Gonçalo Miguel Grenho Rodrigues

Bachelor Degree in Biomedical Engineering Sciences

Real-Time Step Detection Using Unconstrained Smartphone

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

Master of Science in
Biomedical Engineering

Adviser: Hugo Filipe Silveira Gamboa, Professor Auxiliar, Faculdade de Ciências e Tecnologia, Universidade NOVA de Lisboa

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

DRAFT: June 9, 2020

Real-Time Step Detection Using Unconstrained Smartphone

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

Acknowledgements

Without the support of many important people, I would not have been able to reach this important accomplishment in my academic and personal life.

First of all, I would like to thank Professor Hugo Gamboa for accepting me as his master thesis student and for allowing me to develop and work on this project, to develop myself and become a better Biomedical Engineer.

To Ricardo Leonardo, who support me through the whole development of my master thesis project, guiding and helping me every step of the way, I would like to express my deepest gratitude. I would've never been able to achieve and learn as much as I did without your support and knowledge and I'm really grateful for having you as my mentor. Thank you for all the support and patience invested in guiding me.

To *Associação Fraunhofer Portugal Research*, I would like to thank you for giving me everything I could possibly need, for all the support provided and specially for welcoming me into the company in the best possible way.

To BEST Almada, my home away from home where I grew, learned, discovered and developed an European mindset, knowledge, skills and, specially, friendships which I will carry with me for the rest of my life. I would like to thank all of the organization and specially the other 4 Sailors with whom I've been having the honor and pleasure of leading the organization with.

I want to thank my family, who have supported me and helped me deal with the hardships and frustrations which came along this path to become a Biomedical Engineer. My parents and my sister for all being there when I needed, for listening, for advising and for steering me towards success. I want to thank my grandfather, who always supported and motivated me endlessly to work harder and harder every day. Thank you for making all for making me the man I am today.

I want to thank my childhood friends, with whom I've been going through the challenges of university life and who have been with me every step of the way for the past decades. To the friends I made in FCT NOVA, who were always present, supported me and taught me many lessons in life, who spent many days and many nights besides me studying in VII or anywhere else, and helped me get through the toughest situations. To Sofia Gomes, who was there to listen to all the challenges I encountered and for guiding me towards the best path, either in life or work, and allowing me to become a person a better person every step of the way.

*"The moment you accept total responsibility for **everything** in your life is the day you claim the power the change **anything** in your life." - Hal Elrod*

Abstract

Nowadays smartphones are carrying more and more sensors among which are inertial sensors. These devices provide information about the movement and forces acting on the device, but they can also provide information about the movement of the user. Step detection is at the core of many smartphone applications such as indoor location, virtual reality, health and activity monitoring, and some of these require high levels of precision.

Current state of the art step detection methods rely heavily in the prediction of the movements performed by the user and the smartphone or on methods of activity recognition for parameter tuning. These methods are limited by the number of situations the researchers can predict and do not consider false positive situations which occur in daily living such as jumps or stationary movements, which in turn will contribute to lower performances.

In this thesis, a novel unconstrained smartphone step detection method is proposed using Convolutional Neural Networks. The model utilizes the data from the accelerometer and gyroscope of the smartphone for step detection. For the training of the model, a data set containing step and false step situations was built with a total of 4 smartphone placements, 5 step activities and 2 false step activities. The model was tested using the data from a volunteer which it has not previously seen.

The proposed model achieved an overall recall of 89.87% and an overall precision of 87.90%, while being able to distinguish step and non-step situations. The model also revealed little difference between the performance in different smartphone placements, indicating a strong capability towards unconstrained use. The proposed solution demonstrates more versatility than state of the art alternatives, by presenting comparable results without the need of parameter tuning or adjustments for the smartphone use case, potentially allowing for better performances in free living scenarios.

Keywords: Step Detection, Smartphone Sensors, Convolutional Neural Networks, Artificial Intelligence, Deep Learning

Resumo

Atualmente, os smartphones contêm cada vez mais sensores incorporados, entre os quais temos os sensores inerciais. Estes sensores fornecem informações sobre o movimento e as forças que atuam no dispositivo, mas também podem fornecer informações sobre o movimento do utilizador. A deteção de passos está na base de muitas aplicações para smartphones, como localização indoor, realidade virtual, análise de saúde e atividade, e algumas dessas aplicações exigem altos níveis de precisão.

Os métodos de deteção de passos existentes dependem muito da previsão dos movimentos executados pelo utilizador e pelo smartphone ou em métodos de reconhecimento de atividade, para ajuste de parâmetros. Estes métodos são limitados pelo número de situações que os investigadores conseguem prever e não consideram situações de falsos positivos que ocorrem na vida diária, como saltos ou movimentos estacionários.

Nesta dissertação, um novo método de deteção de passos sem restrições para smartphones é proposto usando Redes Neuronais Convolucionais. O modelo utiliza os dados do acelerómetro e do giroscópio do smartphone para deteção de passos. Para o treino do modelo, um data set contendo situações de passos e de falsos passos foi construído com um total de 4 colocações do smartphone, 5 atividades de passos e 2 atividades de falsos passos. O modelo foi testado usando os dados de um voluntário, que não foram sido incluídos no treino.

O modelo proposto alcançou uma recall de 89,87 % e uma precisão de 87,90 %, além de conseguir distinguir situações de passos e de falsos passos. O modelo também revelou pouca diferença entre o desempenho em diferentes posicionamentos do smartphone, indicando uma forte capacidade de uso irrestrito. A solução proposta demonstra mais versatilidade do que as alternativas existentes, apresentando resultados comparáveis sem a necessidade de ajustes de parâmetros para diferentes colocações do smartphone, potencialmente permitindo melhores desempenhos na normal utilização diária.

Palavras-chave: Detecção de Passos, Sensores Inerciais, Smartphone, Redes Neuronais convolucionais, Inteligência Artificial, Deep Learning

Contents

List of Figures

List of Tables

Acronyms

ACC	Accelerometer.
ANN	Artificial Neural Network.
BLSTM-RNN	Bidirectional Long Short Term Memory-Recurrent Neural Network.
CNN	Convolutional Neural Network.
DL	Deep Learning.
GC	Gait Cycle.
GT	Ground Truth.
GYR	Gyroscope.
IMU	Inertial Measurement Unit.
LSTM	Long Short Term Memory.
MAG	Magnetometer.
ML	Machine Learning.
PDR	Pedestrian Dead Reckoning.
ReLU	Rectified Linear Unit.
RNN	Recurrent Neural Network.

SDG Stochastic Gradient Descent.

SLE Step Length Estimation.

SSL Semi Supervised Learning.

SVM Support Vector Machine.

Tanh Hyperbolic Tangent.

