Università degli Studi di Padova

Department of Information Engineering

Master Thesis in Telecommunication Engineering

Deep Recurrent Neural Network based Multi-Modal Gait Anomaly Detection System

Supervisor
Michele Rossi
Università di Padova

Co-supervisor
Riccardo Bonetto
Università di Padova

Master Candidate
Mattia Soldan

4 December 2017

Academic Year 2017/2018

Abstract

Recently, smartphones have become a constant presence in our everyday life. The integration of an increasing number of sensors, together with the interoperability with wearable devices has promoted them as gathering units for data acquisition.This ubiquitous sensing capability has attracted the interest for their use in the Human Activity Recognition (HAR) context.

This thesis is focused on the design and implementation of a gait anomaly detection system able to automatically determine the presence of anomalies in the walking style of a person. A smartphone located in an ad-hoc made chest support is utilized to gather inertial data and video signals from the built-in sensors and rear-facing camera. After a data and video processing phase, a Recurrent Neural Network (RNN), based on Gated Recurrent Units (GRU), is trained to predict the temporal evolution of the multimodal gait information. A gait cycle descriptor is then computed extracting the statistics of the prediction errors obtained during the regression task. The learned descriptors are endowed with remarkable discriminative power and are exploited to assess the presence of anomalies. The involved classification algorithms are Support Vector Machine (SVM), Random Forest (RF), and eXtreme Gradient Boosting (XGB). Final results show that this approach to anomaly detection leads to excellent performance, achieving a classification accuracy as high as 99%.

Sommario

Recentemente, gli smartphone sono diventati una presenza costante nella nostra vita quotidiana. L'integrazione di un numero crescente di sensori, insieme all'interoperabilità con i dispositivi *wearable*, li ha promossi come unità di raccolta dati. Tale capacità di monitoraggio continuo degli utenti ha favorito l'interesse per il loro impiego nel contesto del riconoscimento delle attività umane (HAR).

Questa tesi si concentra sulla progettazione e realizzazione di un sistema di rilevamento automatico di eventuali anomalie nello stile di camminata di un utente. Uno smartphone, posizionato su un supporto toracico, viene utilizzato per il collezionamento di dati inerziali e segnali video, sfruttando i sensori integrati e la fotocamera posteriore del dispositivo. Dopo una fase di *preprocessing* dei dati grezzi, una rete neurale ricorrente (RNN), basata su neuroni *Gated Recurrent Units* (GRU), è allenata a prevedere l'evoluzione temporale dei segnali raccolti. Viene quindi calcolato un descrittore per ogni ciclo di camminata, estraendo le statistiche degli errori di predizione ottenuti durante l'attività di regressione. I descrittori così ottenuti sono dotati di un maggiore potere discriminatorio e vengono quindi sfruttati per valutare la presenza di anomalie. Gli algoritmi di classificazione coinvolti sono *Support Vector Machine* (SVM), *Random Forest* (RF) ed *eXtreme Gradient Boosting* (XGB). I risultati finali mostrano che questo approccio alla rilevazione delle anomalie porta a prestazioni eccellenti, con una precisione di classificazione che raggiunge il 99%.

Contents

Abstract **i**

List of figures **ix**

List of tables **xi**

List of acronyms **xiii**

1 Introduction **1**
- 1.1 Gait analysis formalization 3
- 1.2 Human Activity Recognition 5
- 1.3 Motivations and Contributions 7

2 Related Works **11**

3 Gait Dataset **17**
- 3.1 Data gathering 17
- 3.2 Video information extraction 18
- 3.3 Data preprocessing 20

4 Role of Machine Learning **25**
- 4.1 Notable Algorithms 27
  - 4.1.1 Linear Regression 27
  - 4.1.2 Support Vector Machines 29
  - 4.1.3 Random Forests 32
  - 4.1.4 XGBoost 34
  - 4.1.5 Recurrent Neural Network 35

5 Anomaly Detection System **41**
- 5.1 Data Acquisition 42
- 5.2 Data preprocessing 44
- 5.3 Regression 53
- 5.4 Prediction error statistics extraction 58
- 5.5 Classification 60

6 Final Results **63**

7 Conclusions and Future Work **69**

References **71**

List of figures

1.1 Estimated growth of smartphones number in the United States [1]. From 2010 to 2016 the graph show exact data, from 2017 to 2022 estimations are provided. 2
1.2 Gait phases in a normal gait cycle. 4

3.1 Roll, pitch and yaw angles. 18
3.2 Stride, stance and swing times. 22

4.1 Categorization of Machine Learning problems. 26
4.2 Example of margin in SVM hyperplane separation. 29
4.3 Mapping of non-linear separable training data from $\mathbb{R}^2$ into $\mathbb{R}^3$ . . . 32
4.4 Example of feed-forward network. 35
4.5 Example of recurrent neural network. 37
4.6 Example of unrolled BPTT. 38
4.7 Example of recurrent neural network with Gated Recurrent Units. . 39
4.8 Example of Gated Recurrent Unit. 39

5.1 General scheme of the Gait Anomaly Detection System. 42
5.2 Chest support for smartphone. 43
5.3 Example of recording application home screen. 44
5.4 Comparison of the sampling frequency distribution of the smartphone employed in the data acquisition (Asus Zenfone 2) and another smartphone (LG Nexus 5X). 45
5.5 Power spectral density of the three-axial aceelerometer data. 46
5.6 Frequency response of the Butterworth filter in blue, cutoff frequency in green. 46
5.7 Comparison between raw signal and its filtered version. The considered signal is the yaw angle evolution. 47
5.8 Example of IC (circles) and FC (triangles) detection. 48
5.9 Example of estimation on left or right step. 48
5.10 Example of initial signal transient. 49
5.11 Example of trends in data extracted from video. Green vertical lines separates different acquisition sessions. 50
5.12 Example of video detrended data. Green vertical lines separates different acquisition sessions. 51

5.13 First acquisition sessione, on the left the trend is present, on the right it is removed. Vertical lined separates different gait cycles. . . 52
5.14 Example of underlying data trend, visible after detrending and normalization are performed. 52
5.15 Neural networks structure shown using the tensorboard tool. 55
5.16 Evolution of the MSE score throughout several training epochs. . . 56
5.17 Comparison of regressors performance on two cycles of x-axis of the accelerometer signal. 57
5.18 Comparison of regressors performance on two cycles of z-axis of the accelerometer signal. 58
5.19 Comparison between the distribution of MSE estimation across all cycles. Considered signal is the x-axis of accelerometer. 59
5.20 Comparison between the distribution of standard deviation on prediction error across all cycles. Considered signal is the x-axis of accelerometer. 59
5.21 Visualization of grid search scores for the SVM classification algorithms. 60

List of tables

1.1 Types of human activities studied in literature [2, 3]. 6

5.1 Comparison for mean square of prediction error for different regressors. 57

6.1 Performance comparison of SVM classifier applied to the cycles descriptors obtained using LR and RNN regression algorithms. 64

6.2 Performance comparison of RF classifier applied to the cycles descriptors obtained using LR and RNN regression algorithms 66

6.3 Performance comparison of XGB classifier applied to the cycles descriptors obtained using LR and RNN regression algorithms. 67

List of acronyms

IS	Inertial Sensor
CoM	Center of Mass
BIC	Bayesian Information Criterion
CPU	Central Processing Unit
EM	Expectation-Maximization
GMM	Gaussian Mixture Model
KLT	Kanade, Lucas and Tomasi feature tracker
FC	Final Contact
FIR	Finite Impulse Response
FTP	File Transfer Protocol
HAR	Human Activity Recognition
IC	Initial Contact
IMU	Inertial Measurement Unit
LoG	Laplacian of Gaussian
ML	Machine Learning
OS	Operating System
PSD	Power Spectral Density
CWT	Continuous Wavelet Transform
DoG	Difference of Gaussian
SIFT	Scale-Invariant Feature Transform
SVD	Singular Value Decomposition

PCA	Principal Component Analysis
SVM	Support Vector Machine
ERM	Empirical Error Minimization
RBF	Gaussian-Radial Basis
OOB	Out-of-bag
RF	Random Forest
XGB	XGBoost
LR	Linear Regression
CART	Classification and Regression Trees
AI	Artificial Intelligence
ANN	Artificial Neural Networks
MLP	Multi Layer Perceptron
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
RELU	REctified Linear Unit
BPTT	BackPropagation Through Time
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory Unit
MSE	Mean Square Error
CM	Confusion Matrix
CoD	Coefficient of Determination
IoT	Internet of Things

1 Introduction

In the past decade, smartphones have become the most popular mobile devices. Their growth has made them pervasive in our everyday life. Figure 1.1 shows the estimated increase in the number of smartphones in the United States from 2010 to 2022. This plot shows that the number of smartphones is expected to increase up to 270 million devices in 2022, with an estimated growth of 430% with respect to 2010. Besides these facts, in [4] the authors showed that a mobile phone is considered as essential as keys and wallet when leaving home. This attitude reveals that mobile phones naturally fit people's lives without disturbing them.

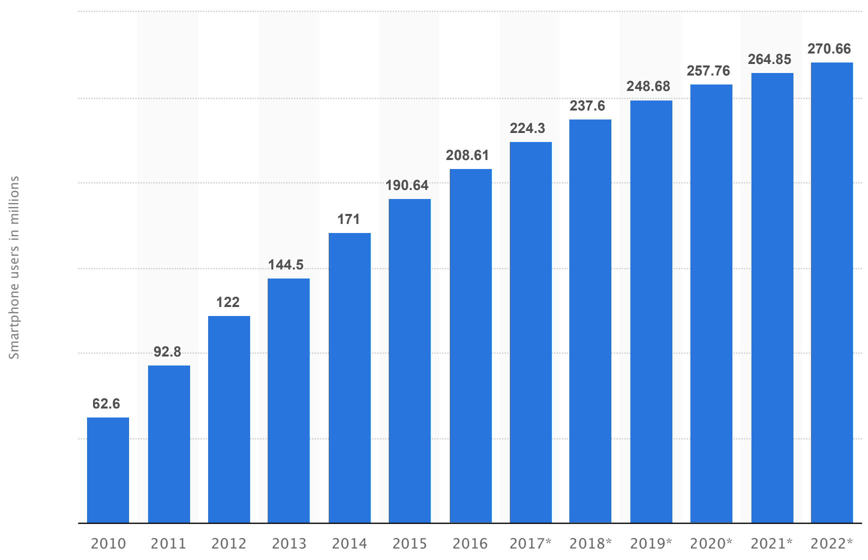


Figure 1.1: Estimated growth of smartphones number in the United States [1]. From 2010 to 2016 the graph show exact data, from 2017 to 2022 estimations are provided.

The primary purpose of these ubiquitous devices is to provide the users with continuously increasing services, ranging from straightforward communications services to more fancy online trading applications. To provide an enhanced user experience, today's smartphones are equipped with several types of sensors that enable all kinds of features.

The accelerometer sensor, for instance, is exploited for tracking the device's movements, which allows to understand the device's orientation, and introduces the automatic landscape orientation change when rotating the smartphone. The gyroscope has been introduced to work in parallel with the accelerometer and increase its precision. Exploiting these sensors, the Android's Photo Sphere camera feature can determine how much a phone has been rotated and in which direction, enabling the possibility to create spheric photos of the user's surrounding. Magnetometer sensors, on the other hand, are capable of sensing magnetic fields, as the earth's one, and are thus involved into compass applications used to point at the planet's magnetic north pole, or in metal detecting applications. In addition to those just mentioned, today's smartphones are equipped with a very rich set of sensors as GPS, Bluetooth, WiFi, microphone, camera, proximity sensor, light sensor, barometer, thermometer, and so forth.

Smartphone devices are not merely communication equipment anymore, but they have become, in all respects, true data gathering machines. The most exciting consequence of this is the possibility to exploit these devices as data processing units to promote them as substitutes for more complex analysis setups. In this context, health monitoring applications are situated, which are a branch of much wider topics: Human Activity Recognition (HAR).
Several techniques could be embedded in a smartphone application for understanding the relationship between physical activity and health conditions. Moreover, thanks to the ever increasing smartphones' computational power, it is possible to develop full fledged frameworks to perform these inferences.

In this thesis, we focus on the possibility of analyzing raw inertial signals extracted from a set of built-in sensors to infer the gait characteristics of a person to automatically detects anomalies.

1.1 Gait analysis formalization

Gait analysis is the study of human locomotion, and it provides useful information in various areas such as healthcare and physiotherapy. Gait study is not a new topic in the research field [5, 6, 7], in fact, several related works are presented in Chapter 2. Nevertheless, continuous advancements in computer vision, mathematics, and computer science have allowed this topic to benefit from continued interest.

Traditionally, multivariate gait data have been challenging to analyze due to the fact that human gait is among the most universal and complex human activities [8]. As shown in [9] and [10], although the global movements of each person are similar, there are specific differences that make each person's manner of walking unique.

To facilitate research and analysis, free gait in humans is usually divided into cycles, each one composed of approximately two human steps. Moreover, each full gait cycle is further segmented into sub-phases. In Figure 1.2, we present the aforementioned phases as defined in [11].

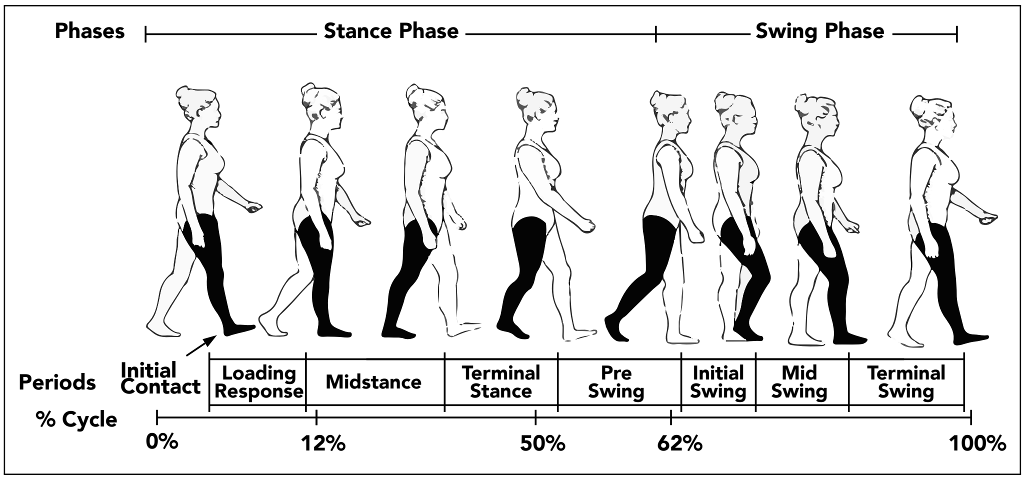


Figure 1.2: Gait phases in a normal gait cycle.

1. **Initial contact**: This is the initial phase of each gait cycle. It comprises the moment when the heel begins to touch the ground but the toes do not touch yet.

2. **Loading response**: This is the initial double-stance period. It begins with the initial floor contact and continues until the other foot is lifted for swing.

3. **Midstance**: In this phase the limb advances over the stationary foot while the knee and hip extend. The toe-off moment is also named the propulsive instant.

4. **Terminal stance**: This phase completes the single-limb support. The stance begins with the heel rising and continues until the other foot strikes the ground. Throughout this phase, body weight moves ahead.

5. **Pre-swing**: This final phase of the stance is the second double-stance interval in the gait cycle. It begins with the initial contact of the opposite limb and ends when the foot lifts off the floor.

6. **Initial swing**: This phase is approximately one-third of the swing period, it begins with a lift of the foot from the floor and it ends when the swinging foot is opposite the stance foot.

7. **Mid-swing**: This phase begins as the swing limb is opposite the stance limb and ends when the swinging limb is forward and the tibia is vertical.

8. **Terminal swing**: This final phase of the swing begins with a vertical tibia and ends when the foot of the first contact strikes the floor.

Due to the complexity of these actions, the gait analysis has traditionally involved laboratories equipped with sophisticated instrumentations and specialized personnel [12]. The main issues related to these setups were the monetary cost of maintenance and the difficulties in data acquisition. In recent years, we assisted to a change in the modus operandi. Wearable devices have been developed to simplify data acquisitions [13]. In particular, low cost and small sensors such as accelerometer, gyroscope, and force sensors were used to perform human gait analysis [14]. Their reduced setup complexity has brought great improvements in data gathering, removing the need for a specialized laboratory.

More recent approaches have managed to exploit smartphones' built-in sensors for this task. The effectiveness of this approach has been shown in [15]. Moreover, smartphones have recently been interfaced with stand-alone monitoring wearable devices. In fact, sensors providing health information like the heartbeat rate or blood pressure are not very useful if embedded on a smartphone, since they require contact with the skin of the user. However, wrist/chest bands and smartwatches that are able to collect this information already exists and can be directly connected via Bluetooth to the smartphone. This integration enables the possibility for several sensors-fusion approaches, and furthermore promoted the involvement of smartphones in gait analysis and HAR in general.

1.2 Human Activity Recognition

The rationale behind HAR is that specific body movements translate into characteristic sensors signal patterns, which can be sensed and classified.

Exploiting the data retrieved from sensors to recognize human activities along with the context in which they are performed is at the core of smart assistive technologies, such as within smart homes [16], rehabilitation [17], health support [18], skill assessment [19] or industrial settings [20].

Human activities can be categorized to ease the approach to their study. In Table 1.1, we summarize the main activity classes, and some of their instances.

Class	Activity types
Locomotion	Walking, running, standing, still, lying.
Mode of transportation	Biking, traveling with a vehicle, riding a bus, driving.
Exercise	Outdoor bicycling, playing soccer, biking on a fitness bike.
Health related activities	Falls, rehabilitation activities, following routines.
Daily activities	Shopping, using a computer, sleeping, going to work, going back home, working, lunch, dinner, breakfast, in a conversation, attending a meeting.
Usage of the phone	Text messaging, making a call, browsing the web, writing an email, using an app.

Table 1.1: Types of human activities studied in literature [2, 3].

HAR has traditionally been tackled in three main manners:

1. vision-based approaches [21, 22];

2. environment interaction sensors strategies [23, 24];

3. wearable sensor-based approaches [25, 26].

