figure 5.4 it is possible to verify that the reproduction of a normal YouTube video consumes more battery when compared to the reproduction of a YouTube video with a static image. Assuming that the user starts playing a normal YouTube video when the battery level is at 100% and continues playback until the battery level reaches 0%, switching to a video with a static image the user will get an additional 3.3 hours of battery life. This value was obtained assuming that the battery discharge is linear with the time and the battery level.

5.2.4 Local Video vs Local Video with Classifiers

The main goal of this experiment is to evaluate the battery consumption associated with playing a video along with the real-time classification and compare it with just the video playback. In the experiment the same mp4 file was used for both situations. The experiment was conducted on the prototype that has been modified to store in a CSV file, the battery level every minute during playback of the video file. For this experiment a frequency of 5 Hz was defined, with an overlap of 0% and a buffer of size 100. During the measurements, the Wi-Fi was disabled. Figure 5.5 shows the measurements obtained.

Experiments and Results

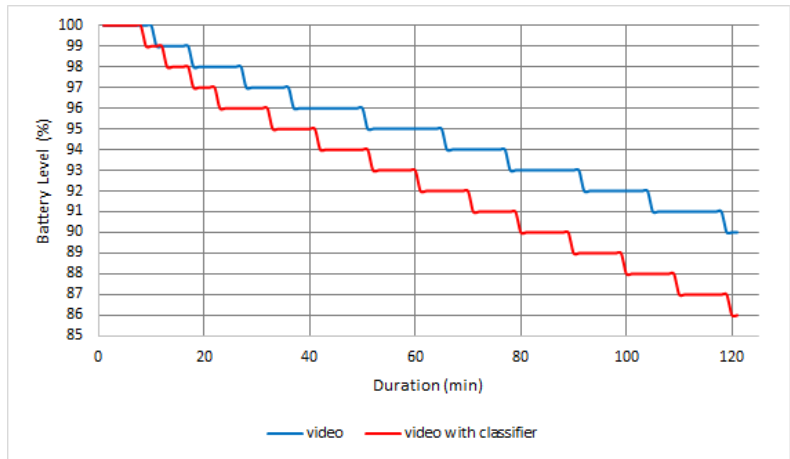


Figure 5.5: Battery consumption during playback of a local video and a local video with real time classification. In this measurement was used only the ensemble classifier. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.

From the analysis of the results in Figure 5.5 it is possible to conclude that the use of the classifiers considerably increases the energy consumption during the visualization of a video file. This extra consumption is due to the fact that the sensors are constantly providing data to the classifiers and also due to the classification process itself. Assuming a battery charge starts at 100% and goes up to 0%, use of the real-time classification decreases the time of use of the smartphone by 5.7 hours when compared to viewing a video but without classification. This value was obtained assuming that the battery discharge is linear with the time and the battery level.

5.2.5 Local Audio vs Local Audio with Classifiers

The main goal of this experiment is to evaluate the battery consumption associated with playing a audio file along with the real-time classification and compare it with just the audio file playback. In the experiment the same mp3 file was used for both situations. The experiment was conducted on the prototype that has been modified to store in a CSV file, the battery level every minute during playback of the video file. For this experiment a frequency of 5 Hz was defined, with an overlap of 0% and a buffer of size 100. During the measurements, the Wi-Fi was disabled. Figure 5.6 shows the measurements obtained.

Experiments and Results

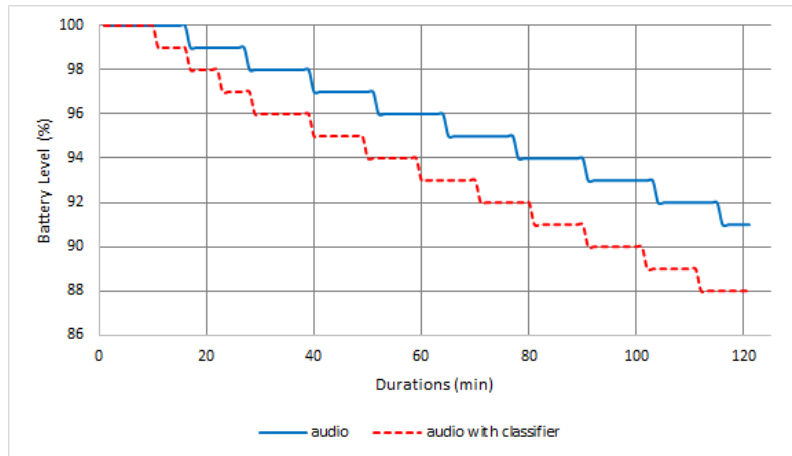


Figure 5.6: Battery consumption during playback of a local audio file and a local audio file with real time classification. In this measurement was used only the ensemble classifier. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.

As in the previous situation, the use of real-time classification leads to higher energy consumption. Assuming a battery charge starts at 100% and goes up to 0%, use of the real-time classification decreases the time of use of the smartphone by 5.5 hours when compared to listen to audio but without classification. This value was obtained assuming that the battery discharge is linear with the time and the battery level.

5.2.6 Local Video with Classifiers vs Local Audio with Classifiers

Using the data from the previous experiments for the use of the real-time classification for audio and video, results in the Figure 5.7.

Experiments and Results

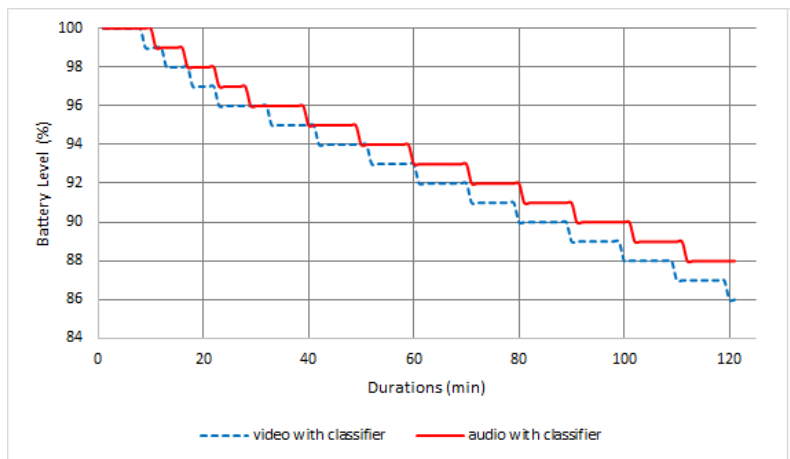


Figure 5.7: Battery consumption during playback of a local audio file with real time classification and a local video file with real time classification. In this measurement was used only the ensemble classifier. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.

From the analysis of the results in Figure 5.7 it is possible to verify that the reproduction of the video file with real-time classification consumes more battery when compared to the respective audio file. Based on the analysis of the Figure 5.7 it is possible to conclude that switching from video to audio will save the battery resources of the smartphone. Assuming that the user starts playing the video when the battery level is at 100% and continues playing until the battery level reaches 0%, when switching to the audio file the user will get an additional 2.4 hours of battery life. This value was obtained assuming that the battery discharge is linear with the time and the battery level.