Chapter 1

Introduction

1.1 Contextualization

Microelectromechanical systems (MEMS) have seen a rapid development and are now present in many of ubiquitous everyday devices such as smartphones. MEMS sensors such as accelerometers, gyroscopes and magnetometers, which constitute an Inertial Measurement Unit (IMU), can give us information about the position, orientation and the overall movement of the device and indirectly, of the user as well. These IMUs have been proven to be a reliable and cheap alternative to laboratory/industrial grade sensors [1, 2] and researchers have been using them in a wide number of fields of study, among which we have stride detection.

With this new and more accessible option, novel walk detection techniques have emerged in an attempt to correctly record the human gait. However, most of the approaches taken have been limited in some aspects such as constrained positioning of the device, the lack of detection of changes of pace, which in a free living situation cause them to fail to perform with high accuracy due to the high number of variables and challenges we encounter in our daily life when we are walking. Some applications of stride detection such as indoor location or gait analysis, require high accuracy and real-time use and current state of the art methods also lack the ability to accurately detect the moment of the step in real-time in some cases. As such, it has become increasingly necessary to find new solutions.

With this work, we propose a step detection algorithm that focus on improving existing smartphone-based step detection by making it impervious to errors derived from different device placements on the body, changes in user movement and false step situations and ensuring the correct functioning in real-time. These improvements will enhance the performance of many applications with step detection at their core, such as

indoor positioning or activity monitoring systems.

1.1.1 Applications

- **Medical Gait Analysis**

  The use of IMU in the study of gait parameters has seen extensive research and applications [3–6]. The study of the human gait is most of the times limited to hospitals or researches centers, where laboratory grade sensors are available. However, the analysis is most of the time limited to the daily life situations the medical professionals can recreate.

  F. Proessla *et al.* [1] extracted gait parameters using a smart device attached to the patient's ankle. The gait parameters were extracted using peak detection in the accelerometer signal. These parameters showed high correlation with results from a validated medical sensor, the APDM Opal. Robust step detection with our everyday smartphone will allow medical professionals to study the patients gait, in free living situations where we encounter the most variability of movements and obstacles, and make better informed diagnosis.

- **Indoor Location**

  Step detection is a major part of step detection algorithms using Pedestrian Dead Reckoning (PDR) [7–11]. PDR is one method used for indoor location where the step length and heading of the user are extracted from the IMU data and used to predict the movement in space. In order to have a high localization accuracy it is necessary to have little to no step misdetection. Accurate indoor location using a smartphone allow for inexpensive location capabilities that can be applied in emergency services, augmented reality, geofencing, worker location in factories or hospitals and as alternative for GPS location services.

1.2 State Of The Art

Over the last decades, many methods have appeared trying to record and analyse the human gait. Video recording of patients walking, force plate measurements, IMUs and other wearable sensors [5] have been employed to the study of human locomotion. With the appearance of smartphones with embedded IMUs, many researchers started to use them in this study and also found new ways of applying this knowledge of the human gait. Step counters and activity monitoring systems are now some of the most common applications we find on our smartphones. The lack of accuracy indoors of GPS location also prompted researchers to look for alternatives for indoor location and found smartphones sensors to be a reliable alternative [7–16]. Nowadays, more and more applications need highly accurate and mobile step detection systems. As algorithms with step detection in

their core continue to grow in complexity, it has become detrimental that these become more precise in order to reduce cumulative errors.

We chose to divide current state of the art algorithms in three main categories, based on the methods used, which are parameter based methods, machine learning methods and deep learning methods.

1.2.1 Parameter Based Methods

One of the most experimented approach are the parameter based methods. These use methods such as Peak and Valley Detection [7, 12, 17], Zero Crossing [18] and Short Term Fourier Transform (STFT) [19] which require manual tuning of the parameters in order to get optimal results. Lee *et al.* [17] proposed to solve the day to day walking variability through a peak and valley detection method with adaptive thresholds. Using the accelerometer signal, the peak and valley detection thresholds are update using the distance in time and the magnitude of the last pair of peak and valley. Using this method, an average accuracy of 99.3% was obtained for a combination of 7 smartphone placements and 3 manners of walking. Poulose *et al.* [7] used a different approach. They applied a sensor fusion algorithm in order to obtain pitch-based step detection algorithm. Then a peak and valley detection was made with temporal and magnitude thresholds. This approach was able to produce a 3,14% error rate. This step detection method was only tested for walking and handheld position. Khedr and El-Sheimy [12] applied peak and valley detection to the acceleration norm to perform step detection. A verification of the detected peaks and valleys was also done using the magnetometer and gyroscope signal in order to compensate variations in the amplitude and frequency of the signal caused by shifts in the user movements speed and smartphone movement. An accuracy of 99.5% was achieved in free walking manner where the smartphone position and the user speed vary. This method provided a more extensive step detection solution with a wide number of walking manners and smartphone positions.

These methods provide a good estimate at step detection but they are limited by the number of positions and movements the researchers can predict. Situations such as climbing up and down stairs and false step situations were not accounted for and they can result in an increase of the error of the algorithms. Also, errors resulting from the changes in position of the smartphone and of the users movement speed are not accounted for. As a possible solution, Kang *et al.* [19] proposed a frequency based algorithm for walk detection and step counting. They adaptively select the most sensitive axis of the gyroscope and extract the most prominent frequency of the users movement using a sliding time window and Fast Fourier Transform. The steps are then counted by multiplying the frequency with the elapsed time. This proposed algorithm exhibits a good performance, with a 95,76% accuracy, and is able to distinguish between walking a non walking activities such as standing and typing.

1.2.2 Machine Learning Methods

Many approaches join the conventional parameter based methods with classification methods such as Support Vector Machines(SVMs) [8, 9, 13], Artificial Neural Networks (ANNs) [10] and Decision Trees [20] in order to increase accuracy and reduce misclassification errors.

Support Vector Machines are often used in device pose recognition and user movement classification due to its high performance [8]. Park *et al.* [13] proposed an algorithm for walk detection using a SVM. The SVM is used to detect the placement of the device by selecting features from the obtained tri-axial accelerometer and gyroscope data, increasing the accuracy of the step counting procedure. The SVM exhibited an overall accuracy of 97.32% in device placement detection and the step detection showed a 97.09% overall accuracy. Zhang *et al.* [8] designed a step counting algorithm supported by a step mode and device pose detection. Two classifiers were used for device pose and step mode recognition, a Artificial Neural Network (ANN) and a SVM. The result of the classifiers will influence the cut-off frequencies of a band pass filter in place to prevent undercounting and overcounting. This method is able to detect step with an accuracy of 98%. Rodríguez *et al.* [9] developed a peak and valley step supported by an ensemble of Support Vector Machines and a Bayesian step mode probability. The support vectors machines and Bayesian probability worked together to produce coherent and correct step mode classification and reduce false classification of strides.

