Università degli Studi di Padova
Facoltà di Ingegneria

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
CORSO DI LAUREA IN INGEGNERIA INFORMATICA
TESI DI LAUREA MAGISTRALE

ONE-CLASS SUBJECT IDENTIFICATION FROM SMARTPHONE-ACQUIRED WALKING DATA

Sistema di Riconoscimento Biometrico da Segnali di Camminate Acquisiti da Smartphone

---

Laureando:
Luca Merelli

Relatore:
Ch.mo prof. Michele Rossi
Correlatore:
Matteo Gadaleta

Anno Accademico 2015/2016
Padova, 7 Marzo 2016

One Class Subject Identification from Smartphone-Acquired Walking Data

by Luca Merelli

Abstract

In recent years, wearable technologies are one of the fastest growing segment of the so-called Internet of Things (IoT), which together with Wireless Sensor Network (WSN) technologies will enable Ubiquitous Sensing. Wearable devices have the distinct property to be worn by users, extending the IoT paradigm to a human-centric sensing scenario. Technologies advances allowed the integration of an increasing number of sensors, able to gather physiological and behavioral biometric information of a user. In this work, I am concerned with the design of a novel type of human identification system with the aim to recognize a user from the biometric traits of his way of walk. A smartphone located in the right pocket of the user is utilized to acquire motion data from the built-in sensors. In particular, accelerometer and gyroscope data are gathered and then processed through a cycle extraction phase, a Convolutional Neural Network (CNN) for feature extraction and a One-Class SVM (OSVM) classifier for the final identification. The proposed system is tested and from quantitave results it shows that is possibile to achieve an Equal Error Rate close to 1%.

Acknowledgements

I would like to express my greatest thanks towards Prof. Michele Rossi, for having agreed to be my supervisor during the course of my Master thesis work, and Matteo Gadaleta for the good advice, support and assistance during all the phases of this work.
In addition, I would like to dedicate this thesis to my beloved family especially my parents, who have given me unconditional support and have been close to me in both good times and difficult ones throughout my university career and beyond.
Finally, I would like to thanks my friends, with whom I shared my course of study.

Contents

List of Figures **iii**

1 Introduction **1**
1.1 Authentication on Smartphones 2
1.2 State of the Art of Gait Recognition 3
1.3 Motivations and Contributions 6

2 Data Acquisition and Preprocessing **9**
2.1 Data Acquisition 9
2.2 Data Preprocessing 11

3 Convolutional Neural Networks **17**
3.1 An overview on Neural Networks 18
3.1.1 Model of a Neuron 19
3.1.2 Artificial Neural Network Architectures 20
3.1.3 Learning the ANN Model 21
3.2 Convolutional Neural Networks 23
3.2.1 Deep Architectures Learning 23
3.2.2 CNN Description 24

4 One Class Support Vector Machine **29**
4.1 Introduction to OCC Problem 29
4.1.1 One-Class Vs. Multi-Class Classification 29
4.1.2 OCC Solution Methods 31
4.1.3 Application Scenarios 31
4.2 One-Class Support Vector Machines 32
4.2.1 Support Vector Machines 32
4.2.2 One-Class SVM 34
4.3 Dimensionality Reduction 37
4.3.1 Principal Component Analysis 38
4.3.2 Sequential Forward Selection 40

5 The Recognition System **41**
5.1 Data Acquisition and Preprocessing 41
5.2 Feature Extraction 43

5.3 Identification 45

6 Results **49**
6.1 CNN Model Optimization 49
6.2 CNN Features Evaluation 52
6.3 OSVM Model Optimization 58

7 Conclusions **69**

Bibliography **71**

List of Figures

1.1 Sub-division of the gait cycle as suggested by Perry 4

2.1 Activity Logger logo and home screen. 10
2.2 Comparison of the sampling frequency distribution of the smartphone employed in the data acquisition (LG G2) and another smartphone. 11
2.3 Power Spectral Density of the raw accelerometer data acquired. 12
2.4 Accelerometer signals (on the left) and gyroscope signals (on the right) from three different users after the cycle extraction step. 13
2.5 Accelerometer data for two different walks with different orientation along the original axes a_x, a_y, a_z and the transformed ones, a_ξ, a_ψ, a_ζ. 14

3.1 Simple biological neuron model. 18
3.2 Mathematical model of an Artificial Neuron. 19
3.3 Examples of ANN Architectures 21
3.4 Two examples of connectivity pattern in a convolutianal layer of a CNN, where each neuron has a receptive field of size 3. 24
3.5 Example of downsampling with max operator, here shown with a stride of 2. 26
3.6 Example of 2-stage CNN architecture. 27

4.1 Example of a conventional and one-class classifier 30
4.2 One-Class SVM Classifier, where the decision bound and the origin, the only member of the negative class, are reported. 35
4.3 Principal Component number 1 and 2 capturing the directions in which the data has the most variance. 39
4.4 Generic diagram of a wrapper approach for feature selection. 40

5.1 General schema of the proposed biometric system for the human gait recognition problem. 41
5.2 Preprocessing block of the proposed biometric system. The internal steps involved in the computation are highlighted. 42
5.3 Complete schema of the CNN architecture proposed, in which is highlighted the block used as feature extractor tool. 45
5.4 General schema of the Identification Block, where the internal steps are highlighted. 46

5.5 Plot of the dependencies of FPR and FNR as a function of the security level. EER is highlighted. 47

6.1 Output test accuracy versus number of cycles per subject employed during the training phase of the CNN. 50
6.2 Output loss function versus number of cycles per subject employed during the training phase of the CNN. 51
6.3 Output test accuracy versus number of features of the first fully-connected layer of the CNN, by varying the number of subjects for the training phase. 52
6.4 Comparison of Machine Learning Algorithms trained on 10 subjects not used in CNN learning phase. Classification test accuracies versus the number of train records for each subject are reported. 54
6.5 Comparison of test accuracy versus the number of train records, obtained from two CNN models trained on 10 and 30 subjects 55
6.6 Comparison of classification algorithms trained on feature vectors obtained from a pre-trained CNN and calculated directly from the data cycle. Test accuracy versus the number of train records is reported. 57
6.7 A confusion matrix for a binary classification problem 59
6.8 Performance measure obtained from a pre-trained OSVM model by varying the number of train data and the ν parameter. 61
6.9 F1-measure performance by varying ν value versus the number of train cycles. 62
6.10 Comparison of dimensionality reduction methods by varying the number of features. 64
6.11 Performance measure of 6 walk acquisitions tested on an OSVM model trained on the remaining 5 walk acquisitions. 65
6.12 Performance measure of 12 walk acquisitions tested on an OSVM model trained on the remaining 11 walk acquisitions. 66
6.13 Average performance comparison by varying the number of features on walk acquisition not used during training. 67
6.14 The EER point for the cumulative score functions FPR e FNR. 68

Chapter 1

Introduction

In recent years, the field of *wearable technologies* is growing at a fast pace and promises to have a widespread influences in the society of coming years. These technologies are now one of the fastest growing segment of the so-called *Machine to Machine* (M2M) communication, better known as *Internet of Things* (IoT) [43], which comprises internet-connected objects that can communicate, compute, sense and potentially actuate, if equipped with artificial intelligence. IoT represents the most disruptive technological revolution to date and together with Wireless Sensor Networks (WSN) it will enable *Ubiquitous Sensing* [40], of the physical world.

Originally, the vision of Ubiquitous Computing, coined by Mark Weiser in 1991 [38], is that, eventually, computers will disappear and become part of our environment, fading into the background of our everyday lives. In the same way, ubiquitous sensing try to fuse the digital and physical worlds, in which users are challenged to find new ways to interact with this new generation of devices, and new uses of them. To facilitate this process, low-power devices rely on sensor technologies as well as existing wireless networking systems and protocols. Moreover, as costs of computation and connectivity continue to fall, these technologies are increasingly embedded into almost all devices that users own and come into contact with.

In particular, wearable devices have the distinct property to be worn by users. Therefore, wearables extend the IoT paradigm to a *human-centric sensing scenario*, where users are involved in the sensing phase participating actively and making it feasible to get information that otherwise is not possible [39]. Technologies advances allowed the integration of an increasing number of sensors in devices such as smartphones, smart watches, wristband, chest straps and clip-on devices. Measurements from accelerometer, gyroscope, magnetometers, GPS, thermometers and cameras can be retrieved from these devices, in order to make users lives easier, safer, healthier, less expensive and more productive. In addition users can have more control on their lives and free up time automating routine tasks. According to a survey, wearable electronics business powers from $20 billion in 2015 to almost $70 billion in 2025 [41], the dominant sector will remain the healthcare sector which merges medical, fitness, wellness, entertainment, security and industrial.

In this thesis, I focus my attention on the security field and I investigate the utilization of inertial sensors, embedded in last generation smartphones, for biometric authentication. In particular, I experiment gait recognition, a biometric method that allows an automatic

verification of a person by the way he or she walks. Gait recognition has been based on the use of video sources and floor sensors, but in recent years a new approach based on wearing motion-recording sensors on the body in different places, such as on the waist, in pockets, at the ankle, is starting to develop. The main advantage of gait recognition using wearable sensors is that it provides an unobtrusive method of authentication for mobile devices that already contain motion sensors.

1.1 Authentication on Smartphones

Authentication is a measure of security, which has the aim to validate the identity of a person provided to a system checking specific characteristics previously stored. Authentication on smartphones is usually performed by three approaches. PIN or password authentication, graphical authentication and the most recent biometric authentication. These methods are used to authenticate a user for accessing the mobile device and its services.

In the contest of mobile devices, knowledge-based authentication via PIN or password is the most common method. A PIN is used for authentication at the Subscriber Identify Module (SIM)-card after turning on the smartphone, but it does not protect the content of the smartphone as it can be eluded by removing the SIM-card form the phone.

Graphical authentication methods have a higher level of security, because the secret needed for authentication is an image or a pattern, that are harder to forge and easier to remember than a password because of the so-called *pictures superiority effect* [42], according to which pictures are easier to remember than words. These methods can be categorized in locimetrics, drawmetrics and cognometrics. For locimetric methods it is necessary to highlight special points in a given image, these chosen locations form the password, that during authentication have to be repeated. Drawmetric methods request the user to reproduce a previously drawn image, the best known method is *Draw-A-Secret* (DAS), in which during enrollment the user draws a pattern on a grid. Cognometric methods require the user to recognize previously selected images in a set of images.

Biometrics are spreading in recent years and they are increasingly used in commercial and private sector thanks to the variety of biometric sensors present in electronic devices that can be used to authenticate individuals based in their intrinsic physiological or behavioral characteristics, that are unique for each person. Physiological means characteristics that are part of the human body, such as fingerprint, hand silhouette, iris pattern or DNA fingerprint. Instead, behavioral characteristics are measurable in human activities. Therefore biometrics can establish a direct relation between the person under identification and unique biometric traits [14].

The most common physiological biometric methods running completely on smartphone are face, speaker or fingerprint recognition. Automatic face recognition uses integrated front camera to authenticate users. In the literature, there are several approaches, such as geometrical based, correlation based, neural network and support vector machine based. They are becoming convenient mobile authentication methods thanks to the increasing computational power available on mobile phones. Speaker recognition is the identification of a person from characteristics of voices. On smartphone speaker recognition can be done in background during phone calls or they require the user to speak or recite a text for the purpose of explicit authentication. Fingerprint recognition on mobile phone is realized in

two ways. The first one is via dedicated sensor, that is not usually integrated in most of the smartphones. The second approach is to use back cameras to capture an image of the finger and then evaluate the fingerprint through image processing algorithms.

Generally, physiological characteristics used in these methods are more stable, because physiological traits are essentially fixed and do not change over time. On the other hand, behavioral biometrics method is based on the recognition of skills, knowledge, style, preference or strategy used by people during everyday activities. Unfortunately, these characteristics are more apt to change, because they depend on factors such as aging, injuries and mood. Because of the prevailing presence of these intra-personal variations, behavior based systems have a discriminatory power smaller than the physiological methods and more efforts need to be done to achieve good reliability performance.

One of the recent studied behavioral biometric is gait recognition, which has the aim to authenticate people based on the way they walk and it has the advantage to be more user-friendly compared with the previous described authentication methods, because it is not an obstructive method since it does not require a direct interaction with the user.

There are three different approaches in gait recognition: machine vision based, floor sensor based and wearable sensor based [12].

In machine vision based gait recognition, subjects are recorded with video cameras. [9] and [10] survey this approach. As stated in these papers, research on this area is well advanced but gait recognition for individual identification is still far from practical applications because of poor recognition rates and evaluation on limited datasets. Floor sensor based approaches uses integrated pressure sensors measuring ground reaction forces or the pressure in the perpendicular direction to the floor surface. Because of the need of these particular sensors the main application of this are access control and smart homes.

In a wearable sensor based approach, a recording sensor is worn or attached to the human body, in order to measure numerous type of data during a walk, depending on which sensor is used, such as gyroscope, accelerometers or telemetry sensors. Thanks to these devices, it is possible to perform two types of identification: continuous and static. In continuous authentication, the gait data are recorded by acceleration sensors, while the user is walking, in a way that the stored information can be used to verify the identity of the users during the walk.

Static authentication is a mechanism that makes decisions about the identity of a user after the user has walked and the aim is to accept or reject this person with the result of allowing or not access to a particular system or service. The interest in performing research on different aspects within wearable-based gait biometrics has increased in recent years. In particular, accelerometers based gait recognition is widely studied, thanks to the fact that last generation mobile smartphones are already equipped with such sensors. In the next section I will cover in depth the state of the art of gait recognition with wearable devices.

1.2 State of the Art of Gait Recognition

Gait recognition is an emerging biometric technology which involves people being identified through the analysis of the way they walk. A particular way or manner of walking is the definition for gait [22] and human gait is the continuous repetition of cycles, which generally consist of two steps each.

Gait is not a new topic in the research and scientific literature, in fact it has been investigated in the past decades in several medical studies, such as those of M.P. Murray in [15] and [16]. Murray stated "Although the excursions of both pelvic and thoracic rotation in repeated trials of the same subject were similar, there were striking differences in these excursions among the individual subjects tested" and also "if all gait movements are considered, gait is unique". In a research work of 1977 [20], J.E. Cutting and L.T. Kozlowski investigated the possibility for a person to recognize friends by their gait. They showed that people recognize others not by using static properties such as height but by dynamic aspects such as length of step, rhythm of the walk, speed, bounciness etc. and people claimed to associate these dynamic aspects with particular individuals.

These examples present a first input to the gait uniqueness of each person. Several human factors, such as aging, weight, injuries, operations etc. may change a person's walking style in a permanent or temporary way [22]. Moreover, Figure 1.1 shows how Dr. J. Perry divided the biological process of the musculo-skeletal system of a gait cycle, which consists five stance phase periods and three swing phase periods [17].

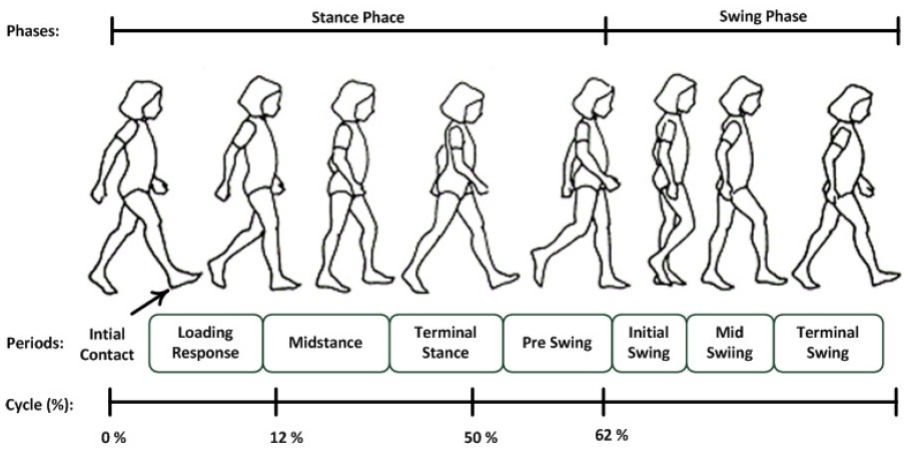


Figure 1.1: Sub-division of the gait cycle as suggested by Perry

Hence, researchers initially studied gait for medical applications, for example with the aim to recognize changes in walking patterns, that can help to identify conditions such as Parkinson's disease and multiple sclerosis in their earliest stages [18]. These early studies were so encouraging and promising to propose gait as a biometric feature to be used for identifying individuals. However, only in recent years gait recognition has attracted interest as a method of identification thanks to reliable and inexpensive sensors, and processing power to handle sizable amount of data.

Many studies have been done and many academic articles have been published about gait recognition. To present a comprehensive overview of the research work in the literature, I divided this discussion about fields of application and methods used to perform gait recognition.

Generally, we can find three common application fields: medical and healthcare, activity recognition, identification and authentication. All these application areas are based on the recognition of a person exploiting unique gait traits.

In the medical field, gait analysis is performed to assess pathologies, that will affect walking gait function and body balance, and to monitor human motor functions. Jang et al. in [23] propose a personal health counseling application, which is able to identify irregular or abnormal walking patterns like asymmetry or skew and to warn the person that he or she may have walking problems. Other researchers focused on more specific diseases, like in [24], where they studied non-specific chronic low back pain (LBP) and they tried to determine whether patients with LBP and healthy controls could be classified based on gait characteristics. In [25] C.Y. Lee and J.J. Lee estimated gait pattern, speed of the subject, total walking distance and improvement of walking function, in order to monitor functional recovery in gait rehabilitation.

Activity recognition is an application field closely related to healthcare and medical ones. Daily activities information is useful for clinicians to improve differentiated treatment on diagnosis of neurological, degenerative and respiratory disorders, but also an accurate quantification of daily physical activity could improve quality of life and prevent pathologies such as obesity, diabetes and cardiovascular disease widespread in today's society. Several research works are performed in this direction. In [26], activity classification using accelerometers in a multi-sensor configuration is performed to acquire information from raw signal to improve activity type and intensity estimation in physical surveillance projects such as UK Biobank and NHANES. Shoaib et al. in [27] proposed an activity recognition study with the aim to use the capability of smartphones for motivating people to be physically active and give them the right motivational feedback. Other activity recognition and classification works can be found in [28], [29] and [30].

However, in most of these the analysis of biometric gait recognition has been studied for use in identification, surveillance and forensic systems, since they provide more reliable and efficient means of identity verification [22].

Gait recognition systems typically have three main components. A low-level sensing module, which gathers raw data using motion sensors. A feature processing and selection module that processes the raw sensor data and transforms them into a reduced set of features. A classification module that uses the features to identify users. Research works differ according to the technique and algorithms used in these three modules, but the common aspect is that gait data acquisition is performed through wearable sensors. This approach is based on attaching or wearing motion recording sensors, that can retrieve numerous data types, in fact most literature has put a great focus on accelerometer based gait recognition [22].

One of the first studies carried out with wearable sensors belongs to Morris [18] in 2004, where a device, called GaitShoe, was developed to be worn on shoes, equipped with an extensive sensor suite. The pattern recognition results suggested that GaitShoes has the ability to extract features useful to recognize individual subjects, as well as groups of subjects with a similar gait and Neural Nets appeared to be a promising method for discriminating between both individual subjects and between groups of subject with normal gait, and groups of subjects with Parkinson's disease.

The first that investigated user identification from accelerometers signals only was

Alisto et al. in [31]. In their study, the recognition is based on the analysis of three-dimensional acceleration signal produced by gait employing acceleration sensors in a portable device worn on a belt with fixed orientation. Preprocessing is performed using peak detection and for identification they used correlation, frequency domain. Performance is quite good, obtaining a mean Equal Error Rate (EER) of 13%.

In other works, data acquisition can be performed placing sensor devices in different body positions leading to unrealistic scenarios, such as ankle [26], hip and waist [33] with fixed orientation. Moreover, inertial sensors are used in combination with other sensors due to their sensing ability. For example, in [34] GPS data are used with accelerometer data and in [27] it is investigated the usage of magnetometer and gyroscope data to improve recognition accuracy.

In recent years, researchers started to study acceleremoter-based gait recognition on mobile devices due to the rapid deployment of their sensing technology and the explosion of their usage in people's daily lives. In 2010, Derawi et al. [30] acquired acceleration data from a mobile phone attached to the belt of the subject. Only the acceleration in the x-direction is used to extract repeating cycles to result in one single average cycle for each person. Dynamic Time Wrapping is used as comparison method. In [21] authors developed a user verification system using smartphone accelerometer. They proposed a more robust cycle extraction phase using a template dynamically updated and the user verification was performed by calculating similarity scores.

In [36], inspired by research done in the area of speaker recognition Nickel et al. applied Hidden Markov Models to the gait data collected using a mobile phone, but this model requires several output in order to provide good results. Other popular signal techniques including the Fourier Transform (FT), Short Time Fourier Transform (STFT), and Hilbert-Huang Transform (HHT) are deployed to analyze the collected data in [35].

Finally, Machine Learning (ML) algorithm are investigated in the classification phase for identification. Supervised algorithms are typically used, such as k-Nearest Neighbor [1], Support Vector Machine, Multi Layer Perceptron and Decision Tree [29]. An important preprocessing phase to apply these algorithms is the feature extraction phase. Most research papers utilized hand-crafted features calculated directly on gait cycles, which are used as input for the training phase of the ML algorithms [2].

In the next section, I will present the proposed method for human gait recognition.

1.3 Motivations and Contributions

This thesis describes the development of a novel type of human identification system. The system uses smartphone-acquired walking data, in order to correctly identify a user from the biometric traits of his way of walking. The proposed system includes four sequential phases: data acquisition, preprocessing, feature extraction and identification. These four phases are typically implemented in every identification system, but they basically differ from the approach present in the literature for techniques and algorithms used to achieve the desired goals.

First of all, I used last generation smartphones with built-in inertial sensors in order to acquire accelerometer data, which are widely used for gait recognition, and also gyroscope data, which in this work are used in combination with the accelerometer ones to investigate

if this new source of information could improve recognition accuracy. Regarding the data acquisition procedure, volunteers walk in a real world scenario, leaving them to walk in a way they consider normal. Acquisitions are performed with a smartphone located in the right pocket of the trousers without a fixed orientation, unlike previous literature approach, where devices are blocked in a fixed position in order to avoid sensor noise and to get accurate orientation. These problems are faced in the preprocessing phase.

Inspired by the work of Ren et al. [21] the preprocessing of the raw data is performed through a template based method used for cycle extraction, which includes template extraction and update, comparison and minimum detection utilizing a suitable correlation distance. As mentioned before, a fixed orientation of the device is not enforced, therefore, it is necessary to implement a coordinates transformation in order to obtain a new independent reference system. The new reference system is evaluated directly from the raw data acquired, extracting gravity versor from the accelerometer data and using Principal Component Analysis.

The feature extraction phase is one of the new key aspects of the proposed system. I use a Machine Learning (ML) approach to extract meaningful features directly from the data cycles of accelerometer and gyroscope. In contrast with the most used approaches, which are based on the calculation of statistical features, such as mean, standard deviation, histogram bins etc. Hence, the aim is to completely eliminate the human effort on this phase of the system, leaving the ML algorithms the freedom to learn the features from the data in a completely automatic way. In particular, to do this. I used a pre-trained Convolutional Neural Network (CNN) as a black-box for feature extraction. CNN is a Deep Supervised Feedforward Neural Network, which recently has become a hot topic in voice and image recognition. In this thesis, I investigate the usage of CNN to transform input signals into a reduced set of features and the ability of these features to represent the essential discriminative traits of gait cycles. Usually, a CNN is designed to take advantage of the 2D structure of an input image, therefore, developing a CNN architecture for this system was more challenging for the different nature of the input data. CNN applied to data signals is not a topic widely studied in the literature and there are no standards regarding the choices to follow in the design phase of the network. For these reasons, CNN architecture is one of the key aspects of the system, because feature extracted by the CNN model affects the performance of the subsequent phase.