5.2.7 Local Video and Audio with and without Classifiers

The following chart was obtained using measurements from previous experiments. The experiments used were: local video; local audio; local video with classifiers; local audio with classifiers.

As has been concluded in previous experiments, real-time classification contributes to higher battery consumption. With a duration of 2:00:30 h, a sampling frequency of 5 Hz, an overlap of 0% and with a buffer size of 100, 361 classifications were performed during the measurements, i.e., per minute were performed, approximately 3 classifications. Reducing the frequency of classifications would reduce battery consumption associated with real-time classification.

One way to reduce the frequency with which classifications are made would be to reduce the frequency of the sensors or increase the size of the buffer that stores the sensor values. In this way, the number of classifications would be reduced, but the classification process would remain constant, that is, despite reducing the number of classifications, it would increase the time the application takes to fill the buffers, not resulting in a substantial reduction in battery consumption.

Another way is to activate the classification process only if a particular event is triggered, such as the user switching off the screen or selecting another file to play. In this way the classification

Experiments and Results

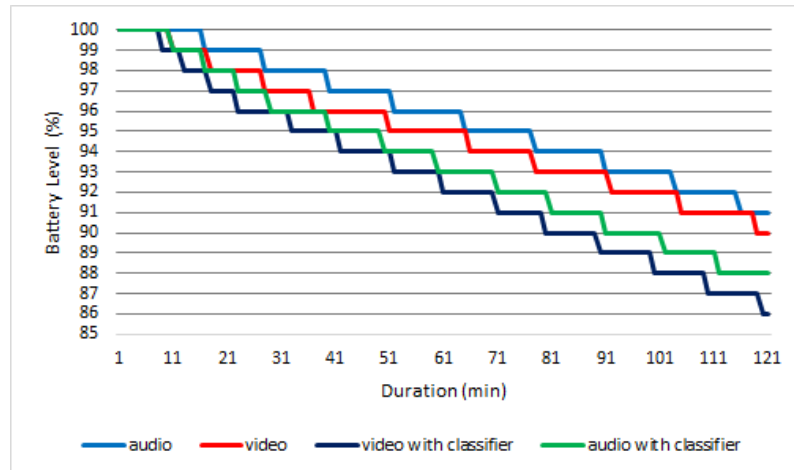


Figure 5.8: Battery consumption for local video and audio with and without real-time classification. For measurements with classifiers the ensemble was used. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.

process would not be continuous which translated in a reduction in the number of classifications and in the time that the application consumes to fill the buffers with the data of the sensors. With this approach it would be possible to substantially reduce battery consumption associated with real-time classification.

5.2.8 Battery Consumption per Classifier

This experiment aims to analyze energy consumption per classifier. This experiment aims to find out which of the classifiers used has the most impact on energy consumption. For this, five measurements of the battery consumption will be conducted at the same time that a video file is being played, one for each individual classifier, another for ensemble (with all classifiers) and one for video only. The same mp4 file used in previous experiments was used. For these measurements, a frequency of 5 Hz was chosen with 0% of overlap and a buffer size of 100.

All experiments were performed under the same conditions. No application was running in the background beyond the prototype. During the measurements, the Wi-Fi was disabled. Figure 5.9 shows the measurements obtained.

Experiments and Results

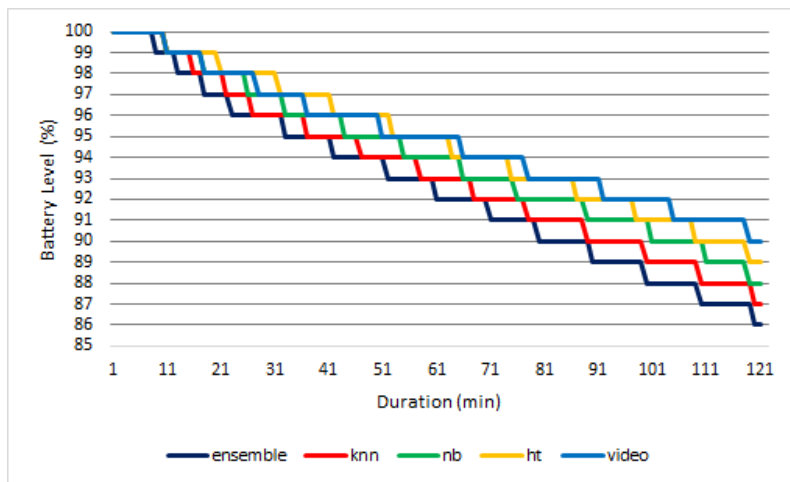


Figure 5.9: Battery consumption for the ensemble classifier and for each of the individual classifiers while playing a video. The chart line has a ladder effect due to the resolution of the values being one unit. Since the resolution is one unit, whenever a battery level drops by 1% a step is generated.

Analyzing the chart it is possible to conclude that the ensemble classifier is the one that consumes the most battery when compared to each of the individual classifiers. Individually the KNN is the one that consumes more battery. The individual classifier that consumes less energy during real-time classification is HT.

Table 5.1 summarizes in hours how long a battery charge would last if a video was played with each of the classifiers tested until the battery reaches the end (assuming the battery is fully charged at the beginning of the playback).

Table 5.1: Time, in hours, that the playback of a video along with each of the classifier would take to completely discharge the battery.

	Just Audio	Just Video	Ensemble	KNN	NB	HT
Hours	22.2	20	14.3	15.4	16.7	18.1

This value was obtained assuming that the battery discharge is linear with the time and the battery level.

5.3 Experiments

In this set of experiments it is tested the accuracy of four types of classifiers: Ensemble, Naive Bayes, Hoeffding Trees and KNN. The Ensemble classifier used in all experiments combines the three classifiers used in this dissertation: Naive Bayes, Hoeffding Trees and KNN.

This Ensemble is based on the confidence that each classifier has in its prediction, that is, if a classifier is 99% certain about the label of an activity, but the other two agree with another label with only 30% certainty, then it is not absurd to infer that the first classifier's opinion should be

Experiments and Results

taken into account, despite its numeric disadvantage. In this case there will always be a classification even if all the classifiers disagree, always prevailing the classifier with greater confidence.

Before starting the experiments to test the accuracy of the classifiers, a study was made to find the window size that best results allows to obtain. For this, the dataset was divided into two distinct sets: the training set (80% of the dataset) and the test set (20% of the dataset). Next, the accuracy of the classifiers for different window sizes was studied. Figure 5.10 shows the results obtained.