In this thesis, we focus on the use of wearable (on-body) sensing, as this allows activity and context recognition regardless of the user's location. In fact, although vision based and environment sensing approaches have achieved high accuracy in recognizing human gestures and activities, they come with several disadvantages. Vision-based systems consist of a set of cameras used to monitor a subject, these setups are often cumbersome and lack of flexibility, sometimes even suffering from illumination changes. Environment sensing systems try to determine human activities by their interactions with smart objects. The underlying assumption is that there exist a direct correlation between objects triggering and actions. This

might be true for actions as ”opening a fridge” or ”sitting on the sofa”. The main issue of this approach lies in the deployment of such smart objects. Although this rationale follows the Internet of Things (IoT) philosophy, it is not yet possible to scale up such systems.
The strength of sensor-based approaches is that the utilized sensing devices are suited to be carried during daily routines, enabling continuous sensing. Data collected from these devices is processed to determine specific patterns corresponding to activities. Some often used approaches are feature extraction or sliding windows followed by classification, template matching [2], or hidden Markov modeling [27, 28]. Sliding window approaches are frequently used for static and periodic activities [29], while sporadic activities are best identified using template matching approaches or hidden Markov modeling [30].
Smartphones can be considered as part of the wearable devices class. They have the advantage to condense several sensing instruments into one single object, avoiding the need for the users to wear additional devices. The involvement of smartphones in HAR tasks have undoubtedly significant advantages, but it is not concerns-free. In fact, enabling continuous sensing on mobile devices should take into account the battery usage. Monitoring application should not heavily affect the user’s experience, and energy-efficient mechanisms need to be adopted [31]. Moreover, several HAR frameworks rely on Machine Learning (ML) algorithms which are often too computationally demanding. Usually the solution to this problem is to perform off-line training on a dedicated remote server. Suitable FTP services are often made available to upload the sensed data.

1.3 Motivations and Contributions

In this thesis, a novel human gait anomaly detection system is proposed. The project aims to completely automate the process of anomaly detection from raw data collection to anomaly assessment. This framework could be used in rehabilitation scenarios [32] to monitor the evolution of patient’s gait during treatments. Moreover, it can be useful in Parkinson’s disease studies to analyze the gait changes along different stages of the disease. Furthermore, this system can be included in a personal health counseling application to warn a person about his possible walking issues. Moving away from health-care, a possible involvement of such system might be in well-being and fitness applications [33, 34, 35], to help athletes improving their habits and performance.

This work departs from the approaches present in the literature for two main reasons:

1. the inclusion of signals extracted from a reference video acquired during a walk;
2. the techniques and algorithms used in the anomaly detection phase.

The goal here, is to develop a personalized application. With this in mind, the data has been collected by a single person throughout several walking sessions. Data gathering has involved both standard and anomalous data, where anomalous data are characterized by the conscious modification of the walking style. All acquisitions were performed with the smartphone located in the chest with a support that blocked it in a fixed vertical position. The considered signals are the tri-axial accelerometer, tri-axial gyroscope and recorded reference video. Computer vision techniques are then used to estimate the optical flow and retrieve the pitch, roll, and yaw angles evolution from the video signal. Raw data is then processed to compute Initial Contact (IC) and Final Contact (FC) instants of every step, and gait cycles are extracted. No high-level gait parameters are computed. Instead, preprocessed raw signals are involved in the anomaly detection system which relies on several ML algorithms.

We exploit a Recurrent Neural Network (RNN) to execute prediction on the raw data and successively evaluate the statistics related to the prediction errors. Differently from other literature approaches, we do not focus on the extraction of high-level patterns or features. On the contrary, we apply supervised classification algorithms to the statistics extracted after the regression tasks have taken place.

This thesis is organized as follows. In Chapter 2, a review of the literature related to the gait analysis topic is presented. In Chapter 3, a detailed characterization of the necessary preprocessing operations applied on raw signals is reported. In Chapter 4, we present a taxonomy on the most relevant supervised machine learning algorithms employed in this thesis. Furthermore, we present a detailed description of each algorithm along with pros and cons. In Chapter 5 the anomaly detection framework is presented. In Chapter 6 the results obtained with the proposed framework are evaluated. Finally, in Chapter 7 conclusions and possible future works are stated.

2 Related Works

In this chapter we review the state of the art on gait analysis.

In the last decade, a lot of works related to the extraction and analysis of gait parameters and anomaly detection have been proposed. The majority is based on the detection of unusual walking patterns with computer vision techniques applied to an input reference video [12]. Other approaches, instead, investigated the processing of raw data collected from Inertial Measurement Unit (IMU) placed in different parts of the body (e.g., ankles, chest, arm or lower-back) [36].
Usually, the computer vision-based approach utilizes stereophotogrammetry, which provides a 3D representation of gait and posture through many cameras. These technologies have proven to be very accurate and reliable, but with the disadvantage of entailing a high monetary costs [12]. Moreover, besides having a non-negligible cost, those systems can only be used in controlled laboratories by expert technicians. On the other hand, the IMU-based approaches have emerged to replace these costly systems. These techniques rely on Inertial Sensor (IS) based systems which are low-cost and are suited to be used in free-living conditions. A large body of works has appeared on the estimation of motion behavior using these sensors [14]. What has emerged is that, while the sensing part of these systems is quite mature, the parameter extraction and processing part can still be improved by employing novel techniques.

Inertial sensors have been used to automatically identify individual gaits from continuous walking traces. For example, in [37] the authors show the accuracy that is reachable with respect to the extraction of IC and FC instants. This, in turn, leads to accurate gait cycle identification. This information is the first step toward the partitioning of the human gait in its sections, namely: initial contact, loading response, midstance, terminal stance, pre-swing, initial swing, mid-swing, and terminal swing [2] briefly presented in the previous chapter.
Based on these advancements, in [38] a machine learning approach to gait-related parameter estimation is proposed. This analysis only produced measures of step length, velocity, swing time and stance time which, however, allowed some initial form of gait classifications. This analysis resulted in simple gait parameters as step length, velocity, swing time and stance time, which consist in simple first or second order measures that allow some initial form of gait classifications.

These first results laid the foundations for different algorithms for accurate extraction of gait parameters from raw accelerometer and gyroscope data. In [39], the authors reviewed five different estimation methods for gait event detection and temporal parameters extraction. For this analysis, they used acceleration signals collected from a single IMU. The results show that all the five algorithms are capable of achieving good performance in the step and stride durations estimation, but the same precision is not met when estimating swing and stance times. This is due to errors in the FC detection. In [39], the authors also showed how the placement of the IMU in the lower trunk does not significantly affect the final accuracy. This empirical proof has directly impacted the thesis work presented in [2] and, in particular, the choice of the on-body smartphone positioning. In [2], moreover, the author implements a gait parameter estimation method similar to the one in [40]. He takes advantage of the continuous wavelet transform to de-noise the raw IMU signals preserving the underlying principal frequencies. When combined with differentiation analysis, this technique shows excellent performance in suppressing noise, correcting baseline drift, and resolving overlapping peaks problems. Moreover, it can detect the IC and FC instants of every gait cycle without errors.

Recently, IMU-equipped smartphones have been widely utilized to collect gait data. This is mainly due to the possibility of using ad-hoc applications. Moreover, a smartphone based system may take advantage of the wireless network connection to send reports or alarms to the person in charge of monitoring the gait signals. For instance, in [41], the authors propose a personal health counseling application that can point out some distinctive features of the user and monitor their weekly or monthly changes. In this study, the smartphone is located in the belt of a user, and the analyzed data comes from a tri-axial accelerometer and a gyroscope. In the data collection phase, to avoid problems related to noise recursion, accelerometer drift, and to remove the effect of the gravity affecting raw data, the authors propose a low-pass filtering and a zero velocity updating technique to assess the drift of the accelerometer with every step. This makes it possible to remove the effects of the drift. Mean step length and mean walking speed are estimated using the upward and downward movement of the trunk according to the inverted pendulum model of the body's Center of Mass (CoM) trajectory.

In [42], a smartphone based system to collect and calculate standard gait parameters is presented. These parameters include step length, velocity, cadence, motion intensity, and walking regularity. Moreover, the prototyped system provides fall and anomaly detection functions. The anomaly detection algorithm is based on a dynamic threshold set according to a normal distribution model. It can determine abnormal gait parameters related to an occasional accident. If the latest data deviates from the normal activity model too much, the data is flagged as an anomalous event. The proposed system shows high accuracy and reliability when determining the number of steps and the walking duration with a global error smaller than 5.45%.

In [43], another anomaly detection system is proposed. It is an accelerometer-based solution based on hand engineered features. This approach uses an unsupervised ML algorithm able to detect abnormal situations with a reasonable detection rate. The anomaly detection procedure is based on a Gaussian Mixture Model (GMM). To define the suitable probability function and determine the optimal number of classes of the GMM [44], the Expectation-Maximization (EM) algorithm [45] is used together with the Bayesian Information Criterion (BIC) score [46]. By using the computed probability density function, an anomalous event is detected whenever the value of the random variable associated with the current instance has a probability below a given threshold.

In [47], the authors introduce a walking model based on normal distributions. The model works by analyzing the signals extracted from a tri-axial accelerometer during walking activities. They tried different placements for the tri-axial accelerometer on the human body. Finally, they found that the sensor on the chest performed best in separating stumbles from normal activities. This supports the smartphone placement considered in this thesis.

In parallel with traditional IS based gait analysis, researchers have recently begun to explore low-cost computer vision technologies. These new techniques transform existing sensing networks into smart visual monitoring systems. The result is an improved framework to assess human behavior for medical diagnosis. One example is the work done by the authors in [48]. They present a computer vision based system able to determine different walking styles and to detect whether a walking action deviates from usual walking patterns or not. In this setup, fixed out-of-body cameras are used. The analysis method starts with the extraction of human silhouettes from the video, and the computation of frame-to-frame optical flows. Then, motion metrics based on histogram representations of silhouettes-masked flow are computed. Finally, gait analysis with eigenspace transformation is performed. This work, based on the analysis of six different walking gaits, suggests good results with an accuracy of about 90%.

Deep learning has also been considered for gait analysis. For example, in [49] the authors use a Multi Layer Perceptron (MLP) network for classifying post-stroke patients gait pattern types.

A more advanced approach is presented in [50]. Here, the authors investigate the problem of people recognition utilizing the gait as a unique identifier. Differently from silhouettes based analysis, they implement a deep learning approach using the optical flow as the primary source of motion information. They exploit a Convolutional Neural Network (CNN) as feature extractor, utilizing the last convolutional layer output as a descriptor which is latter used for classification purposes. State-of-the-art approaches to extract these parameters from inertial sensor data are, however, limited in their clinical applicability due to the underlying assumptions. Namely: stride length, stride time, stance length and stance time. To overcome this, in [51] the authors present a method to translate the information provided by wearable sensors into context-related expert features. This is a paradigm shift for gait analysis, from stride-related extracted data to meta-feature utilization. Current research, in fact, suggests that deep convolutional neural networks are suited to automate feature extraction from raw sensor inputs eliminating the need for conventional parameter extraction [51, 52].

A big step forward has been introduced by the utilization of RNN architectures, which are best suited for time series analysis as the ones related to gait raw data. For instance, a gait feature extractor is implemented in [53] by combining convolutional layers and recurrent Long Short-Term Memory Unit (LSTM) layers. Convolutional layers are used for their ability to extract essential meta-features, while LSTM ones are used to track the temporal evolution of the meta-features.

In [54], the authors present a deep neural network named "DeepConvLSTM". This approach combines convolutional, LSTM, and fully connected layers. This framework allows to perform sensors fusion removing the need for engineered features, this fact makes it suitable for analyzing multimodal wearable sensors signals. It has been tested on two different datasets, the "OPPORTUNITY" [55] and Skoda [56] ones. Their results show that this architecture achieves 4% higher performance than state-of-the-art techniques, on average.

Differently from the existing sensor-based or smartphone-based gait analysis studies proposed in the literature, in this thesis we focus on merging the information coming from the embedded sensors of the chest-worn smartphone together with the signals extracted from the rear phone video camera, recorded while the user is walking. The primary objective is to completely automate the anomaly detection procedure. This is achieved by means of suitable machine learning algorithms. The whole process involves the training of an RNN on standard gait data. For this purpose, an overlapping window strategy is utilized for signals segmentation before the training phase. The RNN is successively employed to predict the temporal evolution of previously unseen data. Prediction error statistics are then computed and fed to several classification algorithms to detect possible anomalies.

3 Gait Dataset

3.1 Data gathering

In this chapter, we revise the principal tools used for data preprocessing. The obtained results will be presented in Chapter 5.

Data collection and video recording for smartphone analysis must be automated by means of a specific application. After data gathering, raw data need to be preprocessed to address some specific problems as long as good practice operations. These steps are fundamental for the subsequent use in machine learning algorithms. These operations are presented and detailed in this chapter.

One of the common problems of inertial data collected from the built-in sensors of a smartphone is that the sampling rate can vary during an acquisition.
The sampling rate, in fact, is related to the computational load of the operating system. Therefore, it can have a distribution with standard deviation greater than zero. This problem leads to the presence of temporal misalignment between the samples of different sensors. Hence, an interpolation phase of all gathered signals is necessary.

Before the signals are preprocessed, the video information must be extracted from the recorded video file.

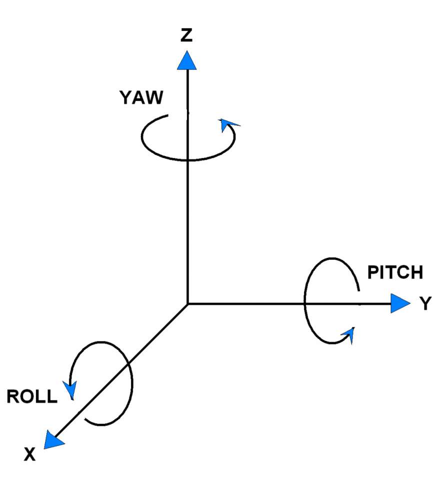


Figure 3.1: Roll, pitch and yaw angles.

3.2 Video information extraction

The video recorded during the acquisition phase is elaborated through *Computer Vision* techniques to extract meaningful information regarding the device orientation in the 3D space.
Video information extraction aims to estimate the optical flow and retrieve the triaxial temporal evolution regarding the smartphone orientation in the space along the whole video acquisition.

Formalization If we consider camera-centered coordinates, each point on a tridimensional surface moves along a tridimensional path. The projection of that points onto an image plane, instead, produces a bi-dimensional path. The velocity of all the points can be computed estimating their instantaneous directions. The bi-dimensional velocities for all the visible surface points are often referred to as the bi-dimensional motion field. The goal of the optical flow estimation is to compute an approximation of the motion field from the time-varying image intensity [2]. The desired output is a tridimensional signal representing the angular evolution of the smartphone, thus the roll, pitch, and yaw angles. These angles are shown in Figure 3.1

Lucas and Kanade In this thesis, Lukas and Kanade optical flow estimation method is exploited. This method is based on the assumption that the optical flow $\mathbf{v}$ is constant in an $n \times n$ neighborhood W of every point. This algorithm solves the basic optical flow estimation for all the pixels in that neighborhood, by the least squares criterion.

By combining information from several nearby pixels, the Lucas–Kanade method can often resolve the possibly present ambiguity of the optical flow equation. It is also less sensitive to image noise than point-wise methods. On the other hand, since it is a purely local method, it cannot provide flow information in the interior of uniform regions of the image. The algorithms, in fact, works considering several key points which must be identified and tracked across different frames. These key points are usually corners and edges, not uniform regions.

Kanade, Lucas and Tomasi Feature Tracker Starting from the Lucas, and Kanade algorithm, Kanade, Lucas and Tomasi feature tracker (KLT) is based on two main steps:

- Frame feature extraction.
- Feature tracking.

Frame feature extraction is performed using the Scale-Invariant Feature Transform (SIFT) technique [57]. The key-point extraction is performed for each frame of the video under analysis. Then the feature tracking phase is performed between every couple of consecutive frames. More in details, the algorithm works as described in Algorithm 3.1.

After the tracking has taken place, the Essential matrix $\mathbf{E}$ is computed. The Essential matrix is a unique 3×3 matrix which captures the geometric relationship between two calibrated cameras or between two locations of a single moving camera. It is then suited to analyze the video of the gait session.
Moreover, the RANSAC algorithm is employed to remove outliers value which are inevitably present due to a non-perfect estimation. The Essential matrix allows estimating two more useful matrices, the rotation $\mathbf{R}$ and translation $\mathbf{t}$ matrices related to the smartphone movements. In the end, it is possible to retrieve the

Algorithm 3.1 KLT Tracker

```
Input: Sequence of frames.
Output: Optical flow estimation.

 1. spatial filtering with Gaussian 2D kernel;
 2. temporal filtering with 1D Gaussian kernel;
 3. first frame keypoints extraction;
 4. for every frame couple $I(t)$, $I(t+1)$:
      • tracking: for every keypoint in $I(t)$ compute $\mathbf{v}$ on a $n \times n$ window
        centered in the keypoint;
      • apply motion to every point and obtain the position in the successive
        frame $I(t+1)$.
```

three angular signals pitch, roll, and yaw from the rotation matrix. A detailed analysis of the Essential matrix and its factorization can be found in [2].

3.3 Data preprocessing

Raw data collected from the accelerometer and gyroscope sensors, together with the signals extracted from the reference video, need to be processed to remove high-frequency noise and to extract gait cycles. Moreover, the raw signals must be interpolated and resampled to synchronize them and pad each extracted cycle to the same length. The padding procedure is performed exploiting a resampling operation, performed after the cycles are identified. The extracted gait cycles are then utilized to form the dataset fed to the machine learning techniques. Those algorithms are detailed in the next chapter.

Data processing includes five main phases: **interpolation, filtering, cycles extraction, signals de-trending, and normalization.**

Interpolation

As discussed above, since the sampling rate is not constant during the acquisition due to the non-real-time nature of the Android Operative System, an interpolation step is needed. Moreover, smartphones output samples whenever there is a change in the sensor and, therefore, the time intervals between two samples are not evenly spaced and differ for each sensor. After this operation, every signal samples result being evenly spaced in time.

Filtering

Noise is a constant presence in signals measured from real quantities evolution, as in this case. In this particular setup, where the smartphone is located in an ad-hoc chest made support, the support itself might introduce noise. This effect is present due to spurious vibrations that are not descriptive of the walking style but are induced by an improper wearing of the support. It is then necessary to filter such signals to remove the noise which is typically related to high-frequency components [58]. The solution to this problem is often provided by the application of a low pass filter.

A specific cutoff value must be specified for the filter. This threshold value can be chosen by inspecting the Power Spectral Density (PSD) of the signals at hand. PSD estimation can be computed through several methods, one of the most used is the Welch's one [59].

Cycles Extraction

After the filter procedure has been computed, it is possible to execute the segmentation of the raw signals into gait cycles. As discussed in [2], human gait follows a cyclic behavior where there is a periodic repetition of a pattern, known as a cycle, that corresponds approximately to two human steps. It is possible then to identify the cycles and singularly analyze them.