Also, for the identification a ML approach is used. To this end, I used the One-Class Support Vector Machine proposed by Schölkopf et al. in 1999 [37], where they designed an algorithm which computes a binary function which is supposed to capture regions in input space where the probability density of the input data lives. Since One-Class problem depends only on positive examples of a target class, this classification problem can be treated as an identification one. Feature vectors of a single subject are given in input, in order to create a model able to discriminate data of a single subject from that of all others subjects. In this way, a person can be identified, if the bound created around his data is precise and robust enough.

The contents of this thesis are organized as follows. In the next Chapter 2, I describe in detail data acquisition and the preprocessing phase of the data collected. In Chapter 3 and 4, I give an introduction on the fundamentals of Convolutional Neural Networks (CNNs) and One Class Support Vector Machine (OSVM). In Chapter 5, I describe the design and

the implementation of the proposed system. Then, in Chapter 6, I present the experiments and tests carried out to assess the performance of the proposed system. Finally, in Chapter 7 I draw the conclusions.

Chapter 2

Data Acquisition and Preprocessing

2.1 Data Acquisition

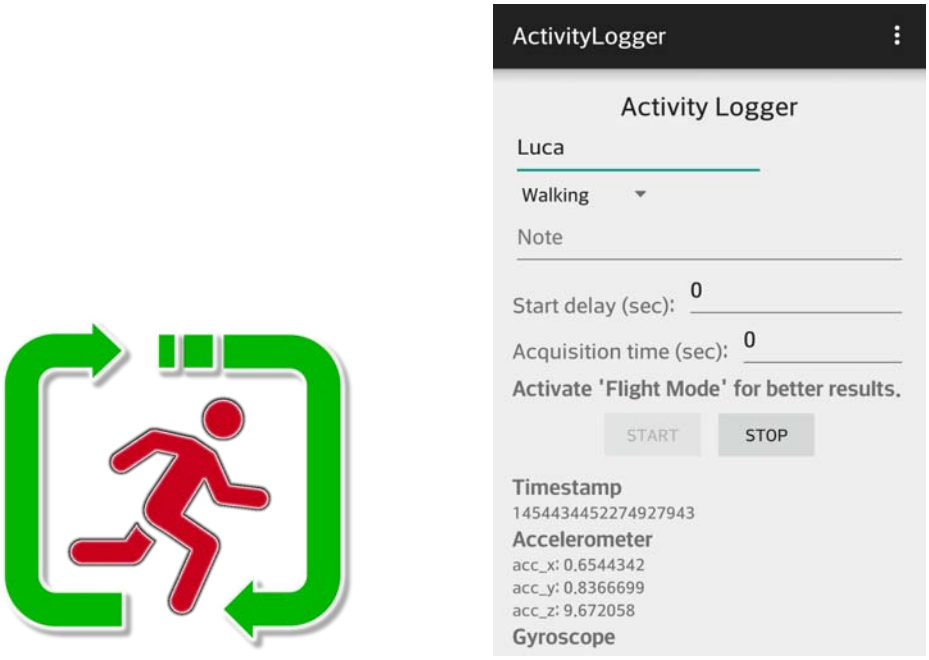
Input data for the designed system were collected using a LG G2 smartphone. It features a 2.26GHz quad-core Qualcomm Snapdragon 800 processor and it comes with 2GB of RAM on which runs Android 4.4.2 KitKat. It is equipped with built-in sensors: GPS, Proximity sensors, light and an Inertial Measurement Unit (IMU). In particular, IMU comprises a tri-axial LGE accelerometer sensor, a gyroscope and a magnetometer, whose specifications are reported in Table 2.1.

Sensor	LGE Accelerometer	LGE Gyroscope	LGE Magnetometer
Vendor	STMicroelectronics	STMicroelectronics	AKM
Range	39.227	34.907	4912
Resolution	0.001 (0.003%)	0.001 (0.003%)	0.15 (0.003%)
Power	0.28mA	6.1mA	5 mA

Table 2.1: UMI specifics of LG G2 used for data acquisition.

Data collection is performed with a own developed Android application, called *Activity Logger*, which is used to set up acquisition parameters, give informations about the user, collect data and save them in non-volatile memory.

Before the acquisition start, in the home screen it is possible to input the user name, to select the activity, which in this case is always *walking* and to insert some useful information about particular conditions of the user or of the walk, such as age, disease or smartphone position problems. Moreover, there is the option to set up a starting delay in seconds, useful to give user time to put the mobile phone in the right position and start walking, and the total acquisition time in seconds. To begin the acquisition, the user must press start and if the starting delay is not set, the application starts immediately to acquire data. In Figure 1.1 we show the logo and the home screen of the Activity Logger application.



(a) Activity Logger logo

(b) Activity Logger home screen

Figure 2.1: Activity Logger logo and home screen.

The application saves in a temporary buffer three-dimensional data gathered from accelerometer and gyroscope, which are then used for the subsequent preprocessing phase, in addition also three-dimensional magnetometer data, gravity acceleration, linear acceleration and rotation vector are retrieved. When the acquisition ends, the buffer is read by saving data for each sensor in text files, where each sensor event is identified with a timestamp in nanoseconds, stored in a directory of the smartphone file system.

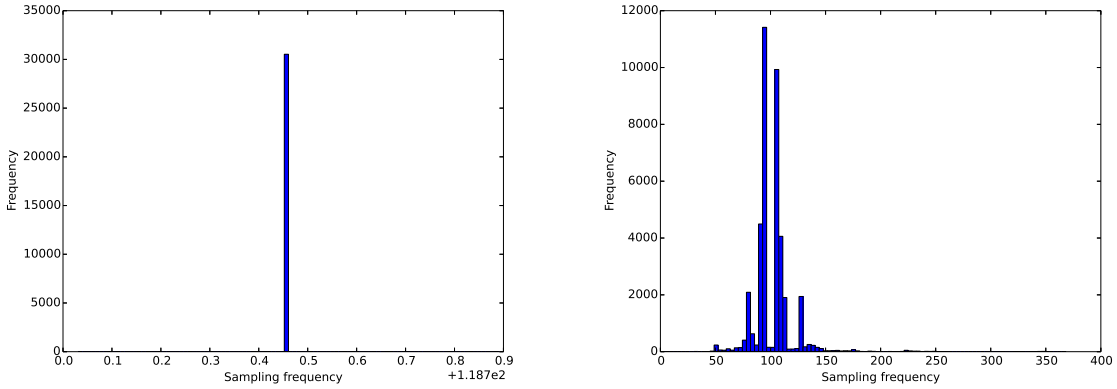
Raw data were collected from 40 volunteers, either males or females, aged between 17 and 84 years with the application just described. Volunteers were asked to walk in normal conditions in a real scenario for a minimum of two acquisitions with a duration comprised between 5 and 10 minutes. The person under examination needs to start the application, input the information he/she considers appropriate, put the smartphone in the right pocket of the trousers in vertical position with the screen pointed towards the legs and start to walk. The phone position in the pocket is not a big issue, because small rotation or displacement of its initial position does not affect the acquisition, but in order to acquire data as uniform as possible, subjects have to wear trousers, which do not let the smartphone move in an uncontrolled manner; jeans were preferred.

At the end of the acquisition, only accelerometer and gyroscope data vectors on each axis are preprocessed in the subsequent phase. In addition, their magnitudes were calculated, that is, given accelerometer vector, $\overrightarrow{a} = (\mathbf{a}_x, \mathbf{a}_y, \mathbf{a}_z)$ and gyroscope vector, $\overrightarrow{\mathbf{g}} = (\mathbf{g}_x, \mathbf{g}_y, \mathbf{g}_z)$, magnitude vectors are defined with the following equations:

$$\mathbf{a}_{mag} = \sqrt{\mathbf{a}_x^2 + \mathbf{a}_y^2 + \mathbf{a}_z^2} \tag{2.1}$$

$$\mathbf{g}_{mag} = \sqrt{\mathbf{g}_x^2 + \mathbf{g}_y^2 + \mathbf{g}_z^2} \tag{2.2}$$

Note that, the sampling frequency of LG G2 has a standard deviation close to zero, hence the sampling frequency is almost constant, as one can see in Figure 2.3.(a), where the sampling frequency distribution is reported, but this is not always the case. In fact, in Figure 2.3.(b) we show the sampling frequency distribution of another smartphone, in which one can see that constant sampling frequency is not guaranteed, probably it depends on the computational load of the operative system stack. Hence, an interpolation phase of all the signals gathered is necessary.



(a) LG G2 case.

(b) Another smartphone case.

Figure 2.2: Comparison of the sampling frequency distribution of the smartphone employed in the data acquisition (LG G2) and another smartphone.

2.2 Data Preprocessing

Raw data retrieved from accelerometer and gyroscope sensors with their magnitude needs to be processed in order to extract gait cycles, which are going to form the data set used in the feature extraction phase. The preprocessing step includes four sequential steps: interpolation, filtering, cycles extraction, reference system transformation and normalization.

Interpolation

An interpolation step is needed because the sampling rate is not constant during the acquisition due to the priority policy of the Android Operative System. Moreover, smartphones output data values whenever there is a change in the sensor, therefore, the time intervals

between two samples are not equal and differ for each sensor. Time interpolation is performed through spline interpolation to ensure a sampling rate of $f_S = 200\,Hz$ and fixed time interval between any two consecutive samples.

Figure 2.2 shows the Power Spectral Density of the raw accelerometer data and withe noise gathered by the smartphone. In particular, the tri-axial accelerometer data of a walk of a random subject is reported together with the tri-axial accelerometer data of the noise. As one can see, noise undergoes a filtering applied by the smartphone, but if the noise were not filtered, it would be reasonable to think that the intersection point with the accelerometer data of the walk is at the red marker dot, highlighted in Figure 2.2. Hence, the bandwidth of the accelerometer walking data is about $70\,Hz$ and the choice to fix the sampling rate to 200 Hz will avoid the presence of aliasing.

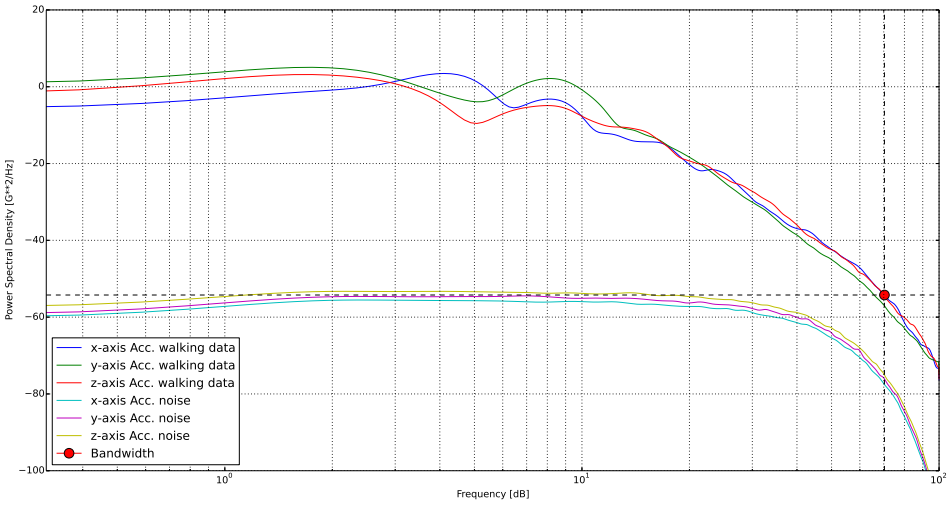


Figure 2.3: Power Spectral Density of the raw accelerometer data acquired.

Filtering

A filtering step is required to deal with possible noise problems of the acquired data. In order to smooth signals and remove high peaks, a lowpass FIR filter of order 30 and cutoff frequency $f_c = 40\,Hz$ is applied to all accelerometer and gyroscope components and to their magnitude.

Cycle Extraction

Filtered signals are passed through a cycle extraction phase. Human gait follows a cyclic behavior, where there is a periodic repetition of a pattern, known as cycle, which corresponds approximately to two human steps, hence it is reasonable to think that cycles are

highly correlated in a walking trace of the same subject. For this reason, gait cycles are recognized by conducting a cycle identification repeatedly in the walking trace. In order to do this, a template of fixed length is utilized. The first template is extracted finding the absolute minimum between the first 0.5 and 2.5 seconds of the accelerometer magnitude. Then, the subsequent cycles are identified using this template as sliding window on the signal and they are then evaluated through a correlation distance measure, defined as follows:

$$corr\_dist(\mathbf{u}, \mathbf{v}) = 1 - \frac{(\mathbf{u} - \bar{\mathbf{u}}) \cdot (\mathbf{v} - \bar{\mathbf{v}})}{\|\mathbf{u} - \bar{\mathbf{u}}\| \, \|\mathbf{v} - \bar{\mathbf{v}}\|} \tag{2.3}$$

Hence, it is possible to find the following minimum looking for local minima in an interval where the correlation distance between the signal and the time-shifted template falls below a certain threshold and it is possible to identify the starting point of the subsequent gait cycle. Then, the template is updated performing a weighted mean between the previous one and the current one, in order to obtain an accurate template to avoid that detection errors are propagated in the subsequent identification. The detection continues until the end of the signal and the result is a vector of indices in the time domain. Two consecutive indices represent the start and end points of a cycle and these values are used to extract cycles from all the accelerometer and gyroscope signals.

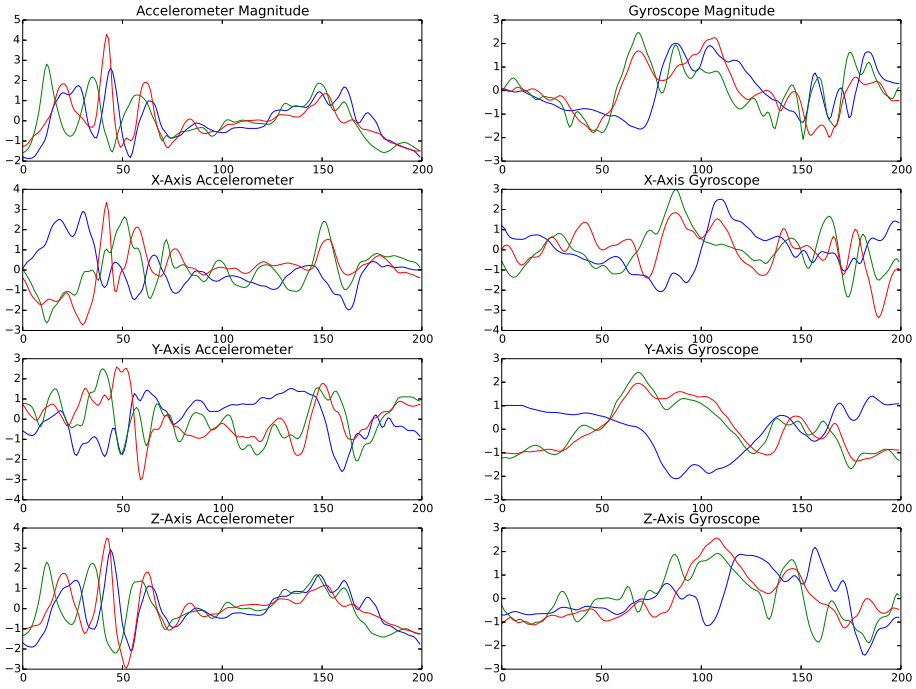


Figure 2.4: Accelerometer signals (on the left) and gyroscope signals (on the right) from three different users after the cycle extraction step.

In Figure 2.4 cycles of three different subjects are shown. As one may notice, gait patterns are different for each subject. This example demonstrates the basic idea that the gait pattern is unique for each person and differs between different people. In particular, signals that differ the most are the gyroscope one. Hence, including in the feature extraction phase also these data could give important information, which may help capture the uniqueness embedded in each gait and to improve discrimination between different people.

Reference System Transformation

A reference system transformation is motivated by the fact that the orientation of the smartphone inside the pocket is unknown and it changes continuously during walking due to the leg movement. Hence, the new reference system is extracted directly from the acquired data, in order to obtain a system independent from the orientation of the phone. Three independent and orthogonal versors are extracted in the following way.

Because of an accelerometer is subject to the effect of gravity acceleration, data have a common direction due to this component, which is the main low frequency component. Hence, the first axis of the new reference system is the gravity direction, which is calculated as the mean direction of the gravity from the tri-axial accelerometer data.

Then, the acceleration data are projected on this new axis. First the acceleration component along the gravity axis is evaluated and by removing this component from the original data, a new flattened component placed on the plane orthogonal to the gravity vector is obtained. Analyzing the flattened component, there is a strong directionality, along which we observed the greatest excursion in the acceleration data. In order extract this direction from the data and to obtain a vector orthogonal to the previous one, Principal Component Analysis (PCA) is applied. Finally, the third axis is obtained simply with a cross product between the two versors just calculated.

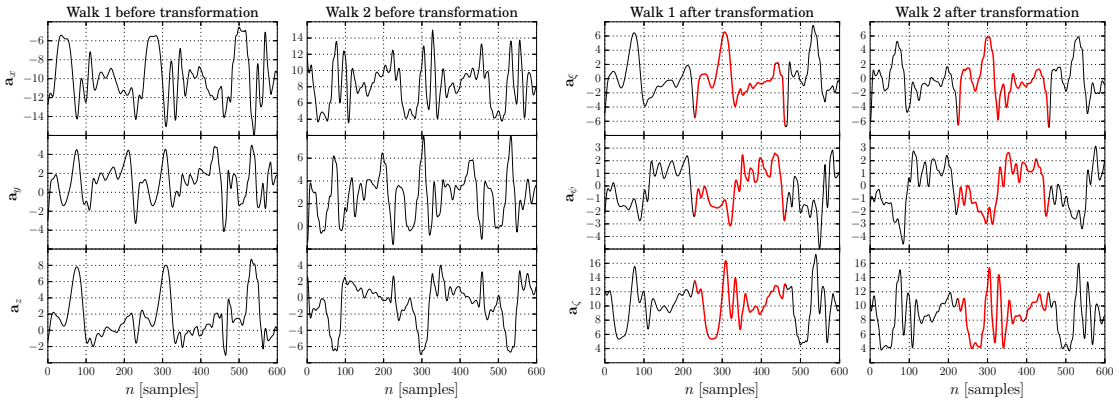


Figure 2.5: Accelerometer data for two different walks with different orientation along the original axes a_x, a_y, a_z and the transformed ones, a_ξ, a_ψ, a_ζ.

The positive effects obtained thanks to this reference system transformation are shown in Figure 2.5, where we show two different walks of the same subjects acquired with different orientation. On the left there are raw accelerometer data along the original axes and on the right one there are the same data in the new reference system. As one can notice, in the original signal the repetitive pattern of the two walks is hard to identify or is completely different, this is not the case of the signals in the new reference system, which show a very similar pattern.

Normalization

As noted before, each type of walk derived from different subjects differs in shape, but different subject also differs for their speed, in fact people have slow, normal and fast gait or they may change speed during walking. This aspect is reflected on the duration of each cycle. Moreover, the subsequent feature extraction phase requires in input a fixed number of samples for each signal, which can affect the performance of the Convolutional Neural Network in terms of computational time spent and final classification accuracy. In fact, a high cycle length could considerably increase the computational time spent in the learning phase and the number of free parameters of the network, without any improvement i terms of classification accuracy. On the other hand, a low cycle length could speed up the feature extraction process, but could led to aliasing problems due to subsampling of the signal. Therefore, a good trade-off has to be found. For all these reasons, each cycle is normalized to a length of k samples, which in this case was set to 200 samples per cycle.

Chapter 3

Convolutional Neural Networks

Feedforward Neural Networks or Multilayer Perceptron with multiple hidden layers in artificial neural networks are usually known as Deep Neural Networks (DNNs). Convolutional Neural Networks (CNN) is one kind of feedforward neural network, proposed in 1960s, when Hubel and Wiesel researched the neurons used for local sensitive orientation-selective in the cat's visual cortex [4]. CNN is an efficient recognition algorithm which is widely used in pattern recognition, image processing [64] and speech recognition [61]. It has many advantages such as a simple structure, a few training parameters and adaptability. In order to use the machine learning algorithms it is necessary to extract from the acquired signals some features to be informative, non-reduntant and able to facilitate the subsequent learning. The extracted features are a reduced representation of the relevant information from the input data, so that the desired task can be performed using this representation instead of the complete initial data.

In this work, I investigate the potential for generic features obtained from a well trained Convolutional Neural Network model to perform the task of walk recognition for data acquired from smartphone motion sensors.

The choice of this new approach is motivated by the broad applicability of CNNs shown in recent years due to the availability of large dataset, growth in computational power, availability of GPUs and efficient algorithms, which have been used to train these large networks. In the field of image processing, several research group have explored the potential of CNNs to outperform more classical approach to object recognition and detection that are based on hand-crafted features, such as in [59] and [60]. The CNN system applied in these large-scale object recognition or detection tasks consists of a feature extractor, the actual CNN, followed by a classifier or a regressor. Razavin et al. [3] showed that combining CNN features with a simple classifier such as a linear SVM is highly competitive or even superior to classical approaches in most visual recognition tasks. Also, in [62] authors showed that CNN feature maps can be used with Random Forest and SVM to yield classification results that outperform the original CNN.

The gait recognition problem using motion sensors is a machine learning problem which has previously been addressed by deriving hand-crafted features. In [1] and [2], the feature extraction step is performed by computing several features for x-, y- and z-direction, such as mean, minimum, maximum, standard deviation, variance, energy, entropy etc. By contrast, the proposed system uses a CNN, that was initially trained for general classification, as

a generic feature extractor. Then the extracted features for a single person are used to perform the recognition task through the One-Class Support Vector Machine algorithm.

In this Chapter, I summarize the fundamental notations and high-level description of the different components of Convolutional Neural Networks (CNNs), starting with a brief review of the basic concept of Artificial Neural Networks (ANNs) where CNNs have their roots.

3.1 An overview on Neural Networks

Artificial Neural Networks (ANNs) are an attempt of modeling the information processing capabilities of the human brain and biological nervous system. The human brain computes in an entirely different way from the conventional digital computer. It consists of nerve cells called neurons, linked together with other neurons via strands of fiber called axons, as illustrated in Figure 3.1. Axons are used to transmit nerve impulses from one neuron to another when the neurons are stimulated. A neuron is connected to the axons of other neurons via dendrites. The contact point between a dendrite and an axons is called synapse. When the signals received are strong enough, the neuron is activated and emits a signal, which might be sent to another synapse, and might activate other neurons [65].

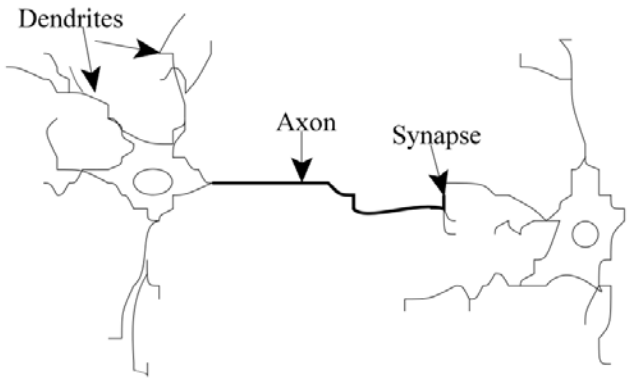


Figure 3.1: Simple biological neuron model.