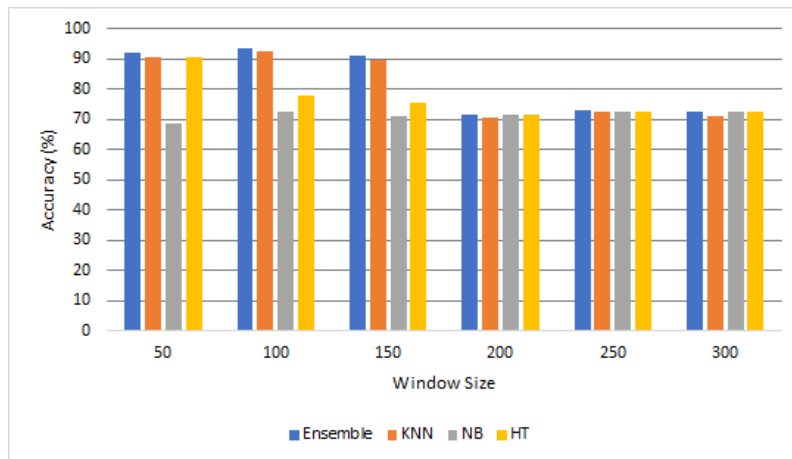


Figure 5.10: Accuracies of the ensemble and of each of the individual classifiers for the different window sizes studied. The window sizes studied range from 50 to 300. A 0% overlap was used. For the kNN the value of k used was 3 ($k=3$).

Analyzing the results in Figure 5.10 it is possible to conclude that the size of the window that best accuracies allows to obtain is size 100. Based on these results it was decided to use window size 100 for all the following experiments.

Tables 5.2 and 5.3 present the characterization of the train and test sets with respect to the number of samples per activity and also the number of windows obtained by activity for the window size chosen for the experiments (window size = 100).

Table 5.2: Number of samples and windows per activity for the train set. The training set corresponds to 80% of the dataset. The window size is set to 100.

Activity	Samples	Windows
Audio	65982	659
Video	65978	659
Other	65984	659

Experiments and Results

Table 5.3: Number of samples and windows per activity for the test set. The test set corresponds to 20% of the dataset. The window size is set to 100.

Activity	Samples	Windows
Audio	16500	165
Video	16500	165
Other	16500	165

5.3.1 Global Accuracy

This first experiment has as main objective to study the accuracy of the classifiers. All data were used in this experiment, without taking into account the scenario to which they belonged.

In this experiment, the data is handled as a stream. As such, a fixed length sliding window was used to iterate the data in the order it was recorded. The window size was set to 100, because it presented the best and most consistent results. Since the data was recorded at a frequency of 5Hz, this means that a window with a size set to 100, possesses 20 seconds of data.

This experiment consisted of splitting the dataset by a split factor (80% for training and 20% for test), creating a train set and a test set, but taking into account that each set must have considerable samples of each activity, to avoid a training set to have no samples of a particular activity. Then, these sets are pre-processed into ARFF files and fed to the classifiers.

Ensemble and their individual classifiers were tested. The results are presented in Table 5.4.

Table 5.4: Accuracy for each of the classifiers. For the kNN the value of k used was 3 ($k = 3$).

%	Ensemble	KNN	NB	HT
Accuracy	93.54	92.53	72.73	77.78

The best result was obtained from the Ensemble Classifier, with an accuracy of 93.54%. Individually, the KNN performed better when compared to the other two classifiers. Another interesting observation is how the Ensemble Classifier and the KNN obtained very similar results.

In order to understand the contribution that the individual classifiers had in Ensemble's classification, we present the following results. Figure 5.11 takes into account only the instances that were correctly classified.

Experiments and Results

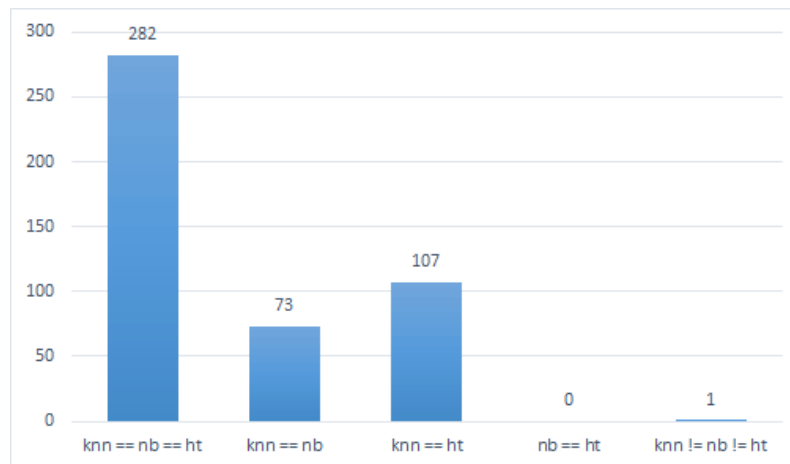


Figure 5.11: Correct Classifications by Combining Classifiers.

Of the 495 windows tested, 463 were correctly classified, which corresponds to an accuracy of 93.54%, as shown in the previous table. From the analysis of the Figure 5.11 it is possible to conclude that in 61% of the times (282) all the classifiers agreed on the classification. In situations where there was no unanimity, the combination of classifiers that contributed the most to a correct classification of the sample was the KNN and Hoeffding Trees. They contributed to the correct classification of 23% of the windows. Note that the combination of Naive Bayes and Hoeffding Trees did not contribute to any correct classification.

Only once did the classifiers disagree with the classification. In this case the classification assumed by the Ensemble was obtained by Naive Bayes, since it showed a greater confidence when compared with the others. The Naive Bayes in addition to having shown greater confidence in its classification was the only one that correctly classified the window.

In the samples that were correctly classified it is highlighted the fact that there was no situation where the classification of a classifier prevailed in relation to the majority. In the cases where the individual classification prevailed over the majority, it eventually led to an incorrect classification of the window.

5.3.2 Accuracy by Scenario

The main objective of this set of experiments is to test the accuracy of the classifiers for each of the scenarios considered in this dissertation.

In this set of experiences, the data were divided in their respective scenarios. The main objective is to analyze each scenario individually. A small program was developed in Java to separate the data by scenario. The execution of the program resulted in a set of data for each scenario that is tested individually.

For these experiments the training set was the same used in the previous experiment, 80% of all data available in the dataset, that is, the data of all the scenarios. For the test set we used only the data referring to the scenario under study and that were not part of the training set.

Experiments and Results

As in the previous experiment a window of size 100 and an overlap of 0% is used. Ensemble and their individual classifiers are tested.

5.3.2.1 Walking

In this specific experiment only the data referring to the “Walking” scenario (see Section 3.1.1 for more information about this scenario) are tested. This experiment has as main goal to study the accuracy of the classifiers only for Walking scenario. In this scenario there is data for when the user is watching a video, listening to audio or none of the above. The results are presented in Table 5.5.

Table 5.5: Accuracy for the Walking Scenario. For the kNN the value of k used was 3 ($k = 3$).

%	Ensemble	KNN	NB	HT
Accuracy	98.89	98.89	82.78	82.78

Ensemble and KNN achieved the best results, with an accuracy of 98.89%. The Hoeffding Trees and Naive Bayes obtained the same results, but inferior when compared with the other classifiers. Using only the data referring to the "Walking" scenario there was an improvement in the accuracy when compared to the results obtained in the first experiment.