Liu *et al.* [10] proposed a step detection method using a feed forward ANN. The proposed solution uses 5 layers and a tanh activation function with Stochastic Gradient Descent (SGD) optimizer. The ANN was able to almost perfectly detect the users steps. The ANN was tested in three different data sets, belonging to the signals of three different individuals walking a total of 80 steps. The ANN results only exhibited one false positive on the third data set and managed to count the exact number of steps correctly in the other two. This shows the applicability of ANNs for step detection, however the proposed method only tested for walking procedure and in a limited manner.

1.2.3 Deep Learning Methods

Deep Learning has risen in importance in the last few years due to its capacity to learn from the data without the need of manually defining parameters and thresholds and its versatility to many different areas of study [21–26].

In step detection and step length estimation for Pedestrian Dead Reckoning (PDR), Edel and Koppe [14] implemented a Bidirection Long Short Term Memory-Recurrent Neural Network (BLSTM-RNN). The model is able to work with the raw data of the smartphones IMU. Using this implementation they were able to obtain a high step detection accuracy (1.48% mean error) for several smartphone placements and walk manners. This approach however does not take into account false step situations and other walking activities such as climbing stairs, which often compromises the accuracy of the step

detection. Also, a BLSTM-RNN makes real-time implementation not possible.

Steinmetzer *et al.* [3] proposed a novel step detection method using an insole with sensors and Convolutional Neural Networks (CNNs) to detect and compare steps of healthy patients and patients with Parkinson Disease. The proposed model used 2 convolutional layers with a Rectified Linear Unit (ReLU) activation function, 2 max pooling layers and 1 dense layer in the end with sigmoid function in order for the output of the model to be a step probability. The network was able to detect the steps and return the beginning and end marks of the steps with an F1-score of 0.974 for daily living activities and 0.939 for the healthy patients and 0.938 for Parkinson Disease patients F1-score in the Time Up and Go Test.

1.2.4 Discussion

As seen, a large variety of different approaches and techniques have been employed in the past in an attempted to detect steps of the smartphone user. However, most algorithms focuses on specific tasks and movements, making adjustments in order to increased the accuracy in those cases. During an ordinary day, a person walks, runs, walks upstairs, downstairs, and makes a series of different other movements which can't all be predicted and adjusted to, in order to have the highest accuracy possible. In these scenarios step detection algorithms which focuses on specific tasks and smartphone placements will have a lower performance and in application such as indoor location or medical monitoring application, this can have a negative effect. As such, a real-time solution which can correctly detect each step with high accuracy and also dismiss false step situations is needed.

1.3 Objectives

With this work we propose a novel step detection method which can perform with high accuracy in free living situations and regardless of the device pose. We aim to make the automatic step detection proposal impervious to false step situations and other variables encountered in day to day situations which can be applied for real-time use. This in turn, will allow for better and more accurate applications such as indoor location which requires high accuracy of step detection and reduce cumulative errors.

1.4 Thesis Overview

This thesis contains 6 Chapters. The first chapter includes all the contextualization, state of the art and objectives of the dissertation. In the second chapter, all the theoretical concepts necessary to understand the work develop are explained. In the third chapter, an explanation of the data collection procedure for the training and testing of the proposed model is presented. In the fourth chapter, the data pre-processing and the final model

and the post processing methods are explained in detail. In the fifth chapter, the results and their discussion are presented. Finally, in the sixth chapter, the conclusions gathered from this work are presented along with future steps.

Chapter 2

Theoretical Background

In order to fully understand the work behind the applied step detection algorithm, some information of the theoretical principles and the technologies behind it are presented, beginning with the physical principles behind the sensors used, a basic introduction to gait cycle and human locomotion, knowledge about data frame of reference rotation, followed by machine learning principles, and finalizing with the functioning and intricacies of artificial neural networks, from the basic mathematical principles all the way to learning mechanisms and some of the different kinds of neural networks existent at the moment and of relevance to this work.

2.1 Sensors

Inertial sensors detect forces resulting from motion. These forces act on inertial masses within the sensors core, without needing any contact with an outside medium. In smartphones, the IMUs used are Micro Eletromechanical Sensors (MEMS). The mechanical movement of a core mass, due to the applied external force, produces an electrical signal that is then converted into an accurate measurement of the force. An Inertial Measuring Unit is typically composed of two inertial sensors: a triaxial accelerometer and a triaxial gyroscope. It is also frequent for an IMU to have a triaxial magnetometer.

2.1.1 Accelerometer

An accelerometer is an inertial sensor that measures the acceleration resulting from external forces. The measurement is conventionally in meters per second squared (m/s^2). The accelerometers present in the smartphones are usually MEMS capacitive accelerometers. In capacitive sensors the core mass is located between fixed beams and a shift in the seismic mass produces a change in the capacitance between these fixed beams. This change

in capacitance is converted into an accurate value of the acceleration.

2.1.2 Gyroscope

Gyroscopes are inertial sensors that measure angular velocity, usually in radians per second (rad/s). They have vibrating elements to detect the angular velocity. MEMS Gyroscopes are based on the transfer of energy between two vibration modes caused by the Coriolis acceleration. They don't have rotating parts which makes its miniaturization easier with MEMS construction techniques [27].

2.1.3 Magnetometer

Most magnetometers use the Hall Effect to obtain information about the magnetic field. The Hall Effect establishes that, in a metal sheet where a current I is being conducted, if a magnetic field is applied, the charges will distribute in reaction to that magnetic field resulting in a voltage. The voltage created is proportional to the applied magnetic field [28].

2.2 Gait Analysis

Gait is defined as the manner of locomotion of the human body. Gait analysis is the quantitative measurement and assessment of human locomotion including both walking and running. The gait cycle is defined as the period from the point of initial contact (also referred to as foot contact) of the subjects' foot with the ground to the next point of initial contact of the same limb. The gait cycle can be divided into two phases, the stance phase and the swing phase, which are portrayed in Figure 2.1. In the stance phase the analyzed foot is in contact with the ground and in the swing phase the foot is off the ground. During each phase we can also define 2 kinds of time intervals, single support and double support, which describes the period where the body is supported by one or two limbs respectively.