This structures has the capability of organizing its constituents, so as to perform certain computations e.g. pattern recognition, perception, and motor control many times faster than the fastest computer in existence today. For these reasons, the study of ANN has been attracting increasing attention in recent years

Analogous to human brain structure, an ANN can be defined as a massively parallel distributed computing cell made up of simple processing units, called neurons, that has a natural propensity for acquiring experiential knowledge through a learning process, storing it through inter-neuron connection strengths, known as synaptic weights, and making it available for use [63].

An ANN derives its computing power through its massively parallel distributed structure and its ability to learn and therefore generalize. Generalization refers to the production of reasonable outputs for inputs not encountered during the learning process. These two information-processing capabilities make it possible for neural networks to find good approximate solutions to complex problems that are intractable and to capture the subtle functional relationships among the data even if the underlying relationships are unknown. This is in contrast to most traditional empirical and statistical methods, which need prior knowledge about the nature of the relationships among the data. In practice, neural networks cannot provide the solution by working individually, but they need to be integrated into a consistent system engineering approach where different networks are assigned a subset of the tasks that match their inherent capabilities.

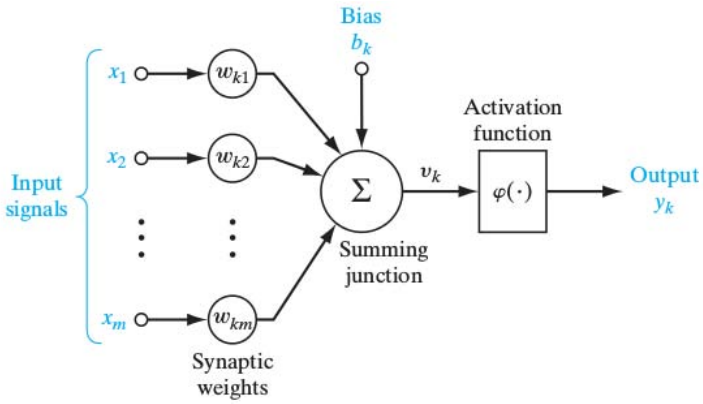


Figure 3.2: Mathematical model of an Artificial Neuron.

3.1.1 Model of a Neuron

The fundamental processing-unit of an ANN is the artificial neuron, a simple abstraction of the complexity of a real neuron. Figure 3.2 shows the structure of an artificial neuron with m inputs, where you can identify its core components:

- A set of synapses, or connecting links, each of which is characterized by a weight or strength of its own. In particular, a signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}.
- An adder for summing the input signals, weighted by the respective synaptic strengths of the neuron.
- An activation function for limiting the permissible amplitude range of the output signal of a neuron to some finite value of a neuron.
- An externally applied bias, denoted by b_k. The bias has the effect of increasing or lowering the net input of the activation function, depending on whether it is positive or negative, respectively.

A neuron computes its output value by performing a weighted sum on its inputs, summing a bias factor to the sum, and then the result of this combined summation is passed through an activation function to produce the output of the processing element. In mathematical terms, the process is summarized in Equation 3.1 and 3.2:

$$v_k = b_k + \sum_{j=1}^{m} w_{kj} x_j \tag{3.1}$$

$$y_k = \varphi(v_k) \tag{3.2}$$

where $x_1, x_2, ..., x_m$ are the input signals; $w_{k1}, w_{k2}, ..., w_{km}$ are the respective synaptic weights of neuron k; v_k is the linear combiner output plus the the bias b_k; $\varphi(\cdot)$ is the activation function and y_k is the output signal of the neuron.

The common activation functions $\varphi(\cdot)$ are the *sign* function, which outputs a value $+1$ if its argument is positive and -1 if its argument is negative; linear function, sigmoid and hyperbolic tangent function which allow nodes to produce output values that are nonlinear in their input parameters.

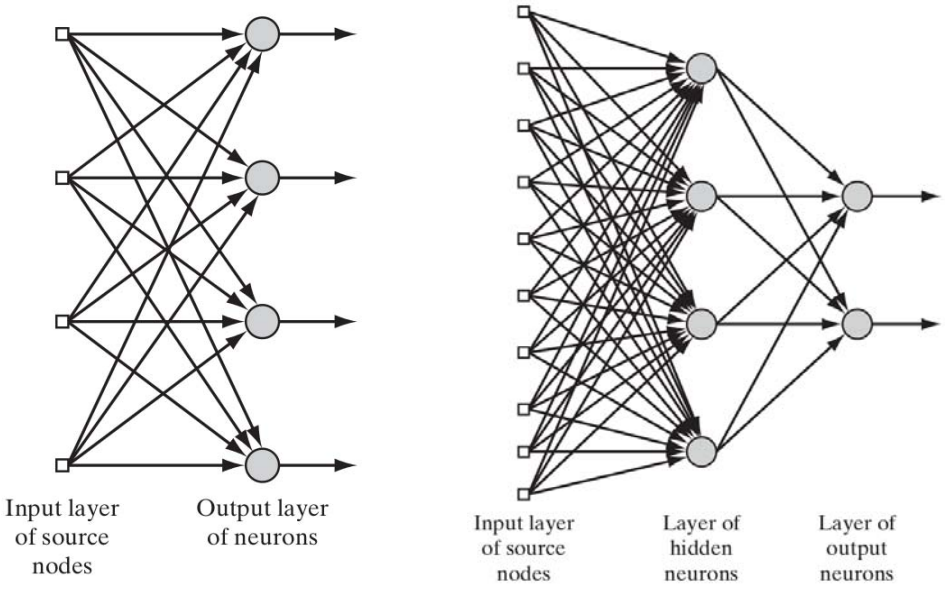
3.1.2 Artificial Neural Network Architectures

ANNs can be categorized on the basis of two major criteria: the learning rule used and the connections between their processing elements. Based on connections, neurons are arranged onto precise structured manners, known as architecture model, in order to fully harvest the benefits of mathematical complexity that can be achieved through their interconnection.

From this point of view three fundamentally different types of ANN architectures may be identified:

- **Single-Layer Feedforward Networks** - In a layered neural network, the neurons are organized in the form of layers. In the simplest form of a layered network, we have an input layer of source nodes that projects directly onto an output layer of neurons, but not vice versa. In other words, this network is strictly of a feedforward type. In Figure 3.3 (a) such a network is called a single-layer network, with the designation "single-layer" referring to the output layer of computation nodes. The Rosenblatt's Perceptron (1958) was the first algorithmically described neural network. It is a single-layer feed-forward network able to classify patterns that are linearly separable.

- **Multilayer Feedforward Networks** - The second class of a feed-forward neural network may contain several intermediary layers between its input and output layers. Such intermediary layers are called hidden layers and the nodes embedded in these layers are called hidden neurons. The resulting structure is known as a multilayer neural network where the nodes in one layer are only connected to the nodes in the next layer. The set of output signals of the neurons in the final layer of the network constitutes the overall response of the network to the input signals supplied by the source nodes in the input layer. By adding one or more hidden layers, the network is capable of extracting higher-order statistics from its input. Figure 3.3 (b) illustrates the layout of a multilayer feed-forward neural network for the case of a single hidden layer.



(a) Feed-forward network with a single layer

(b) Multilayer feed-forward network with one hidden layer.

Figure 3.3: Examples of ANN Architectures

- **Recurrent Networks** - A recurrent neural network distinguishes itself from a feed-forward neural network in that it has at least one feedback loop. For example, a recurrent network may consist of a single layer of neurons with each neuron feeding its output signal back to the inputs of all the other neurons.

3.1.3 Learning the ANN Model

A major task for a neural network is to learn a model of the world in which it is embedded, and to maintain the model consistent with the real world. Knowledge of the world consists of observations and measurements, obtained by means of sensors designed to probe the environment. These observations provide the pool of information, from which the examples used to train the neural network are drawn, referred to as a set of training data.

The learning process aims at acquiring the knowledge about the environment as it is embodied by the training data and at providing a representation of such knowledge by opportunely modifying the values taken on by the free parameters of the network, i.e synaptic weights and biases.

Learning algorithms can be divided into supervised and unsupervised methods. *Supervised learning* denotes a method which infers a function from labeled training data. The output computed by the network is observed and the deviation from the desired output value is measured. The network parameters are corrected according to the the examples and the error signal, calculated by the desired response and the actual response of the

network. This kind of learning is also called *learning with a teacher*, since a control process knows the correct answer for the set of selected input data.

Unsupervised learning uses a training set of unlabeled examples and allows the discovery of hidden structures in the input data. Since the corrections to the network weights are not performed by an external agent, the network itself decides what output is best for a given input and reorganizes accordingly.

In this work, I used a standard fully supervised Convolutional Neural Network model, for this reason, I dedicate the remaining section to the discussion of the principles of supervised learning.

As mentioned before, the network parameters are adjusted under the combined influence of the training data and the expected response in a step-by-step fashion with the aim to optimize in some statistical sense an error signal, defined as $E(\mathbf{w})$, which represents the difference between the desired response and the actual response of the network. Such form of supervised learning is the basis of error-correction learning, where a performance measure for the system in function of the free parameters is defined, i.e. mean-square error, the sum of squared errors or a loss function.

This function may be visualized as a multidimensional error-performance surface, or simply error surface, with the free parameters as its coordinates. Any given operation of the system under the teacher's supervision is represented as a point on the error surface. For the system to improve performance over time and therefore learn from the teacher, the operating point has to move down successively toward a minimum point of the error surface. Since the output of an ANN is a nonlinear function of its parameters because of the choice of its activation function, greedy algorithms such as those based on *gradient descent method* have been developed to efficiently determine the set of weights $\mathbf{w}$ that minimize $E(\mathbf{w})$. The weight update formula, known as delta rule, for node j can be written as follows:

$$w_j \leftarrow w_j + \triangle w_j \tag{3.3}$$

where $\triangle w_j = -\lambda \frac{\partial E(\mathbf{w})}{\partial w_j}$ and λ is a parameter known as learning rate. The second term states that the weight should be increased in a direction that reduces the overall error term. However, because the error function is nonlinear, it is possible that the gradient descent method may get trapped in a local minimum.

The gradient descent method can be used to learn the weights of the output and hidden node of a multilayer ANN. For hidden nodes, the computation is not trivial because it is difficult to asses their error term, $\frac{\partial E(\mathbf{w})}{\partial w_j}$, without knowing what their output values should be. A technique known as *back-propagation* has been developed to address this problem. There are two phases in each iteration of the algorithm: the forward phase and the backward phase. During the forward phase, the weights obtained from the previous iteration are used to compute the output value of each neuron in the network and their derivatives are evaluated at each node. In the forward direction, outputs of the neurons at level k are computed prior to computing the outputs at level $k+1$. During the backword phase, the weight update formula is applied in the reverse direction, that is, the weights al level $k+1$ are updated before those at level k. This back-propagation approach allows to use the errors for neurons at layer $k+1$ to estimate the errors for neurons at layer k.

3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are biologically-inspired variants of MLPs and refer to a sub-field of machine learning, known as Deep Learning, that is based on learning levels of representations, corresponding to a hierarchy of features, factors and concepts.

Feedforward Neural Network or Multilayer Perceptron with multiple hidden layers in artificial neural networks is usually known as Deep Neural Networks (DNNs). Convolutional Neural Networks is one kind of feedforward neural network. In 1960s, when Hubel and Wisel researched the neurons used for local sensitive orientation-selective in the cat's visual system, they found the visual cortex contains a complex arrangement of cells. These cells are sensitive to small sub-regions of the visual field, called receptive field. The sub-regions are tiled to cover the entire visual field. These cells act as local filters over the input space and are well-suited to exploit the strong spatial correlation present in natural images. Being the animal visual cortex the most powerful visual processing system in existence, it seems natural to emulate its behavior as Convolutional Neural Network try to do.

CNN is an efficient recognition algorithm which fileds has become a hot research topic and is widely used in pattern recognition and image processing. It has many desirable properties such as a simple structure, a few training parameters, adaptability and weight shared network structure, which make it more similar to biological neural networks, reducing the network complexity and the number of parameters [66].

3.2.1 Deep Architectures Learning

Human ability to abstract requires that the visual cortex has a deep structure. When humans try to solve an artificial intelligence task, they often exploit their intuition about how to decompose the problem into sub-problems and multiple levels of representation, where models can be reused in different object examples. A computational machine that need to express complex behaviors requires highly varying mathematical functions, that are non-linear in terms of raw inputs and display a very large number of variations across the domain of interest. The raw input to the learning system are high dimensional entities, made of many observed variables, which are related by unknown intricate statistical relationships. If a machine captured the factors that explain the statistical variations in the data, one would be able to say that the machine understands those aspects of the world covered by these factors of variation [67].

The focus of Deep Architecture Learning is to automatically discover such complex behaviors, learning features at multiple levels of abstraction that allows a system to map the input to the output directly from data without completely depending on human-crafted features.

The concept of Deep Learning comes from the study of Artificial Neural Network. Multilayer Perceptron which contains more hidden layers is a Deep Learing Architetures. On the contrary, the model of Multilayer Perceptron, reviewed in the previous section, with zero or only one hidden layer is called *shallow architecture*. These architecture offer little or no invariance to shifting, scaling and other forms of distortion. They ignore completely the topology of the input, yielding similar training result for all permutation of the input vector and adding multiple hidden layers to the Multilayer Perceptron is not

a good approach, since the so called diffusion of gradients problem may occur. These problems are exacerbated by the fact that in a fully connected neural network when the number of hidden layers rises, also the number of connections rises and the network can't be trained effectively in a finite time [68].

Convolutional Neural Networks developed by Yann LeCun [4] is the first truly deep architecture to overcome these problems. It is the architecture that closely resembles human visual system and it can be trained using gradient descent algorithms.

3.2.2 CNN Description

The concept of Convolutional Neural Network was intorduced in 1995 by Yann LeCun and Yoshua Bengio to deal with the variability and identification of two-dimensional image information. The following description in based on [4] by Yann Lecun, on [70] by David Bauchain and on [69] by Wang et al.

At a most basic level, a Convolutional Neural Network is a multilayer, hierarchical neural network. There are three principal factors that distinguish the CNN from a simple feedforward neural network: local receptive fields, weights sharing and spatial pooling or subsampling layers.

In a simple MLP network each neuron was fully connected to the neurons in the subsequent layer. Instead of CNN, which exploits spatially-local correlation by enforcing a local connectivity pattern between neurons of adjacent layers, in other words each neuron is constrained to depend only on a spatially local subset of the neurons of the previous layer, an example is reported in Figure 3.4. The set of nodes that affect the activation of a neuron is referred to as the neuron's *receptive field*. In terms of network architecture, this translates into a sparse set of edges since adjacent layers are not always fully connected.

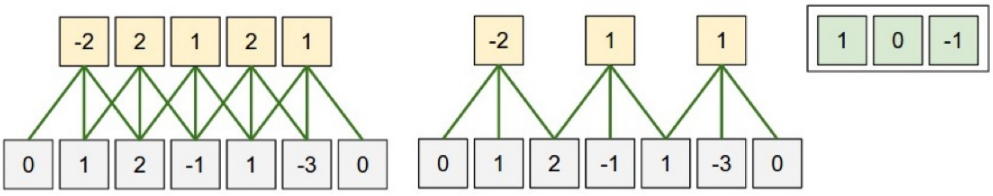


Figure 3.4: Two examples of connectivity pattern in a convolutianal layer of a CNN, where each neuron has a receptive field of size 3.

The second feature that distinguishes CNNs is that the edge weights in the network are shared across different neurons in the hidden layers. Sharing the weights across multiple neurons in a hidden layer translates into evaluating the same filter over multiple subwindows of the input image. Since each neuron computes a weighted linear combination of its inputs, this process is equal to evaluate a linear filter over the input values. Hence, the CNN is effectively learning a set of filters, each of which is applied to all of the sub windows within the input image. Using the same set of filters over the entire image forces the network to learn a general encoding or representation of the underlying data [69].

Constraining the weights to be equal across different neurons also has a regularizing effect on the CNN, because this allows the network to generalize better in many visual recognition tasks. Moreover, weight sharing reduces the number of free parameters in the CNN, making it easier and more efficient to train. Finally, evaluating a filter over each window in the input image corresponds to perform a convolution of the image with a filter. Therfore, in a convolutional step of a CNN, the input image is convolved with each filter to obtain a convolutional respose map.

The final distinguishing component in a CNN is the presence of sub-sampling or pooling layers. The goal here is twofold: reduce the dimensionality of the convolutional responses and confer a small degree of translational invariance into the model. The standard approach is through spatial pooling.

Convolutional Neural Network is composed of several feature map arrays clustered into one, two, or more stages, followed by a classifier at the end. Each stage is used as a feature extractor and it could be made by a convolutional layer, activation layer and feature pooling layer.

- **Convolutional Layer** - It consists of a set of filters. During the forward pass, each filter slides across the width and height of the input feature map. As the filter slides across the input, the dot product between the entries of the filter and the input is computed. Since all neurons use the same weight vector, then the forward pass of the convolutional layer can be computed as a convolution of the neuron's weights with the input. Therefore, it is common to refer to the sets of weights as a filter or a kernel. Suppose to have a set of n two dimensional feature maps of size $H \times W$ as input, each feature map is denoted by x_i and each neuron x_{ijk}, a two dimensional trainable kernel k_{ij} connects feature map x_i with the set of feature maps y_j. A feature map is obtained by convolution of the input feature map with the kernel and adding a bias term.

$$y_j = b_j + \sum_i k_{ij} * x_i \tag{3.4}$$

  where b_j is a trainable bias parameter and $*$ operator performs a two dimensional discrete convolution. The number of features maps at the output is determined by how many different convolutional kernels are used. A Convolutional Layer has two important hyperparameters to be set when building a CNN. The first is the size of zero-padding, that specifies the number of pixels to add to each side of the input to apply the filter to every element of the input matrix and get a larger or equally sized output. Adding zero-padding is also called wide convolution, and not using zero-padding would be a narrow convolution. Another hyperparameter is the stride size that specifies intervals at which to apply the filters to the input. In the literature typically stride is of sizes 1, but a larger stride size may allow to build a model that behaves somewhat similarly to a Recursive Neural Network, i.e., looking like a tree.

- **Activation Layer or Rectification Layer** - It applies an element-wise activation function that introduces non-linearity. This is usually a point-wise hyperbolic tangent $f(x) = \tanh(x_i)$, a sigmoid function $f(x_i) = \frac{1}{1-e^{-x_i}}$ or a Rectified Linear Unit (ReLU) $f(x_i) = \max(0, x_i)$.

- **Feature Pooling Layer** - It is common to periodically insert a pooling layer in-between successive convolutional layers. The purpose of this layer is to progressively reduce the spatial resolution of preceding feature maps. In spatial pooling, the convolutional response map is first divided in a set of m×n blocks, generally disjoint, then a pooling function is evaluated over the responses in each block. This process yields a smaller response map with dimension m×n. The most common form of pooling is the max pooling, that takes the response for each block to be the maximum value over the block response. Figure 3.5 shows an example of max pooling.

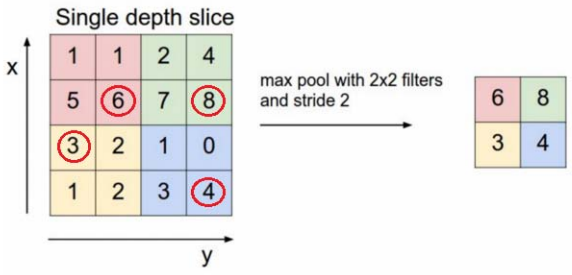


Figure 3.5: Example of downsampling with max operator, here shown with a stride of 2.

- **Fully-Connected Layer** - After several convolutional and max pooling layers, the high-level reasoning in the neural network is achived via fully connected layers. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Artificial Neural Networks.

- **Loss Layer** - The loss layer specifies how the network training penalizes the deviation between the predicted and true labels and is normally the last layer in the network. Various loss functions appropriate for different tasks may be used. Softmax loss is used for predicting a single class of K mutually exclusive classes. Sigmoid cross-entropy loss is used for predicting K independent probability values in range $[0, 1]$. Euclidean loss is used for regressing to real-valued labels $[-\infty, +\infty]$. Typically, in this layer are stored the output class scores o labels.

The most common form of a Convolutional Neural Network architecture stacks a few Convolutional Layer and Rectification Layers, followed by Pooling Layers, and repeats this pattern until the input data has been merged spatially to a small size. At some point, it is common to transition to Fully-Connected layers, that holds the output of the classification task. In Figure 3.6 there is an example of a two stage convolutional network.

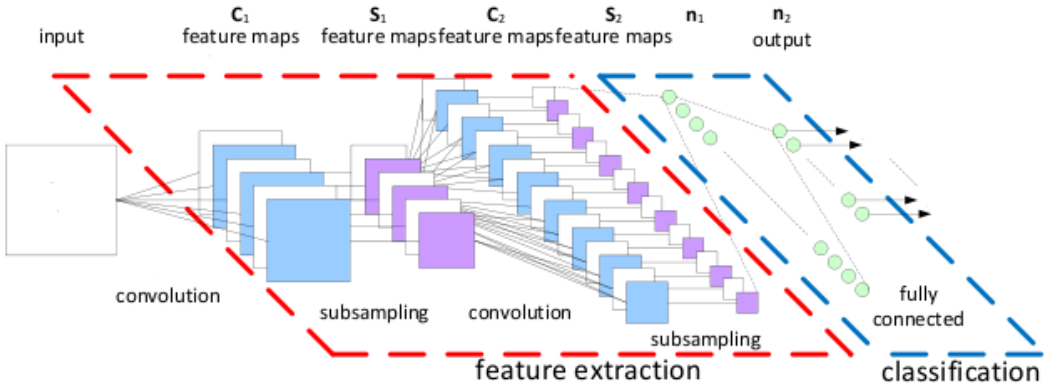


Figure 3.6: Example of 2-stage CNN architecture.

In this way, a deep architecture can be constructed. Intuitively, the low-level convolutional filters, such as those in the first convolutional layer, can be thought of as providing a low-level encoding of the input data. In the case of image data, these low-level filters may consist of simple edge filters. As we move to higher layers in the neural network, the model begins to learn more and more complicated structures. By using multiple layers and large numbers of filters, the CNN architecture can thus provide vast amounts of representational power.

To train a CNN, standard technique of error backpropagation can be used, by iteratively optimizing the weights by calling forward and backward algorithms and updating parameters. However, in practice there are a number of challenges in applying the gradient descent rule, the most important is that the computation of gradient is done for each training input, unfortunately, when the number of training inputs is very large this can take a long time, and learning thus occurs slowly. An idea called *Stochastic Gradient Descent* (SGD) can be used to speed up learning [5].

Following the discussion on SGD used in Caffe[1] and [5]. For a dataset D, the optimization objective is the average loss over all $|D|$ data instances throughout the dataset:

$$L(W) = \frac{1}{|D|} \sum_{i}^{|D|} f_W(X^{(i)}) + \alpha r(W) \tag{3.5}$$

where $f_W(X^{(i)})$ is the loss on data instance $X^{(i)}$ and $r(W)$ is a regularization term to avoid overfitting on the dataset and it is governed by the parameter α. When $|D|$ is very large, the idea behind stochastic gradient descent is to estimate the gradient for a small sample of randomly chosen training inputs, $N \ll |D|$, obtaining a stochastic approximation of the loss function:

$$L(W) = \frac{1}{N} \sum_{i}^{N} f_W(X^{(i)}) + \alpha r(W) \tag{3.6}$$

[1] http://caffe.berkeleyvision.org/

The model computes f_W in the forward pass and the gradient ∇f_W in the backward pass. The parameters update $\triangle W$ is calculated from the error gradient ∇f_W. In particular, SGD updates the weights by a linear combination of the negative gradient $\nabla L(W)$ and the previous weight update V_t. The learning rate λ is the weight of the negative gradient. The momentum μ is the weight of the previous update. Formally, given the previous weight update V_t and current weights W_t, Equations 3.7 and 3.8 are used to respectively compute the update value V_{t+1} and the updated weights W_{t+1} at iteration $t + 1$.

$$V_{t+1} = \mu V_t - \lambda \nabla L(W_t) \tag{3.7}$$

$$W_{t+1} = W_t + V_{t+1} \tag{3.8}$$

The momentum μ tends to make deep learning with SGD both stabler and faster, by smoothing the weight updates across iteration and the learning rate λ determines the learning speed of neurons.