5.3.2.2 Running

In this specific experiment only the data referring to the “Running” scenario (see Section 3.1.2 for more information about this scenario) are tested. In this scenario there is data for when the user is and when he is not listening to audio (see Table 5.6).

Table 5.6: Accuracy for the Running Scenario. For the kNN the value of k used was 3 ($k = 3$).

%	Ensemble	KNN	NB	HT
Accuracy	100	100	100	100

The results obtained were the same for all the classifiers tested, all of them obtained 100% accuracy. As in the previous scenario, there were also improvements in classifier accuracies.

5.3.2.3 Sitting on the Couch

In this specific experiment only the data referring to the “Sitting on the Couch” scenario (see Section 3.1.3 for more information about this scenario) are tested. In this scenario there is data for when the user is watching a video, listening to audio or none of the above (see Table 5.7).

Table 5.7: Accuracy for the Sitting Scenario. For the kNN the value of k used was 3 ($k = 3$).

%	Ensemble	KNN	NB	HT
Accuracy	92.59	90.37	44.44	67.41

Experiments and Results

The best result was achieved using the ensemble of the three individual classifiers, with an accuracy of 92.59%. Individually the KNN was the classifier that produced the best results, registering an accuracy of 90.37%.

The worst result was obtained by Naive Bayes, which obtained an accuracy of 44.44%, a very low value. Hoeffding Trees, though performing better than Naive Bayes, was far below ensemble and KNN, with an accuracy of 67.41%.

5.3.2.4 Kitchen

In this specific experiment only the data referring to the “Kitchen” scenario (see Section 3.1.4 for more information about this scenario) are tested. In this scenario there is data for when the user is watching a video, listening to audio or none of the above (see Table 5.8).

Table 5.8: Accuracy for the Kitchen Scenario. For the kNN the value of k used was 3 ($k = 3$).

%	Ensemble	KNN	NB	HT
Accuracy	80.00	78.89	66.67	63.33

In this experiment the ensemble obtained an accuracy of 80.00%. Individually the KNN obtained better results than the other classifiers, with an accuracy of 78.89%. Naive Bayes and Hoeffding Trees were very close, with Hoeffding Trees registering the worst result, with an accuracy of 63.33%.

5.3.2.5 Conclusions

In this set of experiments, the main objective was to study the accuracies for each individual scenario. Based on the results obtained for the scenarios under study, it is possible to conclude that the ensemble classifier obtained the best result in all the scenarios under study and allowed to improve accuracy by 1.67% (average value) when compared to the best accuracy obtained by individual classifier.

Analyzing the results for the individual classifiers, it is possible to conclude that the KNN presents better accuracies when compared with the other classifiers used. The KNN achieved the best individual result in all the scenarios under study. In the experiments performed Naive Bayes and Hoeffding Trees always obtained the worst accuracies. In general, it can be concluded that Naive Bayes obtained the worst results among all used classifiers.

5.3.3 Impact of Frequency Reduction

The main objective of this set of experiments is to evaluate if the reduction of the sampling frequency has an impact on the accuracy of the classifiers. To perform these experiments it was necessary to reduce the frequency of the data that was collected.

As already mentioned, the data used in this dissertation were collected at a sampling frequency of 5 Hz. To reduce this frequency, for the frequencies that were intended to study, a script was

Experiments and Results

developed in Python. This developed script removes samples of the data with the initial frequency. For example to reduce the data frequency from 5 Hz to 2.5 Hz, every two samples the script removes one.

Two frequency reductions were made: in the first experiment the sampling frequency was reduced from 5 Hz to 2.5 Hz; in the second experiment the sampling frequency was reduced from 5 Hz to 1.66 Hz.

In this first experiment the sampling frequency was reduced from 5 Hz to 2.5 Hz. Accuracy tests were done for all data (global) and for each of the individual scenarios: Walking, Running, Sitting and Kitchen. For this, a Java program was developed to separate the data by scenario. The results are shown in Figure 5.12.

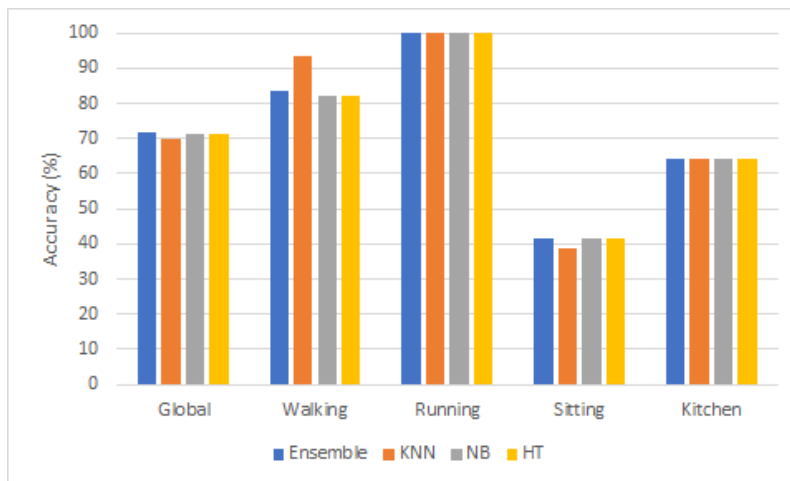


Figure 5.12: Accuracies of the ensemble and each individual classifier for each scenario studied. These accuracies were obtained for a frequency reduction from 5 Hz to 2.5 Hz. A window of size 100 and with an overlap of 0% was used.

From the analysis of the data it is possible to verify that with the exception of the Running scenario, the accuracies lowered significantly in all the classifications. In the Running scenario all classifiers maintained the 100% accuracy that had been initially recorded. The scenario that suffered most from the reduction of frequency was Sitting, where the best result was for the ensemble classifier with 41.79%.

In this second experiment the frequency was reduced from 5 Hz to 1.66 Hz. Accuracy tests were done for all data (global) and for each of the individual scenarios: Walking, Running, Sitting and Kitchen. To do this, the Java program used in previous experience was used to separate the data by scenario. The results are shown in Figure 5.13.

Experiments and Results

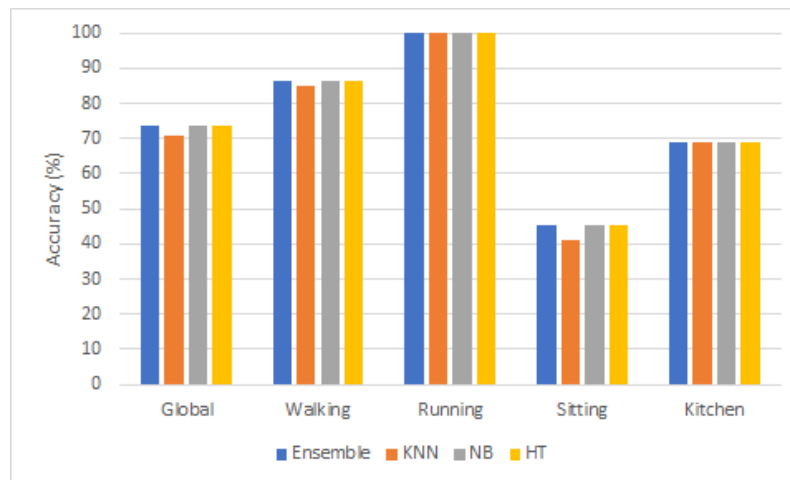


Figure 5.13: Accuracies of the ensemble and each individual classifier for each scenario studied. These accuracies were obtained for a frequency reduction from 5 Hz to 1.66 Hz. A window of size 100 and with an overlap of 0% was used.