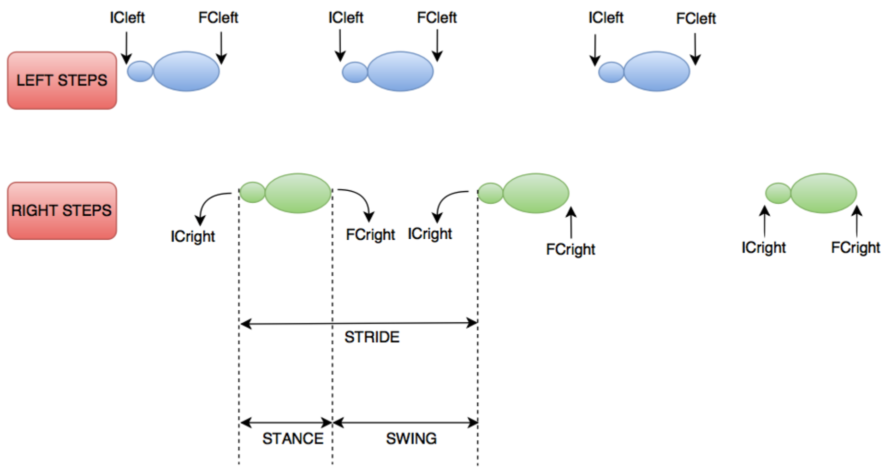


Figure 3.2: Stride, stance and swing times.

To extract the gait cycles, different techniques are present in the literature as recalled in Chapter 2. The one exploited in this thesis is the one implemented in [2] and based on [60]. The IC and the FC events within each cycle are estimated by analyzing the vertical component $\mathbf{a}_y$ (y-axis) of the accelerometer data.

In particular, IC and FC events are determined using a Continuous Wavelet Transform (CWT) of the signal $\mathbf{a}_y$. The IC events are detected as the local minima of the transformed signal. Instead, the FCs events are detected as local maxima of the differentiation of the CWT.

Signals detrending

In data analysis, it is common to encounter data presenting trends. For example, in marketing data, this particular problem is often encountered [61, 62]. A formal way to refer to a time series presenting trends is as non-stationary.
Trends can be defined as a continued increase or decrease in the series over time. To analyze the data, there can be benefits in identifying, modeling, and even removing trend information from time series to expose underlying information.
It is possible to identify two general classes of trends. Namely:

- **Deterministic Trends**: These are trends that consistently increase or decrease.

- **Stochastic Trends**: These are trends that increase and decrease inconsistently.

In general, deterministic trends are easier to identify and remove, but stochastic ones can still be tackled with the same techniques utilized for the previous class. It is also useful to identify trends in terms of their scope of observations.

- **Global Trends**: These are trends that apply to the whole time series.
- **Local Trends**: These are trends that apply to parts or subsequences of a time series.

To detrend a time series, the trend has to be identified and visualized. Visualization helps in determining to which general class it belongs (i.e., deterministic or stochastic). Once the trend has been identified, it can be eliminated. This allows to obtain a stationary time series.

Normalization

In machine learning and statistical learning, it is good practice to normalize the data as investigated in [63]. In Neural Networks applications, input data normalization with certain criteria, prior to the training process, has been found to be crucial to obtain good results as well as to speed up the calculations significantly. Moreover, in several clustering algorithms, data normalization has been determined as a fundamental step to boost the performance [64, 65]. Indeed, data normalization intervene also in regression, avoiding a subset of features to prevail based on their ranges and not on the true data meaning [66]. This summarizes the various motivations behind data normalization which should always be computed.

4

Role of Machine Learning

Over the past two decades, Machine Learning technology has continuously grown, becoming one of the most attractive fields in ICT. The constant increasing data availability, and the extraordinary performance of particular algorithms induce to believe that smart data analysis will become even more pervasive as a necessary ingredient for technological progress [67].

In this chapter, we briefly introduce machine learning basic information regarding its branches. Successively, the algorithms involved in this thesis are revised to present their working principles and their pros and cons.

The concept behind machine learning can be stated as follows. Given a training set S, sampled from an unknown distribution D and labeled by some target function f. Given a model set H containing a set of functions.
The target of learning algorithms is to determine the function $H_S : X \to Y$ that minimizes the error with respect to the unknown D and f and an error function. The subscript S emphasizes the fact that the output predictor depends on S.

Machine learning algorithms can be identified in four different branches depending on the presence or absence of target values and the type of desired output.

	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Figure 4.1: Categorization of Machine Learning problems.

The goal of Supervised Learning is to obtain predictions of an outcome measure based on a number of inputs also said features. For this branch of machine learning the target value is known. Differently, in unsupervised learning, there is no outcome value, and the goal is to describe the associations and patterns among the samples of the dataset.

For the purpose of this work we focus on the two main supervised learning tasks, namely, regression and classification.

- **Regression:** predictions are based on quantitative values. The output domain is $\mathbb{R}^n$;

- **Classification:** predictions are based on qualitative values. The output domain is categorical. Qualitative variables usually assume values from a finite and discrete set with no explicit ordering.

In supervised learning, the mathematical tool for predicting y_i given input x_i is usually referred as a model. For instance, with a linear model, the prediction is given by $y_i = \sum_j \mathbf{w}_j x_{ij}$, or rather a linear combination of weighted input features. The parameters of the model are the undetermined part that we need to learn from the data. Usually we will use $\mathbf{w}$ to denote the parameters.

4.1 Notable Algorithms

Here, we briefly introduce the learning algorithms that are considered throughout this thesis.

4.1.1 Linear Regression

Linear Regression (LR) was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. The reason is to be found in the fact that linear models are easier to understand than non linear ones. Moreover, the statistical properties of the resulting estimators are simpler to determine [68].
Linear regression simplicity is the key to easily tackle a significant number of problems quickly. It is also recommended as a reference model to assess the baseline performance on a problem's solution. It is then possible to determine wheter complex models achieve better results or not. In particular, this is the reason why linear regression has been considered in this thesis.

Linear regression models the relationship between some "explanatory" variables and some real-valued outcome using a set of linear functions. Cast as a learning problem, the domain set X is a subset of $\mathbb{R}^d$, for some d, and the labels set Y is the set of real numbers $\mathbb{R}$. The purpose of linear regression is to learn a **linear function** $h : \mathbb{R}^d \rightarrow \mathbb{R}$ that best approximates the relationship between a set of features and an output value [69].
A linear predictor is a function of the unknown model parameters ($\mathbf{w} \in \mathbb{R}^{d+1}$) which are estimated from the data. The linear function can also be expressed as

$$\mathrm{y} = f(X) = \mathbf{w_0} + \sum_{j=1}^{d} \mathbf{x}_j \mathbf{w}_j, \tag{4.1}$$

where $\mathbf{w}_j$ with $j = 0, \ldots, d$ are the model parameters.
The learning paradigm requires to determine $\hat{f}(X)$ to predict y given X. It is then necessary to penalize the errors made during the prediction phase. This is done introducing a Loss Function.

Several loss functions are available, the most common and widely used are listed below:

- Squared loss function: $L(h,(x,y)) = (h(x),y)^2$, commonly used for regression;
- Absolute loss function: $L(h,(x,y)) = |(h(x),y)|$;
- Binary loss function: $L(h,(x,y)) = [\![(h(x),y)]\!]$, which is the indication function for which $[\![A]\!] = 1$ if A is true, zero otherwise. This loss function is commonly used for classification purposes, outputting one if and only if the predictions is wrong.

The most used loss function is the Mean Square Error (MSE). The corresponding minimization problem is:

$$\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} (\langle \mathbf{w}, \mathbf{x_i} \rangle - \mathbf{y_i})^2, \tag{4.2}$$

where m is the cardinality of X, and $\langle \cdot, \cdot \rangle$ represents the inner product operation. To solve problem 4.2, it is necessary to compute the gradient with respect to $\mathbf{w}$, and compare it to zero.

$$\frac{2}{m} \sum_{i=1}^{m} (\langle \mathbf{w}, \mathbf{x_i} \rangle - \mathbf{y_i}) \cdot \mathbf{x_i} = 0\,, \tag{4.3}$$

and by letting:

$$\mathbf{A} = \left(\sum_{i=1}^{m} \mathbf{x_i} \mathbf{x_i}^\top \right) \qquad \textit{and} \qquad \mathbf{b} = \sum_{i=1}^{m} \mathbf{y_i} \mathbf{x_i} \tag{4.4}$$

it is possible to determine the optimal solution $\mathbf{w}^*$:

$$\mathbf{w}^* = \mathbf{A}^{-1} \mathbf{b} \tag{4.5}$$

The main limitation of the linear regression is the incapacity to approximate non-linear dependencies between the features and the outputs. For this reason, more sophisticated regressors have been introduced.
When the output is multidimensional, linear regression is also referred as multivariate linear regression [70].

4.1.2 Support Vector Machines

A Support Vector Machines (SVMs) is a supervised learning algorithm serving both classification and regression purposes. However, since SVMs are more commonly used in classification problems, this is what we will focus on in this section.

SVMs are based on the idea of finding a hyperplane that best divides a dataset into two classes, as shown in Figure 4.2.

Support Vector Machine (SVM) are based on the concept of support vectors. These are the vectors connecting the origin to the data points that are closer to the dividing hyperplane.

The SVM paradigm tackles the sample separation challenge by searching for "large margin" separators. Roughly speaking, a halfspace separates a training set with a large margin if all the examples are not only on the appropriate side of the separating hyperplane but also far away from it.

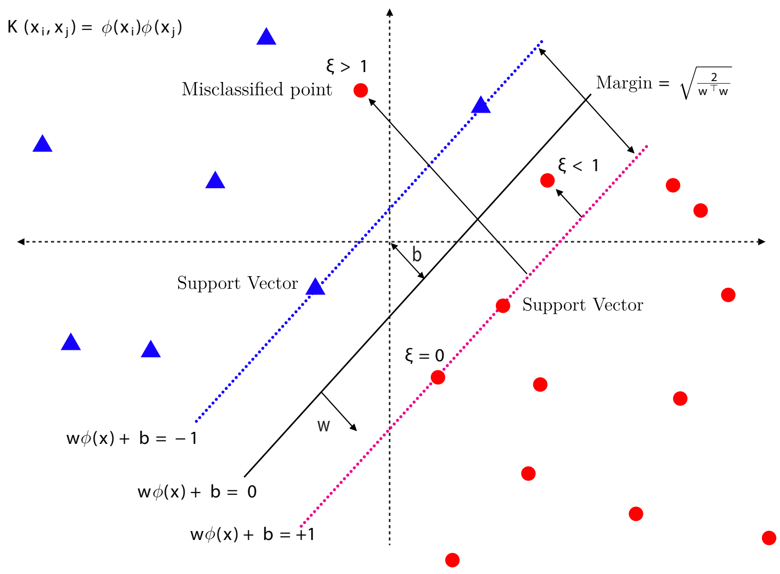


Figure 4.2: Example of margin in SVM hyperplane separation.

More formally, a SVM constructs a hyperplane in a high or infinite-dimensional space. The separating hyperplane is defined as $H = \{\mathbf{x} \,|\langle \mathbf{w}, \mathbf{x}\rangle + \mathbf{b} = 0\}$, where $\mathbf{w}$ is a normal to the hyperplane as shown in Figure 4.2. The distance between the plane to the origin is $|\mathbf{b}|$, and $||\mathbf{w}||$ is the Euclidean norm of $\mathbf{w}$. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin). This is due to the fact that the larger the margin, the lower the generalization error of the classifier. This happens because the functional margin reduces the sensitivity of the classification to the noise that may affect the data. This maximum margin is sought during the training phase. The margin is limited by the two hyperplanes $H_1 = \{\mathbf{x} \,|\, \langle \mathbf{w}, \mathbf{x} \rangle + \mathbf{b} = 1\}$ and $H_2 = \{\mathbf{x} \,|\, \langle \mathbf{w}, \mathbf{x} \rangle + \mathbf{b} = -1\}$, hence, it is of size two. Moreover, these hyperplanes are the ones where the support vector points lie. The output values of this classification are $\mathrm{y}_i \in \{-1, 1\}$ which specify the class corresponding to training vector $\mathbf{x}_i$, $i \in \{1, ..., m\}$, where m is the cardinality of the dataset X.

It is possible to formalize two different cases of separations:

- **Hard-SVM**: In this case, the algorithm seeks the best margin that separates the classes, with the constraint that all the points are correctly classified. This is the case presented so far. It is implicitly assumed that the training set is linearly separable, which in real life data is a rather strong assumption. The hard SVM optimization problem may be expressed as follows [69]:

HARD-SVM

input: $(\mathbf{x}_1, \mathrm{y}_1), \ldots, (\mathbf{x}_m, \mathrm{y}_m)$
solve:

$$(\mathbf{w}_0, b_0) = \underset{(\mathbf{w},b)}{\operatorname{argmin}} \|\mathbf{w}\|^2 \;\; s.t. \;\; \forall i, \;\; \mathrm{y}_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \tag{4.6}$$

output: $\hat{\mathbf{w}} = \frac{\mathbf{w}_0}{\|\mathbf{w}_0\|}$, $\hat{b} = \frac{b_0}{\|\mathbf{w}_0\|}$

- **Soft-SVM**: In this second case, the algorithm allows the presence of misclassified points to which it assigns a weight proportional to the distance ξ from the hyperplane related to the correct classification region, as shown in Figure 4.2. In this case, the weights ξ are considered part of the error minimization procedure that produces the solution. We can see from the soft margin optimization problem definition, how the weights ξ have the role of additional terms within the function to minimize. On the other hand, they also are involved in the constraint definition.

SOFT-SVM

input: $(\mathbf{x}_1, \mathrm{y}_1), \ldots, (\mathbf{x}_m, \mathrm{y}_m)$
parameter: $\lambda > 0$
solve:

$$\min_{(\mathbf{w},b,\xi)} \left(\lambda \|\mathbf{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i \right) \qquad (4.7)$$
$$s.t.\ \forall i,\ \ \mathrm{y}_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \ \ and\ \ \xi_i \geq 0$$

output: $\mathbf{w}$, b

Kernel trick

Whereas the original problem may be stated in a low dimensional space, it often happens that the data are not linearly separable. For this reason, it was proposed to map the original problem into a much higher dimensional one.

Kernel methods are a class of algorithms which owe their name to the use of kernel functions $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})\phi(\mathbf{x}')$, where $\phi(\mathbf{x})$ represents the feature mapping transformation into the higher dimensional space. The most known algorithm of this class is the SVM one.
Kernel functions are designed to enable the mapping of the data without the need of computing the coordinates in the higher dimensional space. In fact, a kernel function applies the transformation by solely computing the inner products between all pairs of data, obtaining a transformation in a much more computationally efficient manner.

This approach is called the kernel trick [71]: by using it any linear model can be turned into a non-linear one replacing its features (predictors) by a kernel function. The principal kernel functions are:

- Linear kernel: $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}$;
- Sigmoid kernel: $K(\mathbf{x}, \mathbf{x}') = (\gamma \mathbf{x}^\top \mathbf{x} + \zeta)^d$ for $d, \gamma, \zeta > 0$;

- Degree-Q polynomial kernel: $K(\mathbf{x}, \mathbf{x}') = (\gamma \mathbf{x}^\top \mathbf{x} + \zeta)$;
- Gaussian-Radial Basis (RBF) kernel: $K(\mathbf{x}, \mathbf{x}') = e^{(-\gamma \|\mathbf{x}-\mathbf{x}'\|^2)}$ for $\gamma > 0$.

where γ, ζ, d are suitable constants.

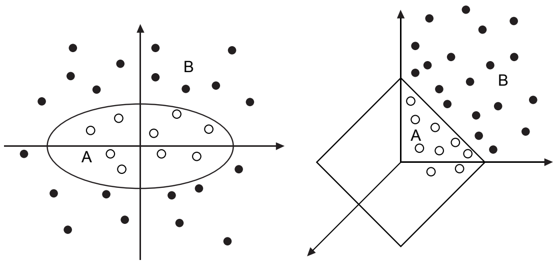


Figure 4.3: Mapping of non-linear separable training data from $\mathbb{R}^2$ into $\mathbb{R}^3$

SVMs have the advantage of not making any strong assumption on the data. Moreover, the likelihood of overfitting is reduced with respect to other classifiers. On the other hand, SVMs are not well suited neither for large datasets, due to the high training time, nor for noisy datasets with very overlapping classes.

4.1.3 Random Forests

Random Forests (RFs) or random decision forests [72] are an ensemble and voting based machine learning algorithms for classification, regression and other tasks. RFs construct a multitude of decision trees [73] which are all used to compute a prediction. The final prediction is the mode of the classes (classification) or the mean (regression) of the individual trees predictions. Moreover, RFs resolve the decision trees's habit of overfitting to their training set [69]. Ensemble methods aim at improving the classification accuracy by aggregating predictions from multiple classifiers (bagging). The more diverse and the less correlated the base classifiers, the more accurate is the ensemble result.

Definition: *A random forest is a classifier consisting of a collection of tree-structured classifiers* $\{h(\mathbf{x}, \boldsymbol{\theta}_k), k = 1, 2, \dots\}$ *where the* $\{\boldsymbol{\theta}_k\}$ *are independent identically distributed random vectors, and each tree casts a unit vote for the most popular class at input* $\mathbf{x}$.

The generation of each tree in the forest is performed according to the following steps which introduce in the forest two sources of randomization:

1. **Boostrap sampling:** If the cardinality of the training set is N, then N different records are sampled at random, but with replacement, from the original data. This sample will be the training set for growing the tree;

2. **Feature sampling:** If M is the number of input variables or features, at each node, $m \ll M$ variables are chosen at random out of M. These features are then used to determine the best split of the node. The value of m is held constant during forest growth and each tree is grown to the largest extent possible.

The number of trees, the number of variables (m) selected at each node and the depth of each tree are hyper-parameters that have to be tuned.

Once the forest is trained, the classification procedure on new data is obtained by means of a voting system.

State of the Art RF [74] are characterized by:

- The use of Out-of-bag (OOB) error as an estimate of the generalization error. In the forest building process, when the bootstrap sample set is drawn, about $1/3^{rd}$ of original data instances are left out. This set of instances, different for each tree, is called OBB data and is used for error estimation on each tree;

- The usage of proximities score in replacing missing values and outliers. The proximity score is computed for each pair of data samples. This metric is extracted after each tree is built. In particular, proximities are calculated in this way: if two samples are run down the tree and they end up occupying the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees.