Chapter 4

One Class Support Vector Machine

In One-Class Classification (OCC), only information of one of the classes is available, the instances of this class will be called the target or positive instances. This means that just the target class is well characterized by instances in the training data, while the objects of other class, called the negative class, are either absent, poorly sampled or not well defined to form a statistically-representative sample of the other concept. OCC algorithms aim to build classification models using data from the target class only. These models should be able to distinguish examples of the target class from those of the negative class. Although this problem can be viewed as a standard binary classification problem, the absence of counter-examples prevents the direct use of traditional binary classification techniques [51].

OCC problems are prevalent in real world where positive and unlabeled data are widely available but negative data are hard or expensive to acquire. In the literature a large number of different terms have been used for this problem, because OCC has been considered and applied under many research themes, such as outlier or novelty detection and concept learning.

In this chapter, I will introduce the problem of one-class classification with respect to conventional multi-class classification problem, considering some possible one-class scenarios to motivate their importance. After a review of the basic approaches to solve OCC problems in the literature, I formalize the fundamental concepts of One Class Support Vector Machine.

4.1 Introduction to OCC Problem

4.1.1 One-Class Vs. Multi-Class Classification

A general multi-class classification problem can be decomposed into several two-class classification problems, therefore I am going to consider for the rest of the discussion the two-class problem as the basic classification problem.

In a two-class scenario, usually the two classes are labeled by -1 and $+1$. A training set is a set of objects which for each object $\mathbf{x}_i$, a label $y_i \in \{-1, +1\}$ is attached:

$$X_{tr} = \{(\mathbf{x}_i, y_i) | i = 1, ..., N\} \tag{4.1}$$

A function $f(\mathbf{x})$ has to be inferred from the training set, such that for a given feature vector $\mathbf{x}$ the label is obtained, $y = f(\mathbf{x})$, where:

$$f : \Re^d \rightarrow \{-1, +1\} \tag{4.2}$$

Considering Figure 4.1, a training set comprising of instances of apple, indicated by stars, and pears, indicated by pluses, is reported. Each object has two feature values, so they can be represented in a 2-dimensional feature space. In this example any binary classifier can be applied, with the result that the two classes can be separated without errors by a solid line. But if the test data object is from an entirely different domain, like the outlier apple in the right lower corner, the binary classifier will always classify this object as an apple or a pear, which results in both cases a classification error. This problem arises because in a conventional multi-calss classification problem, data from two or more classes are available and the decision boundary is supported by the presence of data objects from each class.

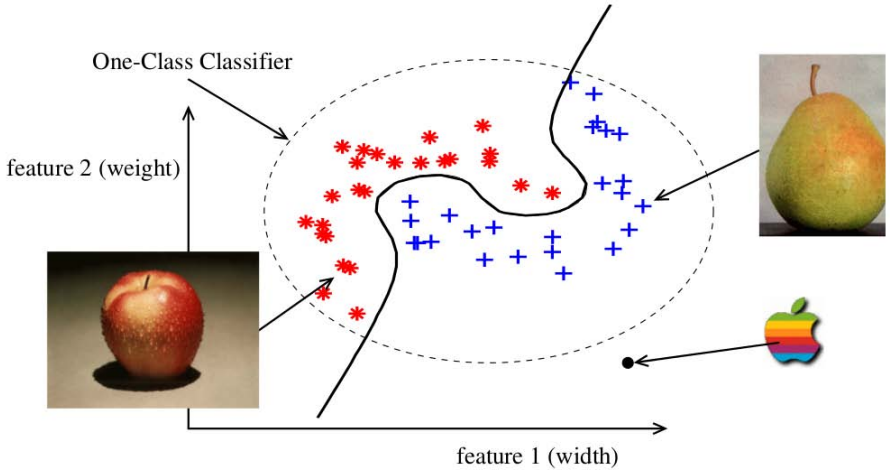


Figure 4.1: Example of a conventional and one-class classifier

To identify the outlier, a one-class classifier should be trained [55]. An example of a one-class classification is given by the dashed line. As stated earlier, in OCC problems the negative data objects are absent or available in limited amount, so only one side of the classification boundary can be determined only using the target data. This makes the problem of OCC harder than the problem of conventional two-class classification. The task in OCC is to define a classification boundary around the positive class, such that it accepts as many objects as possible from the positive class, while it minimizes the probability of accepting the outlier objects. In OCC, since only one side of the boundary can be determined, it is hard to decide on the basis of just one-class how tightly the boundary should fit in each of the directions around the data. It is also harder to decide which features should be used to find the best separation of the positive and outlier class objects.

4.1.2 OCC Solution Methods

For one-class classification several models have been proposed and most often the methods focus on outlier detection [51]. Three basic approaches to solve OCC may be identified:

- **Outlier Generation Methods** - Conceptually it is the simplest solution for outlier detection, because it generates outlier data around the target data, also named artificial counter-examples. The problem then becomes a two-class classification problem that can be solved by standard binary classification techniques. This approach may have poor predictive performance on new data in high dimensional problems, since it is necessary to assume a distribution for the outlier data.

- **Density Methods** - They directly estimate the density of the target data by assuming a uniform outlier distribution and by the application of Bayes rule. Some methods used for the estimation are Parzen Density Estimator [52] and Gaussian Mixture Models [53]. The need to assume a particular distribution for the target data limits this approach because the training data should be a typical sample from the true data distribution.

- **Boundary Methods** - This approach looks for a boundary or frontier around the target data. It tries to avoid the estimation of the complete density of the data and therefore also works with uncharacteristic data set, this not only provides advantages when just a limited number of samples is available, but it is even possible to learn from data when the exact target density distribution in unknown. For the boundary methods, it is sufficient that the user can indicate just the boundary of the target class by using examples. Among the machine learning techniques adapted for this purpose, one frequently used is the Support Vector Machine, such as One-Class Support Vector Machine [37] and Support Vector Data Description [54].

4.1.3 Application Scenarios

To motivate the importance of One-Class Classification, let us consider some scenarios. One of the most important application for OCC is outlier detection, with the aim to detect uncharacteristic objects from a dataset. An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism [56]. Some applications of outlier detection could be: fraud detection, for example purchasing behavior of a credit card owner usually changes when the card is stolen and abnormal buying patterns can characterize credit card abuse. In public health and medicine the occurrence of a particular disease, unusual symptoms and test results may indicate potential health problems of a patient. Detecting measurement errors, for example in data derived from sensors may contain measurement errors, removing such errors can be important in other data mining and data analysis tasks.

Another scenario in which OCC can be used is when one of the classes is sampled very well, while the other class is severely undersampled, due to the difficulty and the expensiveness to obtain such measurements. For instance, in a machine monitoring system where the current condition of a machine is examined to detect when it shows a problem. Measurements on the normal working conditions of a machine are very cheap and easy to

obtain. On the other hand, measurements of outliers are very expensive and it is impossible to generate all faulty situations. Another example is the automatic diagnosis of a disease. It is relatively easy to obtain positive data, all patients who are known to have a common disease, but negative data may be difficult to obtain since other patients in the database cannot be assumed to be negative cases if they have never been tested, and such tests can be expensive. Alternatively, if the disease is rare it is difficult to collect positive samples until a sufficiently large group has contracted that disease, which is an unsatisfactory approach.

There are other scenarios in which it may seem sensible to suggest that one-class problems should be reformulated into two-class ones because there is actually data from other classes that can be used for training. However, there are genuine one-class applications where it is inappropriate to make use of negative data during training.

For example, consider password hardening [57], which is a biometric system that strengthens the login process on a computer by not only requiring the correct password to be typed, but also requiring it to be typed with the correct rhythm. Password hardening is clearly a one-class problem, because a single user must be verified and during training time only data from that user is available, we cannot ask anyone else to provide data without supplying them with the password. Even in applications where instances from multiple classes are available at training time, it may be opportune to focus solely on the target class under consideration and contemplate a one-class setup.

4.2 One-Class Support Vector Machines

The technique I used for walk recognition is One-Class Support Vector Machine, described by Schölkopf et al. (1999) in [37]. In their work they construct a hyper-plane around the data, such that this hyperplane is maximally distant from the origin and can separate the regions that contain no data. They propose to use a binary function that returns $+1$ in a small region containing the data and -1 elsewhere. Moreover, they introduce a variable that controls the effect of outliers, i.e., the hardness or softness of the boundary around the data and suggest the use of different kernels, corresponding to a variety of non-linear estimators.

In this section, before starting with the theory dissertation about One-Class SVM, I present a brief summary of the basic concepts of Support Vector Machines (SVMs), based on the theory in [19] and [49], which are useful to understand One-Class SVM.

4.2.1 Support Vector Machines

A classification technique that has received considerable attention is Support Vector Machine (SVM). This technique has its roots in statistical learning theory and has shown promising empirical results in many practical applications. The basic idea behind SVM is the concept of decision boundary, a maximal margin hyperplane that separates two data classes. Maximizing the margin and thereby creating the largest possible distance between the separating hyperplane and the instances on either side of it, tends to have better generalization errors than those with small margins.

Intuitively, if the margin is small, then any slight perturbation to the decision boundary can have a significant impact on its classification performance. Classifiers that produce

decision boundaries with small margins are more susceptible to model overfitting and tend to generalize poorly on unseen examples.

If the training data is linearly separable, then a binary classification problem of N training examples can be considered. Each example is denote by a tuple $(\mathbf{x}_i, y_i)$ where $i = 1, 2, ..., N$ and $x_i = (x_{i1}, x_{i2}, ..., x_{id})^T$ corresponds to the attribute set for the i^{th} example. By convention, let $y_i = \{-1, 1\}$ denote its class label. The decision boundary of a linear classifier can be written in the following form:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{4.3}$$

where $\mathbf{w}$ and b are parameters of the model.

The training phase of SVM involves estimating the parameters $\mathbf{w}$ and b of the decision boundary from the training data. If the training data is linearly separable, then a pair $(\mathbf{w}, b)$ exists such that the following two conditions are met:

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b \geq 1 & if\ y_i = 1 \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 & if\ y_i = -1 \end{cases} \tag{4.4}$$

SVM imposes an additional condition requirement that the margin of its decision boundary must be maximal. An optimum separating hyperplane can be found minimizing the squared norm of the separating hyperplane, equivalent to the following constrained optimization problem:

$$\begin{cases} \min\limits_{\mathbf{w}} \dfrac{\|\mathbf{w}\|^2}{2} \\ y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, ..., N \end{cases} \tag{4.5}$$

Since the objective function is quadratic and the constraints are linear, this is known as a convex optimization problem, which can be solved using standard Lagrange multiplier method. First, the objective function must be rewritten in a form that takes into account the constraints imposed on its solutions. The new objective function is known as the Lagrangian for the optimization problem:

$$L_P = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i \left(y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) - 1 \right) \tag{4.6}$$

This lead to the following optimization problem:

$$\begin{cases} \max\limits_{\lambda_i} \left(\min\limits_{\mathbf{w}, b} L_P \right) \\ \lambda_i \geq 0 \qquad\qquad\qquad i = 1, ..., N \end{cases} \tag{4.7}$$

where the parameters λ_i are called the Lagrange multipliers. It is shown that the solution of problem 4.7 satisfies the following condition:

$$\lambda_i \left[y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) - 1 \right] = 1 \tag{4.8}$$

The constraint states that the Lagrange multiplier λ_i must be zero unless the training instance $\mathbf{x}_i$ satisfies the equation $y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) = 1$. Such training instances with $\lambda_i \geq 0$

lies along the hyperplanes $\mathbf{w} \cdot \mathbf{x}_i + b = 1$ or $\mathbf{w} \cdot \mathbf{x}_i + b = -1$, parallel to the decision boundary with maximum margin, and is known as a support vector. Training instances that do not reside along these hyperplanes have $\lambda_i = 0$.

Solving the optimization problem of Equation 4.6 is very difficult because it involves a large number of parameters. The problem can be simplified by transforming the Lagrangian into a function of the Lagrangian multipliers only, known as the dual problem, and solved using numerical techniques such as quadratic programming.

Once the λ_i are found, a feasible solution for $\mathbf{w}$ and b can be found and a test instance $\mathbf{z}$ is classified with the following decision function:

$$f(\mathbf{z}) = sign(w \cdot z + b) = sign\left(\sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \cdot \mathbf{z} + b\right) \tag{4.9}$$

If $f(\mathbf{z}) = 1$, then the test instance is classified as a positive class; otherwise, it is classified as a negative class.

In some cases SVM may not be able to find any separating hyperplane at all, because the data contains missclassified instances. This problem can be addressed by using a method known as the soft margin approach, that accepts missclassification of the training instances. This can be done by introducing positive-valued slack variables ξ_i into the constrains of the optimization problem 4.7.

Nevertheless, most real world problems involve non-separable data for which no hyperplane exists that successfully separates the instances in the training set. One solution is to transform the data from its original coordinate space in $\mathbf{x}$ into a new higher-dimensional space $\Phi(\mathbf{x})$, called the transformed feature space, so that a linear decision boundary can be found to separate the instances in the transformed space. Then, the training algorithm would only depend on the data through dot products in the form $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, such computation can be computationally expensive, because it may suffer from the curse of dimensionality problem. However, there were functions, called kernel functions, K such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, that can be used in the training algorithm instead of the dot product on Φ. This method is known as *kernel trick*, that allows inner products to be calculated directly in feature space, without performing any mapping.

4.2.2 One-Class SVM

Schölkopf et al. (1999) suggested a method of adapting the SVM methodology to the one-class classification problem [37]. Essentially, they propose an algorithm which computes a binary function which is supposed to capture regions in input space where the probability density lives, i.e. a function such that most of the data will live in the region where the function is nonzero. They applied the new proposed method on artificial and real-world data to provide a solid foundation to the theoretical result outlined in their work.

The Problem

The problem can be formulated in the following terms: suppose that a dataset has a probability distribution P in the feature space. Find a subset S of the feature space such that the probability that a test point from P lies outside S is bounded by some a priori

specified value. Supposing that there is a dataset drawn from this underlying probability distribution P, one needs to estimate a subset S of the input space such that the probability that a test point from P lies outside of S is bounded by some a prior specified value in range $(0, 1)$. The solution for this problem is obtained by estimating a function f that takes the value $+1$ in S and -1 on the complement $\bar{S}$ [50]. See Figure 4.2 for an example. The strategy of Schölkopf et al. is to map the data into the feature space corresponding to a kernel function, and to separate them from the origin, which is the only member of the second class, with maximum margin, employing standard two-class SVM techniques. For a new point $\mathbf{x}$, the value $f(\mathbf{x})$ is determined by evaluating which side of the hyperplane it falls on.

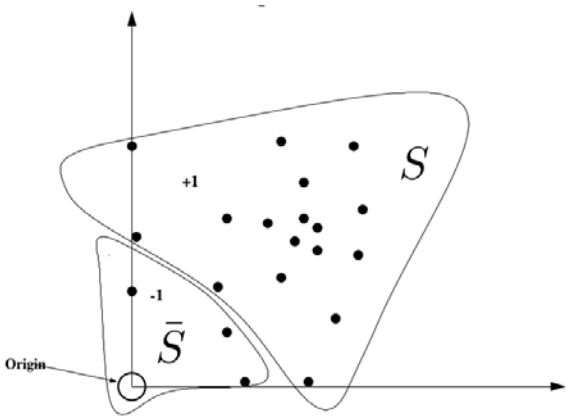


Figure 4.2: One-Class SVM Classifier, where the decision bound and the origin, the only member of the negative class, are reported.

The Algorithm

Consider a training data $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_l \in X$, where x_i is a d-dimensional vector, $l \in \mathbb{N}$ is the number of observations and X is some set. Also let Φ be a feature map $X \longrightarrow F$, i.e. a map into an inner product space F such that the inner product in the image of Φ can be computed by evaluating some simple kernel function $K(\mathbf{x}, \mathbf{y})$.

To separate the data set from the origin, one has to solve the following quadratic problem:

$$\begin{cases} \min\limits_{w,\xi,\rho} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{\nu l} \sum\limits_i \xi_i - \rho \\ (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) \geq \rho - \xi_i \\ \xi_i \geq 0 \end{cases} \tag{4.10}$$

Here, $\nu \in (0, 1]$ is a parameter that regularizes the fraction of outliers and Support Vectors (SVs).

Since nonzero slack variables ξ_i are penalized in the objective function, one can expect that if $\mathbf{w}$ and ρ solve this problem, then the decision function is:

$$f(\mathbf{x}) = sgn\left((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) - \rho\right) \tag{4.11}$$

Equation 4.11 will be positive for most examples $\mathbf{x}_i$ contained in the training set, while the SV type regularization term $\|\mathbf{w}\|$ will still be small. The actual trade-off between these two goals is controlled by ν.

Using multipliers $\alpha_i, \beta_i \geq 0$,they introduce a lagrangian formulation of the objective function of problem 4.10:

$$L\left(w, \xi, \rho, \alpha, \beta\right) = \frac{1}{2}\left\|\mathbf{w}\right\|^2 + \frac{1}{\nu l}\sum_i \xi_i - \rho - \sum_i \alpha_i\left((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) - \rho + \xi_i\right) - \sum_i \beta_i \xi_i \tag{4.12}$$

and set the derivatives with respect to the primal variables w, ξ, ρ equal to zero, yielding:

$$\mathbf{w} = \sum_i \alpha_i \Phi(\mathbf{x}_i) \tag{4.13}$$

$$\alpha_i = \frac{1}{\nu l} - \beta_i, \sum_i \alpha_i = 1. \tag{4.14}$$

In Equation 4.13, all parameters are called Support Vectors. Together with the kernel function, the SV expansion transforms the decision function 4.11 into the following kernel expansion

$$f(\mathbf{x}) = sgn\left(\sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho\right) \tag{4.15}$$

Substituting 4.13 and 4.14 in 4.12, one can obtain the dual problem:

$$\begin{cases} \min\limits_{\alpha} \dfrac{1}{2}\displaystyle\sum_{i,j} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \\ 0 \leq \alpha_i \leq \frac{1}{\nu l} \\ \displaystyle\sum_i \alpha_i = 1 \end{cases} \tag{4.16}$$

One can show that at the optimum, the two inequality constraints of problem 4.10 become equalities if α_i and β_i are nonzero, i.e., if $0 < \alpha_i < \frac{1}{\nu l}$. Therefore, one can recover by exploiting that for any such α_i, the corresponding pattern $\mathbf{x}_i$ satisfies

$$\rho = (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) = \sum_j \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) \tag{4.17}$$

Note that if ν approaches 0, the upper boundaries on the Lagrange multipliers tend to infinity, i.e. the second inequality constraint in 4.16 becomes void. The problem then resembles the corresponding hard margin algorithm, since the penalization of errors becomes infinite, as can be seen from the primal objective function in problem 4.10. It is still a feasible problem, since they have placed no restriction on the offset ρ, so it can become a large negative number in order to satisfy the constrains of 4.11.

To understand the meaning of the parameter ν, suppose to use a kernel which can be normalized as a density in input space, such as the Gaussian. If $\nu = 1$ is used the constrains in 4.16 only allows the solution $\alpha_1 = ... = \alpha_l = \frac{1}{l}$. Thus the kernel expansion reduces to an estimate of the underlying density. For $\nu < 1$, the equality constraint in 4.16 still ensures that the decision function is a thresholded density. However, in that case, the density will only be represented by a subset of training examples, the Support Vectors. Therefore, they show that ν is an upper bound on the fraction of outliers and a lower bound on the fraction of SVs. With probability 1, asymptotically, ν equals both the fraction of SVs and the fraction of outliers. The modification proposed to the SVM algorithm allows for the possibility of outliers and they have incorporated this softness of the decision rule using the ν-trick, obtaining a direct handle on the fraction of outliers.

The problem obtained has formulated in terms of quadratic programs (QPs) for computing region that capture a certain fraction of the data. These constrained optimization problems can be solved via an off-the-shelf QP package to compute the solution. Schölkopf describes a modified version of Sequential Minimal Optimization (SMO), an SV training algorithm proposed for classification, in which sequential optimization over pairs of input patterns is carried out. The optimization algorithm is a topic beyond the scope of this thesis, so for further details on this proposed technique can be found in [37].

4.3 Dimensionality Reduction

The term Dimensionality Reduction (DR) is often reserved for those techniques that reduce the dimensionality of a data set by creating a new set of features. DR is an important task in the preprocessing of data sets with a large number of features, that has been widely investigated for different purposes, such as clustering, classification or function approximation, becoming the focus of many research works where datasets can contain hundreds of features. Since the learning gets harder quickly as the dimensionality of the problem increases, using too many features will lead to different problems such as curse of dimensionality, decreased performances and overfitting [48]. These problems worsen with the presence of many irrelevant, noisy and redundant features.

There are many potential benefits of dimensionality reduction: facilitating data visualization and data understanding, reducing the measurement and storage requirements, reducing training and calculation times, defying the curse of dimensionality to improve prediction performance in term of accuracy of the model.

The subset of the potential input features can be defined through two different approaches: feature selection and feature transformation or feature extraction [47]. *Feature selection* reduces dimensionality by selecting a subsets of relevant features to the existing features. *Feature transformation or feature extraction* performs a transformation of the original features to a new reduced set of features which are more significant. Since in this thesis I used the term feature extraction to refer to the extraction of informative and non-reduntant features from an initial set of measured data, I will refer to this second technique of dimensionality reduction as feature transformation.

DR techniques in a supervised learning context are well posed in that there is a clear objective of discovering a reduced representation of the data where the classes are well separated. By contrast, DR in an unsupervised context is ill posed in that the overall ob-

jective is less clear. While one-class classification falls somewhere between the supervised and unsupervised learning categories, supervised DR techniques appear not to be directly applicable to OCC problems because of the absence of a second class label in the training data. In supervised learning the objective of DR is to optimize the performance of the final system, that is, minimize the classification error. However, in one-class classification performance estimation is difficult because the absence of counterexamples makes the estimation of the false positive rate hard and assumption-based.

There are several research works that tries to address this problem for OCC. For example, in [46] Villalba and Cunningham demonstrated the potential improvements by applying carefully selected DR techniques to one-class classification. They choose four different OCCs: support vector data description, a k-nearest neighbors approach, a k-means clustering approach and a Gaussian model. Then, they applied them to real datasets from different domains in combination with principal component analysis, Q-α algorithm, locality preserving projections and Laplacian score. The results obtained shows that DR algorithms have to be applied to a dataset in combination with a domain specific knowledge, because a global solution for all OOC problems does not exist. In [45] authors proposed a new filter feature selection approach for one-class classification, which uses the feature ranking of five feature selection measures from different paradigms adapted to the one-class scenario combined using different aggregation strategies.

In order to select the best subset of features, in the remaining part of this section I present the two technique used to face the problem of the DR in OCC. The first is *Principal Component Analysis* (PCA), a technique for unsupervised dimensionality reduction, that falls in the category of feature transformation techniques. The second one is a feature selection technique, known as *Sequential Feature Selection*.