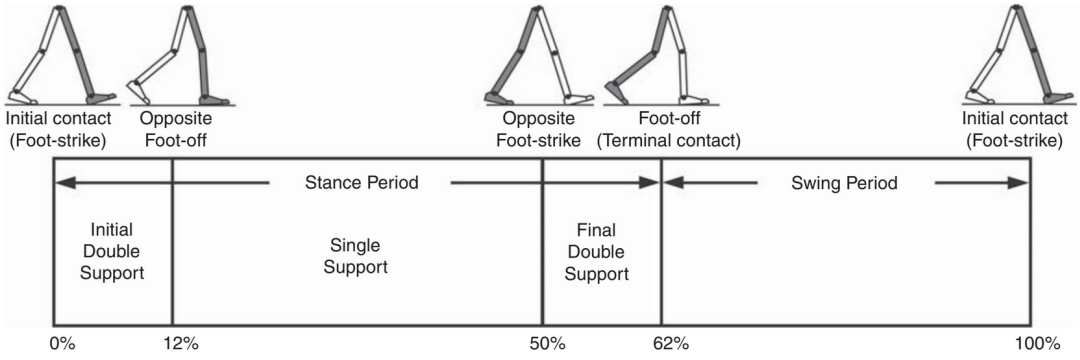


Figure 2.1: Gait cycle temporal division. Retrieved from [29]

The gait cycle can also be divided into 8 functional phases, known as the Rancho classification [29], related to the movement of the leg during the one stride:

1. **Initial contact** - when the foot touches the ground, starting with the contact of the heel with the ground (heel strike);
2. **Loading Response** - weight acceptance and double support period. Opposite limb ending stance phase and starting the initial swing;
3. **Mid-stance** - the foot is well placed in the ground to support the movement of the opposite limb that is in the swing phase (single support period);
4. **Terminal Stance** - the opposite limb is in the end of the swing phase (ends with initial contact of the opposite limb);
5. **Pre-swing** - the analyzed limb starts to leave the ground. Final double support period;
6. **Initial swing** - begins the moment the foot stops being in contact with the ground and starts the swing. Characterized by highest knee flexion;
7. **Mid-swing** - second third of the swing. Begins in the end of the initial swing and ends in the begging of the terminal swing. Characterized by maximum hip flexion;
8. **Terminal swing** - last third of the period of the swing. Ends with initial contact of the foot [29].

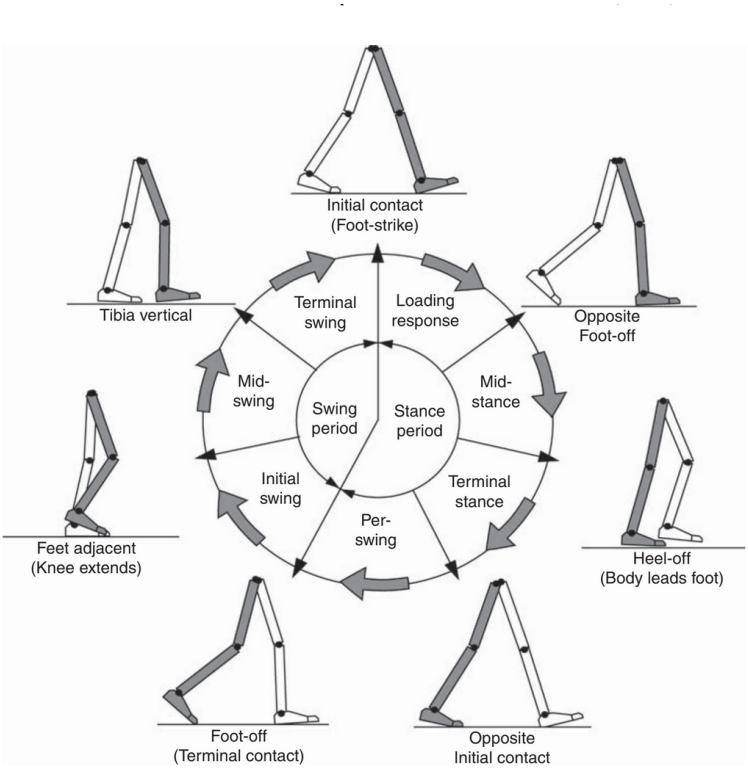


Figure 2.2: Gait cycle Rancho classification. Retrieved from [29]

There are many parameters that are extracted from the observation and analysis of gait cycle. Some gait parameters with high relevance in step detection are:

- **Cycle time:** number of cycles per unit of time;
- **Cadence:** number of steps per unit of time;
- **Stride length:** distance covered per step;
- **Velocity:** stride length/cycle time; [30]

2.3 Quaternions

Quaternions are a 4 dimensional complex numerical system created by William Rowan Hamilton in 1843. Quaternions have high application for orientation and heading algorithms since they do not suffer from problems such as gimble lock present in Euler angles. Gimble lock is experienced when pitch angle approaches +-90 degrees and prevents a correct orientation measurement.

A quaternion is represented by a real quantity q_0 and 3 complex numbers i, j and k:

$$q = q_0 + iq_1 + jq_2 + kq_3 \tag{2.1}$$

These complex numbers follow these relationships:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{2.2}$$

$$ij = i \times j = k = -j \times i = -ji \tag{2.3}$$

$$jk = j \times k = i = -k \times j = -kj \tag{2.4}$$

$$ki = k \times i = j = -i \times k = -ik \tag{2.5}$$

2.3.1 Quaternion Rotation

Any unit quaternion q may be written as

$$q = q_0 + \mathbf{q} = \cos\frac{\theta}{2} + u \sin\frac{\theta}{2}$$

where

$$\mathbf{q} = iq_1 + jq_2 + kq_3, \qquad u = \frac{\mathbf{q}}{|\mathbf{q}|} \qquad and \qquad \tan\frac{\theta}{2} = \frac{|\mathbf{q}|}{q_0} \tag{2.6}$$

Theorem For any unit quaternion

$$q = q_0 + \mathbf{q} = \cos\frac{\theta}{2} + u \sin\frac{\theta}{2} \tag{2.7}$$

and for any vector $v \in \mathbb{R}$ the action of the operator

$$L_q(v) = qvq^* \tag{2.8}$$

on v may be interpreted geometrically as a rotation of the vector v through an angle θ about q axis of rotation [31]. In Figure 2.3, we can see a representation of this rotation.

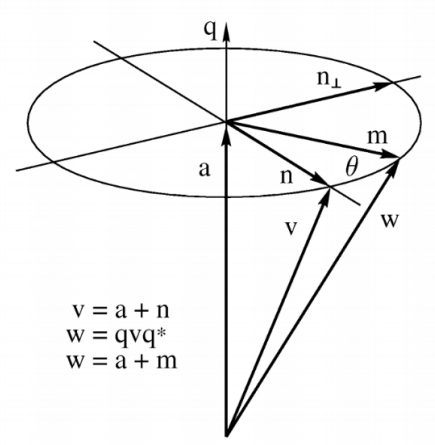


Figure 2.3: Quaternion rotation of vector v through an angle θ around the q axis of rotation, resulting in vector w. Adapted from [31].