The RF algorithm is characterized by several excellent properties such as their robustness to noise and computational speed. RFs do not require any input preparation, and are capable of handling numerical, binary and categorical features. At the same time they can also handle outliers and missing values. Their capacity to achieve high classification accuracy without the risk of overfitting have made them quite notorious.

Despite all these good qualities, some critical issues remain. In fact, their theoretical analysis is quite difficult. In some cases, the algorithm can get biased and the variable importance scores (Z-score) do not seem to be reliable. As last, when using RF for regression, it does not predict beyond the range of the response values in the training data, thus reducing its capacity to generalize.

4.1.4 XGBoost

XGBoost (XGB) [75] implements the gradient boosting decision tree algorithm which was presented in [76]. Gradient boosting takes its name from the use of the gradient descent algorithm to minimize the loss when adding new models to the ensemble during the training phase.
As for RF, XGB is based on decision trees. The main differences between these two algorithms consist in the ensembling method and in the way they make decisions.

RF uses parallel ensembling, meaning that after data and feature sampling, all the trees are built simultaneously. The final model is the parallel ensemble of all the trees.

Differently, XGB exploits sequential ensembling, in this case, the trees are sequentially added to the model, and each of them rely on the performances of the model at the previous step. XGB starts with a shallow tree and uses it to model the original target. Then it takes the error from the first round of predictions and uses it as a new target for a second tree. This procedure is then repeated until a stopping condition is met. XGB essentially focuses on modeling prediction errors from previous trees.

The way the model is created also reflects the way predictions are made: RF implements a simple majority vote for classification, while XGB gets the advantage of a weighted majority vote. This usually tends to provide better performances.

During the training phase, RF works by overfitting subsamples of the training data and then reduces the overfit averaging the predictor's output. XGB successively trains trees on the residuals error of the previous models. Due to this procedure RF results to have low bias and high variance, on the contrary, XGB results to be a low variance and high bias model.

In the literature, XGB is one of the best algorithms available today, and it is almost always outperforming RF [77].

4.1.5 Recurrent Neural Network

RNNs extend the concept of Feedforward Neural Networks (NNs) by adding loops connections to the standard input-output flow. NNs feed information straight through, never touching a given node twice, hence they are usually represented by means of Direct Acyclic Graphs (DAGs). RNNs instead, cycle data through loops.

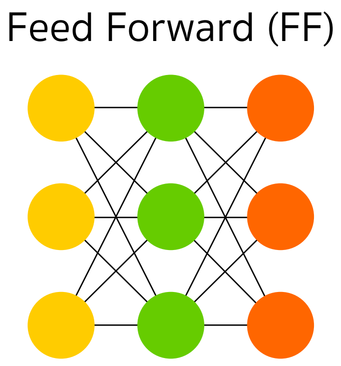


Figure 4.4: Example of feed-forward network.

We can now introduce some formalism: the first and last layers (yellow and red in Figure 4.4) are called respectively *Input Layer* and *Output Layer*. Each layer in between (green) is referred to as a *Hidden Layer*.
Nodes in the Input Layer are referred to as X_i with $i = 1, \ldots, p$, where p is the input dimension.
Nodes in the Output Layer are referred as Y_k with $k = 1, \ldots, K$. Depending on the task being performed (i.e. classification or regression) K defines the number of possible classes or the number of predicted features
The nomenclature for a Hidden Layers node is $Z_{d_i}^{(l)}$, where l identifies the hidden layer and d_i identifies the specific node in that layer.

Using graph theory nomenclature, we can name each connection between the nodes as an arc. Each arc is associated with a different weight $\mathrm{w}_{i,j}^{(l)}$ where i, j, l are the indices of the nodes connected by the arc and the specific layer we are considering, respectively. In particular, using the notation $\mathbf{w}_{i,j}^{(l)}$ we refer to the weight of the arc that connects the node i of layer $(l-1)$ to the node j of layer (l). The output of node j in layer l is:

$$Z_j^{(l)} = \phi \left(\mathbf{w}_{0,j}^{(l)} + \sum_{i=1}^{d^{(l-1)}} \mathbf{w}_{i,j}^{(l)} \, Z_i^{(l-1)} \right) , \tag{4.8}$$

where ϕ is called *Activation Function*, and it is applied to each node and introduces a transformation on the input of the node. Notable activation functions are:

- Linear: $\phi(x) = x$;
- Sigmoid: $\phi(x) = \dfrac{1}{1+e^{-x}}$;
- Hyperbolic Tangent (Tanh): $\phi(x) = \dfrac{2}{1+e^{-2x}} - 1$;
- REctified Linear Unit (RELU): $\phi(x) = \begin{cases} 0 & for\, x \leq 0 \\ x & for\, x \geq 0 \end{cases}$

The choice of the number of hidden layers (L) as long as the number of nodes per layer $[d^{(1)}, d^{(2)}, \dots, d^{(L-1)}]$ and the activation function Θ defines the network model set H.
The training procedure of the network modifies the weights ($\mathbf{w}$) to minimize a specific loss function. This procedure is performed through the application of the *Backpropagation Algorithm* [78].

The main difference between a feedforward network and an RNN is that the former has no notion of order in time, and the only input it considers is the current example it has been exposed to. RNNs, on the other hand, take as input, not only the current example they see, but also their past states, as shown in Figure 4.5.

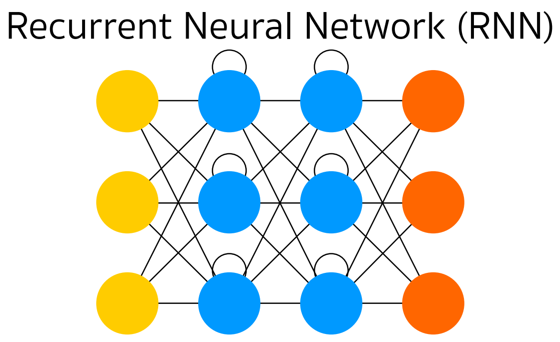


Figure 4.5: Example of recurrent neural network.

RNNs have two sources of input, the present and the recent past, which are combined to determine how to respond to new data. It is often said that recurrent networks have memory. Adding memory to neural networks has a purpose: there is information in the sequence itself, and RNNs manages to find peculiar patterns in it. This is the main reason why RNN are best suited to time series analysis.
The sequential information is preserved in the RNN's hidden states. The hidden states affect the processing of new samples according to a cascade effect. This cascade allows the RNNs to determine correlations between noncontiguous events in the sequence. These correlations are called "long-term dependencies", meaning that an event depends upon, and is a function of, one or more events that came before.
For RNNs architectures the nomenclature is usually different from the one of feedforward NNs. In fact, the nodes output are no longer referred as $\mathbf{Z}$. On the contrary, the nomenclature $\mathbf{h}$ is used. It represents both the current output of the cell and its hidden state or memory, it can be expressed as:

$$\mathbf{h}_t = \Theta(W\mathbf{x}_t + U\mathbf{h}_{t-1}). \tag{4.9}$$

The hidden state at time step t is $\mathbf{h}_t$. It is a function of the input at the same time step $\mathbf{x}_t$, modified by a weight matrix W. The hidden state also depends on its previous value, $\mathbf{h}_{t-1}$, multiplied by the hidden-state-to-hidden-state matrix U. Matrix U is also known as a transition matrix.

The weight matrices are filters that determine how much importance to accord to both the present input and the past hidden state. The error they generate will return via backpropagation and be used to adjust their weights until the error cannot go any lower.

RNNs rely on an extension of the previously cited backpropagation, called BackPropagation Through Time (BPTT). Time, in this case, is simply expressed by a well-defined, ordered series of calculations linking one-time step to the next, which is all backpropagation needs to work. In practice Truncated BPTT is used, it is an approximation of full BPTT that is preferred for long sequences since full BPTT's forward/backward cost per parameter update becomes very high over many time steps. The downside is that the gradient can only flow back so far due to that truncation, so the network cannot learn dependencies that are as long as in full BPTT. In Figure 4.6 is shown a compact version of an RNN cell and its "unrolled" version, which is the one to consider when applying the BPTT algorithm.

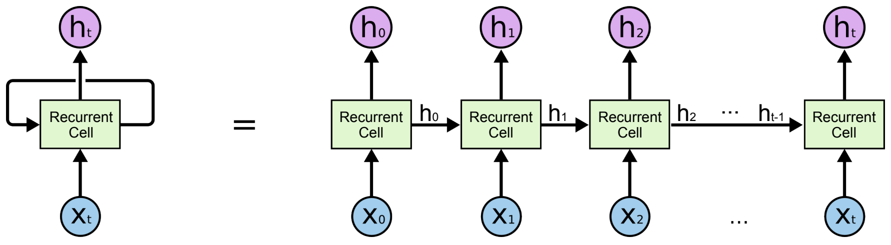


Figure 4.6: Example of unrolled BPTT.

Analyzing Figure 4.6, we can track the dependency of an output state $\mathbf{h}_t$ at time t by all its predecessors, in time. Moreover, it is clear how the output of a cell is wired both toward other cells and toward itself.

GRU cell

A Gated Recurrent Unit (GRU) was proposed in [79] to make each recurrent unit to adaptively capture dependencies of different time scales.

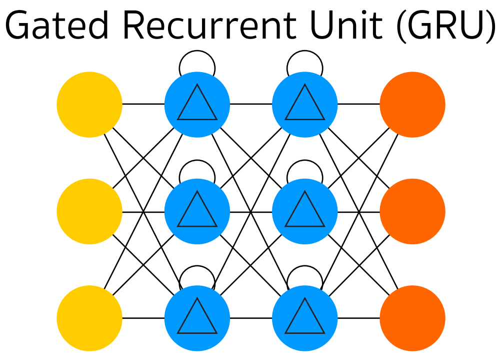


Figure 4.7: Example of recurrent neural network with Gated Recurrent Units.

As previously introduced RNNs cell, GRU units maintains a memory in the form of internal state $\mathbf{h}$. This cell, moreover, implements a gating system that modulates the flow of information inside the unit to handle the interaction between state and current sample. The gates are two, namely: update ($\mathbf{z}_t$) and reset ($\mathbf{r}_t$).

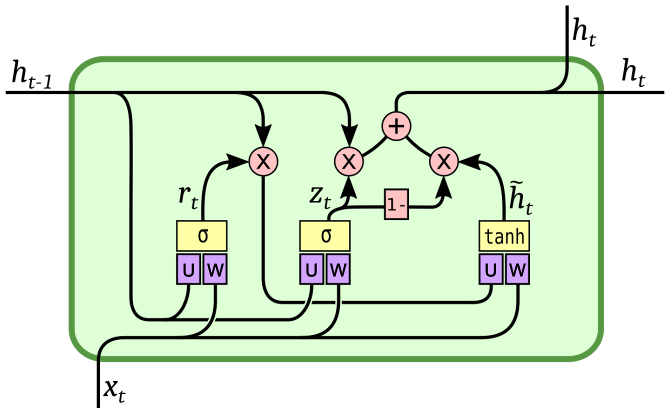


Figure 4.8: Example of Gated Recurrent Unit.

In Figure 4.8 the internal structure of a GRU cell is sketched. Every element of the cell and their relationship is here formally presented. First we start by defining the variables involved:

- $\mathbf{x}_t$: input vector;
- $\mathbf{h}_t$: output vector, activation vector;
- $\tilde{\mathbf{h}}_t$: candidate activation;
- $\mathbf{z}_t$: update gate vector;

- $\mathbf{r}_t$: reset gate vector;
- W, U and b: parameter matrices and vector.

The activation $\mathbf{h}_t^j$ of the GRU at time t is a linear interpolation between the previous activation $\mathbf{h}_{t-1}^j$ and the candidate activation $\tilde{\mathbf{h}}_t^j$: [79]

$$\mathbf{h}_t^j = \left(1 - \mathbf{z}_t^j\right) \tilde{\mathbf{h}}_{t-1}^j + \mathbf{z}_t^j \mathbf{h}_t^j \, , \tag{4.10}$$

where an update gate $\mathbf{z}_t^j$ decides how much the unit updates its activation, or content. The update gate is computed by:

$$\mathbf{z}_t^j = \sigma \left(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1}\right)^j \, . \tag{4.11}$$

The candidate activation $\tilde{\mathbf{h}}_t^j$ is computed similarly to that of the traditional recurrent unit:

$$\tilde{\mathbf{h}}_t = \tanh(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))^j \, , \tag{4.12}$$

where $\mathbf{r}_t$ is a set of reset gates and $\odot$ is an element-wise multiplication. The reset gate $\mathbf{r}_t^j$ is computed similarly to the update gate:

$$\mathbf{r}_t^j = \sigma \left(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1}\right)^j \, . \tag{4.13}$$

Gating has revealed itself as a fundamental technique to tackle one of the most famous problems of standard recurrent neural networks: the vanishing gradient problem.
GRUs relates very closely to LSTMs which are so far more famous. These two gated cells, however, have been found to have almost the same performance. The real and crucial difference is that GRUs involve fewer gates, therefore, they are computationally more efficient and easier to train. GRUs, indeed, are best suited to the case of smaller datasets [79].

5 Anomaly Detection System

This chapter describes, in full detail, the Anomaly Detection System developed by the author. Here it is presented how previously introduced algorithms and dataset are combined to perform the task at hand.
Firstly, the structure of the framework is outlined, then all the implementation details and numerical evaluations are reported.
The system is composed of five main blocks:

1. Data acquisition;
2. Data and video processing;
3. Regression;
4. Prediction error statistics extraction;
5. Classification.

Each building block will be characterized and, if present, intermediate results will be discussed.

A conceptual representation of the framework is shown in Figure 5.1.

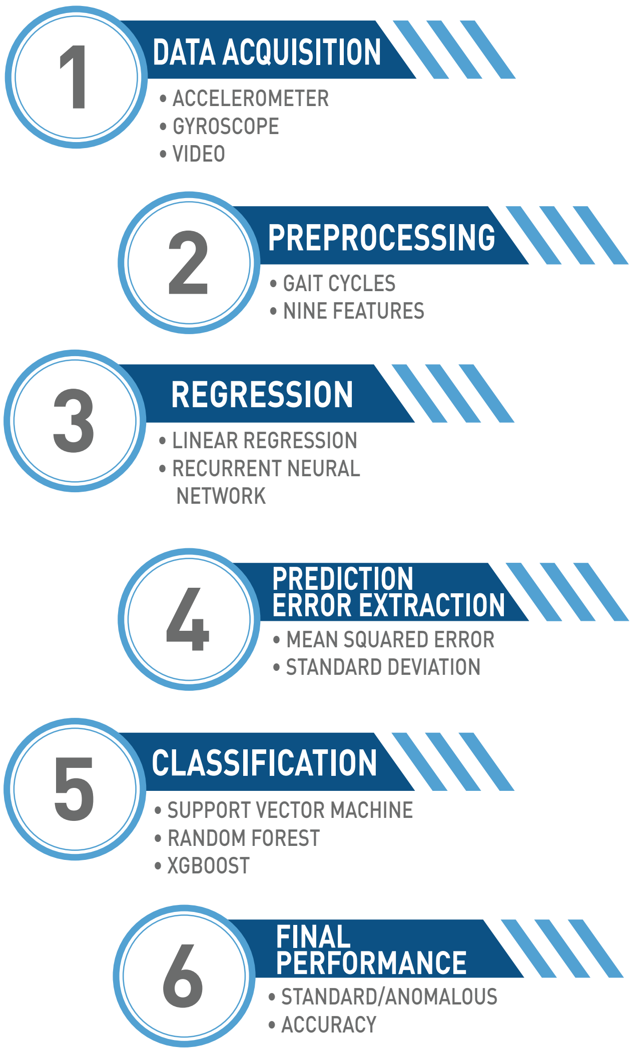


Figure 5.1: General scheme of the Gait Anomaly Detection System.

5.1 Data Acquisition

The dataset of this thesis has been created by the author throughout the acquisition of data from several walking sessions.
The device involved in data collection is an Asus Zenfone 2 which has been located in ad hoc made chest support (see Figure 5.2). The smartphone is used to collect

inertial data and video signals from the built-in sensors and rear-facing camera during several walking sessions. This smartphone features a 2.3 GHz quad-core Intel Atom Central Processing Unit (CPU) and it comes with 4GB of RAM. The Operating System (OS) is Android 5 Lollipop.

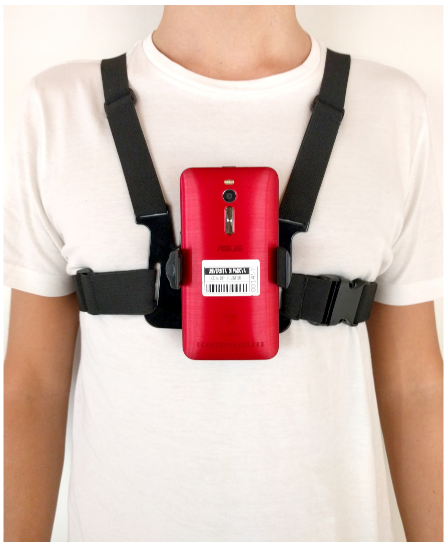


Figure 5.2: Chest support for smartphone.

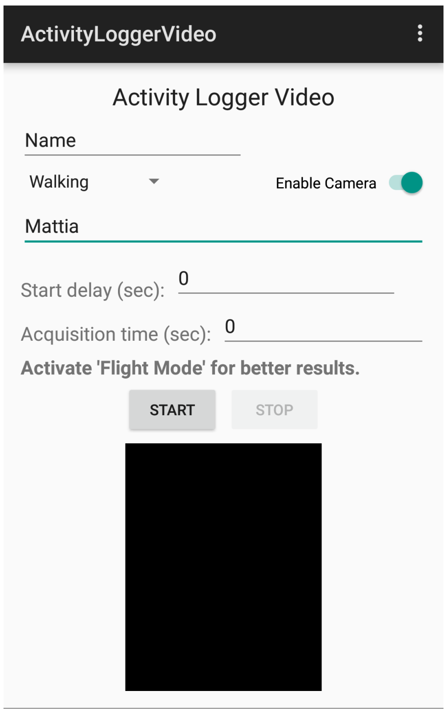
As anticipated in the previous chapters, the sensors of interest for this work are the accelerometer and the gyroscope, as well as the camera.
Accelerometers will measure the directional movement of a device but will not be able to resolve its lateral orientation. In fact the accelerometer tracks acceleration (m/s^2) or senses device vibration measuring linear movement acceleration.

The gyroscope, on the other hand, is used to track device's rotation measuring the angular rotational velocity ($°/s$). This information is then complementary to the one supplied by the accelerometer.