4.3.1 Principal Component Analysis

Principal Component Analysis (PCA) is a dimension-reduction tool that can be used to reduce a large set of variables to a small set that still contains most of the information of the large set [19]. PCA is a mathematical procedure that transforms a number of correlated features into a number of uncorrelated features called principal components, where each pair of new distinct features has 0 covariance, features are ordered with respect to how much of the variance of the data each feature captures, the first feature captures as much of the variance of the data as possible and subject to the orthogonality requirement, each successive attribute captures as much of the remaining variance as possible.

The variability of a collection of multivariate data con be summarized by computing the covariance matrix $\mathbf{S}$ of the data, which has entries defined as $s_{ij} = covariance\,(d_{*i}, d_{*j})$ that is the covariance of the i^{th} and j^{th} features of the data, in other words it is a measure of how strongly the features vary together. The covariance between two jointly distributed real-valued random variables X and Y is defined as:

$$Cov(X,Y) = \mathbb{E}\left[(X - \mathbb{E}(X))\,(Y - \mathbb{E}(Y))\right] \tag{4.18}$$

A transformation of the data that has the properties mentioned before can be obtained by using eigenvalue analysis of the covariance matrix. Let $\lambda_1, ...\lambda_n$ be the eigenvalues of $\mathbf{S}$. The eigenvalues are all non-negative and can be ordered such that $\lambda_1 \geq \lambda_2 \geq .... \geq \lambda_n$.

Let $U = [u_1, ..., u_n]$ be the matrix of eigenvectors of $\mathbf{S}$. These eigenvectors are ordered so that the eigenvector i^{th} corresponds to the i^{th} largest eigenvalue. Finally, if the original data matrix $\mathbf{D}$ has been preprocessed so that the mean of each attribute is 0, then the following properties are valid:

- the data matrix $D' = DU$ is the set of transformed data that satisfies the conditions posed above.
- Each new features is a linear combination of the original features, that is the weights of the linear combination for the i^{th} feature are the components of the i^{th} eigenvector. The variance of the i^{th} new feature is λ_i.
- The sum of the variance of the original features is equal to the sum of the variance of the new features, which are called principal components.

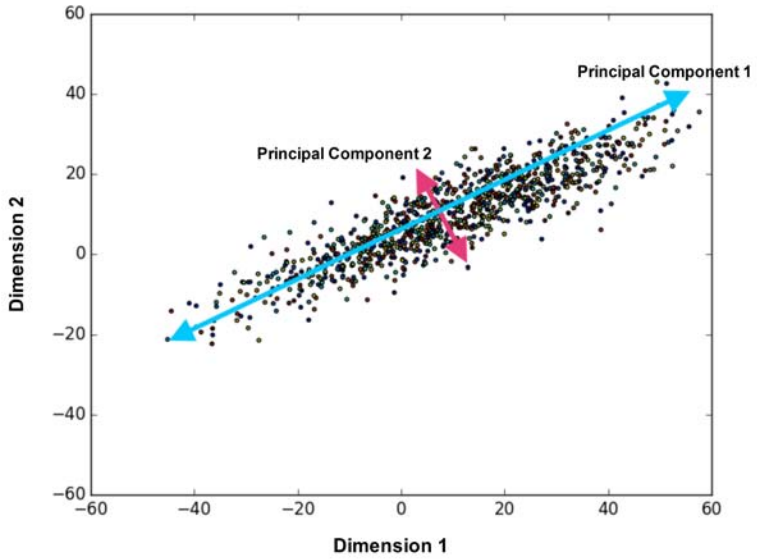


Figure 4.3: Principal Component number 1 and 2 capturing the directions in which the data has the most variance.

The eigenvector associated with the largest eigenvalue indicates the direction in which the data has the highest variability. In other words, if all the data vectors are projected onto the line defined by this vector, the resulting values would have the maximum variance with respect to all possible directions. The eigenvectors of $\mathbf{S}$ define a new set of axes. Indeed, PCA can be viewed as a rotation of the axes to a new set of axes that are aligned with the variability of the data, as shown in Figure 4.3 with the example of the first two principal components.

4.3.2 Sequential Forward Selection

Feature selection approach reduces the dimension of a dataset by finding the best minimum subset without transforming the data into a new set. The difficulty of extracting the most relevant variables is mainly due to the large dimension of the original variable set, the correlations among inputs which cause redundancy and the presence of features which do not have any predictive power. In the literature, variable selection methods are classified into three categories: filter, embedded and wrapper methods [47].

The filter approach consists of a pre-processing phase which is independent of the learning algorithm, and the subset of relevant variables is extracted by evaluating the relation between input and output of the considered system. An embedded approach performs the feature selection in the learning machine, where the features are selected during the training phase, by thus reducing the computational cost and improving the efficiency associated with the selection of variables.

Wrapper approaches consider the machine learning as a black box in order to select subsets of variables on the basis of their predictive power. The basic idea of the wrapper approach is to use the prediction performance or the classification accuracy of a given learning machine to evaluate the effectiveness of the selected subset of features. A generic scheme concerning wrapper approach is shown in Figure 4.4.

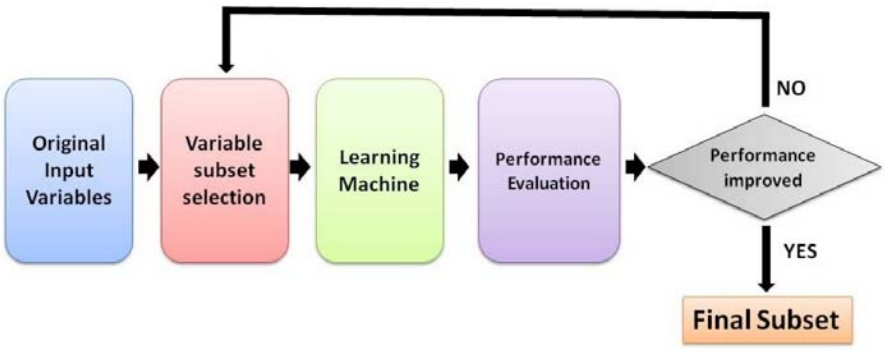


Figure 4.4: Generic diagram of a wrapper approach for feature selection.

Wrapper approaches generally achieve better recognition rates than filters since they are tuned to the specific interactions between the classifier and the dataset, but they are seen as a brute force approach, so they suffer from being computationally expensive, because the exhaustive search becomes unaffordable if the number of features is too large. The used Sequential Forward Selection (SFS) is a greedy search strategy [47]. Starting with an empty set of features, SFS sequentially adds the feature that results in the highest performance function when combined with the features that have already been selected. The performance measure, used in this thesis, in combination with the SFS method is the square Root of the Mean Square Error (RMSD).

Chapter 5

The Recognition System

This chapter describes how the methods presented in the previous chapters are combined to develop a recognition system based on smartphone-acquired walking data, from the data acquisition to the final identification of a subject. There are four main blocks involved in the identification process: data acquisition, preprocessing, feature extraction and the final identification. The main building blocks of the proposed technique are reported in Figure 5.1 and in the next sections I will highlight the fundamentals steps involved in each block and the interaction between them.

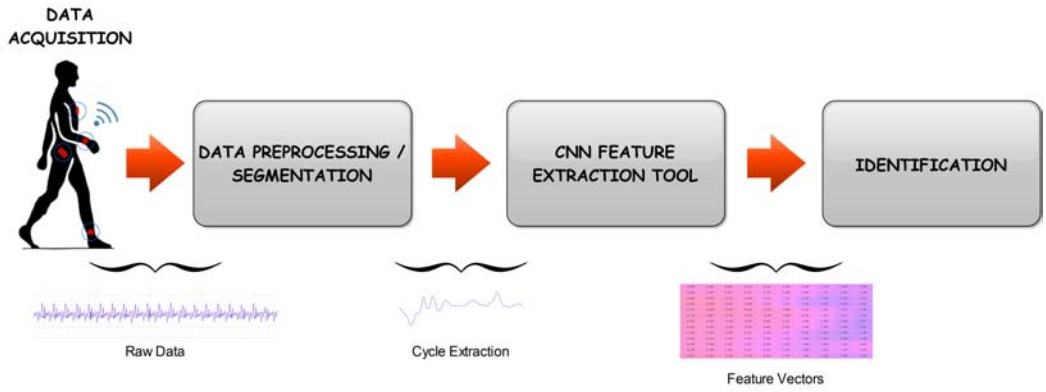


Figure 5.1: General schema of the proposed biometric system for the human gait recognition problem.

5.1 Data Acquisition and Preprocessing

Data acquisition and preprocessing have already been described in Chapter 2. Now I want to only present these two phases in a comprehensive, user-friendly and visual manner.

Data are acquired via a smartphone located in the right pocket of the trousers of a subject. The gathered data are: tri-axial-accelerometer, -gyroscope and -magnetometer

signals, gravity acceleration, linear acceleration and rotation vector. Only the acceleration and gyroscope data are used in the subsequent phase and I called them *raw data*.

Note that, all the computations of the preprocessing steps are performed on a PC. Therefore, the proposed system performs *static identification* of the subject, in which the identification mechanism will make a decision about the claimed identity after the person has walked.

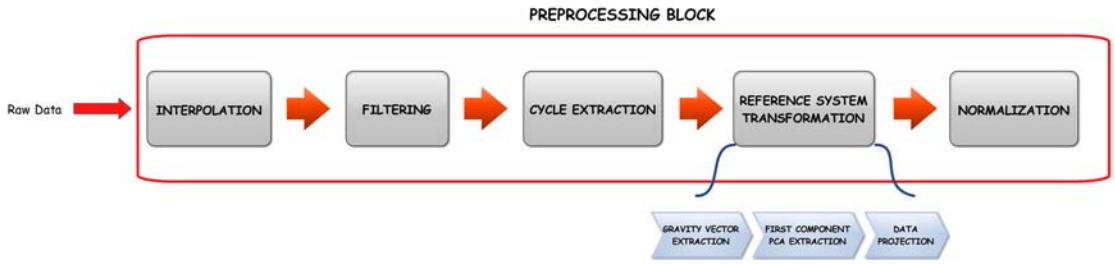


Figure 5.2: Preprocessing block of the proposed biometric system. The internal steps involved in the computation are highlighted.

As shown in Figure 5.2, after transferring the data on a PC, the preprocessing phase starts and it consists of the five sequential steps, listed below:

- **Interpolation** - It has the aim to ensure a fixed time interval between two consecutive samples and to deal with smartphones, that do not have a constant sampling frequency.
- **Filtering** - It is performed for the purpose of smoothing signals and removing high peaks, in order to eliminate noise problems.
- **Cycle Extraction** - This step has the aim of extracting the repetitive patterns, called cycles, in each data trace. In this manner, each walk is divided in a set of gait cycles.
- **Reference System Transformation** - This step is required, because during data acquisition a fixed orientation of the smartphone is not enforced. Therefore, a new reference system is extracted directly from the data. The axes of this new system are the gravity vector and the first principal components obtained from the flattened acceleration data.
- **Normalization** - It is used to force a fixed length for the extracted cycles, in order to respect the constraint imposed by the subsequent feature extraction phase, that requires data of fixed length.

Hence, this preprocessing block transforms the raw data acquired from the smartphone into a set of gait cycles of the same length and orientation, ready for the subsequent feature extraction phase.

5.2 Feature Extraction

In this thesis, I use a pre-trained CNN models as a generic feature extractors. To this purpose, the first step is to train a CNN model for classification and the second step is to use the model as a feature extractor by removing the final layers of the network.

For the development of the network model, I used Caffe[1], an open framework for deep learning algorithms and a collection of reference models. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors. The code is written in C++, with CUDA used for GPU computation and bindings to Python/Numpy and MATLAB. Caffe provides a complete toolkit for training, testing, finetuning and deploying models for the purpose of object classification, object detection and learning semantic features. The key aspects of the Caffe architectures are data storage and layers. Caffe stores and communicates data in 4-dimensional arrays called *blobs*. Blobs provide a unified memory interface, holding batches of images, other data or parameters. Blobs simplify the computational and mental processes of dealing with mixed CPU/GPU operations by synchronizing from the CPU host to the GPU device as needed. A Caffe layer has two key responsibilities for the operation of the network: a forward pass that takes the inputs and produces the outputs, and a backward pass that takes the gradient with respect to the output, and computes the gradients with respect to the parameters and to the inputs, which are in turn back-propagated to preceeding layers [6].

The network architecture, training and testing phases, and feature extraction are developed with the Python interface of Caffe and to speed up operations, the Caffe model is accelerated by drop-in integration of NVIDIA cuDNN[2] for GPU computation.

The CNN architecture developed is summarized in Table 5.1.

Layer	Layer Type	Size	Output Shape
1	Input Layer	8×200	-
2	Convolutional	10 1×10 filters	(10,8,50)
3	Convolutional	20 4×10 filters	(20,5,12)
4	Max Pooling	1×3, stride (1,2)	(20,5,6)
5	Fully Connected + Activation	40 hidden neurons	40
6	Fully Connected + Activation	40 hidden neurons	40 (labels)
7	SoftMax	40 way	40

Table 5.1: Caffe based CNN architecture.

The considered Convolutional Neural Network comprises seven layers, counting the input and softmax layers. Since CNNs are powerfull models for image recognition, they take advantage of the fact that the input consists of images, in particular the layers of a CNN have neurons arranged in 3 dimensions: width, height and depth. For this reason, the different signals acquired for each motion sensors are stacked in order to obtain a matrix

[1] http://caffe.berkeleyvision.org/

[2] https://developer.nvidia.com/cudnn

of dimension 8×200, in which the eight rows correspond respectively to acceleration and gyroscope in the x-,y- and z-direction, acceleration magnitude and gyroscope magnitude. 200 is the number of samples of the normalized cycles, that is the length of the signals. In this manner, each extracted cycle is transformed into a 2D-matrix and can be used as input for the CNN. Usually the depth parameter is equal to 3, which corresponds to the channels of an image, in our case depth is equal to 1.

In the following, I will refer to the Convolutional Layer as Cx, to the Max Pooling Layer as Px and to the Fully-Connected Layer as FCx, where x is the layer index.

Layer C2 is a convolution layer with 10 feature maps. Each neuron in each feature map is obtained by convolving the input with a filter of dimension (1×10). Each filter slides across the input with a stride equals to 4 in the width direction and 1 in the height direction, adding a zero-padding of 3 in the width direction. Therefore, the resulting shape of the feature maps is 8×50. This first layer can be seen as a filtering on the x axis of the input signals, since a linear activation function is applied to the output values of neurons. Moreover, the filter has height 1, hence the convolution is carried out considering a single signal of the input matrix at a time, therefore the low-level features learned are specific for each signal.

Layer C3 is a convolutional layer with 20 feature maps, the convolution step is performed with filters of dimension (4×10) with the same stride and zero-padding of C1. The resulting shape is 5×12. Hence, compared to the previous convolutional layer, the filter used in C3 has height 4. Then C3 learns features that are a combination of those of the accelerometer and gyroscope acquired signals.

P4 is a pooling layer and it performs a downsampling of the input feature maps using the max operation, with filters of size 1×3 applied with a stride of 2 along width direction and of 1 along height direction, therefore the resulting feature maps have dimension 5×6. Its function is to reduce the spatial size of the representation to reduce the amount of parameters and computation time of the network, and consequently to control overfitting.

After P4, there are two fully-connected layers FC5 and FC6, that compute the dot product between their input vector and their weight vector, to which a bias is added. Each output of the neurons in FC5 depends on all the neurons of the feature maps obtained from P4. FC6 computes the output values of the network each depending on the 40 hidden neurons of the previous layer. The result of FC5 and FC6 are passed through a hyperbolic tangent function in order introduce non-linearity.

As mentioned before, the first step is to train the network described for classification, the used learning algorithm is Stochastic Gradient Descent (SGD), which is driven by a loss function and specifies the goal of learning by mapping the current network weights to a scalar value specifying the goodness of the parameter settings. Hence, the goal of learning is to find a setting of the weights that minimizes the loss function, computed in the SoftMax layer.

A CNN model is obtained by training the network for classification. For this reason, the proposed CNN architecture has to be trained with gait cycles of a set of subjects, derived from the preprocessing phase. When the performance obtained are satisfying in terms of accuracy and loss function minimization, the CNN weights are frozen and the network is then used as a black-box tool for feature extraction. As shown in Figure 5.1, the feature extractor consists of the two convolutional layers, the max pooling layer and the first fully

connected layer, so the output values of the FC5 layer are the features used in the next step of the biometric system. Hence, this step transforms a gait cycle onto a finite set of features, which are arranged as a vector.

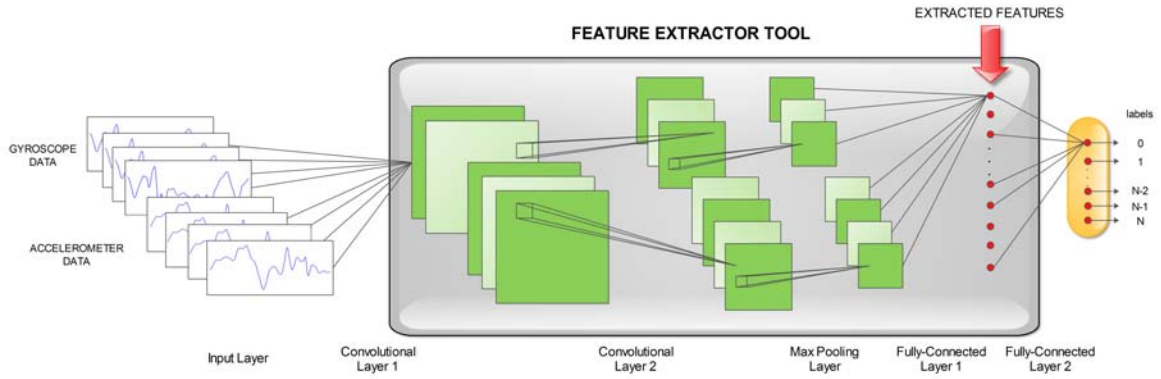


Figure 5.3: Complete schema of the CNN architecture proposed, in which is highlighted the block used as feature extractor tool.

5.3 Identification

The identification phase of the biometric authentication system is performed through the One-Class Support Vector Machine (OSVM) of Chapter 4. The features extracted from the CNN model, which are a reduced representation of the relevant information of the gait cycles, derived from the preprocessing phase, are arranged in vectors. So that to each feature vectors corresponds a walk cycle of a person and they become the input of the identification phase.

The purpose of identification is to recognize one person at a time. Therefore, an OSVM model has to be trained on the feature vectors extracted from the walks of a target subject. After the learning phase is completed, the model is frozen and it is used for identification. Since the model only depends on examples of a target subject, it is clear that to identify different subjects, it is necessary to train different OSVM models on the data of each subject.

Figure 5.4 shows the internal step of the identification phase. This block involves a first step of dimensionality reduction of the features, in order to reduce the complexity of the problem for the subsequent steps. Then, a pre-trained OSVM model is applied to identify the subject and at the end I evaluated the results obtained through the Equal Error Rate (EER), which is one of the best single description of the error rate of an algorithms and a widely used performance metric for biometric systems [44].

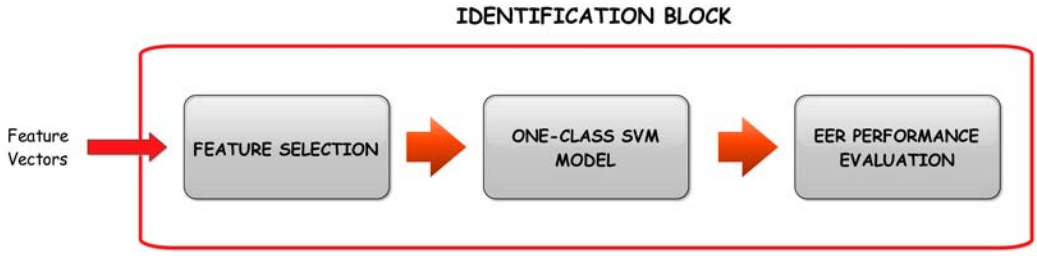


Figure 5.4: General schema of the Identification Block, where the internal steps are highlighted.

Each block shown in Figure 5.4 has to be optimized, in terms of algorithms used, parameter tuning and performance evaluation. In the dimensionality reduction step I tested two different approaches: Principal Component Analysis (PCA) and Sequential Forward Selection, with which I have set the optimal number of features used in the next step. Also the OSVM have to be optimized in terms of its intrinsic parameters and on the number of train cycles needed to build a model able to identify a subject.

All the test are carried out with MATLAB R2015b[3], a high-level technical computing language and interactive environment for algorithm development, data visualization and data analysis. In particular for the OSVM model, I used the `fitcsvm` method, which can train binary support vector machine classifiers and can handle one-class SVM using Schölkopf algorithm.

The last step utilizes Equal Error Rate (EER), a performance measure widely used for identification systems. More specifically, the OSVM model assigns a score for each predicted cycle, indicating how well the model recognized the subject. The OSVM score is the signed distance from an instance to the decision boundary ranging from $-\infty$ to $+\infty$. A positive score for an instance indicates that it is predicted to be in that class, a negative score indicates otherwise. For every score value, the following two error statistics can be calculated:

- **False Positive Rate** (FPR) or **False Matching Rate** - A false match occurs when the system incorrectly approves an identity. Hence, FPR is equal to the fraction of negative examples predicted as a target example.
- **False Negative Rate** (FNR) or **False non-Matching Rate**. A false non-match or rejection occurs when the system incorrectly denies the target subject identity. Hence, FNR is the fraction of target examples predicted as a negative example.

In a perfect biometric system with no errors, FPR and FNR are both equal to zero. Unfortunately, perfect biometric systems do not exist and a variable security level has to be defined. In order to attain a desired trade-off between FPR and FNR, the security level, λ, is increased to make it harder for an impostor to be identified or it is decreased to make it easier for a rightful subject to be identified. In Figure 5.5, the point at which the the

[3]http://uk.mathworks.com/products/matlab/

curves of FPR and FMR intersect is known as Equal Error Rate, the rate at which the number of subjects, who are incorrectly identified and incorrectly rejected is equal.

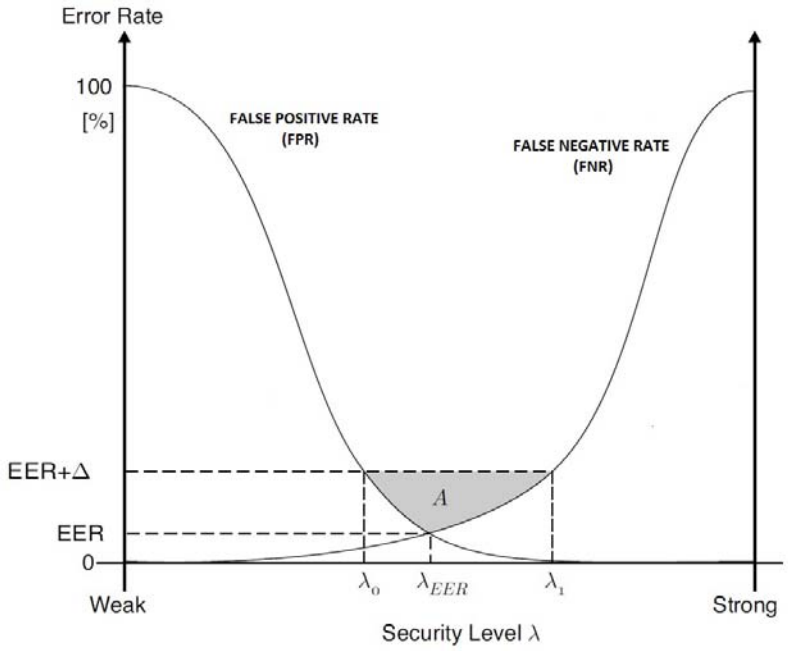


Figure 5.5: Plot of the dependencies of FPR and FNR as a function of the security level. EER is highlighted.