2.4 Sensor Fusion

Sensor Fusion is the process of combining the information captured from multiple sensors in order to increase the accuracy of the measurement. In the case of the smartphone sensors, the information from the accelerometer, gyroscope and magnetometer is often used to rotate the reference frame of the measurements to one more adequate to the interpretation and processing of data, thus resulting in better data for analysis and use in many applications.

2.4.1 Complementary Filter

Complementary filters are applied to the smartphones IMU to compensate for the accelerometer's high frequency noise and the gyroscopes low frequency drift in attitude estimation. They are also capable of dealing with artefacts resultant from magnetic interferences. The complementary filter acts both as a low pass filter for the accelerometers data and a high pass filter for the gyroscopes data [32]. The filter can be formally represented as

$$q_t = \alpha * (q_{t-1} + q_{gyr}) + (1 - \alpha) * q_{acc+mag} \tag{2.9}$$

$q \rightarrow$ quaternion representing the heading angle in degrees with respect to the reference frame

$q_{gyr} \rightarrow$ quaternion representing the contibution of the gyroscope for the final heading angle

$q_{acc+mag} \rightarrow$ quaternion representing the contribution of the accelerometer am magnetometer to the final heading angle

$\alpha = \frac{\tau}{\tau+dt} \rightarrow$ time-related constant which defines the boundary between the contributions of the accelerometer and the gyroscope signals. τ is time a constant which represents how fast we want our filter to respond. dt is the period between acquisitions ($\frac{1}{samplingfrequency}$)

In our situation, we want the filter to able to apply to all the possible step situations which can have a frequencies of up to 2Hz [19]. So a good value for α, with $dt = \frac{1}{100Hz}$ and $\tau = \frac{1}{2Hz}$, will be

$$\alpha = \frac{0.5}{0.5 + \frac{1}{100}} \approx 0.98 \tag{2.10}$$

The filter is also often used in the rotation of reference frames of the sensors in order to obtain more stable data.

2.4.2 Reference Frame

There are several reference frames applied to the analysis of the smartphones orientation and the acquisition of the sensors data. In some cases, a reference frame independent

of the smartphone position can provide an improvement in the stability of the collected data. This can be achieved through quaternion rotations and the data collected from the inertial sensors of the device.

2.4.2.1 Smartphone's Reference Frame

The reference frame of the smartphone is the reference frame of the IMUs it contains. Usually, the reference frame of the smartphone is as the one represented in Figure 2.4 where the X axis points towards right of the smartphone, the Y axis towards top and the Z axis is pointing upwards, perpendicular to the smartphones screen.

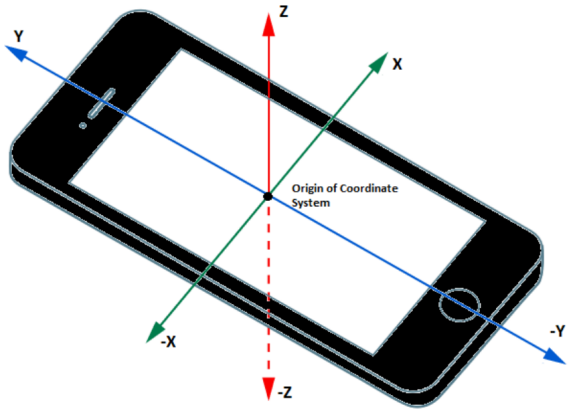


Figure 2.4: Smartphone reference system. Retrieved from [11].

2.4.2.2 Earths Reference Frame

The Earth Reference frame is a reference frame where the z-axis is in the vertical direction, perpendicular to the earths surface and in an upwards direction, the y-axis is in the direction of the magnetic north pole and the x-axis is obtained through the outer product of the z and y axis vectors. This reference frame can be obtained using the accelerometer data to define the vertical direction, *i. e.* the Z-axis and the magnetometer data to define the Y-axis, which points in the direction of the magnetic north pole.

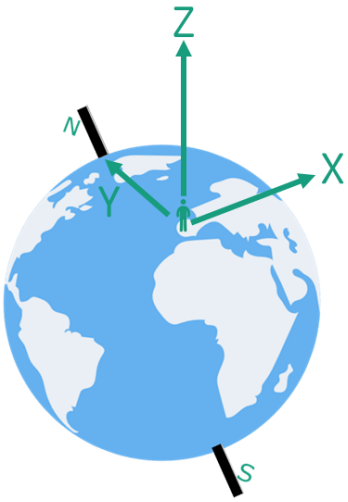


Figure 2.5: Earth coordinates reference system.

2.4.2.3 User Reference Frame

The user Reference Frame is a reference frame where the the z-axis is in the vertical direction, pointing from the centre of the earth, the y-axis is in the direction the user is facing and the x-axis is obtained with the outer product of the z and y axis. The Z-axis can be estimated using the accelerometer data and the y-axis is obtained using a heading algorithm.

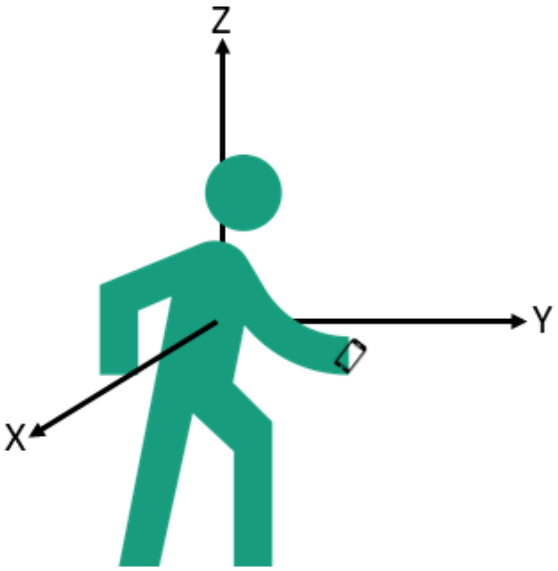


Figure 2.6: User reference system.

2.5 Machine Learning

Machine learning (ML) can be described as *a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty* [33]. ML has seen vast application in many fields of study and through time, a wide number of methods which have the ability to learn these patterns have emerged in many forms, varying in the type of action they perform and the way they learn. The learning process can occur in various manners and we can divide machine learning algorithms according to that.

2.5.1 Supervised Learning

In Supervised Learning we find algorithms which learn through the comparison of the output given by the ML algorithm and the target output, *i. e.* the algorithm is given a set of input and target output (x, y) and it learns through an error estimate calculated with the difference between the estimated output $(\hat{y})$ and the target output (y). Amongst Supervised Learning methods, we find Artificial Neural Networks, Decision Trees and Support Vector Machines.