The reference video is recorded with a frame rate of 30 fps with an H.264 compression and a resolution of 720×576 pixels.

Data collection and video recording are performed using an ad hoc developed, Android application, called *Activity Logger Video*. Given basic information about the user, the application collects data from the built-in sensors, records a reference video from the rear-facing camera and saves all the acquired data in the non-volatile memory. This application is an extension of the *Activity Logger* Android application developed in [80].

The home screen of the application is shown in Figure 5.3. Here we can see the menus where the necessary acquisition parameters can be set up. In particular, the final user is only asked to specify the name and the type of activity along with the delay between pressing the "START" button and initializing the recording, and the duration of the recording itself.


ActivityLoggerVideo

Activity Logger Video

Name

Walking

Enable Camera

Mattia

Start delay (sec): 0

Acquisition time (sec): 0

Activate 'Flight Mode' for better results.

Figure 5.3: Example of recording application home screen.

Anomalous gait cycles are collected by modifying the walking style in several ways. Examples of these modifications are: excessive stride length, excessive chest torsion, excessive knee lifting, sussultory chest movement, and limps.

5.2 Data preprocessing

After data collection, the first step of data preprocessing consists in the extraction of the roll, pitch, and yaw signals from the reference video. This procedure is performed as detailed in Chapter 3. The signals extracted correspond to the temporal evolution of the smartphone angular orientations in a three-dimensional space.

Successive steps are interpolation, filtering, cycles extraction, signals de-trending, and features normalization.

Interpolation

As already discussed in Chapter 3, the sampling rate of the smartphone is not constant during the acquisition. This leads to non-constant inter-sample spacing. An interpolation procedure is then needed. The solution for this problem is a spline interpolation and a resampling operation which ensures a sampling rate of $f_s = 200$ Hz and a fixed time interval between samples.

In Figure 5.4 is shown the distribution of the sampling frequency for the smartphone utilized during our tests, on the left, and a comparison model, on the right. We can see that the Asus results have a greater variance in the sampling rate.

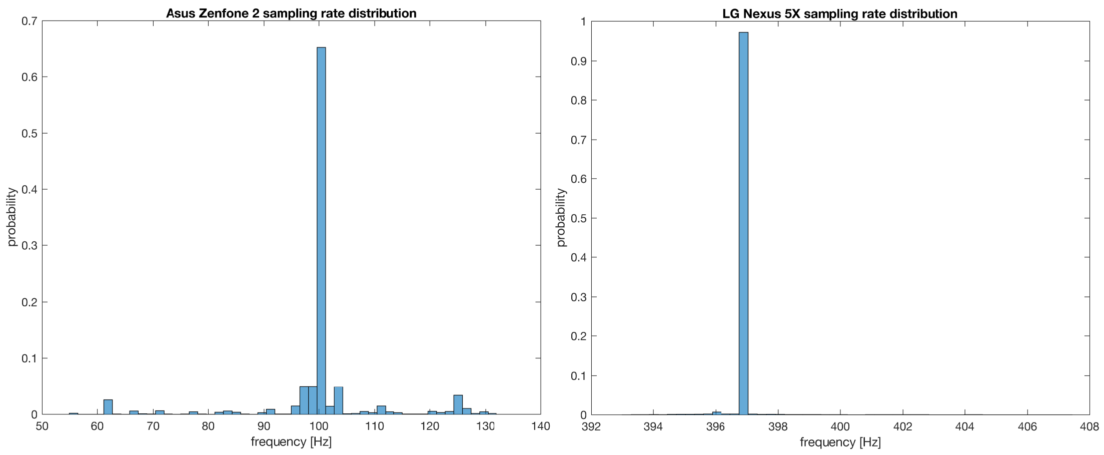


Figure 5.4: Comparison of the sampling frequency distribution of the smartphone employed in the data acquisition (Asus Zenfone 2) and another smartphone (LG Nexus 5X).

Filtering

In Figure 5.5 the PSD of the three-axial data of the accelerometer sensor is shown. It can be seen that most of the information is located at low frequencies and just a very small amount of power characterizes frequency above 40 Hz. The PSD is computed through the Welch's method [59], considering a full walking trace and setting the Hanning window length to 1 s, with half window overlap as in [2].

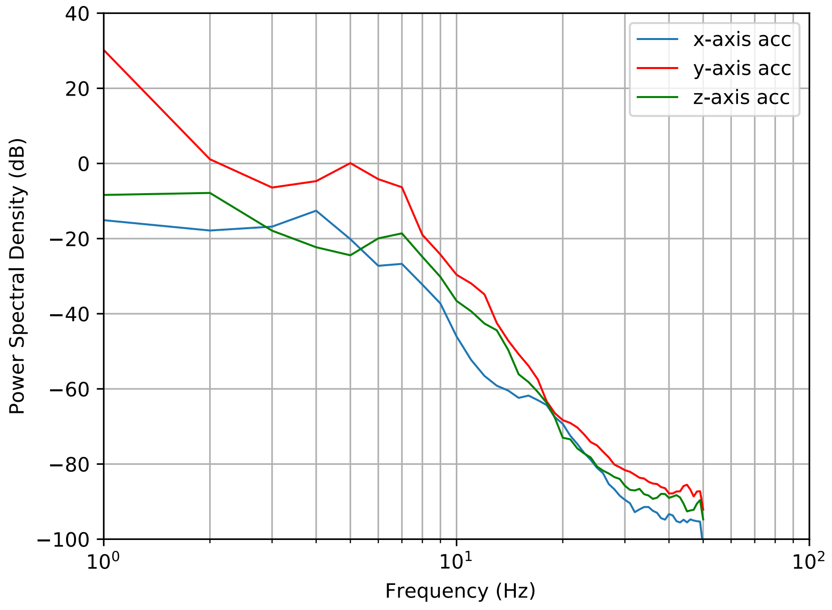


Figure 5.5: Power spectral density of the three-axial aceelerometer data.

Based on previous considerations, each signal is filtered with a Butterworth low pass filter of order 10 and cutoff value $f_c = 40$ Hz. This operation smooths the signals and removes high-frequency motion artifacts.

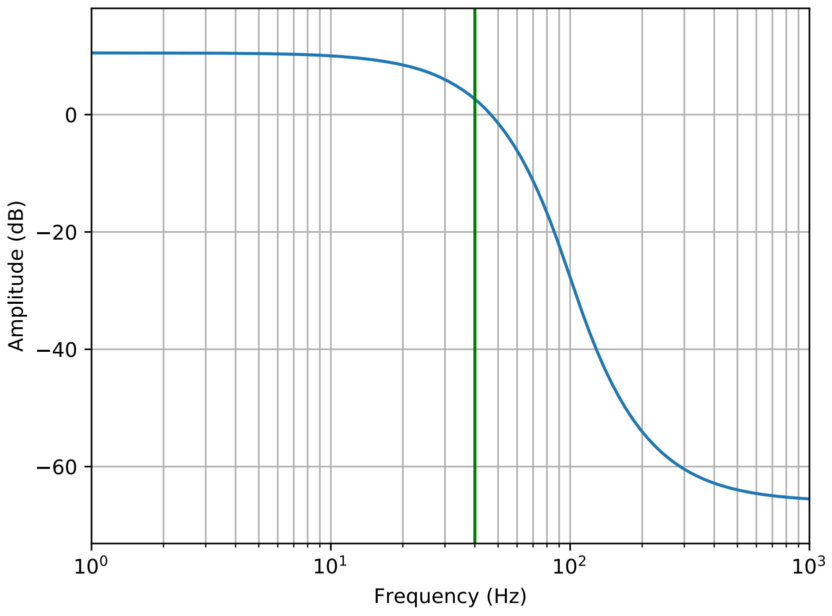


Figure 5.6: Frequency response of the Butterworth filter in blue, cutoff frequency in green.

This operation is applied to all the available signals. The frequency response of the filter is presented in Figure 5.6, while in Figure 5.7 the comparison between the yaw raw signal and its filtered version is shown.

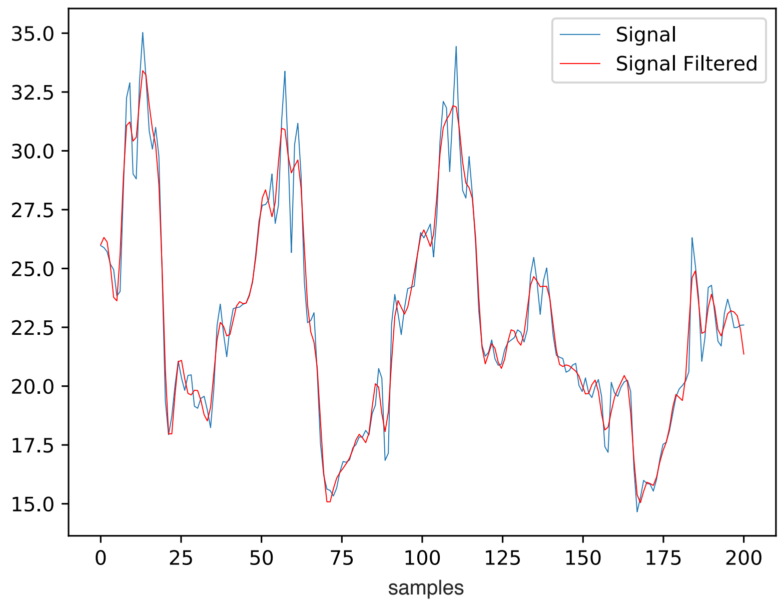


Figure 5.7: Comparison between raw signal and its filtered version. The considered signal is the yaw angle evolution.

Cycles extraction

As presented in Chapter 3, the determination of gait cycles is performed exploiting the CWT transformation. In particular, IC and FC events are extracted as local minima and local maxima of the first and second order derivation of the CWT transform of the y-axis signal of the accelerometer. The result of the procedure is shown in Figure 5.8, where the circles represent the IC instants while the triangles are the FC instants. Moreover, the solid line represents the signal $\mathbf{a}_y$ on which the analysis is based. Instead, the dashed lines represent the CWT transformations.

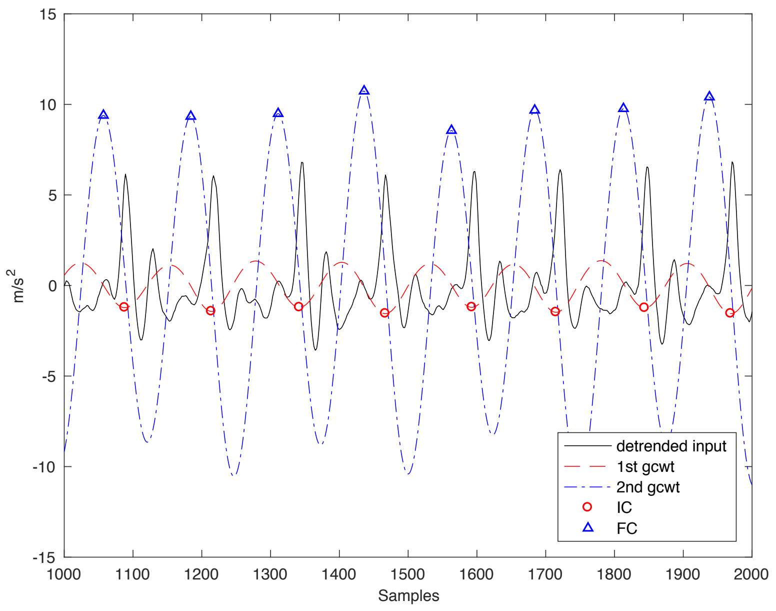


Figure 5.8: Example of IC (circles) and FC (triangles) detection.

After the ICs and FCs time instants have been individuated, the left and right steps can be identified by looking at the sign of the signal obtained by filtering the y-axis of the gyroscope signal $\mathbf{g}_y$.

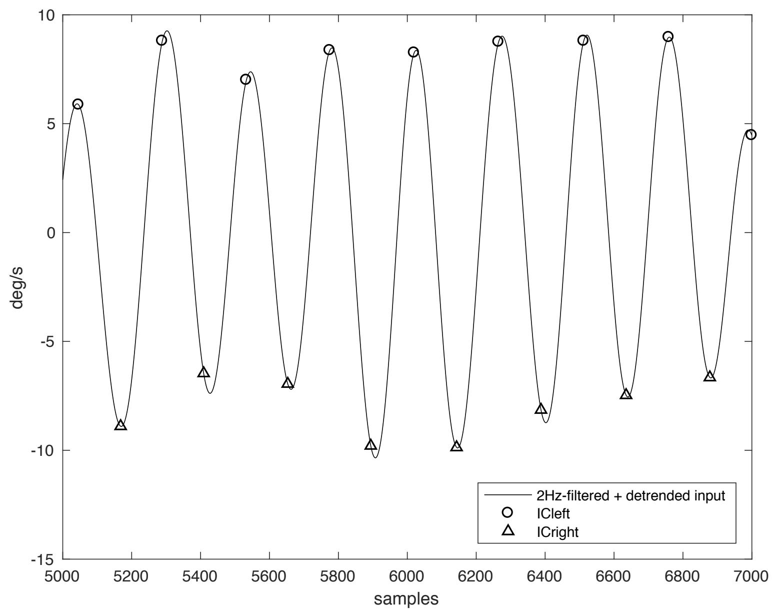


Figure 5.9: Example of estimation on left or right step.

The filtering is computed considering a low-pass 4th-order Butterworth filter with cutoff frequency $f_c = 2$ Hz. As demonstrated in [2, 60] this choice of parameters leads to good estimation results.

After this processing phase, gait cycles instants can be easily identified. In fact, a generic walking cycle i starts at $IC(i)$ and ends at $IC(i+2)$. Moreover, in order to avoid possible errors and transient phenomena, at the beginning and at the end of each acquisition, the first and last 5 cycles are excluded from the computation.

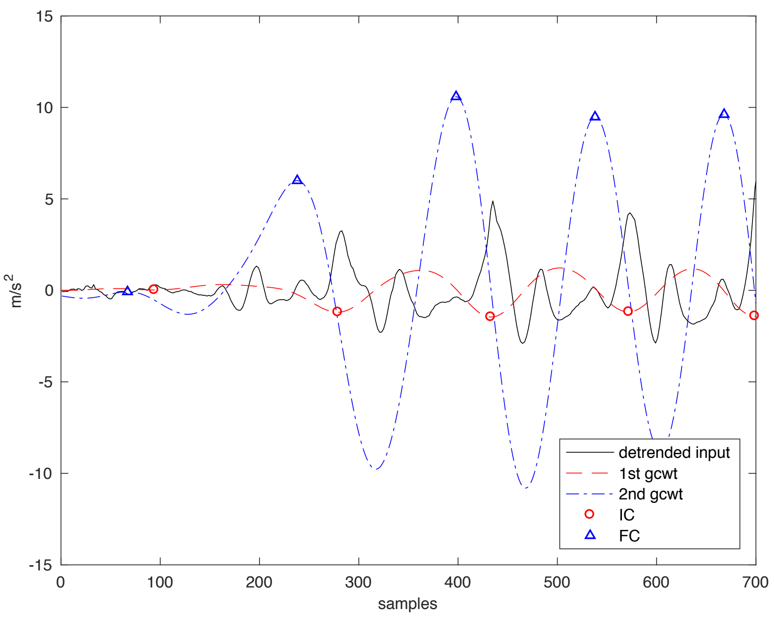


Figure 5.10: Example of initial signal transient.

In Figure 5.10 we present an example of the transient we want to discard. In this region the IC and FC identification are often erroneous. This particular fragment is extrapolated from the first acquisition, however, every acquisition shows similar behavior, therefore this approach is chosen as rule to eliminate initial cycles.

Once the IC instants are known it is possible to determine the cycle vectors in every signal.

Other useful parameters like stance, swing time, step length and step velocity can be extracted from the IC and FC events and from the accelerometer data related to every walking cycles. The detailed extraction, which is out of the scope of this work, is fully presented in [2].

Each extracted gait cycle has a different duration, which depends on the walking speed and the stride length. Because of that, the accelerometer data and the signals extracted from the video, have variable sizes and, hence, further adjustment is necessary. This is done using a spline interpolation to represent all the walking cycles through vectors of $N = 200$ samples each. This value of N is selected in order to avoid aliasing. In fact, assuming a maximum duration of $\tau = 2$ seconds, and a signal bandwidth of $B = 40$ Hz, consequence of the filtered previously performed, a number of samples $N > 2B\tau = 160$ samples/cycle is required.

De-trending

Signals extracted from the reference video shows the presence of different trends on each axis. Figure 5.11 represents an extract (a very small subset of the data) of the signals, showing several different local trends. In Figure 5.11, each vertical line separates data acquired in different sessions.

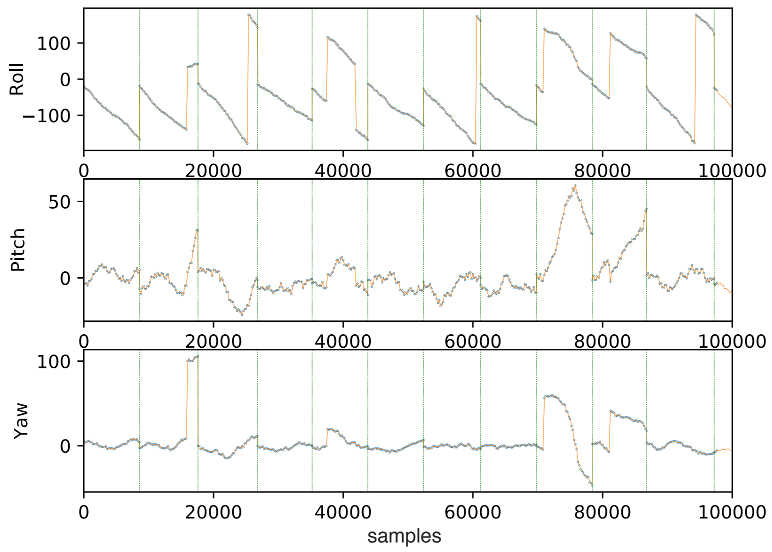


Figure 5.11: Example of trends in data extracted from video. Green vertical lines separates different acquisition sessions.

The presence of these trends may cause inefficiency in grasping the underneath hidden information once the dataset is fed into the machine learning algorithms.

In fact, before feeding data to the algorithms, it is good practice to normalize it, as explained in the next paragraph. The presence of the trend might mask the true information the algorithms seek to learn. In this specific case, it is then necessary to remove the trend to uncover relevant information in the series.