I used this parameter to assess the final performance of the designed biometric system, whereby the lower the EER is, the lower is the error rate of the model. An important issue is to decide the decision threshold based on the score. Select the EER score value as decision threshold is common choice because it guarantees the same FPR and FNR, but for a high security scenario the decision threshold must be increased, and as a consequence the FPR decreases and the FNR simultaneously increases. Instead, for a low security scenario the decision threshold can be small.

Chapter 6

Results

In this chapter, I present the analysis and the experiments carried out in order to set up the biometric system blocks. In particular, I present the optimization performed on the Convolutional Neural Network used as a feature extractor tool and an evaluation of the extracted features combined with other classical machine learning algorithms in order to assess how they generalize on data that was never used. Then, I report how the One-Class SVM (OSVM) model parameters have been set to obtain good identification results.

6.1 CNN Model Optimization

The CNN architecture described in Chapter 5 is the result of several tests carried out in order to obtain a network with the ability to have good prediction performance on the input data. Since there is no one-size-fit-all network architecture and no magical formulas, with which to set all the free hyper-parameters of each layer, the common way to design a suitable CNN architecture is to perform trial-and-error tests. The number of hyper-parameters to be taken into account is not small. There are parameters concerning the whole architecture, such as the number of filters per convolutional layer, filter dimensions, the number of overall layers, namely, convolutional, pooling, activation and fully-connected layer and the way to stack them together. Moreover, each layer has specific parameters to deal with. The effectiveness of each parameter setting depends on the network objective, on the intrinsic complexity of input data, on the generalization accuracy and training time. In this biometric system, CNN is used as a feature extractor from the input data, in order to obtain a set of features able to identify a single subject among all the others.

In this section, the optimizations performed to come up with our final network architecture are not reported, because it is not the purpose of this work. What I want to highlight is the ability of the network to extract good discriminative features from a small set of data.

The CNN was trained on 40 subjects for classification. Data cycles are divided into a train and a test set. In Figure 6.1 I show the classification accuracy on the test set, defined as the number of correct predictions divided by the total number of predictions. Results are obtained by varying the number of cycles considered for each subject in the train set and for each number of train cycles eight runs of the network are performed, plotted as scatter

points with a x-marker. The red curve and dots represent the median and its evolution. Then, the standard deviation is calculated on the data and it is reported in the figure as error bars to show how closely the median is likely to reflect the true values of each run.

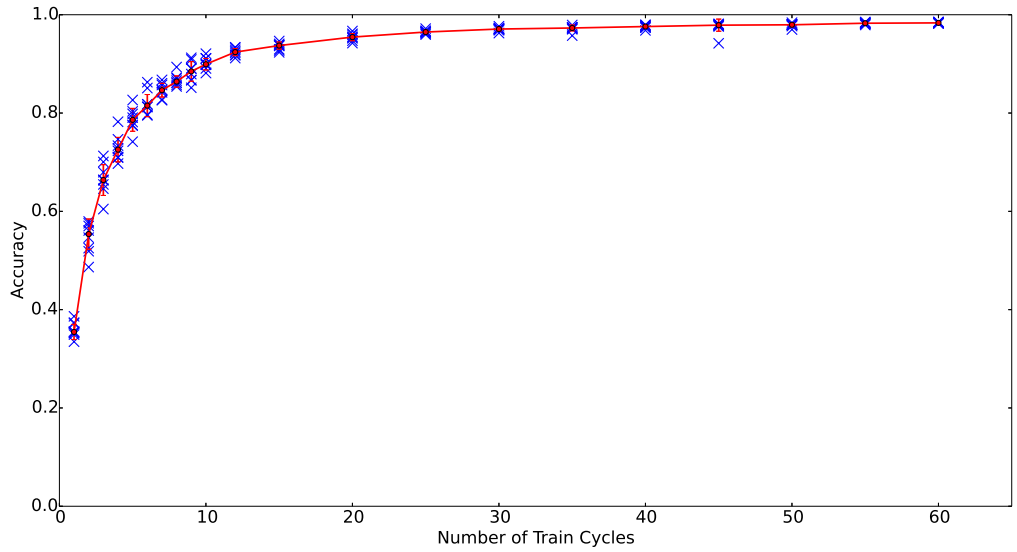


Figure 6.1: Output test accuracy versus number of cycles per subject employed during the training phase of the CNN.

It can be seen that test accuracy increases as the number of training cycles increases, towards a saturation point, that can be identified between 20 and 30 cycles per subject. This behavior shows that with a small number of cycles for each subject the CNN model can achieve a prediction accuracy of up to 95%. Note that, the total number of cycles of 40 subjects is more than 51.000 and 40 cycles per subject are equal to only 1600 cycles for the training set. Despite a training set of small size, the network is able to achieve an accuracy around the median value of 97,6%. This is an important result, because a small number of gait cycles can be representative of more walks for the same subject. Hence, the correlation discovered in the preprocessing phase, continues to exist also across different walks performed by the same subject. This correlation is well captured by the features learned during the training phase of the CNN. In fact, they have a high predictive potential, even if they are learned from a small set of data. In contrast with classical machine learning algorithms that need high amount of training data to learn a model with the risk to incur in overfitting.

Moreover, as the test accuracy and the number of training cycles increases, the standard deviation grows smaller. Error bars represent a description of how confident the median value is to represent the impact of the measurements. Hence, the more the original value ranges close to the median value, the narrower the error bars are and more confident the

median value is. This means that the CNN model is able to make more stable predictions and to learn a model, that represents better the input data and can generalize on data not seen.

The same considerations can be made for the loss function, which during the learning phase of the CNN is minimized and represents the price paid for inaccuracy of predictions.

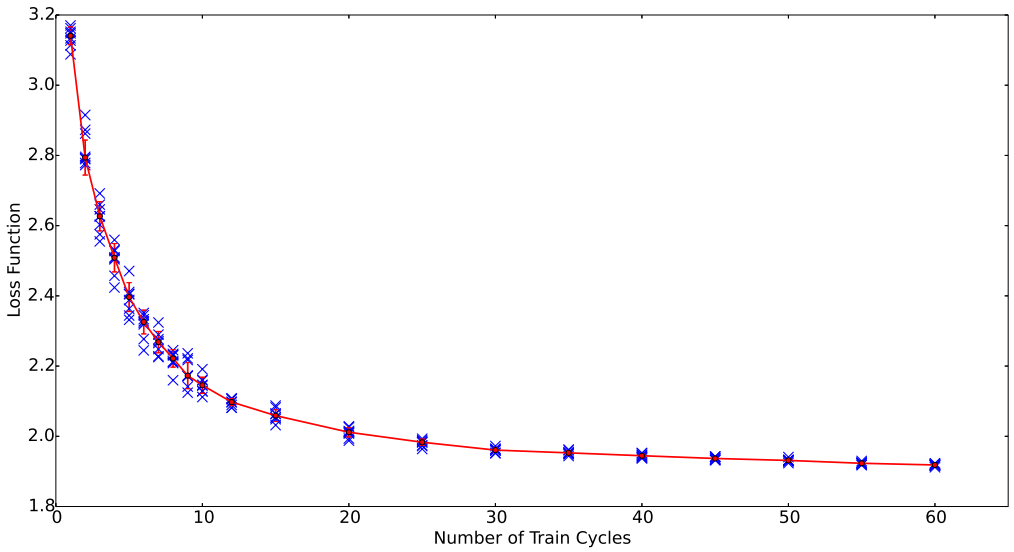


Figure 6.2: Output loss function versus number of cycles per subject employed during the training phase of the CNN.

Based on the results of Figure 6.1 and 6.2, I fixed the number of train cycles per subject equal to 40. This is an acceptable value, since the two performance measures derived from the simulations have already reached a saturation point, at which there is a much lower likelihood that these two measures differ significantly. Moreover, it has been shown that any significant improvements can be gained adding more than 40 cycles to the training set.

Another important parameter to optimize is the number of features at the output of the first fully-connected layer, which are the features to be extracted and used in the identification step. Previous simulations are done with this parameters fixed to 40, but I investigated if this number is a good choice, in order to assess how a greater or lower number of features affects the prediction performance.

For this reason, I trained eight CNN models with a different number of subjects, as it can be seen in the legend of Figure 6.3, where I reported the obtained accuracy. I used a training set with the fixed dimension of 40 cycles per subject and I varied the number of output features of the first fully-connected layer, starting from 5 and adding five features at each iteration up to 60.

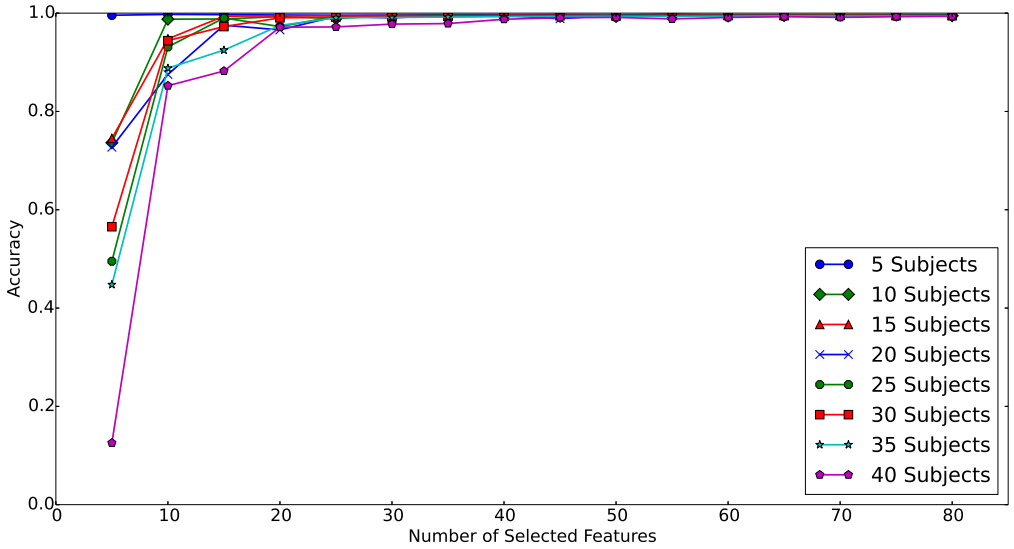


Figure 6.3: Output test accuracy versus number of features of the first fully-connected layer of the CNN, by varying the number of subjects for the training phase.

From the simulations of Figure 6.3, when the number of features is low the models that performs better are those trained with a small number of subjects. This is a reasonable behavior because to discriminate between a small number of subjects, a high number of features is unnecessary. When the number of features increases also the models trained on a high number of subjects reach comparable test accuracies, which are close to the maximum possible value. This demonstrates that 40 features are a good choice, with which all the models reached a saturation point and no improvement were obtained increasing it.

In the next section I evaluated how these 40 features generalize on data not seen from a CNN model using different classification algorithms.

6.2 CNN Features Evaluation

One of the key aspects of the biometric system proposed is the features extracted from the CNN, that can positively or negatively affect the performances of the subsequent identification step. As shown in the previous section, I extracted 40 features relying on the accuracy and loss function metrics achieved testing the pre-trained CNN for a multi-class classification problem. This method is a good approach to understand what is a good number of features to be extracted from the network, but it tells us nothing about the real ability of the features to generalize on other classification problems and what they represent in relation with the input data.

Intuitively, low-level learned features of top convolutional layers provide a low-level representation of the input data. In the case of image data, these low-level features may

consist of simple edge filters. In the higher layers of the network, the model begins to learn more complicated patterns, like object parts that would be useful to represent high-level non-linear complex functions. For image classification, in the literature, several research works try to understand how CNNs work, to achieve a better knowledge of the internal operations and to visualize the extracted features from each convolutional layer, in order to develop better models that are able to extract meaningful properties from the data.

M. Zeiler et. al in [8] introduced a novel visualization technique that gives insight into the function of intermediate feature layers and the operation of the classifier. They reveal that the features learned are far from random and uninterpretable patterns. Rather, they showed many intuitively desirable properties such as compositionality, increasing invariance and class discrimination as one ascends the layers. This approach and others, [7] and [58], rely on the introduction of deconvolutional, unpooling layers and on visualization techniques that allow to observe the evolution of features during training and to identify the correspondence between learned features and input data.

Since the input of the proposed biometric system are not images, but time series, a visualization approach to understand what the 40 features represent is not a viable option, but the mentioned results are encouraging. Therefore, in this thesis I evaluated the significance of the extracted features using them as input for other classification methods and I measured the relative importance of them on the basis of how much influence the output prediction.

The first experiment carried out was to compare the accuracy measure for 4 classification models, namely, Binary Classification Decision Tree, Naive Bayes with kernel smoothing density estimate, K-nearest neighbors (kNN) with 10 nearest neighbors and Support Vector Machines (SVM) with linear, polynomial and radial basis function (RBF) kernels. I did not perform any parameters tuning on the models and the default parameters are used according to the MATLAB documentation.

First of all, I trained a CNN model with data acquired from 30 subjects leaving out 10. The model obtained is then used to extract the features from the data of the 10 subjects not seen during the training phase. The aim of this approach is to use features of people that the CNN have never seen, because it is reasonable to think that features extracted from data not seen are worse in terms of predictive power than those already seen. I am also interested in the generalization errors on these unseen data.

I divided the feature vectors in a training and a test set and I trained the four classification models by varying the dimension of the training set. Then I calculated the classification accuracy on the remaining data, which form the test set. The classification results are shown in Figure 6.4.

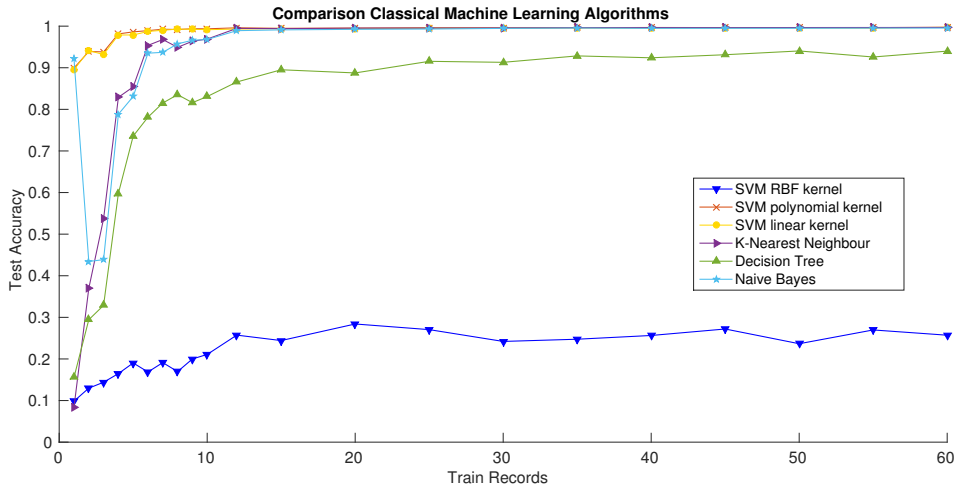


Figure 6.4: Comparison of Machine Learning Algorithms trained on 10 subjects not used in CNN learning phase. Classification test accuracies versus the number of train records for each subject are reported.

Figure 6.1 shows the resulting test accuracy on varying the number of train records for each person in the training set. From these results, we can see how all the classification algorithms, except for the SVM with RBF kernel, share a fast evolution towards the saturation value. In fact, the accuracy reached in these cases is near to the maximum value. SVM with linear and polynomial kernel are the methods with the best performance, because they have an accuracy of over 90% and they reach the saturation point with less than 10 records for each subjects in the training set. kNN and Naive Bayes grow less quickly with respect of the first two algorithms, in fact they need more training records to reach their saturation point, however the performance are similar when more than 20 records are used. The overall performance of the decision tree model are excellent, but worse than previous methods.

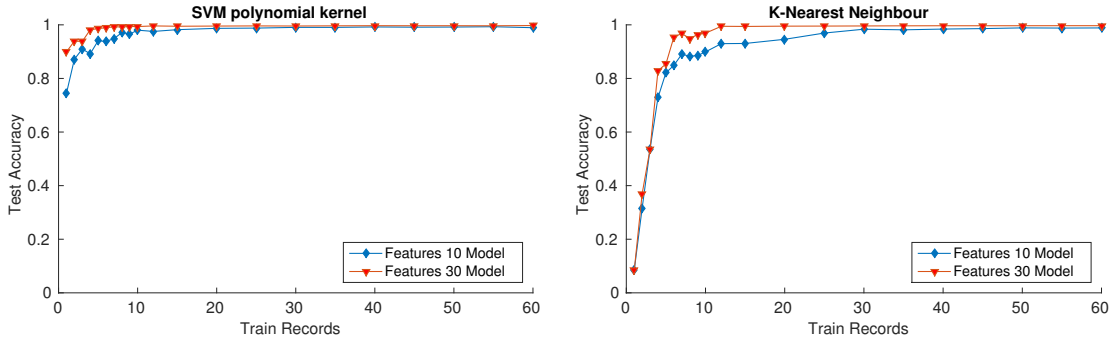
The worst performance are obtained by SVM with a radial basis kernel. The poor classification accuracy can be motivated by the small number of training records involved in the training phase, because in general to induce an accurate class boundary SVM requires a larger amount of data. This drawback arises when the classification problem has a high dimensionality and when the number of features is close or larger than the number of training records, as in our case.

However, except for the SVM model with RBF kernel, accuracies obtained show excellent results with a few records, like the results obtained in the previous section with CNN. This means that the extracted features have a strong predictive power and although the models are trained with a small number of training records, they suffice to produce a model that has good performance.

This is a first proof that CNN can extract useful features that give competitive prediction accuracy combined with other classification algorithms to yield classification results

that are consistent with the original CNN or even better.

The second test carried out has the purpose of evaluating the difference between the performance obtained on the features extracted from CNN models trained on different number of subjects. It is reasonable to think that features learned from a small set of subjects are less robust and predictive than those learned from a larger set. For this purpose, I trained two CNN models: one trained on 10 subjects and one trained on 30. I extracted the features of 10 person never used in both models and I calculated the classification accuracy only for SVM with polynomial kernel model that has obtained the best performance in the previous tests and K-nearest neighbor on varying the train records for each subject.



(a) SVM with polynomial kernel case

(b) K-Nearest Neighbour case

Figure 6.5: Comparison of test accuracy versus the number of train records, obtained from two CNN models trained on 10 and 30 subjects

In Figure 6.5, the red curve rapresents the test accuracy obtained on features extracted from the CNN model trained on 30 subjects and the blue curve on 10 subjects. As expected there is a gap between the two curves, which is more visible when the number of records is less than 20 records in case of SVM with polynomial kernel and less than 40 records in the case of K-nearest neighbour, but in both cases the gap decreases as the number of records increases. However, surprisingly the accuracy obtained from the CNN model trained on 10 subjects has a competitive prediction accuracy compared with the CNN model trained on 30 subjects. This is an interesting result, because although the CNN model is trained on the walks of a few subjects, one can extract meaningful features from the data, which have the potential to distinguish different subjects that have not yet been seen. Increasing the number of subjects in the training of the CNN, the predictive quality of the features is reinforced as is shown by the better prediction accuracy obtained using features extracted from the CNN model trained on 30 subjects.

The most significant experiment carried out aims to further assess the importance of the CNN extraced features with respect to hand-crafted features directly calculated on the extracted cycles after the segmentation phase.

For this reason, I used a CNN model trained on 30 subjects to extract the features data of 10 unseen subjects. On the other hand, data of the same subjects are used to calculate a

set of features, which are shown to be effective using triaxial accelerometers as reported in [2] and [1]. The features were calculated on the extrated cycles of the eight signals, which are accelerometer and gyroscope data on x-,y-z-axis as well as the magnitude vectors. The features calculated are mean, variance, the difference between the maximum amplitude value and the minimum one and the following:

- **Mean Trend** - Each cycle is divided in 20 windows with no overlap. The mean of each of these windows is calculated and subtracted from the mean of the next window. The mean trend is calculated as follows:

$$\mu T = \sum_{i=2}^{20} (|\mu_i - \mu_{i-1}|) \tag{6.1}$$

  where μ_i is the is the mean value of the i_{th} window.

- **Windowed Mean Difference** - It is calculated subtracting from the overall mean μ of the cycle the mean of each window as follows:

$$\mu D = \sum_{i=1}^{20} (|\mu - \mu_i|) \tag{6.2}$$

- **Variance Trend and Windowed Variance Difference** - They are computed similarly to the mean treand and windowed mean difference, except for the variance within each window, which is calculated as follows:

$$\sigma T^2 = \sum_{i=2}^{20} (|\sigma_i - \sigma_{i-1}|) \tag{6.3}$$

  and

$$\sigma D^2 = \sum_{i=1}^{20} (|\sigma - \sigma_i|) \tag{6.4}$$

- **Spectral Entropy** - It is defined as follows:

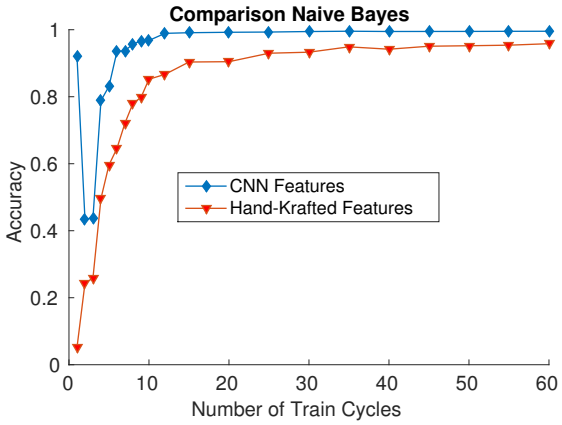
$$PSE = -\sum p_i \log_2 p_i \tag{6.5}$$

  where p_i is the normalized Power Spectral Density so that it can be viewed as a Probability Density Function.

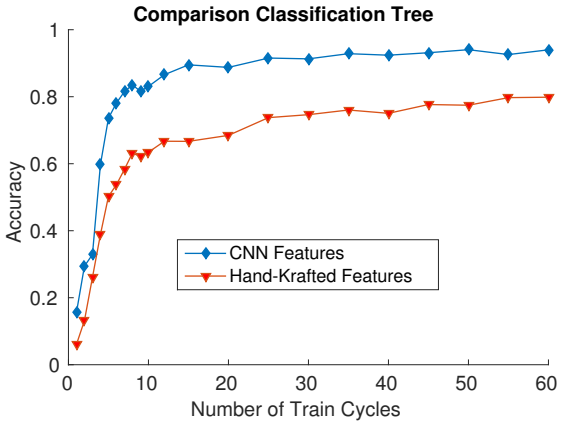
- **Zero Crossing Rate** - It is equal to the number of times the signal changes sign in a given cycle segment. If the cycle has length L, it can be calulated as:

$$ZCR = \frac{1}{2} \sum_{i=1}^{L} |sgn(x(i)) - sgn(x(i-1))| \tag{6.6}$$

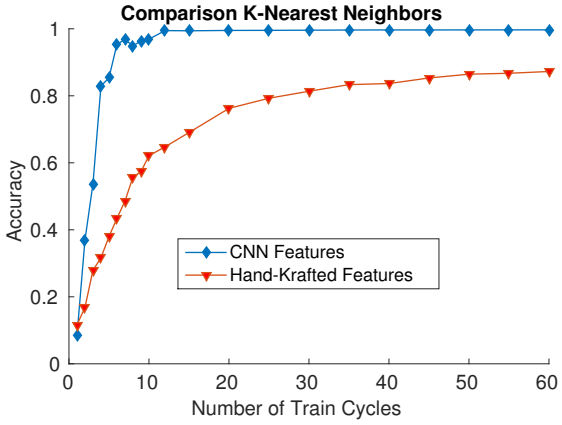
- **Bin Count** - A histogram is constructed by dividing the entire range of amplitude values into a series of consecutive non-overlapping bins and then the values falling into each interval are counted. We use 5 bins in order to obtain 5 features.