Analyzing Figure 5.13 it is possible to conclude that once again the frequency reduction did not affect the accuracy for the Running scenario, remaining at 100%. On the other hand, the Sitting scenario was again the most affected with the reduction of frequency, with the accuracies falling below 50%. Except for the Running scenario, this frequency reduction significantly lowered the accuracies of the classifiers for the other scenarios studied.

5.3.3.1 Conclusions

Figure 5.14 summarizes the accuracies of the classifiers for each of the sampling frequencies studied.

Experiments and Results

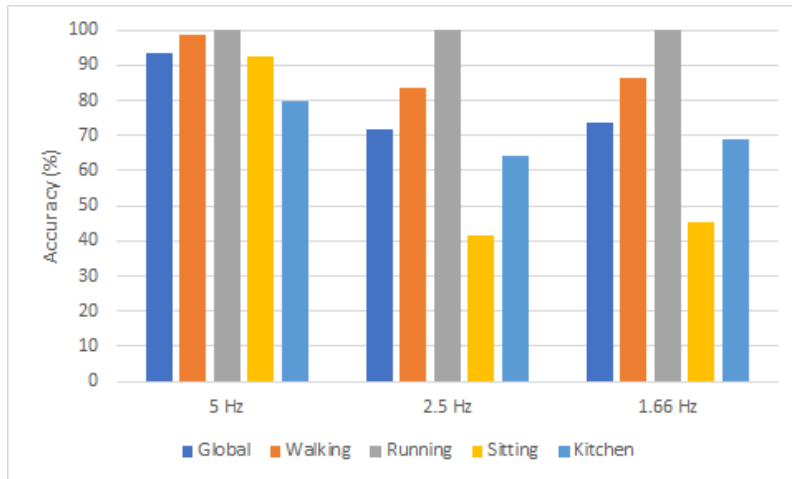


Figure 5.14: All sampling frequencies studied: the frequency at which data was collected (5 Hz) and the two frequency reductions performed (2.5 Hz and 1.66 Hz). Each frequency has 5 bars, 4 for each scenario studied and 1 for all data. The accuracies presented concern only the ensemble classifier. A window of size 100 and with an overlap of 0% was used.

When comparing the reductions to 2.5 Hz and to 1.66 Hz, there was a small improvement in all classifiers for the 1.66 Hz reduction. There were improvements on the order of 3% between the two experiments. With this it can be concluded that reducing to frequencies below 2.5 Hz seem to not have improvements on the accuracy of the classifiers.

In general, frequency reduction negatively affects the accuracies of the classifiers. However, the accuracies for the Running scenario were not affected by the frequency reduction to 2.5 Hz and 1.66 Hz, since in both reductions the accuracies remained at 100% for all classifiers.

The Sitting scenario suffered a significant reduction in accuracy in both reductions. In both cases the accuracies fell below 50%. In the Kitchen scenario, there were also significant falls in accuracies, however, there was no such abrupt drop when compared to the Sitting scenario. Still the accuracies dropped to less than 70% in both reductions for the Kitchen scenario.

By analyzing Figure 5.14, it is possible to conclude that the scenarios where there are fewer movements (Sitting and Kitchen) were the ones that were most affected by the frequency reductions when compared to the scenarios where there is more movement (Running and Walking).

It is possible to conclude that for the scenarios where there are fewer movements the frequency reduction is not feasible, since they register more pronounced falls in the accuracies, dropping below 50% in the Sitting scenario. On the other hand, even though there were decreases in accuracy, frequency reduction did not show a great impact for the Running and Walking scenarios. In these scenarios the accuracies remained above 80%.

One way to reduce the impact that lower frequencies have on accuracy is to reduce the size of the window. A frequency of 5 Hz and a window size of 100 means that each window contains approximately 20 seconds of data. In order to maintain the number of seconds in each window for lower frequencies, one solution is to also reduce the size of the window. For the frequency of 2.5

Experiments and Results

Hz, a window of 50 holds the 20 seconds of data. For the frequency of 1.66 Hz to maintain the 20 seconds a window size of 30 is used.

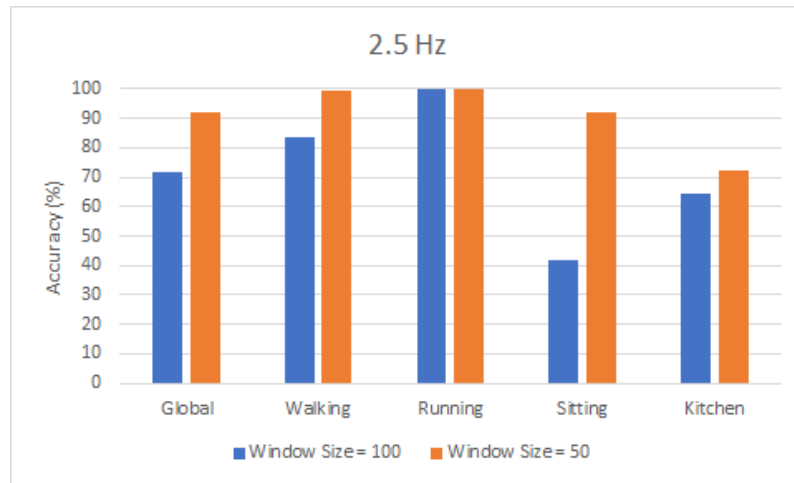


Figure 5.15: Frequency reduction from 5 Hz to 2.5 Hz. For each scenario, the accuracies for the ensemble classifier are presented for the two window sizes used. The size 100 window contains 40 seconds of data and the size 50 window has 20 seconds of data. A 0% overlap was used for both windows.

From the analysis of the Figure 5.15 and the Figure 5.16 it is possible to conclude that by keeping the number of seconds of data, that is, the size of the window is reduced, it is possible to significantly improve the accuracies for the different scenarios. Highlight for the Sitting scenario whose accuracy of ensemble classifier registered an improvement in the order of 50%.