2.5.2 Semi-supervised Learning

Semi-supervised learning (SSL) is a learning method between supervised and unsupervised methods. In some cases, the labelling of large data sets can be unfeasible due to the high amount of time and resources necessary. In these situations SSL can be of great use. In SSL algorithms a small amount of labeled data is mixed with the rest of the unlabeled data. This mixture assists the learning procedure greatly, which will result in higher accuracy results.

2.5.3 Unsupervised Learning

In Unsupervised Learning, the ML algorithms analyse the data and learn patterns according to the characteristics of data and not through reference values or targets. In Unsupervised Learning we find Hidden Markov Models, k-Means Clustering and Gaussian Mixture Models.

2.5.4 Reinforcement Learning

In Reinforcement Learning, an agent (*e.g.* a bot) tries to learn the sequence of actions which will result in the highest cumulative reward. The concept of reward is similar to the reward given to dogs when trying to teach them a command, you give them a treat every time they respond correctly to the command and in time they will respond correctly more often, resulting in a higher cumulative reward.

2.5.5 Evaluation Metrics

The evaluation of the results can be made through several metrics. In the case of a binary classification problem, and using our case as an example, we classify each result, in comparison with the ground truth data, as:

- **True Positive** - a step is correctly identified where there is a step
- **False Positive** - a step is incorrectly identified where there is no step
- **True Negative** - a step is not identified where a step was not taken
- **False Negative** - a step is not identified where one occurs

Accuracy It is a metric which measures the percentage of correct classifications (True Positives + True Negatives) that occurred. Accuracy, can be formally represented as

$$Accuracy = \frac{True\ Positives + True\ Negatives}{All\ Test\ Data} \tag{2.11}$$

Recall It's a percentage of correct classification ($True\ Positives$) which occured among all which should result in positive classifications ($True\ Positives + False\ Negatives$) and is formally represented as

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{2.12}$$

Specificity It's the rate of correct negative classifications, *i.e.* the True Negative Rate, and can be formally represented as

$$Specificity = \frac{True\ Negative}{True\ Negative + False\ Positive} \tag{2.13}$$

Precision It's the rate of True Positives among all positive results. Precision is formally represented as

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{2.14}$$

F-score In the evaluation of a classifiers performance, a F-score can be used. This F-score is the combination of precision and recall and can be formally represented as

$$F - score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} \tag{2.15}$$

2.5.6 Data Sets

In order to attain the best generalization from machine learning algorithms it is important to evaluate how the algorithm should learn from the data. Taking this into account, separation of the data into at least two different data sets, the training and testing data sets, is needed. This separation is important in order for the machine learning algorithm to learn properly from the provided information and to analyse how it deals with new information it has not seen before after the training procedure. It is important that these two data sets do not contain data in common in order to prevent issues such as overfitting. One further group of data can be employed, the validation data set, which is used in order to analyse the learning procedure into further detail.

Training Data Set The training data set is the group of data the model is going to use to learn from. This data set usually amounts for the highest percentage of the data in order to give the model to opportunity to learn as much as possible.

Validation Data Set The validation data set is used to analyse the training procedure of the model. This data set amounts for a small percentage of the training data, which is removed from the training data set. Through the analysis of the metrics from the validation data set throughout the training procedure, we can withdraw conclusions about the training of the model such as if it is overfitting, if some parameters need to be adjusted or even analyse what the best amount of training time for the model is.

Testing Data Set This data set is used to test the model after training. This data set has data that the machine learning model has not seen yet and through the testing data we can analyse if the resulting model can generalize well or if some adjustment is needed.

2.6 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a type of machine learning algorithm based on the functioning of the biological neural networks. In the human brain, a neuron receives information from other neurons. That information is then processed and sent to next neuron, and this happens until the information reaches its target. ANNs are a simplified version of the biological neural networks.

ANNs use layers of neurons. Inside each neuron, the N inputs (x_i) are processed and summed through linear combination with the neurons weight (w) and a bias ($bias$) parameters. Then, this value goes through an activation function (σ) which gives us the output of the neuron. One common activation function is the sigmoid function [34].

$$a = \sum_{i=0}^{N} x_i * w_i + bias \tag{2.16}$$

$$\sigma(a) = \frac{1}{1 + e^{-a}} \tag{2.17}$$

2.6.1 Activation Functions

Activation functions, as the name suggests, activate the neuron depending on the input values. These functions dictate the level of activity of the neuron through threshold values which, in term, will result in its output value. Several activation functions have been proposed through time in order to adapt neural networks to new challenges and applications and to improve its processing speed.

2.6.1.1 Linear

A linear activation function picks up the input multiplied by the weight and the added bias values and outputs a value proportional to it. This activation function allows for multiple output values which are linearly proportional to the input.

$$\sigma(a) = constant * a \tag{2.18}$$

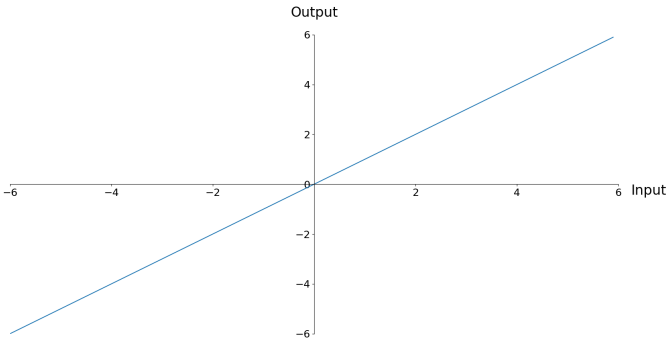


Figure 2.7: Linear activation function

2.6.1.2 Sigmoid or Logistic function

The sigmoid or logistic function is mathematically described by

$$\sigma = \frac{1}{1 + e^{-a}} \tag{2.19}$$

This function has output values in the interval [0, 1] and for that reason has seen multiple applications in fields of study where a probabilistic output of the neural network is needed [35].

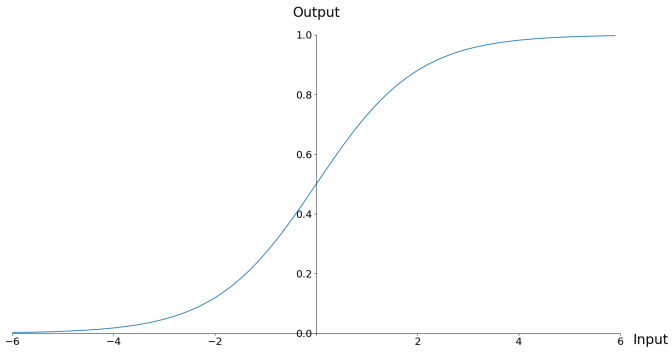


Figure 2.8: Sigmoid activation function

2.6.1.3 Rectified Linear Unit (ReLU)

The Rectified Linear Unit or ReLU, is an activation function which only activates with positive input values and exhibits a linear behaviour. Its commonly used in convolutional neural networks due to allowing faster convergence speeds, helping to prevent exploding or vanishing gradients and for its computational efficiency [36].