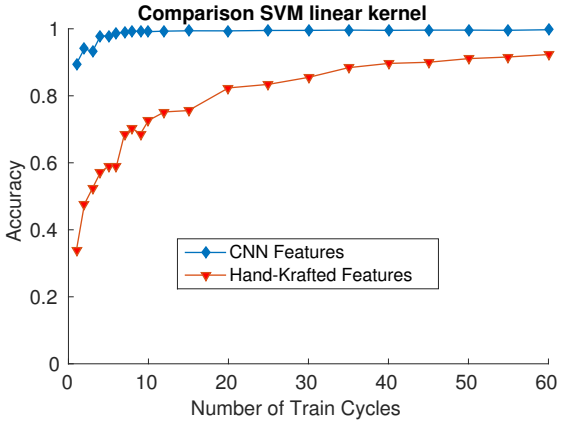
(a) Naive Bayes case



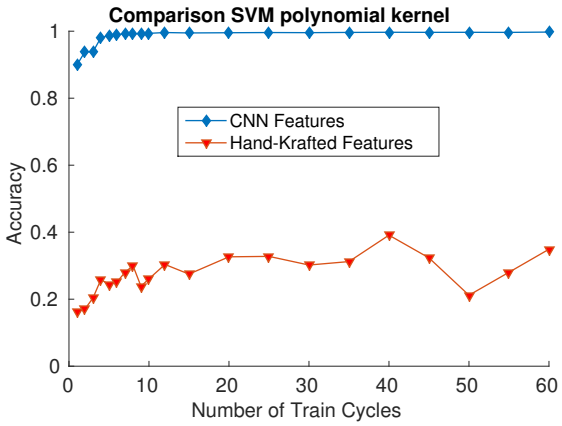
(b) Binary classification decision tree case



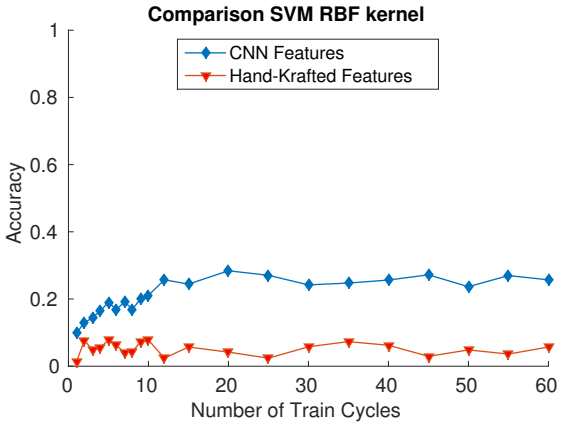
(c) K-Nearest Neighbour case



(d) SVM with linear kernel case



(e) SVM with polynomial kernel case



(f) SVM with RBF kernel case

Figure 6.6: Comparison of classification algorithms trained on feature vectors obtained from a pre-trained CNN and calculated directly from the data cycle. Test accuracy versus the number of train records is reported.

The two sets of features are used to train and test the six classification algorithms used in the first experiment. Figures 6.6 compares the classification accuracy for each algorithms achieved when using the CNN feature extractor versus commonly-used hand-crafted features.

From these results all the classification algorithms trained on the set of CNN extracted features outperform those trained with the set of hand-crafted one. The classification accuracies achieved using the features extracted by the CNN were consistently better than the classification accuracies achieved when using the hand-crafted features. This a remarkable result, which demonstrates that a pre-trained CNN extracts features that are meaningful and effectively performs generalized feature extraction also on data of subjects never seen. Additionally, these features have a much higher predictive power than the hand-crafted one, considering how they influence the output prediction accuracy.

Moreover, the prediction accuracy obtained with hand-crafted features seems to be influenced by the training set size. Considering only the classification models that has good performance, one can notice that the evaluated measure does not reach a saturation point, but they tend to grow with the number of training records. In contrast with the behavior of the models trained with CNN features, which has a fast growth of the accuracy measure with a small amount of training records.

The tests reported in this section are an attempt to demonstrate that other classification algorithms can be used successfully with features extracted from a pre-trained CNN to yield excellent prediction accuracy. Even if we do not have a precise interpretation about what these features represent in relation with the input data, there are no doubts that they are an impressive baseline for classification, prediction and recognition tasks. Additionally, these results show the effectiveness and generality of the features learned with Convolutional Neural Networks, which have to be considered strong candidates for feature extraction on time signals and not only on images.

Moreover, the choice to use a machine learning algorithm in order to automatically extract features from the data has proved successful, in particular from the last result obtained and the good performance achieved by the SVM models are promising for subsequent identification phase with One-Class SVM model.

K-Nearest Neighbor performance tells us something on the local distribution of the data. K-NN is sensitive to the local structure of the data, since it is based on the principle that the samples within a dataset will generally exist close to other instances with common properties. When the label of an unclassified instance is unknown it can be determined observing the class of its nearest neighbors. Therefore, k-NN can be used to estimate arbitrary distributions. The results obtained by this algorithms tell us that instances of different subjects are close to each other and not skewed in the feature space. This a good initial basis for one class classification problem.

6.3 OSVM Model Optimization

The identification process is performed via a pre-trained One-Class Support Vector algorithm, whose theoretical fundations are described in Chapter 4. In order to obtain

satisfying performance, a model optimization phase is needed to show how the parameter settings influence the identification of the target subject. In particular, during the training phase, I carried out an optimization on the model and on the kernel parameters, on the cycle number included in the training set and on the number of features used. Finally, several tests on walk data never used in the training phase were evaluated to assess the final classification performance.

For these experiments, the number of correct predictions made, i.e, the classification accuracy, does not provide specific performance information on the single classes involved in the classification task. Since an OSVM problem can be viewed as a binary classification problem, because of the presence of a target and a negative class (i.e., all the other subjects), a more suitable way to present prediction results has to be used. A confusion matrix, whose structure is shown in Figure 6.7, provides all the information needed to determine how the model performs on both classes. It is a 2×2 matrix, where each entry denotes the number of records of class i predicted to be of the class j.

	Target Class (Predicted)	Negative Class (Predicted)
Target Class (Actual)	TRUE POSITIVE (TP)	FALSE NEGATIVE (FN)
Negative Class (Actual)	FALSE POSITIVE (FP)	TRUE NEGATIVE (TN)

Figure 6.7: A confusion matrix for a binary classification problem

In particular, true positive (TP) and false negative (FN) correspond respectively to the number of target cycles correctly and wrongly predicted. True negative (TN) and false positive (FP) corresponds respectively to the number of negative cycles correctly and wrongly predicted. The performance measures used to evaluate the experiments are: recall, precision and F1-measure, which are listed below.

- **Recall or True Positive Rate** (TPR) measures the fraction of target cycles correctly predicted by the classifier and the formal definition is the following:

$$TPR = \frac{TP}{TP + FN} \tag{6.7}$$

- **Precision** determines the fraction of cycles that actually results to be positive, among those that the classifier predicted to belong to the target class.

$$P = \frac{TP}{TP + FP} \tag{6.8}$$

- **F1-measure** (F1) summarizes into one metric precision and recall. F1 represents a harmonic mean between recall and precision.

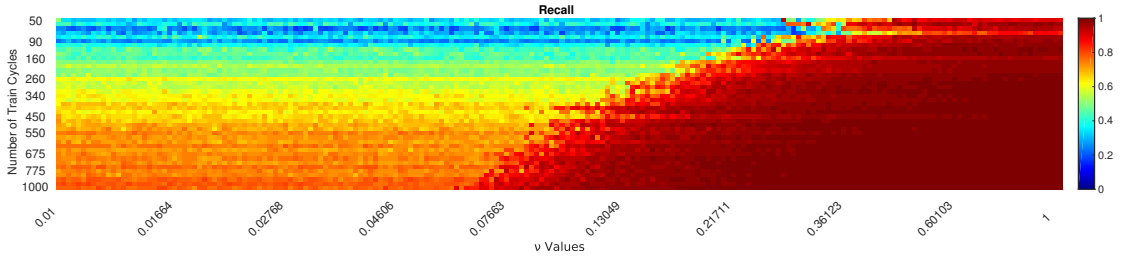
$$F_1 = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision} = \frac{2}{\frac{1}{Recall} + \frac{1}{Precison}} \tag{6.9}$$

The OSVM algorithm takes in input the feature vectors extracted from a Convolutional Neural Network trained on 39 subjects. For the subsequent experiments, the feature vectors of a single subject not seen by the CNN is used as the target data for the OSVM. The negative data are the feature extracted from the others 39 subjects already used in the training phase of the CNN.

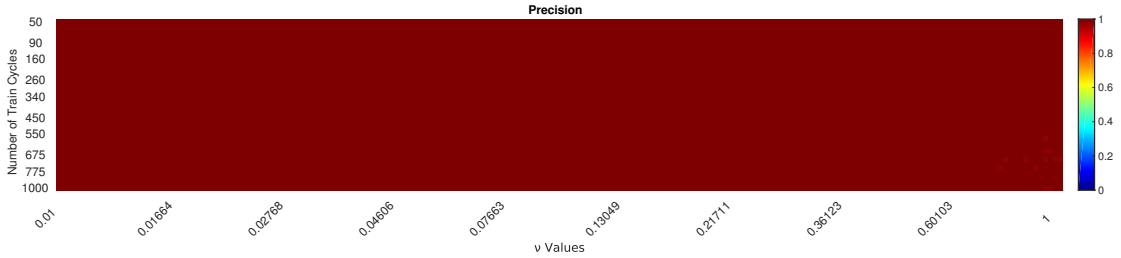
The OSVM algorithm is trained on the walking data of one subject with radial basis kernel and one of the key parameters of OSVM algorithm is ν, which represents the trade-off between the fraction of Support Vectors (SVs) and the fraction of outliers considered during the learning phase. Togheter with the kernel parameter, usually known as γ, ν drives the solution and the shape of the bound learned. According to the MATLAB formulation of the algorithm, a small value of ν leads to fewer support vectors, and, therefore, a smooth, crude decision boundary. A large value of ν leads to more support vectors, and therefore, a curvy, flexible decision boundary. Also, $\nu \in (0, 1]$.

On the other hand, γ can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. The interaction between these two parameters is not trivial and practical means to determine them does not exist, hence an optimization step is needed. However, to simplify the discussion γ is set by default using a heuristic procedure implemented in the MATLAB software.

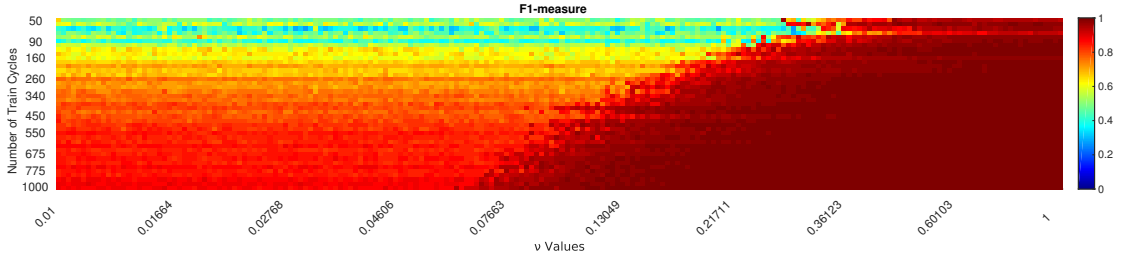
In the first experiment of this section I used a target class, composed of six walk data acquired from a single subject, which corresponds to about 7000 target cycles. Data are divided into a training set, which consists only of targeted data cycles randomly chosen and a test set, which is composed of target data cycles plus a balanced number of data cycles of the negative class, that are data belonging to other 39 subjects. Varying parameter ν, I evaluated the performance obtained from models trained on varying the number of the targeted data cycles. In particular, I considered 200 ν values equally spaced in the interval $[0.01, 1]$ and the train set dimension varies in the range $[50, 1000]$. Figure 6.8 reports the results obtained by the models on the test set. In particular, each entry of the matrices corresponds to a performance measure obtained in correspondence of the i_{th} ν value and the j_{th} dimension of the training set.



(a) Recall performance matrix.



(b) Precision performance matrix.



(c) F1-measure performance matrix.

Figure 6.8: Performance measure obtained from a pre-trained OSVM model by varying the number of train data and the ν parameter.

Recall measure represents how well the model is predicting the target cycles. The matrix of Figure 6.6.(a) shows an optimization region corresponding to high ν values and high number of training cycles. Good values of ν can be found in the interval $[0.6, 1]$ for every training cycle number, because increasing ν means to include more Support Vectors for the definition of the decision boundary built by the model. This interval becomes wider when the number of training cycles increases, because the fraction of SVs increases proportionally to the number of training cycles.

Surprisingly, every combination of the two parameters tested does not affect the precision measure, Figure 6.6.(b), which in most cases reaches the maximum possible value. The higher the precision is, the lower the number of false positive errors committed by the trained model. Hence, also with a low number of cycles the model has low probability to incorrectly identify subjects belonging to the negative class, but on the other hand recall has poor performance. For this reason to maximize both precision and recall, F1 measure,

in Figure 6.6.(c), is used to summarized the behavior of the two metrics into a single one. As stated before, F1-measure is the harmonic mean of two numbers, hence a high value of F1-measure ensures that both precision and recall are reasonably high.

The aim of this test is to set the parameter ν and the number of cycles for the subsequent test. Figure 6.7 shows better how F1-mesure varies as a function of the number of cycles and fixing three different value of ν.

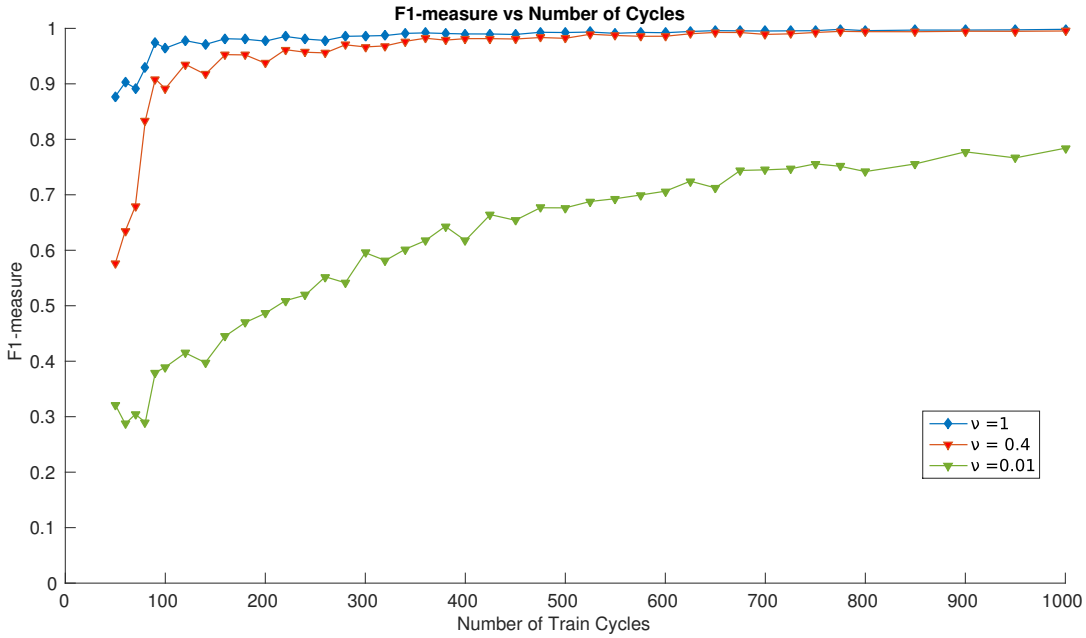


Figure 6.9: F1-measure performance by varying ν value versus the number of train cycles.

In Figure 6.7, it can be seen that the F1-measure for $\nu = 0.4$ and $\nu = 1$ tends to have the same performance when the number of the training cycles is high enough, instead of a model with low ν values needs to be trained with a high number of training cycles to obtain good performance. For this reasons, in subsequent tests I set ν equal to 0.6, which ensures excellent performance and avoids overfitting that may occur if ν value is too high. Moreover, I set the number of training cycles to 300, which corresponds to 2- or 3-minute walk. It is a reasonable amount of data that are needed to capture the data complexity. Remember that the 300 data cycles are selected randomly between more than 7000 target cycles of six walking acquisitions.

Previous experiments are obtained using all the features extracted from the CNN, but one-class classification problem can be viewed as a binary classification one. In fact, a target and a negative class are present. Therefore, due to the reduced complexity of the problem with respect to that faced during CNN training, it is reasonable to think that a smaller number of features can be used to achieve the same or even better classification results. As mentioned in Chapter 4, dimensionality reduction is performed utilizing PCA

and Subsequent Forward Selection. In particular, I use PCA in three different ways, listed below:

- The first approach is to apply PCA just on a subset of target data disjointed from the train and test set. Once PCA technique returns the principal component coefficients, also known as loadings, they are used to convert training and test set features into sets of linearly uncorrelated features called principal components (PCs). The most common practice is to retain high variance features in descending order, as it was done. For brevity, I called this method Classical PCA Approach.

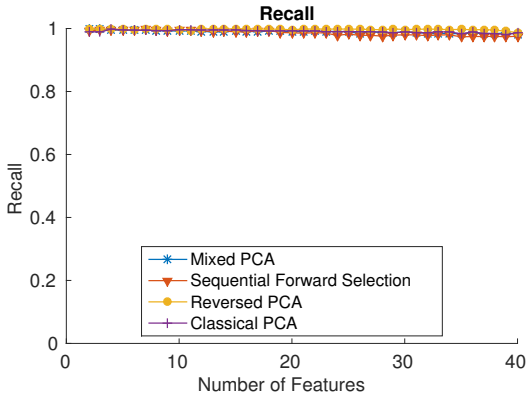
- The second approach used is inspired by the work of Tax and Müller [11], where they stated that retaining the high variance features is not always the best option for one-class classification, instead retaining the low variance features provide smaller classification error. Following this approach and unlike the previous case, I retained PCA features starting from the low variance one in ascending order. I called this method Reversed PCA Approach.

- In the last approach, PCA technique is applied on a set including both target and negative data cycles, in order to obtain more robust coefficients thanks to the presence of negative instances, which are not present in the previous methods. In classical one-class problems examples of the negative class are difficult to obtain, but this is not the case, because we have collected several data from other subjects. Hence, this is a viable approach and I called it Mixed PCA Approach.

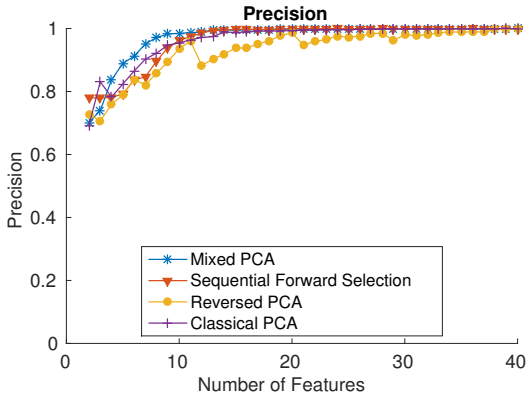
In addition, I also investigated Sequential Forward Selection technique, which starting from an empty feature set selects a subset of features by sequentially selecting them until there is no improvement in predictions, as explained in Section 4.3. Also in this case, cycle data used during feature selection are not considered in the subsequent training and testing phases of the model.

Since a suitable reduced dimension of the problem cannot be chosen a priori, in this work, I addressed the dimensionality reduction problem in two phases. First, I tested the four described methods with the same approach of the previous experiment, that is, I trained the OSVM model on 300 training cycles randomly chosen from six walking data acquisitions of the target subject with $\nu = 0.6$. Then, I tested the performance on target data and on data of the negative class. In the second phase, I selected the method with the best performance and I tested a pre-trained OSVM model on completely new walking data of the target subject in order to evaluate how the model generalizes.

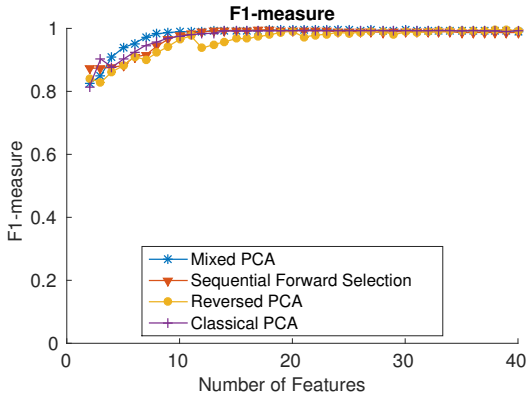
In Figure 6.8, I show the results of the first phase. In particular, the OSVM models are trained on varying the number of features obtained by the tested reduction methods and I calculated recall, precision and F1-measure.



(a) Recall measure performance.



(b) Precision measure performance.



(c) F1-measure performance.

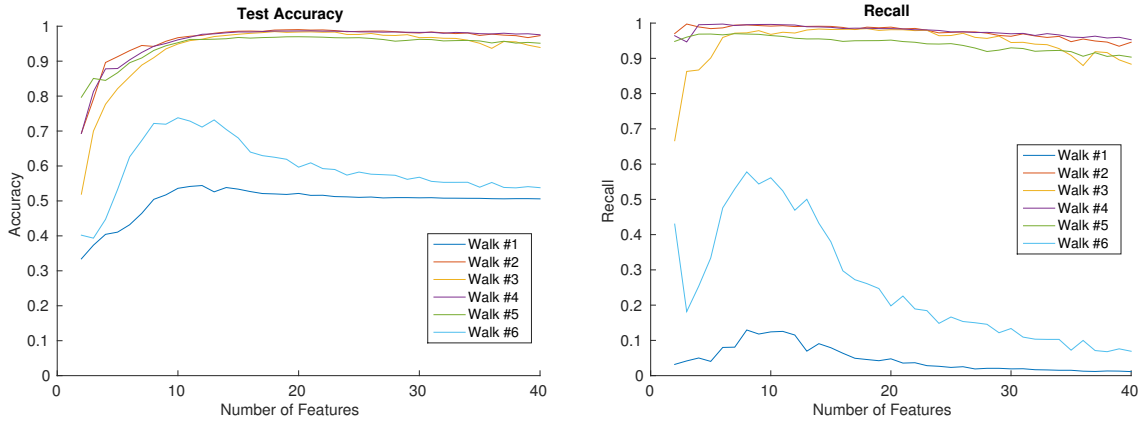
Figure 6.10: Comparison of dimensionality reduction methods by varying the number of features.

All the tested methods achieve excellent performance for the recall measure, shown in Figure 6.8.(a). This is due to the selection of the ν parameter and the number of training cycles, that allow to achieve recall values close to the maximum value reachable. On the contrary, important considerations have to be done on the precision measure. For a number of selected features below 15, all the methods share a gradual evolution to a saturation point very near to the maximum value, that all reach when the number of features is equal or grater than 20. This behavior is justified by the fact that density of the training cycles decreases exponentially when the dimensionality of the problem increases, so due to this sparsity, it is possible to find a decision bound around the target data, able to easily separate them from the negative data. Note that, mixed PCA performs better than the other methods, even when the number of features is low, because the PCA coefficients are calculated, not only from target cycles, but also on negative data. This helps the PCA transformation to consider the variability of the negative data, so that PCA works much better. Unfortunately, in a pure one-class settings, with no negative cycles at training time

Reversed PCA obtains excellent overall performance, but compared to the other methods is slightly inferior, in particular for the precision measure. Intuitively retaining principal components, which explain less variance of the data, has the result to project them in less space as possible. This is a good approach for the target cycles, because the distribution of the negative data is unknown and it seems to maximize the chance that the model will accept as few negative cycles as possible. But when negative data have to be tested, the PCA transformation is applied on these data and the result is that we are packing the negative cycles in the same way of the target one. Hence, the precision on the target class worsen, because the false positives increase.