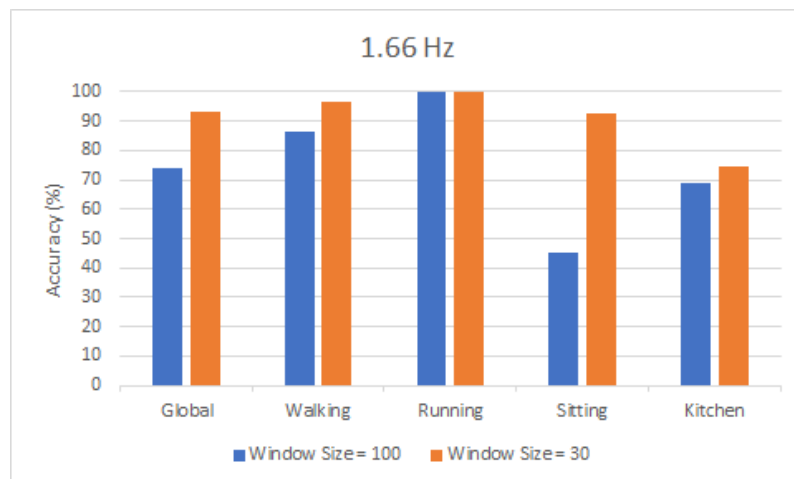


Figure 5.16: Frequency reduction from 5 Hz to 1.66 Hz. For each scenario, the accuracies for the ensemble classifier are presented for the two window sizes used. The size 100 window contains 60 seconds of data and the size 30 window has 20 seconds of data. A 0% overlap was used for both windows.

Experiments and Results

The Figure 5.17 summarizes the results obtained for each studied sampling frequency but maintaining the number of seconds per window according to the sampling frequency.

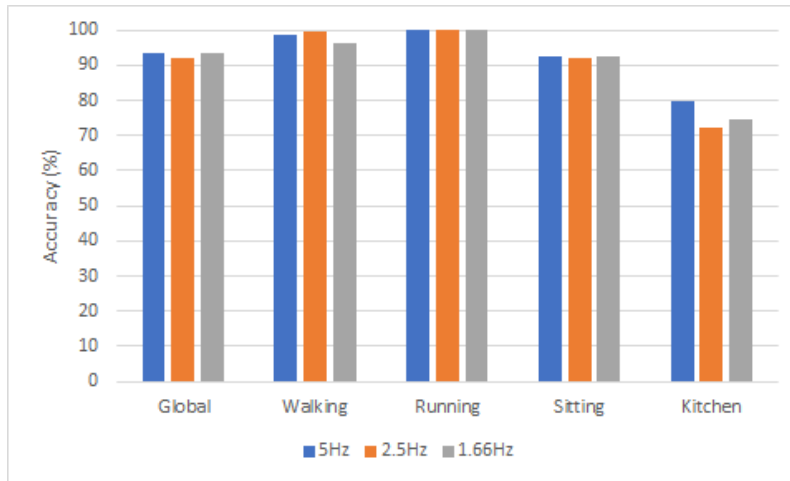


Figure 5.17: Accuracy of the ensemble classifier for the various frequencies studied. The window size for the frequency of 5 Hz is 100, for the frequency of 2.5 Hz is 50 and for the frequency of 1.66 Hz is 30. A 0% overlap was used for all frequencies windows.

Figure 5.17 allows to conclude that, maintaining the number of seconds of data, the different frequencies studied maintain very similar accuracies, not registering significant reductions. In this way it is possible to reduce the frequencies of the data without having a significant impact on the accuracies.

5.3.4 Impact of the use of Headsets

Initially only the sensor data from the smartphone was used to construct the dataset. However, during the stage of identification of the scenarios, it was verified that many situations contemplated the use of the headsets by the user during the accomplishment of certain activity. For this reason it was decided to include the values coming from the Android API that reveal whether or not the headset is connected to the device.

This experiment has as objective to study the impact that the addition of the state values of the headsets (connected or disconnected) had in the accuracy. For this, we use all the data in the dataset. Two experiments are carried out: the first experiment takes into account all the data collected including the data of the state of the headsets; the second experiment takes into account all data collected but ignoring the state of the headset data. In other words, the input from the Android API that reveals whether or not the headset is connected is disabled, either in the training set or in the test set. To be able to perform these experiments, the possibility of activating or deactivating the desired sensors was added to the desktop program. In this case only the one corresponding to the headset has been deactivated.

Experiments and Results

In this way the experience with the values of the headsets have seven sensors with three dimensions and three sensors of a dimension, resulting 76 features. In the experience without the values of the headset we have seven sensors with three dimensions and two sensors of a dimension, resulting in 74 features.

In order to perform this experiment, we assume that when the headset is connected to the smartphone, the user is listening, whether an audio or video file is being played. We did not consider situations where the user has the headset connected but is not listening at all.

As in the previous experiment a window of size 100 was used and the data were divided into 80% for training and 20% for testing. Ensemble and their individual classifiers were trained and tested. The results are presented in Table 5.9.

Table 5.9: Accuracy (%) with and without headsets difference between them.

%	With Headsets	Without Headsets	Absolute Difference
Ensemble	93.54	59.60	-33.94
KNN	92.53	58.79	-33.74
NB	72.73	43.43	-29.30
HT	77.78	33.54	-44.24

From the analysis of the obtained results, we can conclude that With Headsets are significant and consistently better than Without Headsets. The inclusion of the headset significantly improves the accuracy of Ensemble and each of the individual classifiers. Highlight for the classifier Ho-ffding Trees that got an average improvement of 44.24%. From the results it can be concluded that the addition of headset state data is an added value for this project since it has significantly improved the accuracy of the classifiers.

5.4 Summary

In this chapter we present all the experiments related to the battery consumption and the accuracy of the classifiers that were carried out throughout this dissertation. The methods, results and conclusions of each experiment are presented. We tried to perform all the experiments on a step-by-step basis, so that each result obtained contributed to the conclusion of the proposed solution.

Experiments and Results

Chapter 6

Conclusion

This chapter summarizes the work that was done during this dissertation as well as the conclusions that were drawn from the experiments carried out. In addition, the chapter describes reviewed problems and issues yet to be solved, as well as some guidelines on how to tackle them.

6.1 Final Conclusions

The objectives of this dissertation were to investigate the methods that obtained the best results in the real-time classification, to study the overhead associated to the real-time recognition and to analyze the energy consumption of the reproduction of video files and audio with and without classification in real time.

At the beginning of this dissertation some user scenarios were identified where the user could be watching videos or just listening to audio. The scenarios identified were: Running, Walking, Sitting on the Couch and Cooking in the Kitchen. These scenarios served as a basis for constructing the dataset used in this project. The dataset was later collected by the author.

A series of experiments were carried out to study battery consumption. The experiments carried out aimed to: identifying if the video playback consumes more battery than playing an audio file; study the impact that real-time classification has on battery consumption; identify the classifier that consumes less battery.

Experiments were performed to test the accuracy of the ensemble and of each of the individual classifiers with the collected dataset. In a first experiment it was tested with all dataset. In a second experiment the accuracy of the classifiers was tested with the specific data of each scenario.

Finally, a prototype capable of reproducing video and audio files was developed. It is also able to classify activities in real time and based on the result of this rating switch between video and audio. Using the prototype measures were taken to assess the overhead that the real-time classification has on Android.