$$\sigma(a) = \begin{cases} a, & \text{if } a \geq 0 \\ 0, & \text{if } a < 0 \end{cases} \tag{2.20}$$

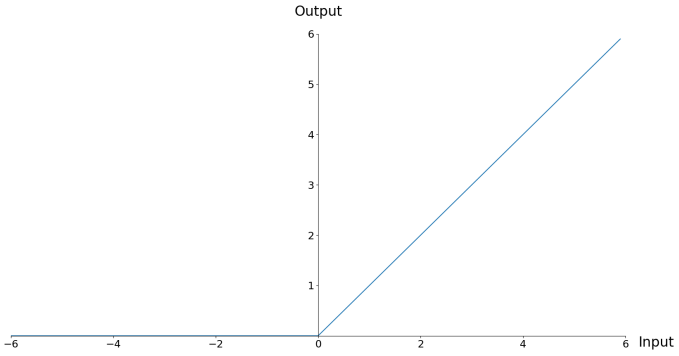


Figure 2.9: Rectified Linear Unit (ReLU)

2.6.1.4 Softmax

Softmax is an activation function used to give a probabilistic output in multiple class classification problems. This activation function is applied in the next to last layer of the network in order to transform the non-normalized output of the neurons into a probability value and the sum of the resulting probabilities assigned to each class are equal to 1. This activation function is applied in classification problems with 3 or more classes [34].

$$p_i = \frac{e^{a_i}}{\sum_{k=0}^{N_k} e^{a_k}} \tag{2.21}$$

$a_i \rightarrow$ input label for the label used to calculate the loss
$N \rightarrow$ Number of input values
$p_i \rightarrow$ Predicted probability value

2.6.2 Loss/Cost Function

In this work we will define loss functions as a mathematical expression used to give a scalar value to the error between the desired output and the estimated one of the training data of the network in 1 epoch.

These functions play a very important role in the learning process of neural networks, since the gradients of the loss functions are used in backpropagation to adjust the parameters of the network.

2.6.2.1 Mean Squared Error (MSE)

This loss function is frequently used in regression problems using neural networks.

$$Mean\ Squared\ Error = \frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - t_i)^2 \tag{2.22}$$

$N \rightarrow$ Size of training data
$t_i \rightarrow$ Target output
$\hat{y}_i \rightarrow$ Predicted output

2.6.2.2 Cross-Entropy

Cross-entropy loss function is used in classification models where the output is expected to be binary.

Binary Classification In the cases where we have a binary classification problem, *i. e.* we only have two classes as possible outputs, *e. g.* dog/not dog, the cross-entropy loss can be calculated as

$$Loss = -(t \cdot \log(\hat{y}) + (1-t) \cdot \log(1-\hat{y})) \tag{2.23}$$

$t \rightarrow$ Target output
$\hat{y} \rightarrow$ Predicted output

Multiple Classes When we have 3 or more classes, loss in given by the sum of the loss values of each individual class

$$Loss = -\sum_{i=0}^{N} t \cdot \log(\hat{y}) \tag{2.24}$$

$N \rightarrow$ Number of classes

$t \rightarrow$ Target output for the class
$\hat{y} \rightarrow$ Predicted output

2.6.3 Backpropagation

In order for the network to learn, an adjustment of the neurons parameters has to be made in an amount proportional to the loss. This is done through backpropagation of errors through the network. The calculation of the change needed to the parameters starts in last layer (output layer) and works backwards in order to compute how they should alter in order to improve the performance of the network, *i.e.* minimize the loss function.

We retrieve information about the changes the network needs to receive through the gradient of the loss function. We know that loss, assuming we are using equation 2.22, is

$$Loss = \frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - t_i)^2 \tag{2.25}$$

and that the output value of our network are given by

$$\hat{y}_i = \sigma(a_i^{L-1} * w_i^L + bias_i^L) \tag{2.26}$$

with σ representing the activation function, a_i^{L-1} representing the output of the previous neuron, w_i^L representing the weight assigned, $bias_i^L$ representing the bias value of the neuron and with the superscript L representing the number of the layer of the neuron. The gradient of the loss will be given by

$$Loss = \frac{1}{N}\sum_{i=0}^{N}\nabla(\hat{y}_i - t_i)^2 = \frac{1}{N}\sum_{i=0}^{N}\frac{\partial Loss}{\partial \hat{y}_i} \tag{2.27}$$

This gradient of the loss tells us the direction of highest increase of the loss function. We're are looking to minimize it so we need the negative value of the gradient. This minimization of the loss function is called **Gradient Descent**.

To figure out how the parameters of the network affect the gradient, *i.e.* the partial derivatives of the Loss with respect to each weight and bias of the network, we start by analysing the output layer. In order to simplify the explanation, we will consider a ANN with only one neuron in each layer and we will represent the linear combination of the input of the neuron with the parameters $(a^{L-1} * w^L + bias^L)$ as z^L resulting in

$$\hat{y} = \sigma(z^L) \tag{2.28}$$

We want to know the impact that changing the weights of the neuron will make to the loss function and in which direction. We know changing the weight will cause a change in the value of the function z^L. That change can be represented by

$$\frac{\partial z^L}{\partial w^L} = a^{L-1} \tag{2.29}$$

This shift in z^L will then cause a shift in the value of our output $\hat{y}$ which can be represented as

$$\frac{\partial \hat{y}}{\partial z^L} = \sigma\,'(z^L) \tag{2.30}$$

and as a consequence, this shift in the output $\hat{y}$ will cause a change in the loss value which can be expressed as

$$\frac{\partial Loss}{\partial \hat{y}} = 2 * (\hat{y} - t) \tag{2.31}$$

Through the *chain rule*, we can then express the derivative of the loss function with respect to the weight as

$$\frac{\partial Loss}{\partial w^L} = \frac{\partial z^L}{\partial w^L} \frac{\partial \hat{y}}{\partial z^L} \frac{\partial Loss}{\partial \hat{y}} \tag{2.32}$$

The partial derivative of the Loss function in respect with the weight of the neuron will be

$$\frac{\partial Loss}{\partial w^L} = a^{L-1} \sigma\,'(z^L) 2 * (\hat{y} - t) \tag{2.33}$$

The chain rule is applicable to the calculation of the partial derivatives of the loss of the rest of the network. In the case of a neuron in the second to last layer, the change in the weight of the neuron would change the value of z^{L-1} represented by

$$\frac{\partial z^{L-1}}{\partial w^{L-1}} \tag{2.34}$$

the change in z^{L-1} would shift a^{L-1} in the manner

$$\frac{\partial \hat{a}^{L-1}}{\partial z^{L-1}} \tag{2.35}$$

the shift in the output of the neuron will change the value of z of the following neuron in the manner

$$\frac{\partial \hat{z}^L}{\partial a^{L-1}} \tag{2.36}$$

and these changes will influence the loss value through the chain rule, resulting in

$$\frac{\partial Loss}{\partial w^{L-1}} = \frac{\partial z^{L-1}}{\partial w^{L-1}} \frac{\partial \hat{a}^{L-1}}{\partial z^{L-1}} \frac{\partial \hat{z}^L}{\partial a^{L-1}} \frac{\partial \hat{y}}{\partial z^L} \frac{\partial Loss}{\partial \hat{y}} \tag{2.37}$$

To study the influence of neurons in subsequent layers, we just need to follow the same procedure. This process allows us to obtain the partial derivatives we find in the gradient of the Loss function and as such, find the amount and direction of the adjustment the neurons need to improve performance.