Since the four methods have comparable performance, for the test of the second phase I decide to choose the classical PCA approach, which is computationally less expensive than the Sequential Forward Selection method and it involves only target data to compute the dimensionality reduction, a desirable property in a real world scenario where data only of the target subject are available.

In order to evaluate how the OSVM model generalizes on new walking data never used in the CNN and OSVM training phase, I sequentially left out all the cycles of a single walk acquisition of the target subject from the six initial walking data and I trained the OSVM model on the cycles of the remaining five walks. So that, all the cycles of the excluded walk acquisitions became the test set, to which I added a balanced number of cycles of the negative class. In Figure 6.9, I show the results obtained on the test set thus created, where I evaluated accuracy and recall.



(a) Test accuracy performance by varying the number of features.

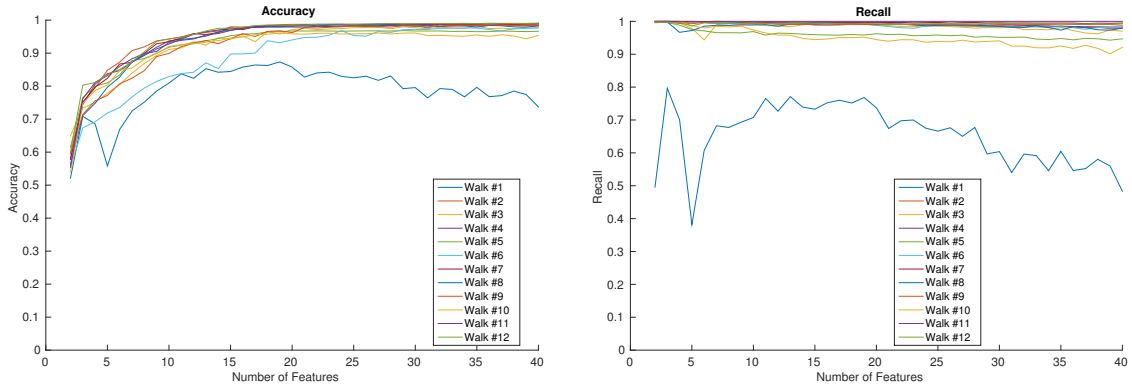
(b) Recall measure performance by varying the number of features.

Figure 6.11: Performance measure of 6 walk acquisitions tested on an OSVM model trained on the remaining 5 walk acquisitions.

From the accuracy results, shown in Figure 6.9.(a), all tested walk data achieve excellent performance, except for two OSVM models, that have bad accuracy on walk number 1 and 6. In particular, from the recall results shown in Figure 6.9.(b), the bad accuracy

performance is obtained due to classification errors on the target class, meaning that the models trained on the remaining walk acquisitions are not able to recognize the subject. The miss-classifications are a problem of acquisition, because from other experiments, not reported in this thesis, the predictions remains poor changing the parameter settings, that is ν, number of training cycles and dimensionality reduction approach. Hence, these walks are not performed in the proper manner, probably the subject walked too fast stopping several times or during the acquisition the smartphone was not always in the same position. Another aspect to point out is the decrease that recall tends to have as the number of selected features is increased. This behavior is a direct result of the curse of dimensionality, because using too many features results in overfitting of the model on the training data.

In the previous tests, there are two walks that achieve bad performance due an improper data acquisition. To avoid this problem, I investigated the possibility of learning more robust models including new cycles from new acquisitions and I evaluated whether a larger number of acquisitions may reinforce the model predictions in the presence of high variability in the data. To this purpose, I added six more walks of the target subject to the previous six and, as before, I trained the model leaving out one walk acquisition, which is considered as the test set. In this case, I use 600 training cycles, because the number of walk acquisitions is doubled. Again, I reported accuracy and recall on varying the number of selected features, as shown in Figure 6.10.



(a) Test accuracy performance by varying the number of features.

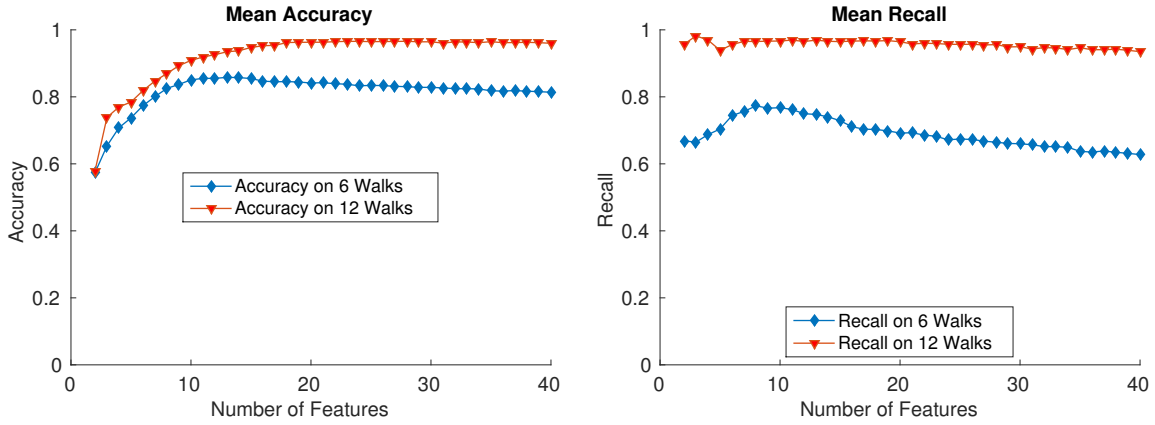
(b) Recall performance measure by varying the number of features.

Figure 6.12: Performance measure of 12 walk acquisitions tested on an OSVM model trained on the remaining 11 walk acquisitions.

These results confirm what we expected. The model trained on a set of acquisition large enough can recognize better noisy data and reinforce ita predictions on data never used during the training phase. In particular, walk number 6 is now recognized. Instead, walk number 1, still obtains poor performance but greatly improved compared to the results of Figure 6.9.

The improvement obtained can be better assessed from the comparison between the results obtained when using 5 walk acquisitions and 11 walk acquisitions to train the model.

In Figure 6.11, I show the accuracy and recall results obtained on each test walk divided by the number of walks. On average, the accuracy obtained with a model trained on 11 walk acquisitions is improved by approximately 10% and the recall measure by 26%.



(a) Test accuracy performance.

(b) Recall measure performance.

Figure 6.13: Average performance comparison by varying the number of features on walk acquisition not used during training.

Considering the whole biometric system, this result means that the model used in the identification step can be trained periodically on data of the target subject to reinforce the classifier and every time that new acquired data are included in the training set, the number of training cycles has to be incremented proportionally, in order to capture the data complexity. Hence, based on the results obtained I choose the number of features equal to 20, which represents a good trade-off between the prediction accuracy on the target subject and negative class.

Finally, I evaluate the performance of the whole system on a target subject using the Equal Error Rate (EER) measure, previously described in Section 5.3. Hence, the final CNN model is trained with data of 39 subjects, where I used 40 cycles for each subject. I applied the pre-trained CNN network on the data cycles of a target subject, with which I extracted 40 features from the output of the first fully-connected layer for each cycle. Feature vectors are used as input of the OSVM algorithm in order to learn a model to identify the target subject. I used $\nu = 0.6$, 300 target cycles and a radial basis kernel to train the OSVM. Then, I created a test set, which comprises cycles of the target subject never used during the training and cycles belonging to negative subjects. Then, I applied the pre-trained OSVM and I calculated the False Positive Rate (FPR) and False Negative Rate (FNR) measures to find the Equal Error Rate, shown in Figure 6.15.

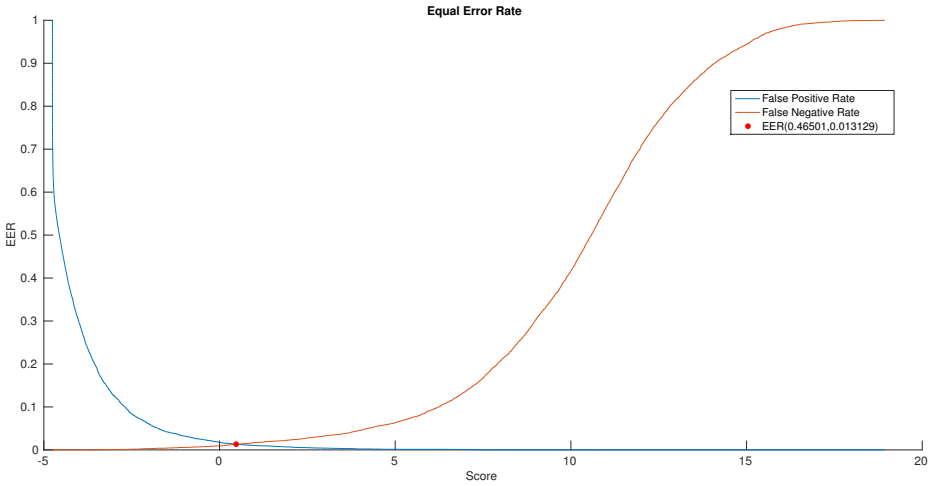


Figure 6.14: The EER point for the cumulative score functions FPR e FNR.

The EER point (0.465, 0.013) has been found for the score value 0.465, i.e. the decision threshold, at which FPR e FNR functions have the same value of 0.013. As expected, the EER value is low and the designed biometric system can be defined accurate and precise. Given the output scores of the model, the system can be more safe shifting to the right the decision threshold and as a consequence minimize the identification errors. On the other hand, for low security the decision threshold must be shifted to the left.

Chapter 7

Conclusions

In this thesis, I have investigated a human identification system based on biometric walking data acquired from a smartphone located in the pocket of the trousers of a person. In particular, the proposed system exploits accelerometer and gyroscope data signals in order to capture the behavioral characteristics, that are unique to each person's gait. After a walking cycle segmentation, a pre-trained Convolutional Neural Network is used as an innovative feature extraction technique. This novel machine learning approach allows to reduce original data complexity by extracting meaningful and general features in a fully-automated way, outperforming commonly hand-crafted statistical features calculated directly on the data signals and classical machine learning approaches. For the final subject identification, only the feature vectors from a target user are considered as input of a one-class classifier based on a support vector machines. Combined with a dimensionality reduction phase, the model achieves excellent performance results also when new walking data are inputted. Moreover, a model can be easily reinforced simply increasing the number of target data used in the training phase, intentionally kept low, in order to classify target walks, which are not performed in a proper manner, while preserving the discrimination ability with respect to the other subjects.

Bibliography

[1] C. Nickel, T. Wirtl, C. Bush. *Authentication of Smartphone Users Based on the Way They Walk Using k-NN Algorithm*, International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2012.

[2] P. Gupta, T. Dallas. *Feature Selection and Activity Recognition System Using a Single Triaxial Accelerometer*, IEEE Transactions on Biomedical Engineering, 2014.

[3] A. S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson. *CNN Features off-the-shelf: an Astounding Baseline for Recognition.* arXiv preprint arXiv:1403.6382, 2014.

[4] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. *Gradient-Based Learning Applied to Document Recognition*, Proc. of the IEEE, November, 1998.

[5] L. Bottou. *Stochastic Gradient Descent Tricks. Neural Networks: Tricks of the Trade.* Springer, 2012.

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell. *Caffe: Convolutional Architecture for Fast Feature Embedding.* arXiv preprint arXiv:1408.5093, 20 June 2014.

[7] M. D. Zeiler, R. Fergus. *Visualizing and Understanding Convolutional Networks*, arXiv preprint arXiv:1311.2901v3, 28 November 2013.

[8] M. D. Zeiler, G. W. Taylor, R. Fergus. *Adaptive Deconvolutional Networks for Mid and High Level Feature Learning.*

[9] V.G.M. Guru, V.N. Kamalesh. *Vision based Human Gait Recognition System: Observations, Pragmatic Conditions and Datasets.* Indian Journal of Science and Technology, Vol 8(15), 71237, July 2015.

[10] J. Wang, M. She, S. Nahavandi, A. Kouzani. *A Review of Vision-Based Gait Recognition Methods for Human Identification.* Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference, pp. 320 - 327. December 2010.

[11] D.M.J. Tax, K. R. Müller. *Feature Extraction for One-class Classification.* ICANN/ICONIP: joint international conference on artificial neural networks and neural information processing. 26-29 June 2003.

[12] C. Nickel. *Accelerometer-based Biometric Gait Recognition for Authentication on Smartphones.* TU Darmstadt, Ph.D. Thesis 2012.

[13] I. Staff. *Technical Evaluation Criteria for the Assessment and Clasification of Biometric Systems*. Technical report draft ver. 0.5-1, Fraunhofer Institute of Graphical Data Processing. August 2000.

[14] N.V. Bougouris, K.N. Plataniotis, E. Micheli-Tzanakou. *Biometrics. Theory, Methods, and Applications.* Wiley, 2010.

[15] M.P. Murray. *Gait as a total pattern of movement*. American Journal of Physical Medicine, vol. 46, pp. 290–332, June 1967.

[16] M. P. Murray, A. B. Drought, and R. C. Kory. *Walking Patterns of Normal Men.* The Journal of Bone and Joint Surgery, 46:335–360, 1964.

[17] J. Perry. *History of the Study of Locomotion.* [Online]. Available: http://www.clinicalgaitanalysis.com/history/modern.html

[18] S.J. Morris. *A shoe-Integrated Sensor System for Wireless Gait Analysis and Real-Time Therapeutic Feedback.* Submitted to the Harvard-MIT Division of Health Science and Technology, May 2004.

[19] P.N. Tan, M. Steinbach, V. Kumar. *Introduction to Data Mining*. Pearson Education, 2013.

[20] J. E. Cutting, L. T. Kozlowski. *Recognizing Friends by their Walk: Gait Perception without Familiarity Cues.* Psychonomic Society, vol. 9(5):pp. 353–6. 1977.

[21] Y. Ren, Y. Chen, M.C. Chuah, J. Yang. *Smartphone Based User Verification Leveraging Gait Recognition for Mobile Healthcare Systems*. Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2013 10th Annual IEEE Communications Society Conference on, June 2013, pp. 149–157.

[22] M.O. Derawi. *Accelerometer-Based Gait Analysis, A Survey*. Norwegian Information Security Conference, November 2010.

[23] S. Jiang, B. Zhang, G. Zou, D. Wei. *The Possibility of Normal Gait Analysis based on a Smart Phone for Healthcare.* IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, 2013.

[24] H. Chan, H. Zheng, H. Wang, R. Sterritt, D. Newell. *Smart Mobile Phone Based Gait Assessment of Patients with Low Back Pain*. Ninth International Conference on Natural Computation (ICNC), 2103.

[25] C.Y. Lee, J.J. Lee. *Estimation of Walking Behavior Using Accelerometers in Gait Rehabilitation*. International Journal of Human-friendly Welfare Robotic Systems, 2002.

[26] A. Mannini, S.S. Intille, M. Rosenberger, A.M. Sabatini, W. Haskell. *Activity recognition using a single accelerometer placed at the wrist or ankle*. Med Sci Sports Exerc, November 2013.

[27] M. Shoaib, H. Scholten, P. J. M. Havinga. *Towards Physical Activity Recognition Using Smartphone Sensors.* 2013 IEEE 10th International Conference on Ubiquitous Intelligence & Computing and 2013 IEEE 10th International Conference on Autonomic & Trusted Computing.

[28] X. Long, B. Yin, R. M. Aarts. *Single-Accelerometer-Based Daily Physical Activity Classification.* 31st Annual International Conference of the IEEE EMBS Minneapolis, Minnesota, USA, September 2-6, 2009.

[29] F. Miao, Y. He, J. Liu, Y. Li, I. Ayoola. *Identifying typical physical activity on smartphone with varying positions and orientations.* BioMedical Engineering OnLine, 2015.

[30] M. Derawi, P. Bours. *Gait and Activity Recognition using Commercial Phones.* Computers & Security Volume 39, Part B, November 2013, Pages 137–144.

[31] J. Mantyjarvi, M. Lindholm, E. Vildjiounaite, S. Makela, and H. Ailisto. *Identifying Users of Portable Devices from Gait Pattern with Accelerometers.* Acoustics, Speech, and Signal Processing, vol. 2, March 2005, pp. 973–976.

[32] M. Derawi, C. Nickel, P. Bours, and C. Busch. *Unobtrusive User-Authentication on Mobile Phones using Biometric Gait Recognition.* Intelligent Information Hiding and Multimedia Signal Processing (IIH- MSP), Oct 2010, pp. 306–311.

[33] S.G. Trost, Y. Zeng, W.K. Wong. *Machine Learning for Activity Recognition: Hip versus Wrist Data.* IOP Publishing, Physiological Measurements 35, p. 2183–2189, October 2014.

[34] M. Ermes, J. Pärkkä, J. Mäntyjärvi, I. Korhonen. *Detection of Daily Activities and Sports With Wearable Sensors in Controlled and Uncontrolled Conditions.* IEEE Transactions on information tech- nology in biomedicine, Vol. 12, No. 1, January 2008.

[35] G.S. Huang, C.C. Wu, J. Lin. *Gait Analysis by Using Tri-Axial Accelerometer of Smart Phones.* Computerized Healthcare (ICCH), 2012 International Conference.

[36] C. Nickel, C. Busch, S. Rangarajan, M. Möbius. *Using Hidden Markov Models for Accelerometer-Based Biometric Gait Recognition.* 2011 IEEE 7th International Colloquium on Signal Processing and its Applications.

[37] B. Schölkopf, J.C. Plattz, J. Shawe-Taylory, A.J. Smolax, R.C. Williamson. *Estimating the Support of a High-Dimensional Distribution.* Neural Computation archive Volume 13 Issue 7, July 2001 Pages 1443 - 1471.

[38] M. Weiser. *The computer for the 21st century.* ACM SIGMOBILE Mobile Computing and Communications Review - Homepage archive Volume 3, July 1999 Pages 3-11.

[39] M. Srivastava, T. Abdelzaher, B. Szymanski. *Human-centric Sensing.* Philosophical Transcation of Royal Society, 370 ser. A (1958), 2012, pp. 176-197.

[40] A. Zaslavsky. Internet of Things and Ubiquitous Sensing. September 2013. [Online]. Available: http://www.computer.org/web/computingnow/archive/september2013

[41] P. Harrop, J. Hayward, R. Das, G. Holland. *Wearable Technology 2015-2025: Technologies, Markets, Forecasts.* IDTechEx, 2015. [Online]. Available: http://www.idtechex.com

[42] G. Stenberg. *Conceptual and perceptual factors in the picture superiority effect.* European Journal of Cognitive Psychology, 2006.

[43] O. Vermesan, P. Friess. *Internet of Things - Converging Technologies for Smart Enviroment and Integrated Ecosystems.* River Publisher, 2013.

[44] J.L. Wayman. *Error Rate Equations for the General Biometric System.* Robotics & Automation Magazine, IEEE, Volume 6, pp. 35 - 48. March 1999.

[45] L.H.N. Lorena, A. Carvalho, A.C. Lorena. *Filter Feature Selection for One-Class Classification.* Journal of Intelligent & Robotic Systems December 2015, Volume 80, Supplement 1, pp 227-243.

[46] S.D. Villalba, P. Cunningham. *An Evaluation of Dimension Reduction Techniques for One-Class Classification.* Artificial Intelligence Review April 2007, Volume 27, Issue 4, pp 273-294.

[47] S.Cateni, M. Vannucci, M. Vannocci, V. Colla. *Variable Selection and Feature Extraction Through Artificial Intelligence Techniques.* InTechOpen, published on: January 2013.

[48] C.M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[49] S.B. Kotsiantis. *Supervised Machine Learning: A Review of Classification Techniques.* Informatica 31, p.249–268, 2007.

[50] L. M. Manevitz, M. Yousef. *One-Class SVMs for Document Classification.* Journal of Machine Learning Research 2 (2001) 139-154.

[51] S. Khan, M.G. Madden. *One-Class Classification: Taxonomy of Study and Review of Techniques.* The Knowledge Engineering Review, pp 1-30, 2014.

[52] G. Cohen, H. Sax, A. Geissbuhler. *Novelty Detection using One-class Parzen Density Estimator. An Application to Surveillance of Nosocomial Infections.* Studies in Health Technology and Informatics. Volume 136, pp. 21 - 26, 2008.

[53] M. Kemmler, E. Rodner, E.S. Wacker, J. Denzler. *One-Class Classification with Gaussian Processes.* Pattern Recognition Volume 46, Issue 12, December 2013, Pages 3507–3518.

[54] D.M.J. Tax, R.P.W. Duin. *Support Vector Data Description.* Machine Learning, 54, 45 - 66, 2004.

[55] D.M.J. Tax. *One-Class Classification: Concept-learning in the Absence of Counter-examples.* Ph.D. thesis, Technische Universiteit Delft, 2001.

[56] I. Ben-Gal. *Outlier Detection.* Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers," Kluwer Academic Publishers, 2005.

[57] K. Hempstalk, E. Frank, I.H. Witten. *One-Class Classification by Combining Density and Class Probability Estimation.* In Proceedings of European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part I (pp. 505-519). Berlin: Springer.

[58] L. Xu, J.SJ. Ren, C. Liu, J. Jia. *Deep Convolutional Neural Network for Image Deconvolution.* Advances in Neural Information Processing Systems (NIPS), 2014.

[59] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun. *OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks.* arXiv preprint arXiv:1312.6229. February, 2014.

[60] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell. *DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition.* arXiv preprint arXiv:1310.1531. October, 2013.

[61] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, D. Yu. *Convolutional Neural Networks for Speech Recognition.* IEEE/ACM Transactions on Audio, Speech, and Language Processing, Vol. 22, No. 10, October 2014.

[62] B. Athiwaratkun, K. Kang. Feature Representation in Convolutional Neural Networks. arXiv preprint arXiv:1507.02313. July, 2013.

[63] S. Haykin. *Neural Networks and Learning Machines.* Prentice Hall, 2009.

[64] A. Krizhevsky, I. Sutskever, and G.E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks.* Vol. 25, pp. 1097–1105. December, 2012.

[65] R. Stufflebeam. *Neurons, Synapses, Action Potentials, and Neurotransmission.* Consortium on Cognitive Science Instruction, 2008.

[66] T. Liu, S. Fang, Y. Zhao, P. Wang, J. Zhang. *Implementation of Training Convolutional Neural Networks.* arXiv preprint arXiv:1506.01195. June, 2015.

[67] Y. Bengio. *Learning Deep Architectures for AI.* Foundations and Trends in Machine Learning: Vol. 2: No. 1, pp 1-127. 2009.

[68] P. Hála. *Spectral Classification using Convolutional Neural Networks.* arXiv preprint arXiv:1412.8341, December, 2014.

[69] T. Wang, D.J. Wu, A. Coates, A.Y. Ng. *End-to-end text recognition with convolutional neural networks.* Pattern Recognition (ICPR), 2012 21st International Conference on Date of Conference: 11-15 Nov. 2012 Page(s): 3304 - 3308.

[70] D. Bouchain. *Character Recognition using Convolutional Neural Networks.* Institute for Neural Information Processing, 2006.