For removing the trend and obtaining a stationary series, several techniques can be used. The one used in this work is the ”Detrend by Model Fitting”. To operate this method a further simplification assumption must be made, the trend that affects a gait cycle is considered to be linear. Therefore, the trend affecting each cycle is approximated by a linear model. This assumption allows us to remove the global trend without interfering with the cycle's inner trend. After the model is fitted, the de-trending is computed just by subtracting the model from the dataset.

The result shows the good performances of the procedure. In fact, in Figure 5.12 it can be seen that the trends have been removed with good approximation.

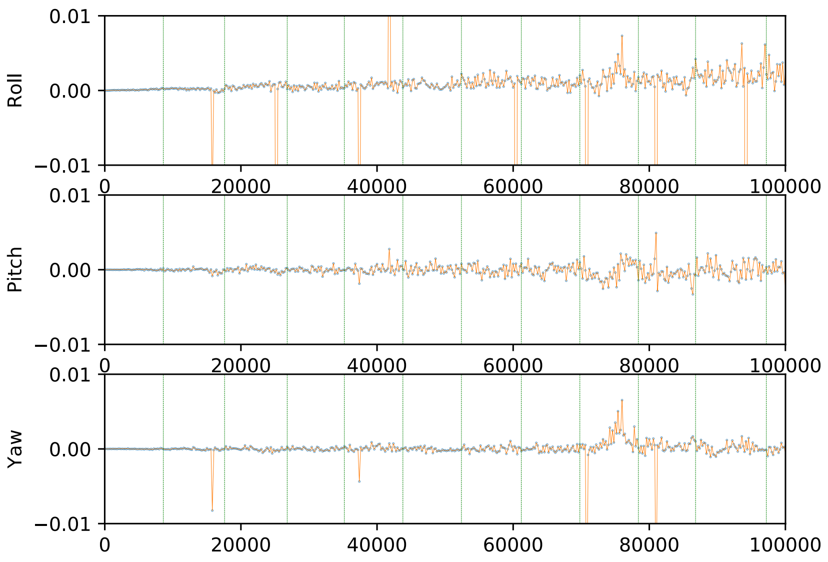


Figure 5.12: Example of video detrended data. Green vertical lines separates different acquisition sessions.

If we consider the first acquisition session, we can see in Figure 5.13 the improvements achieved thanks to the trend removal.

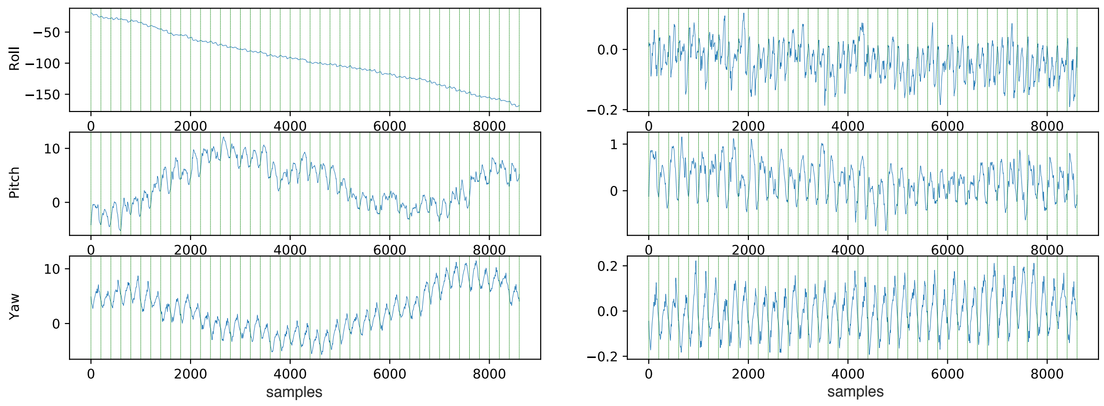


Figure 5.13: First acquisition sessione, on the left the trend is present, on the right it is removed. Vertical lined separates different gait cycles.

Figure 5.12 does not show the underlying trend we seek to reveal. However, if we normalize the data set after the trending removal we obtain a clear visualization of the inter-cycles trend, as shown in Figure 5.14. The comparison with previous Figures gives a solid understanding of how much the true relevant information was covered by the trends.

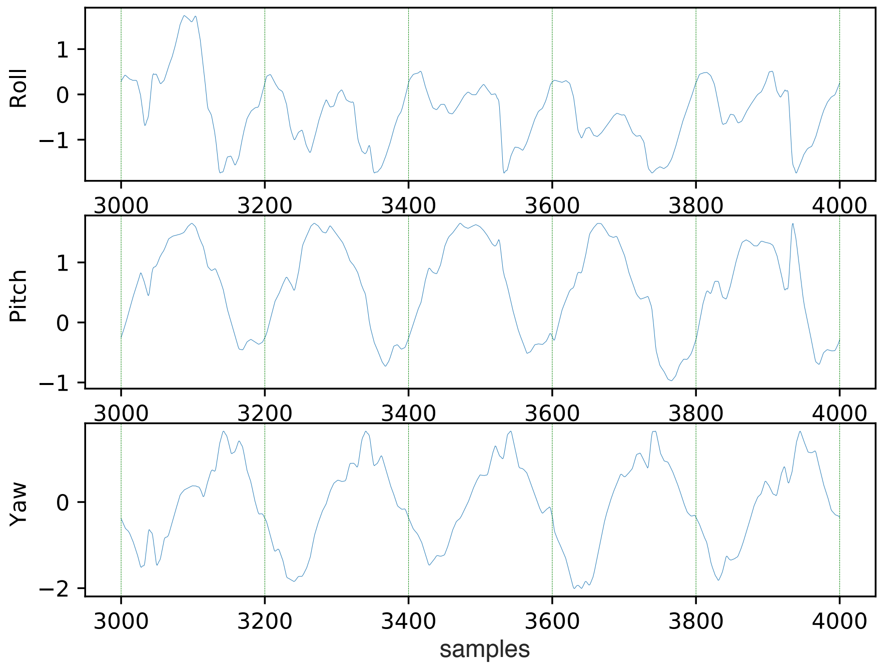


Figure 5.14: Example of underlying data trend, visible after detrending and normalization are performed.

Normalization

In this work, two normalization techniques have been employed. The first technique is a standard normalization, which is performed by removing the mean and by dividing the data by its standard deviation. This normalization is applied to each feature of the data used for the regression task. A different approach has been adopted for preprocessing the data used in the classification module. In particular, the data fed to the classification algorithms are rescaled in the $[0, 1]$ range. Practically, these operations are performed exploiting the powerful API: scikit-learn [81].

Dataset division

The dataset is divided into two main parts. The first one is dedicated to the training, validation, and testing of the regression algorithms. This portion of the dataset is composed of a large number of standard gait cycles, precisely this part is composed of 78 different acquisitions used for training and validation, for a total of 4388 cycles, and 10 acquisitions used for testing, which amount for 578 total cycles. The division of the samples between training and validation data is performed by randomly selecting the acquisitions with a (75%/25%) split. The second part of the dataset, instead, is reserved for the training, validation, and testing of the classification algorithms. This latter portion of the dataset is composed both of standard and anomalous cycles. The number of acquisitions is 53 for standard data and 61 for anomalous data, which contain respectively 2975 and 2744 cycles. Therefore the two classes are almost balanced (50%/50%). The split between train and test is again set to (75%/25%). However, the split is executed directly on the cycle descriptors and not on the acquisitions. A k-fold cross-validation approach is employed to determine the best set of parameters for each classifier. During this phase, the training data is then used both for training and validation.

5.3 Regression

Once the collected data has been preprocessed, it is possible to train an RNN to learn to predict the temporal evolution of each signal.

RNN Architecture: Several architectures have been tested for this task varying the number of layers and their size, which is represented by the number of neurons. Moreover, all the notable hyper-parameters have been tuned and will be presented afterward. The final architecture is composed of four layers, one input layer, two recurrent layers and one output layer. Recurrent layers are composed each of 400 GRU neurons which, as presented in the previous chapter, are well suited for modeling the temporal evolution of time series such as the signals at hand. With this configuration, the network is described by $1,489,209$ parameters which are the total weights and biases of all the neurons.

The implementation of this network is performed exploiting the Keras API [82] which is a high-level neural networks API acting as a tensorflow frontend [83]. TensorFlow is an open source software library for numerical computation using data flow graphs. Each node of the graph represents an instance of a mathematical operation, and each edge is a multidimensional data set called *tensor* on which the operations are performed. Understanding these core concepts of Tensorflow is fundamental also in the context of high-level API involvement. In fact, knowing the core structure of tensorflow it is possible to adapt the code to best exploit its computational power.

Tensorflow gives the users several tools for debugging neural networks, one of them is Tensorboard. This tool allows to visualize the layer structures and dependencies. An example is shown in Figure 5.15. We can see the fundamental building blocks of the computational graph. A useful scope organization technique helps to organize the view of the layers, in particular, each GRU component may be exploded to investigate the internal structure. Although very interesting, it may be very complicated for non-expert users, which brings us back to the Keras API that helps us creating layers of abstraction on top of the computational graph. Finally, Tensorflow natively supports computation on GPU devices, which is fundamental for Neural Networks training.

Training Several other parameters must be set up before the training phase can take place. One of these hyper-parameters is the initialization of the weights and biases of each neuron. For this purpose, two different methods are used. Biases are initialized with constant values equal to $b = 1$, while weights are

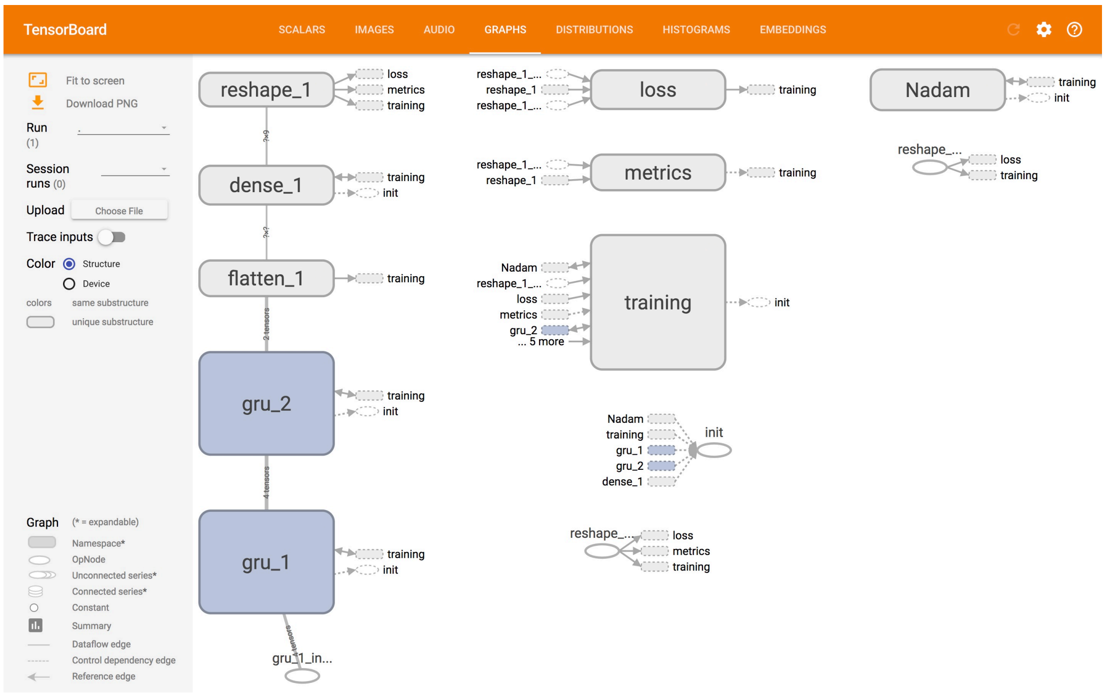


Figure 5.15: Neural networks structure shown using the tensorboard tool.

initialized by randomly drawing samples from a truncated normal distribution. The distribution is characterized by zero mean standard deviation $std = \sqrt{\frac{1}{n}}$, where n is the average of the numbers of input and output units. This choice is made to maintain variance of activations throughout the network. This reflects on a faster convergence of the weights to the optimal value that minimizes the loss function [84].

The optimizer involved in training procedure is the Nadam [85], which has demonstrated to offer the best performance.

The activation functions have been chosen to be the hyperbolic tangent in the hidden layers and linear in the output one. The network has also been tested with RELU activation function, showing a reduced performance in comparison with tanh. Finally, the last parameters to be set are the batch size and the number of training epochs, which have been chosen to be respectively $batch\_size = 100$ samples and $epochs = 10$. At this point, data from each gait cycle is segmented following an overlapping window technique. The input window size is set to be equal to 10 samples for each feature/signals, each input to the neural network is then a matrix of (10×9) values. The neural network is then trained to predict the next sample of each signal, outputting a matrix (1×9).

During the training phase, the MSE is evaluated at the end of each epoch. The result is presented in Figure 5.16. The number of gait cycles involved in the training phase are $N_1 = 3301$ while the ones left out for validation are $N_2 = 1087$. Considering the sliding window procedure, the number of samples available for training are $s_1 = 659620$ and for validation $s_2 = 217200$.

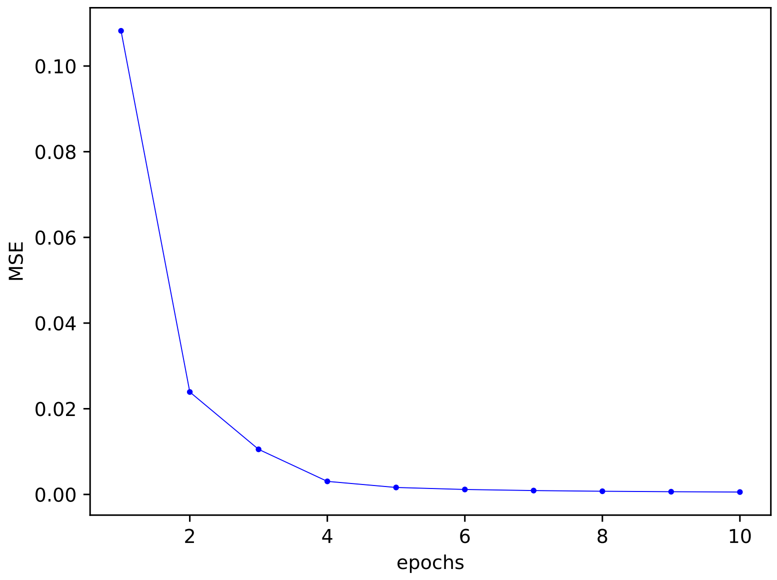


Figure 5.16: Evolution of the MSE score throughout several training epochs.

Comparison: The value of the MSE itself is not an absolute measure of quality. To better assess the prediction capacities of the Neural Network an additional regressor is trained, an LR model. The Coefficient of Determination (CoD) (R^2) score for this regressor is equal to one, which is the maximum achievable. It provides a measure of how well observed outcomes are replicated by the model, based on the proportion of the total variation of the outcomes explained by the model. The two models are used to perform predictions on a certain number of cycles, and the respective MSE are compared. The performance are compared on the three parts of the dataset. In Table 5.1 we summarize the performances of the two algorithms on all the dataset parts.

		Training	Validation	Test
	LR	0.00073	0.00054	0.00065
MSE	RNN	0.0018	0.00133	0.00116

Table 5.1: Comparison for mean square of prediction error for different regressors.

This first result might suggest that the neural network does not introduce improvements upon simpler approaches. This intuition will be contradicted by the numerical evaluations presented in Chapter 6.

In Figures 5.17 and 5.18 a visual comparison of the two regressors performance are shown. The considered signals are respectively x-axis and z-axis of the accelerometer device. It can be seen that both the techniques achieve excellent performance with just minor differences, which are appropriately highlighted.

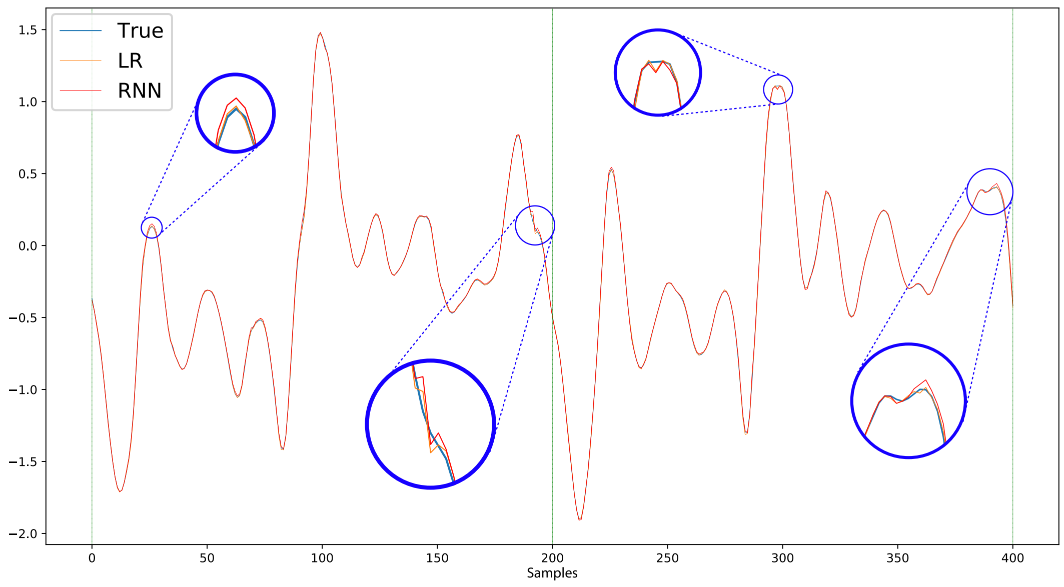


Figure 5.17: Comparison of regressors performance on two cycles of x-axis of the accelerometer signal.