Conclusion

The experimental results show that the method that obtained the best results in the classification experiments was the ensemble classifier, obtaining an average accuracy of about 93% for all experiments. Individually the best performance classifier obtained was the kNN ($k = 3$), obtaining an average accuracy of about 92%. With ensemble it is possible to obtain on average an accuracy improvement of 1% when compared to the best single classifier (kNN).

The results of energy consumption experiments indicates and as expected that playing a video consumes more battery than playing an audio file. At the end of a full charge, switching the video over audio increases the battery life by 5 hours without real-time classification. On the other hand and with real-time classification, switching the video over the audio increases the battery life by 2.4 hours.

It is possible to conclude that, with the real-time classification with the ensemble classifier, there is a higher energy consumption. Real-time classification requires a 4% increase in energy consumption when used together with the playback of a video and a 3% when used together with the playback of a audio file. This results in an average decrease of 7 hours in the time of using a full battery charge using the ensemble classifier. Individually the HT is the classifier that has the least impact on battery consumption. This classifier consumes 1% extra when used in conjunction with playing a video.

Although high accuracies have been obtained for the ensemble classifier, from the energy point of view it is necessary to reduce the consumption associated with the classification in real time in order to save energy when switching from video to audio. Otherwise it does not compensate for switching from video to audio, since the extra consumption associated with real-time rating is higher than the savings obtained by switching from video to audio.

The real-time classification process has an average execution time of 98.95 ms. 16% of this time is used to extract the 76 features of the sensor data and 84% refers to the classification of the feature vector from the feature extraction phase.

It was also studied the impact that the reduction of the sampling frequency has on the accuracies of the classifiers. From the results it was possible to conclude that reducing the frequency while maintaining window size significantly reduces the accuracy of the classifiers. There was a fall of approximately 20% in ensemble accuracy for the global scenario. On the other hand, keeping the number of seconds per window (i.e., reducing the size of the window), the fall recorded in ensemble accuracy for the global scenario was less than 2%. Thus, it is possible to conclude that for frequency reduction to be feasible, from the point of view of the accuracies of the classifiers, it is necessary to reduce the size of the window in order to maintain the same number of seconds per window.

6.2 Future Work

Throughout the development of this dissertation, it was felt that there were some aspects that should have been dealt with, but if this were done it would not be possible to focus on the specific

Conclusion

objectives defined at the beginning of this dissertation. This section addresses these aspects as a possible way to improve the work already done.

As previously mentioned, the dataset used in this dissertation was constructed only by the author. Due to this fact it has become difficult to cover all possible use situations for each scenario. Enriching the existing dataset with the situations not contemplated initially would contribute to a better accuracy of the classifiers for the scenarios considered. In addition, only four scenarios were covered for this dissertation due to time constraints. A more comprehensive dataset that contemplates more usage scenarios would contribute to better real-time recognition and thus predict whether the user is watching a video or just listening to the audio more accurately.

One of the limitations of the dataset is that the data has been collected by only one user. Adding data from additional users would increase the variability of the dataset. In this way, other types of validation tests, such as Leave-One-Out Cross-Validation, could be performed and more consistent and valid results obtained.

For the Kitchen scenario, the user follows certain steps until he/she starts playing a media file. For example, if the user wants to watch a video, he switches the screen on, opens the file explorer, chooses the file he want to play, the player starts, the video starts to play, and finally the user places the smartphone on the table. These observation sequences can be used to help improve the accuracy of the classification process. Since the Hidden Markov Models [STJ14] allow capturing transitions between different activities, we believe that their use would improve the accuracy of the classifications for the Kitchen scenario. In addition to having to implement the Hidden Markov Models in the desktop application and the prototype, it is also necessary to collect data from the various sequences of steps and convert them to the format that the HMM classifier accepts.

Regarding the sampling frequency, the results obtained make it necessary to analyze even lower frequencies in order to better base the conclusions obtained.

Although a prototype of the intended solution has been developed in this dissertation, it is not yet completely finished. Certain aspects can still be further developed in order to increase the robustness of the application. The prototype only allows playback of local files. One way to make the prototype more complete would be to add the possibility of using the prototype with online streaming platforms such as YouTube.

Due to the fact that the prototype is constantly collecting and processing the data it receives from the sensors, there is a significant increase in energy consumption. One way to reduce the extra consumption caused by the constant classification would be to trigger data collection and consequent classification when a certain event was detected by the prototype. In this way, the prototype would not have to constantly collect data from the sensors, which would reduce the impact that the real-time classification process has on the smartphone's battery.

One way to reduce the computational weight that real-time classification has is to reduce the number of features that have to be calculated before being classified. It is therefore necessary to find out which features are most relevant and which lead to a better accuracy. In this way it is possible to discard the features that have less importance in the classification process, thus reducing the computational weight.

Conclusion

According to Zheng et al. [ZWR⁺17] lower sampling frequencies consume less energy. Based on this, the use of lower sampling frequencies in the prototype will lead to a reduction in the energy consumption associated with the real-time classification.

One way to improve the accuracy of online classifiers is to continually update models with new instances using an incremental approach. These new instances are used to extend the knowledge of existing models, that is, to further train the models. The MOA classifiers used in this dissertation are incremental, which will facilitate implementation. The biggest problem is whether the new instances, which will be used to update the models, were correctly classified by the classifiers. One way to resolve this problem is to ask the user if the instance has been properly classified. If it has been correctly classified it is used to update the models, otherwise it is discarded. In order not to be constantly bothering the user to confirm that the instance was correctly classified, it would be useful to establish how often the models should be updated.

References

- [AI13] Alvina Anjum and Muhammad U. Ilyas. Activity recognition using smartphone sensors. In *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*, pages 914–919, 2013.
- [AMD⁺15] Ferhat Attal, Samer Mohammed, Mariam Dedabrishvili, Faicel Chamroukhi, Latifa Oukhellou, and Yacine Amirat. Physical human activity recognition using wearable sensors. *Sensors*, 15(15):31314–31338, 2015.
- [And18a] Android. Android studio. <https://developer.android.com/studio/>, 2018. [Online; accessed 10-September-2018].
- [And18b] Android. MediaPlayer. <https://developer.android.com/reference/android/media/MediaPlayer>, 2018. [Online; accessed 10-December-2018].
- [BBS14] Andreas Bulling, Ulf Blanke, and Bernt Schiel. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Comput. Surv.*, 246(3):33:1–33:33, 2014.
- [BHKP10] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604, 2010.
- [Car16] Hugo Louro Cardoso. Predicting activities from smartphones. Master thesis in informatics and computing engineering, Faculdade de Engenharia da Universidade do Porto, July 2016.
- [CM17] Hugo Cardoso and João Mendes Moreira. Improving human activity classification through online semi-supervised learning. *Workshop StreamEvolv co-located with ECML/PKDD 2016*, CEUR Vol-2069:15–26, 2017.
- [CMV18] João MP Cardoso, João Moreira, and Luís Veiga. Contextwa - middleware and context inference techniques from data-streams for the development of context-aware services using mobile devices. *Impact*, 2018(1):18–20, 2018.
- [CN09] Liming Chen and Chris Nugent. Ontology-based activity recognition in intelligent pervasive environments. *International Journal of Web Information Systems*, 5(4):410–430, 2009.
- [CS17] Yufei Chen and Chao Shen. Performance analysis of smartphone-sensor behavior for human activity recognition. *IEEE Access*, 5:3095–3110, 2017.