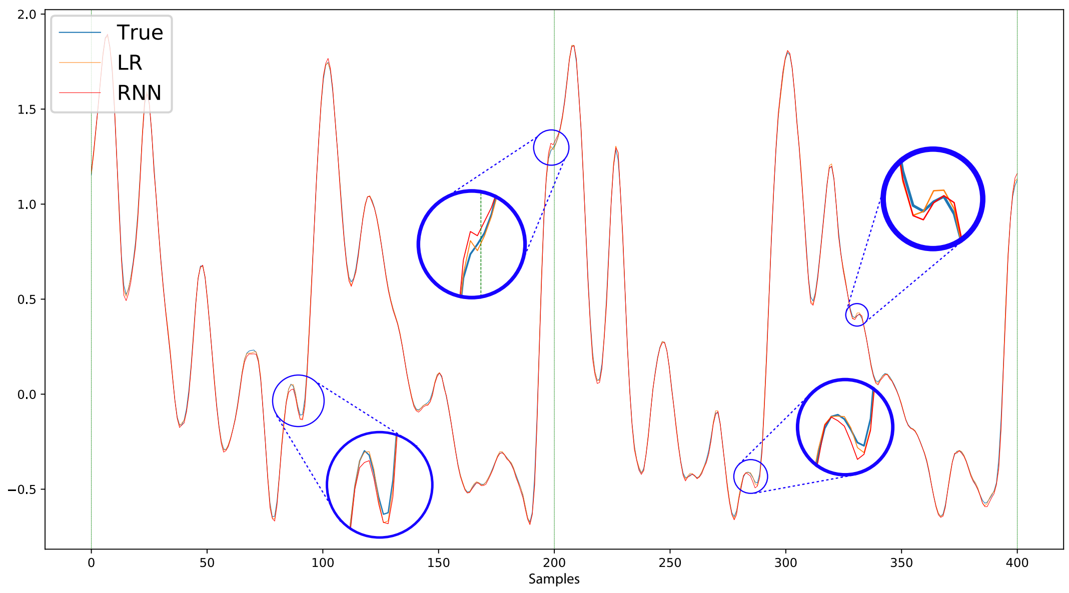


Figure 5.18: Comparison of regressors performance on two cycles of z-axis of the accelerometer signal.

5.4 Prediction error statistics extraction

In this module of the framework, the second part of the dataset is considered. As in the previous step, each gait cycles is further segmented using the sliding window technique, and each obtained sample is fed to all the regression algorithms.

Each cycle is then reconstructed using the prediction outputs, and the prediction error is computed by simply subtracting the predicted signals $\hat{\mathbf{s}}_i(t)$ evolution with the true signals $\mathbf{s}_i(t)$ evolution: $\mathbf{E}_i(t) = \hat{\mathbf{s}}_i(t) - \mathbf{s}_i(t)$ with $i \in \{1, \dots 9\}$.

For each cycle and each feature, the MSE and the standard deviation σ of the prediction errors are computed, leading to a cycle descriptor of dimensions (2×9).

It is possible to further investigate the prediction error before proceeding to present the classification module. In Figure 5.19 it is presented the comparison between standard data and anomalous data regarding MSE estimation for first feature (x-axis of accelerometer signal). What is shown is the histogram of the error values of all cycles. It can be easily seen that prediction on anomalous data induces a higher prediction error, as expected. The regressor used in the RNN.

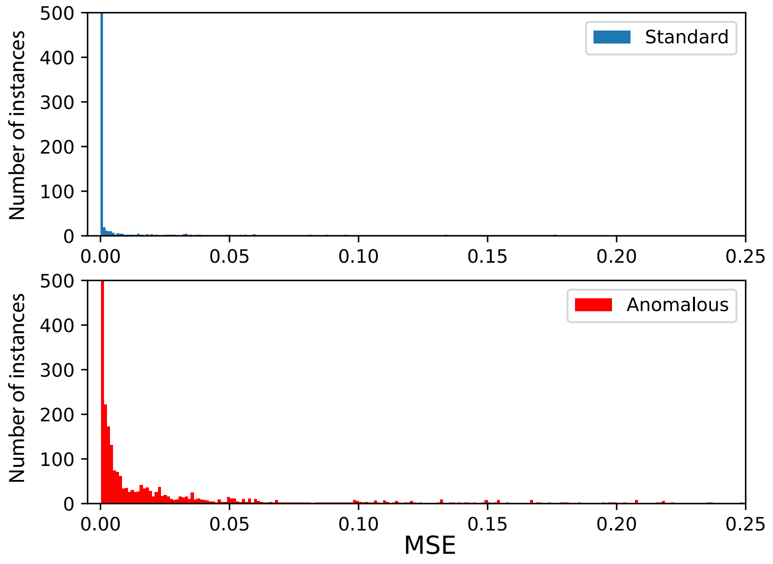


Figure 5.19: Comparison between the distribution of MSE estimation across all cycles. Considered signal is the x-axis of accelerometer.

Greater differences are shown for the standard deviation estimation of the prediction error as shown in Figure 5.20

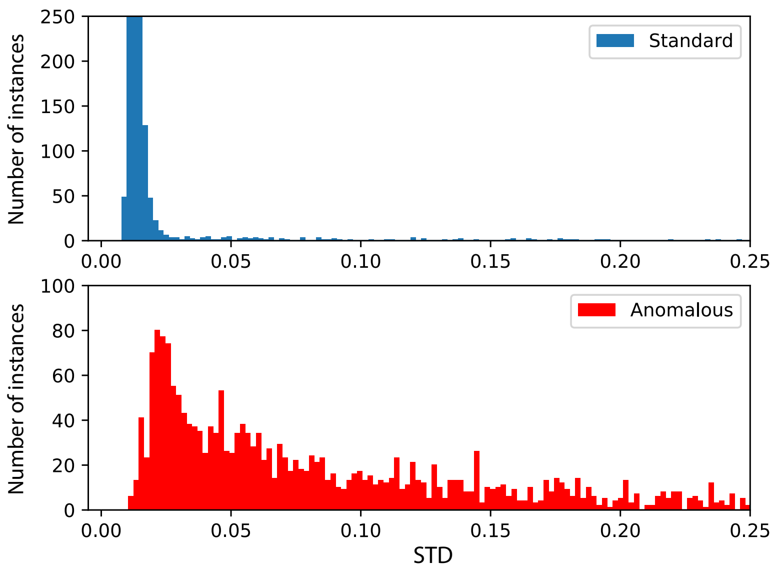


Figure 5.20: Comparison between the distribution of standard deviation on prediction error across all cycles. Considered signal is the x-axis of accelerometer.

In this phase, the cycles are labeled, ”1” for standard data and ”0” for anomalous data.

5.5 Classification

Based on the cycle descriptors computed at the previous step, in this module, several classification techniques are considered to distinguish between standard and anomalous cycles automatically. The algorithms involved are the ones described in Chapter 4.

A grid search technique is exploited to tune the hyper-parameters of each classification algorithm, to obtain the best set of parameters and maximize the correct classification rate.
For this procedure, a feature space, is chosen for each algorithm. Moreover, a k-fold cross-validation scheme with k = 3 is employed to determine which set of parameters reach the highest value of the accuracy score.

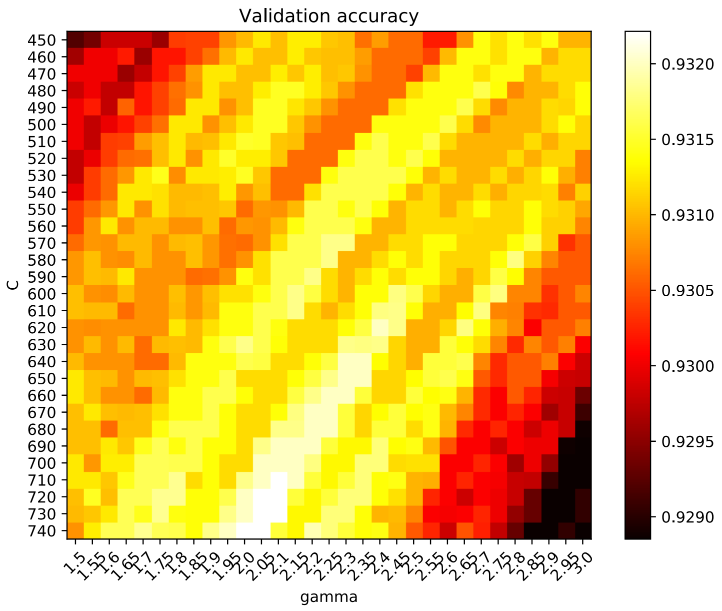


Figure 5.21: Visualization of grid search scores for the SVM classification algorithms.

In Figure 5.21 a visual example of grid search is shown. The algorithm involved is the SVM one and the parameters investigated are C, the penalty parameter, and γ, the kernel coefficient. In this figure is shown the validation score of each combination of the two parameters. Similar images can also be obtained for the other algorithms.

With the intent to produce a complete analysis, not only the *accuracy* is investigated but also the *precision*, *recall*, and their combination F_1 *score* are reported for completeness.
Before presenting the results, we recall the definitions of these quantities:

- **Confusion matrix:** A confusion matrix (CM) is a table used to describe the performance of a classification model on a set of data for which the correct labels are known. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class. The name comes from the fact that it makes it easy to see if the system is confusing two classes. It is not only used in the case of binary classification where its interpretation is trivial, but also in cases of multi-class classification. An example of confusion matrix for the case of binary classification is show by Equation 5.1.

$$CM = \left(\begin{array}{c|c} TN & FN \\ \text{True Negative} & \text{False Negative} \\ \hline FP & TP \\ \text{False Positive} & \text{True Positive} \end{array} \right) \tag{5.1}$$

  True Positive (TP) and True Negative (TN) represents the number of instances correctly classified. On the contrary False Positive (FP) and False Negative (FN) express the number of misclassified samples. The accuracy measure is then computed from the table as the total number of correct classified cases over the total population:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{5.2}$$

- **Precision:** the precision (P) or Positive predictive value (PPV) is the proportion of the predicted positive cases that were correct,

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.3}$$

- **Recall:** the recall or true positive rate (TPR) is the proportion of positive cases that were correctly identified,

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{5.4}$$

- F_1 **score:** this score is the harmonic average of the precision and recall,

$$F_1 = \frac{2}{\frac{1}{\mathrm{Precision}} + \frac{1}{\mathrm{Recall}}} \tag{5.5}$$

After the grid search is completed, each algorithm is trained on the whole training set using the selected parameters. The performance is then evaluated on a test set. The performance of the three classification algorithms is reported in Chapter 6.

6 Final Results

In this chapter, the final results and benchmarks are presented. Three tables summarize the classification performance of each algorithm. In each Table, we compare the performance of a specific classifier (namely, SVM, RF, and XGB) applied to the cycle descriptors obtained using the two regressors discussed in Chapter 4. The objective is to determine whether a specific cycle is anomalous or not, based on the prediction errors. As a result of the training of the regressors on standard data, it is expected that the prediction errors of anomalous cycles are characterized by greater values of MSE and standard deviation with respect to those of standard cycles. This, in turn, allows to correctly identify anomalous data.

In Table 6.1 the benchmarks of the SVM classifier are summarized. This algorithm reveals to be the weakest among the adopted techniques. It can be noticed that the utilization of the RNN brought consistent improvements. Indeed, we observe an increment of the accuracy score of 6.5% with respect to the LR technique. This result implies that the utilization of the RNN leads to error statistics which are much more separable. Predictions obtained using the neural network on the anomalous data diverge more than the ones executed using the linear regression.

SVM				
		Training	Validation	Test
LR	Accuracy	0.9294	0.9322	0.9133
	Precision	0.9030	–	0.8933
	Recall	0.9687	–	0.9443
RNN	Accuracy	0.9811	0.9736	0.9790
	Precision	0.9724	–	0.9669
	Recall	0.9920	–	0.9932

Table 6.1: Performance comparison of SVM classifier applied to the cycles descriptors obtained using LR and RNN regression algorithms.

Moreover, it is worth noting that in both cases the accuracy score evaluated on the three parts of the dataset are coherent, meaning that no overfit nor underfit have occurred. It is also possible to infer that no perfect separation has been achieved by the algorithm and a number of misclassified points are also present in the training set.

As mentioned in Chapter 5, a cross-validation scheme is used to determine the hyper-parameters that optimize the accuracy score on the validation data. For SVM, the hyper-parameters involved in this phase are C, the penalty term, and the kernel coefficient γ. Moreover, the utilized kernel is the RBF one, which was presented in Chapter 4. The parameters that allow obtaining the best validation score are respectively: $C = 710$ and $\gamma = 2.10$ for LR, and $C = 570$ and $\gamma = 2.45$ for RNN. These parameters have been obtained by means of grid search during the model selection phase. It is worth mentioning that the cross-validation technique was set to maximize the F_1 score, in all the experiments. This methodology then seeks to balance precision and recall.

To analyze the precision and recall metrics, we also refer to the confusion matrices which are directly related to the metrics as presented in Chapter 5. Equation 6.1 reports the confusion matrices obtained by the SVM algorithm operating on MSE generated by the LR predictions. The notation related to the confusion matrices is X_Z^Y, where X is the classifier, Y is the regressor, and Z is the part of the dataset to which the matrix refers to.

$$SVM^{LR}_{TRAIN} = \left(\begin{array}{c|c} 1811 & 233 \\ \hline 70 & 2169 \end{array}\right) \quad , \quad SVM^{LR}_{TEST} = \left(\begin{array}{c|c} 611 & 83 \\ \hline 41 & 695 \end{array}\right) \tag{6.1}$$

Using the same notation as in Equation 6.1, in Equation 6.2 the performance obtained exploiting the RNN as regression algorithm are reported.

$$SVM^{RNN}_{TRAIN} = \left(\begin{array}{c|c} 1987 & 63 \\ \hline 18 & 2221 \end{array}\right) \quad , \quad SVM^{RNN}_{TEST} = \left(\begin{array}{c|c} 669 & 25 \\ \hline 5 & 731 \end{array}\right) \tag{6.2}$$

This representation of the classification performance helps to better understand the significant improvement introduced by the RNN technique. The information contained in these matrices can be interpreted as follows:

- the first row contains the number of correct classified anomalous cycles in the first cell and the number of misclassified anomalous cycles in the second cell;

- the second row contains the same information for the standard cycles, but the cells are swapped: the first cell counts the misclassified samples and the second cell counts the correctly classified ones.

By comparing the confusion matrices obtained during the test phase, we can see that the number of misclassified anomalous data cycles decreases from 83 to 25, which is less than 1/3. Even better performance is achieved when classifying the standard gait cycles. In fact, the number of the misclassified standard cycles is reduced from 5.6% to 0.7%.

Considering the RF algorithm, the results are reported in Table 6.2. Differently from previous results, in this case, the algorithm is able to perfectly discriminate between standard and anomalous data during the training phase.

RF				
		Training	Validation	Test
LR	Accuracy	1.0	0.9717	0.9664
	Precision	1.0	–	0.9649
	Recall	1.0	–	0.9701
RNN	Accuracy	1.0	0.9884	0.9867
	Precision	1.0	–	0.9904
	Recall	1.0	–	0.9837

Table 6.2: Performance comparison of RF classifier applied to the cycles descriptors obtained using LR and RNN regression algorithms

Despite the equal performance of the training phase, during validation and test the RNN technique outperforms the LR as in the previous classifier. The gap between the two approaches is reduced, but still it is about 2% which is quite an improvement when dealing with such high performance. In Equations 6.3 and 6.4 the confusion matrices are reported.

$$RF^{LR}_{TRAIN} = \left(\begin{array}{c|c} 2050 & 0 \\ \hline 0 & 2239 \end{array}\right) \quad , \quad RF^{LR}_{TEST} = \left(\begin{array}{c|c} 668 & 26 \\ \hline 22 & 714 \end{array}\right) \tag{6.3}$$

We see from these matrices that the RF algorithm outperforms the results obtained with the SVM one. The only hyper-parameter involved in the cross-validation phase is the number of trees in the forest. The selected best values are 230 and 270 respectively for LR and RNN data.

$$RF^{RNN}_{TRAIN} = \left(\begin{array}{c|c} 2050 & 0 \\ \hline 0 & 2239 \end{array}\right) \quad , \quad RF^{RNN}_{TEST} = \left(\begin{array}{c|c} 687 & 7 \\ \hline 12 & 724 \end{array}\right) \tag{6.4}$$

It is worth observing that, when using the RF algorithms the misclassified samples are more balanced between the two classes with respect to the results obtained using the SVM.

Finally, the performance of XGBoost is shown in Table 6.3. This algorithm reaches the best performance pushing the accuracy up to 98.88% on the test set.

XGB				
		Training	Validation	Test
LR	Accuracy	1.0	0.9749	0.9706
	Precision	1.0	–	0.9715
	Recall	1.0	–	0.9715
RNN	Accuracy	1.0	0.9901	**0.9888**
	Precision	1.0	–	0.9932
	Recall	1.0	–	0.9851

Table 6.3: Performance comparison of XGB classifier applied to the cycles descriptors obtained using LR and RNN regression algorithms.

Like the RF, the XGB algorithm is capable of perfectly classifying all the cycles during the training phase. Differently from RF, XGB shows better generalization capabilities offering improved performance when dealing with previously unseen data. In equations 6.5 and 6.6 the last confusion matrices are reported.

$$XGB^{LR}_{TRAIN} = \left(\begin{array}{c|c} 2050 & 0 \\ \hline 0 & 2239 \end{array}\right) \quad , \quad XGB^{LR}_{TEST} = \left(\begin{array}{c|c} 673 & 21 \\ \hline 21 & 715 \end{array}\right) \tag{6.5}$$

For what concerns the XGB algorithm, the hyper-parameters involved in the cross-validation are the number of boosted trees to fit, the maximum tree depth and the boosting learning rate. The optimal results are $(90, 6, 0.7)$ for LR and $(80, 6, 0.9)$ for RNN.

$$XGB^{RNN}_{TRAIN} = \left(\begin{array}{c|c} 2050 & 0 \\ \hline 0 & 2239 \end{array}\right) \quad , \quad XGB^{RNN}_{TEST} = \left(\begin{array}{c|c} 689 & 5 \\ \hline 11 & 725 \end{array}\right) \tag{6.6}$$

With these results we have demonstrated the successful of the proposed approach and the high accuracy performance of the whole system in automatically recognizing between standard and anomalous data. Moreover we have verified that the utilization of RNN helps in boosting the final score. In fact, these findings support the hypothesis that the RNN models are best suited when analyzing raw time-serie signals with respect to baseline models.

7 Conclusions and Future Work

In this thesis, we have presented a gait anomaly detection system based on inertial and video signals acquired from a smartphone located in an ad-hoc made chest support during different walking sessions. In particular, the proposed method exploits raw signals obtained from accelerometer and gyroscope sensors together with signals extracted from a reference video to estimate the presence of anomalies in walking cycles. After the preprocessing and cycles segmentation phase, an RNN is trained to learn the time evolution of the signals within each walking cycle. A windowing procedure is applied for training and predictions. Predicted signal values are then compared against true signals evolutions, and the prediction error statistics of each cycle are then fed into classification algorithms which are trained to recognize anomalies. The process is completely automated and free from the need of further human tuning starting from the preprocessing procedure on raw data to regression and classification tasks.
The performance of the proposed regression and classification algorithms has been investigated with separated datasets to eliminate any possible data dependence. Results showed that the application of RNN is best suited to learn temporal evolution mechanics and leads to better performances regarding classifications in respect to more straightforward regressive approaches. This analysis might be involved in future applications such as home-based health care and monitor of patients with Parkinson or Alzheimer, rather than detection of an alcoholic state

in subjects and several other tasks. Part of this work may also be employed in future framework of HAR.

Future works should address a systematic and complete analysis of the features sensitivity to noise to assess if any feature might be left out. Principal Component Analysis (PCA) may also be applied to address this problem. The results may help to reduce the complexity of the system speeding up the training procedure and computational time needed for operability. Other research might discuss the possibility to deploy slightly pre-trained networks to accelerate the training phase of final users. Regardless of the unicity of every user walking patterns, similarities are inevitably present. This framework could then take advantages of these similarities to exploit transfer learning [86].

References

[1] "The statistics portal - number of smartphone users in the united states from 2010 to 2022." https://www.statista.com/statistics/201182/forecast-of-smartphone-users-in-the-us/.

[2] A. Lanaro, "Unsupervised and multi-modal gait analysis through growing neural gas networks," *Master Thesis, University of Padova*, 2017.

[3] O. D. Incel, M. Kose, and C. Ersoy, "A review and taxonomy of activity recognition on mobile phones," *Journal of BioNanoScience*, vol. 3, no. 2, pp. 145–171, Jun 2013. [Online]. Available: https://doi.org/10.1007/s12668-013-0088-3

[4] Y. Arase, F. Ren, and X. Xie, "User activity understanding from mobile phone sensors," in *Proceedings of the 12th ACM International Conference Adjunct Papers on Ubiquitous Computing - Adjunct*, ser. UbiComp '10 Adjunct. New York, NY, USA: ACM, 2010, pp. 391–392. [Online]. Available: http://doi.acm.org/10.1145/1864431.1864452

[5] T. Seel, J. Raisch, and T. Schauer, "Imu-based joint angle measurement for gait analysis," *Sensors*, vol. 14, no. 4, pp. 6891–6909, 2014. [Online]. Available: http://www.mdpi.com/1424-8220/14/4/6891

[6] A. Muro-de-la Herran, B. Garcia-Zapirain, and A. Mendez-Zorrilla, "Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications," *MDPI Journal of Sensors*, vol. 14, no. 2, Feb. 2014.

[7] S. Chen, J. Lach, B. Lo, and G. Z. Yang, "Toward pervasive gait analysis with wearable sensors: A systematic review," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 6, pp. 1521–1537, Nov 2016.

[8] T. Chau, “A review of analytical techniques for gait data. part 2: neural network and wavelet methods,” *Gait & Posture*, vol. 13, no. 2, pp. 102 – 120, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0966636200000953

[9] M. Murray, “Gait as total pattern of movement,” *American Journal of Physical Medicine*, vol. 46, pp. 290– 333, 1967.

[10] M. Murray, A. B. Drought, and R. C. Kory, “Walking patterns of normal men,” *Journal of Bone and Joint Surgery*, vol. 46, 1964.

[11] J. Perry, “History of the study of locomotion.” [Online]. Available: http://www.clinicalgaitanalysis.com/history/modern.html

[12] D. Volpe, D. Pavan, M. Morris, A. Guiotto, R. Iansek, S. Fortuna, G. Frazzitta, and Z. Sawacha, “Underwater gait analysis in parkinson’s disease,” *Journal of Gait and Posture*, vol. 52, pp. 87 – 94, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0966636216306488

[13] M. Henriksen, H. Lund, R. Moe-Nilssen, H. Bliddal, and B. Danneskiod-Samsøe, “Test–retest reliability of trunk accelerometric gait analysis,” *Journal of Gait and Posture*, vol. 19, no. 3, pp. 288 – 297, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0966636203000699

[14] I. H. López-Nava and A. Muñoz-Meléndez, “Wearable inertial sensors for human motion analysis: A review,” *IEEE Sensors Journal*, vol. 16, no. 22, pp. 7821–7834, Nov 2016.

[15] S. Nishiguchi, M. Yamada, K. Nagai, S. Mori, Y. Kajiwara, T. Sonoda, K. Yoshimura, H. Yoshitomi, H. Ito, K. Okamoto, T. Ito, S. Muto, T. Ishihara, and T. Aoyama, “Reliability and validity of gait analysis by android-based smartphone,” *Telemedicine and e-Health*, vol. 18, pp. 292–296, March 2012.

[16] P. Rashidi and D. J. Cook, “Keeping the resident in the loop: Adapting the smart home to the user,” *IEEE Journal of Transactions on Systems,*

Man, and Cybernetics - Part A: Systems and Humans, vol. 39, no. 5, pp. 949–959, Sept 2009.

[17] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers, "A review of wearable sensors and systems with application in rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 9, no. 1, p. 21, Apr 2012. [Online]. Available: https://doi.org/10.1186/1743-0003-9-21

[18] S. Mazilu, U. Blanke, M. Hardegger, G. Tröster, E. Gazit, and J. M. Hausdorff, "Gaitassist: A daily-life support and training system for parkinson's disease patients with freezing of gait," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '14. New York, NY, USA: ACM, 2014, pp. 2531–2540. [Online]. Available: http://doi.acm.org/10.1145/2556288.2557278

[19] M. Kranz, A. Möller, N. Hammerla, S. Diewald, T. Plötz, P. Olivier, and L. Roalter, "The mobile fitness coach : towards individualized skill assessment using personalized mobile devices," *Journal of Pervasive and Mobile Computing*, vol. 9, no. 2, pp. 203–215, 2013.

[20] T. Stiefmeier, D. Roggen, G. Ogris, P. Lukowicz, and G. Tröster, "Wearable activity tracking in car manufacturing," *IEEE Journal of Pervasive Computing*, vol. 7, no. 2, pp. 42–50, April 2008.

[21] M. Goffredo, I. Bouchrika, J. N. Carter, and M. S. Nixon, "Performance analysis for automated gait extraction and recognition in multi-camera surveillance," *Journal of Multimedia Tools and Applications*, vol. 50, no. 1, pp. 75–94, Oct 2010. [Online]. Available: https://doi.org/10.1007/s11042-009-0378-5

[22] J. Little and J. E. Boyd, "Recognizing People by Their Gait: The Shape of Motion," *Videre Journal of Computer Vision Research*, vol. 1, pp. 1–32, 1996. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.528

[23] D. Geerse, M. Roerdink, B. Coolen, J. Marinus, and J. van Hilten, "The interactive walkway: Towards assessing gait-environment interactions in a clinical setting," *Journal of Movement Disorders*, vol. 31, no. 2, Jun. 2016.

[24] L. Middleton, A. A. Buss, A. Bazin, and M. S. Nixon, “A floor sensor system for gait recognition,” in *IEEE Workshop on Automatic Identification Advanced Technologies (AutoID’05)*, Buffalo, NY, US, Oct 2005.

[25] K. Basterretxea, J. Echanobe, and I. del Campo, “A wearable human activity recognition system on a chip,” in *Proceedings of the 2014 Conference on Design and Architectures for Signal and Image Processing*, Oct 2014, pp. 1–8.

[26] M. Zubair, K. Song, and C. Yoon, “Human activity recognition using wearable accelerometer sensors,” in *in Proceedings of the 2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, Oct 2016, pp. 1–5.

[27] M. H. Kolekar and D. P. Dash, “Hidden markov model based human activity recognition using shape and optical flow based features,” in *in Proceedings of the 2016 IEEE TELCON Conference*, Nov 2016, pp. 393–397.

[28] M. Z. Uddin, T.-S. Kim, and J.-T. Kim, “A spatiotemporal robust approach for human activity recognition,” *International Journal of Advanced Robotic Systems*, vol. 10, no. 11, p. 391, 2013. [Online]. Available: https://doi.org/10.5772/57054

[29] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou, and Y. Amirat, “Physical human activity recognition using wearable sensors,” *MDPI Journal of Sensors*, vol. 15, no. 12, pp. 31 314–31 338, 2015. [Online]. Available: http://www.mdpi.com/1424-8220/15/12/29858

[30] S. Preece, J. Y Goulermas, L. Kenney, D. Howard, K. Meijer, and R. Crompton, “Activity identification using body-mounted sensors - a review of classification techniques,” vol. 30, pp. R1–33, 05 2009.

[31] O. Yurur, C. H. Liu, X. Liu, and W. Moreno, “Adaptive sampling and duty cycling for smartphone accelerometer,” in *in Proceedings of the 2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems*, Oct 2013, pp. 511–518.

[32] S. Schülein, J. Barth, A. Rampp, R. Rupprecht, B. M. Eskofier, J. Winkler, K.-G. Gaßmann, and J. Klucken, “Instrumented gait analysis: a measure of gait improvement by a wheeled walker in hospitalized geriatric patients,” *Journal of NeuroEngineering and Rehabilitation*, vol. 14, no. 1, Feb. 2017.

[33] S.-M. Lee, S. M. Yoon, and H. Cho, “Human activity recognition from accelerometer data using convolutional neural network,” in *Proceedings of the 2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Feb 2017, pp. 131–134.

[34] S. K. Dhar, M. M. Hasan, and S. A. Chowdhury, “Human activity recognition based on gaussian mixture model and directive local binary pattern,” in *IEEE International Conference on Electrical, Computer, and Telecommunication Engineering (ICECTE)*, Rajshahi, BD, Dec 2016.

[35] V. Ghasemi and A. A. Pouyan, “Human activity recognition in ambient assisted living environments using a convex optimization problem,” in *Proceedings of the 2016 IEEE International Conference of Signal Processing and Intelligent Systems (ICSPIS)*, Tehran, IR, Dec 2016.

[36] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou, and Y. Amirat, “Physical human activity recognition using wearable sensors,” *MDPI Journal of Sensors*, vol. 15, no. 12, pp. 31 314–31 338, 2015. [Online]. Available: http://www.mdpi.com/1424-8220/15/12/29858

[37] J. Taborri, E. Palermo, S. Rossi, and P. Cappa, “Gait partitioning methods: A systematic review,” *MDPI Journal of Sensors*, vol. 16, no. 1, 2016.

[38] S. D. Din, A. Hickey, S. Woodman, H. Hiden, R. Morris, P. Watson, K. Nazarpour, M. Catt, L. Rochester, and A. Godfrey, “Accelerometer-based gait assessment: Pragmatic deployment on an international scale,” in *Proceedings of the 2016 IEEE Statistical Signal Processing Workshop (SSP)*, June 2016, pp. 1–5.

[39] D. Trojaniello, A. Cereatti, and U. D. Croce, “Accuracy, sensitivity and robustness of five different methods for the estimation of gait temporal parameters using a single inertial sensor mounted on the lower

trunk," *Journal of Gait and Posture*, vol. 40, no. 4, pp. 487 – 492, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0966636214006432

[40] J. Mccamley, M. Donati, E. Grimpampi, and C. mazzà, "An enhanced estimate of initial contact and final contact instants of time using lower trunk inertial sensor data," *Journal of Gait and posture*, vol. 36, pp. 316–8, 03 2012.

[41] S. Jiang, B. Zhang, G. Zou, and D. Wei, "The possibility of normal gait analysis based on a smart phone for healthcare," in *Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, August 2013.

[42] D. Qin and M.-C. Huang, "A smart phone based gait monitor system," in *Proceedings of the 10th EAI International Conference on Body Area Networks*, 2015, pp. 32–38. [Online]. Available: http://dx.doi.org/10.4108/eai.28-9-2015.2261519

[43] M.-S. Lee, J.-G. Lim, K.-R. Park, and D.-S. Kwon, "Unsupervised clustering for abnormality detection based on the tri-axial accelerometer," in *Proceedings of the 2009 ICCAS-SICE*, Aug 2009, pp. 134–137.

[44] D. Reynolds, *Gaussian Mixture Models.* Boston, MA: Springer US, 2009, pp. 659–663. [Online]. Available: https://doi.org/10.1007/978-0-387-73003-5_196

[45] G. Xuan, W. Zhang, and P. Chai, "Em algorithms of gaussian mixture model and hidden markov model," in *Proceedings of the 2001 International Conference on Image Processing (Cat. No.01CH37205)*, vol. 1, 2001, pp. 145–148 vol.1.

[46] H. S Bhat and N. Kumar, "On the derivation of the bayesian information criterion," 01 2010.

[47] N. H. Chehade, P. Ozisik, J. Gomez, F. Ramos, and G. Pottie, "Detecting stumbles with a single accelerometer," in *Proceedings of the 2012 Annual*

International Conference of the IEEE Engineering in Medicine and Biology Society, Aug 2012, pp. 6681–6686.

[48] L. Wang, "Abnormal walking gait analysis using silhouette-masked flow histograms," in *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, 2006, pp. 473–476.

[49] K. Kaczmarczyk, A. Wit, J. Zaborski, J. Piłsudski, and M. Krawczyk, *Artificial Neural Networks (ANN) Applied for Gait Classification and Physiotherapy Monitoring in Post Stroke Patients.* IN-TECH Open Access Publisher, 2011. [Online]. Available: https://books.google.it/books?id=0_WqoAEACAAJ

[50] A. Sokolova and A. Konushin, "Gait recognition based on convolutional neural networks," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2/W4, pp. 207–212, 2017. [Online]. Available: https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-2-W4/207/2017/

[51] J. Hannink, T. Kautz, C. F. Pasluosta, K. G. Gaßmann, J. Klucken, and B. M. Eskofier, "Sensor-based gait parameter extraction with deep convolutional neural networks," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 1, pp. 85–93, Jan 2017.

[52] J. Hannink, T. Kautz, C. Pasluosta, J. Barth, S. Schulein, K. G. Gassmann, J. Klucken, and B. Eskofier, "Mobile stride length estimation with deep convolutional neural networks," *IEEE Journal of Biomedical and Health Informatics*, vol. PP, no. 99, pp. 1–1, 2017.

[53] Y. Feng, Y. Li, and J. Luo, "Learning effective gait features using lstm," in *Proceeding of the 2016 23rd International Conference on Pattern Recognition (ICPR)*, Dec 2016, pp. 325–330.

[54] F. J. Ordóñez and D. Roggen, "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition," *MDPI Journal of Sensors*, vol. 16, no. 1, 2016.

[55] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, M. Creatura, and J. d. R. Millàn, "Collecting complex activity datasets in highly rich networked sensor environments," in *Proceedings of the 2010 7th International Conference on Networked Sensing Systems (INSS)*, June 2010, pp. 233–240.

[56] P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Tröster, "Activity recognition from on-body sensors: Accuracy-power trade-off by dynamic sensor selection," in *Proceedings of the 5th European Conference on Wireless Sensor Networks*, ser. EWSN'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 17–33. [Online]. Available: http://dl.acm.org/citation.cfm?id=1786014.1786017

[57] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

[58] J.-K. Shiau, C.-X. Huang, and M.-Y. Chang, "Noise characteristics of mems gyro's null drift and temperature compensation," vol. 15, pp. 239–246, 09 2012.

[59] P. D. Welch, "The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms," *IEEE Journal of Transactions on audio and electroacoustics*, vol. 15, no. 2, pp. 70–73, 1967.

[60] S. D. Din, A. Godfrey, and L. Rochester, "Validation of an accelerometer to quantify a comprehensive battery of gait characteristics in healthy older adults and parkinson's disease: Toward clinical and at home use," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 3, pp. 838–847, May 2016.

[61] W. Shi, P. Shang, J. Wang, and A. Lin, "Multiscale multifractal detrended cross-correlation analysis of financial time series," *Physica A: Statistical Mechanics and its Applications*, vol. 403, no. Supplement C, pp. 35 – 44,

2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437114001241

[62] Y. Teng and P. Shang, "Detrended fluctuation analysis based on higher-order moments of financial time series," *Physica A: Statistical Mechanics and its Applications*, vol. 490, no. Supplement C, pp. 311 – 322, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437117308439

[63] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Journal of Transactions on Nuclear Science*, vol. 44, no. 3, pp. 1464–1468, Jun 1997.

[64] I. Mohamad and D. Usman, "Standardization and its effects on k-means clustering algorithm," *Journal of Applied Sciences, Engineering and Technology*, vol. 6, no. 17, pp. 3299–3303, 9 2013.

[65] G. W. Milligan and M. C. Cooper, "A study of standardization of variables in cluster analysis," *Journal of Classification*, vol. 5, no. 2, pp. 181–204, Sep 1988. [Online]. Available: https://doi.org/10.1007/BF01897163

[66] S. Raschka, "About feature scaling and normalization – and the effect of standardization for machine learning algorithms," 2014-07-11.

[67] E. Alpaydin, *Introduction to Machine Learning (Adaptive Computation and Machine Learning).* The MIT Press, 2004.

[68] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04. New York, NY, USA: ACM, 2004, pp. 116–. [Online]. Available: http://doi.acm.org/10.1145/1015330.1015332

[69] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms.* New York, NY, USA: Cambridge University Press, 2014.

[70] A. C. Rencher and W. F. Christensen, *Multivariate Regression.* John Wiley & Sons, Inc., 2012, pp. 339–383. [Online]. Available: http://dx.doi.org/10.1002/9781118391686.ch10

[71] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *Ann. Statist.*, vol. 36, no. 3, pp. 1171–1220, 06 2008. [Online]. Available: https://doi.org/10.1214/009053607000000677

[72] T. K. Ho, "Random decision forests," in *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ser. ICDAR '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 278–. [Online]. Available: http://dl.acm.org/citation.cfm?id=844379.844681

[73] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar 1986. [Online]. Available: https://doi.org/10.1007/BF00116251

[74] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[75] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[76] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001. [Online]. Available: http://www.jstor.org/stable/2699986

[77] "XGBoost github repository readme." https://github.com/dmlc/xgboost/blob/master/demo/README.md.

[78] S. Sathyanarayana, "A gentle introduction to backpropagation," 07 2014.

[79] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[80] M. Gadaleta and M. Rossi, “Idnet: Smartphone-based gait recognition with convolutional neural networks,” *CoRR*, vol. abs/1606.03238, 2016. [Online]. Available: http://arxiv.org/abs/1606.03238

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

[82] F. Chollet *et al.*, “Keras,” https://github.com/fchollet/keras, 2015.

[83] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[84] S. K. Kumar, “On weight initialization in deep neural networks,” *CoRR*, vol. abs/1704.08863, 2017. [Online]. Available: http://arxiv.org/abs/1704.08863

[85] T. Dozat, “Incorporating nesterov momentum into adam,” 2015.

[86] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” *CoRR*, vol. abs/1411.1792, 2014. [Online]. Available: http://arxiv.org/abs/1411.1792