REFERENCES

- [dOL12] Alexandre de Oliveira Lopes. Activity recognition from smartphone sensing data. Master thesis in informatics and computing engineering, Faculdade de Engenharia da Universidade do Porto, July 2012.
- [FDFC10] Davide Figo, Pedro C. Diniz, Diogo R. Ferreira, and João M. P. Cardoso. Pre-processing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14(7):645–662, 2010.
- [GCM⁺18] Kemilly Dearo Garcia, Tiago Carvalho, João Mendes Moreira, João M. P. Cardoso, and André C. P. L. F. de Carvalho. A preliminary study on hyperparameter configuration for human activity recognition. *CoRR*, abs/1810.10956, 2018.
- [Goo18] Google. Android developers. <https://developer.android.com/>, 2018. [Online; accessed 4-July-2018].
- [KHRM06] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng. Some effective techniques for naive bayes text classification. *IEEE transactions on knowledge and data engineering*, 18(11):1457–1466, 2006.
- [KIE12] Mustafa Kose, Ozlem Durmaz Incel, and Cem Ersoy. Online human activity recognition on smart phones. In *Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data*, pages 11–15, 2012.
- [KLG11] Simon Kozina, Mitja Lustrek, and Matjaz Gams. Dynamic signal segmentation for activity recognition. In *proceedings STAMI 2011, IJCAI 2011*, pages 93–98, 2011.
- [KMG⁺17] Bartosz Krawczyk, Leandro L. Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.
- [KS12] Sahak Kaghyan and Hakob Sarukhanyan. Activity recognition using k-nearest neighbor algorithm on smartphone with tri-axial accelerometer. *International Journal of Informatics Models and Analysis (IJIMA), ITHEA International Scientific Society, Bulgaria*, 1:146–156, 2012.
- [LaL13] Oscar D. Lara and Miguel a. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys Tutorials*, 15(3):1192–1209, 2013.
- [LCB06] Jonathan Lester, Tanzeem Choudhury, and Gaetano Borriello. A practical approach to recognizing physical activities. In *Pervasive Computing*, pages 1–16. Springer Berlin Heidelberg, 2006.
- [LZY⁺12] Yunji Liang, Xingshe Zhou, Zhiwen Yu, Bin Guo, and Yue Yang. Energy efficient activity recognition based on low resolution accelerometer in smart phones. In *Advances in Grid and Pervasive Computing*, pages 122–136. Springer Berlin Heidelberg, 2012.
- [MGARdlC15] Luis Miguel Soria Morillo, Luis Gonzalez-Abril, Juan Antonio Ortega Ramirez, and Miguel Angel Alvarez de la Concepcion. Low energy physical activity recognition system on smartphones. *Sensors*, 15(3):5163–5196, 2015.

REFERENCES

- [MRSS06] Uwe Maurer, Anthony Rowe, Asim Smailagic, and Daniel P. Siewiorek. *Location and Activity Recognition Using eWatch: A Wearable Sensor Platform*, pages 86–102. Springer Berlin Heidelberg, 2006.
- [Oza05] N.C. Oza. Online bagging and boosting. In *Systems, Man and Cybernetics, 2005 IEEE International Conference*, volume 3, pages 2340–2345. IEEE, 2005.
- [Plo18] Plotly. Plotly python open source graphing library. <https://plot.ly/python/>, 2018. [Online; accessed 15-December-2018].
- [Qui96] J. R. Quinlan. Bagging, boosting, and c4.s. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 1, AAAI'96*, pages 725–730. AAAI Press, 1996.
- [RB11] Daniele Riboni and Claudio Bettini. Cosar: hybrid reasoning for context-aware activity recognition. *Personal and Ubiquitous Computing*, 15(3):271–289, 2011.
- [Ris01] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligenc*, volume 3, pages 41–46, 2001.
- [SBI⁺15] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul J.M. Havinga. A survey of online activity recognition using mobile phones. *Sensors*, 15(1):2059–2085, 2015.
- [SCF⁺10] André C. Santos, João M.P. Cardoso, Diogo R. Ferreira, Pedro C. Diniz, and Paulo Chaínho. Providing user context for mobile and social networking applications. *Pervasive and Mobile Computing*, 6(3):324–341, 2010.
- [SR12] Pekka Siirtola and Juha Röning. Recognizing human activities user-independently on smartphones based on accelerometer data. *International Journal of Interactive Multimedia and Artificial Intelligence*, 1(5):38–45, 2012.
- [SSH16] Gheorghe Sebestyen, Ionut Stoica, and Anca Hangan. Human activity recognition and monitoring for elderly people. In *2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 341–347, 2016.
- [STJ14] Xing Su, Hanghang Tong, and Ping Ji. Activity recognition with smartphone sensors. *Tsinghua Science and Technology*, 19(3):235–249, 2014.
- [Utg89] Paul E. Utgoff. Incremental induction of decision trees. *Machine learning*, 4(2):161–186, 1989.
- [WDT16] Kishor Walsea, Rajiv Dharaskar, and Vilas Thakare. A study of human activity recognition using adaboost classifiers on wisdm dataset. *The Institute of Integrative Omics and Applied Biotechnology Journal*, 7(2):68–76, 2016.
- [You] Youtube. <https://www.youtube.com>. [Online; accessed 1-December-2018].
- [You15] Youtube. Youtube android player api. <https://developers.google.com/youtube/android/player/>, 2015. [Online; accessed 10-December-2018].

REFERENCES

- [Zhu08] Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):1–60, 2008.
- [ZL05] Zhi-Hua Zhou and Ming Li. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541, 2005.
- [ZWN⁺17] Shugang Zhang, Zhiqiang Wei, Jie Nie, Lei Huang, Shuang Wang, and Zhen Li. A review on human activity recognition using vision-based method. *Journal of Healthcare Engineering*, vol. 2017(Article ID 3090343):31 pages, 2017.
- [ZWR⁺17] Lingxiang Zheng, Dihong Wu, Xiaoyang Ruan, Shaolin Weng, Ao Peng, Biyu Tang, Hai Lu, Haibin Shi, and Huiru Zheng. A novel energy-efficient approach for human activity recognition. *Sensors*, 17(9):2064–2085, 2017.

Appendix A

A.1 Conversion of Labels

Table A.1 shows how the conversion between the labels obtained during the data acquisition phase and the labels used as training and test of the classifiers is done. This conversion is used in the desktop program and is mentioned in section 4.2.2.

Table A.1: Conversion of Labels between Files

CSV file label	Exemple	ARFF file label
<scenario_name>_audio	Running_audio	Audio
<scenario_name>_video	Walking_video	Video
<scenario_name>_other	Sitting_other	Other
<scenario_name>	Kitchen	Other