For the realistic situation, where we have multiple neurons in each layer, lets use the network in figure 2.10 as an example. In this case, the neurons in the second to last layer

will influence loss through all the neurons in the subsequent layer, so we need to take this into account in the calculation of the derivative of the first weight of the neuron. This can be formally expressed by

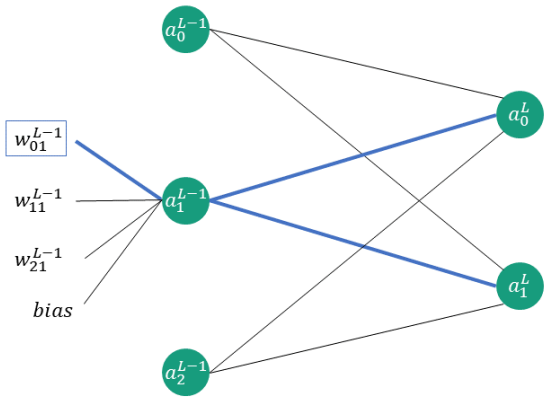


Figure 2.10: The weight of the neuron will impact the loss value through all the neurons it is connected to.

$$\frac{\partial Loss}{\partial w_{01}^{L-1}} = \frac{\partial z_1^{L-1}}{\partial w_{01}^{L-1}} \frac{\partial \hat{a}_1^{L-1}}{\partial z_1^{L-1}} \frac{\partial \hat{z}_0^L}{\partial a_1^{L-1}} \frac{\partial \hat{y}_0}{\partial z_0^L} \frac{\partial Loss}{\partial \hat{y}_0} + \frac{\partial z_1^{L-1}}{\partial w_{01}^{L-1}} \frac{\partial \hat{a}_1^{L-1}}{\partial z_1^{L-1}} \frac{\partial \hat{z}_1^L}{\partial a_1^{L-1}} \frac{\partial \hat{y}_1}{\partial z_1^L} \frac{\partial Loss}{\partial \hat{y}_1} \tag{2.38}$$

which can be simplified as

$$\frac{\partial Loss}{\partial w_{01}^{L-1}} = \frac{\partial z_1^{L-1}}{\partial w_{01}^{L-1}} \frac{\partial \hat{a}_1^{L-1}}{\partial z_1^{L-1}} \sum_{i=0}^{N_L} \frac{\partial \hat{z}_i^L}{\partial a_1^{L-1}} \frac{\partial \hat{y}_i}{\partial z_i^L} \frac{\partial Loss}{\partial \hat{y}_i} \tag{2.39}$$

and the expressing inside the sum is the various contributions of the output of the neurons to the loss through the neurons in the subsequent layer, resulting in

$$\frac{\partial Loss}{\partial w_{01}^{L-1}} = \frac{\partial z_1^{L-1}}{\partial w_{01}^{L-1}} \frac{\partial \hat{a}_1^{L-1}}{\partial z_1^{L-1}} \sum_{i=0}^{N_L} \frac{\partial Loss}{\partial a_1^{L-1}} \tag{2.40}$$

This process can be applied to all neurons in the network.

2.6.3.1 Learning Rate

Learning rate is a parameter used to adjust how fast we want the network to learn, **i.e.** how fast we want to lower the loss value and adjust the weights and biases of the network. The learning rate parameter is applied at the moment of update of the weights and biases in the following manner

$$w_{t+1} = w_t - \alpha \frac{\partial Loss_t}{\partial w_t}, \qquad \alpha \rightarrow learning\ rate \tag{2.41}$$

Learning Rate Decay When we are nearing the convergence, it's more useful to start reducing the amount of the changes performed through backpropagation. This is possible through a reduction of the learning rate through time/epoch and is called learning rate decay. Learning rate decay will reduce the learning rate each epoch and allow for smaller and smaller adjustments as training progresses. It can be formally represented as

$$\alpha = \frac{1}{1 + DecayRate * epoch} \cdot \alpha \tag{2.42}$$

2.6.4 Network Optimization

The optimization functions are used in order to bring the neural network to a minimum value of loss in the most efficient manner possible. They act in the moment of the update of the parameters of the network in an attempt to optimize the process. Many optimization algorithms have been proposed through time trying to improve several aspects of the learning process of ANNs and for many applications.

2.6.4.1 AdaGrad

Adaptive Gradient algorithm or AdaGrad is an optimization algorithm proposed by *Duchi et al.* [37] which incorporates knowledge from previous iterations to improve the optimization process. It's mathematically represented by

$$w_{t+1} = w_t - \alpha * \frac{1}{\sqrt{\varepsilon I + diag(G_t)}} \cdot \frac{\partial Loss_t}{\partial w_t} \tag{2.43}$$

$\alpha \rightarrow$ learning rate
$\varepsilon \rightarrow$ constant
$I \rightarrow$ Identity matrix
$G_t \rightarrow$ sum of the outer products of the gradient until t
$Loss_t \rightarrow$ Loss at time t

2.6.4.2 RMSProp

RMSProp is an adaptive learning rate optimization method. It uses the moving average of the squared gradient of the weight in order to adaptively modify the learning rate. This dampens the oscillations of the loss function during optimization and allows for faster convergence. It is formally represented by

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{E[g^2]_t}} \frac{\partial Loss}{\partial w_t} \tag{2.44}$$

$$E[g^2]_{t+1} = \beta E[g^2]_t + (1 - \beta)(\frac{\partial Loss}{\partial w_t})^2 \tag{2.45}$$

$E[g^2] \rightarrow$ moving average of squared gradients
$\alpha \rightarrow$ learning rate

$w \rightarrow$ weight
$\beta \rightarrow$ moving average constant

2.6.4.3 Adam - Adaptive Moment Estimation

Adam is a optimizer proposed by [38] and combines the strong point of both AdaGrad, which works well with sparse gradients, and RMSProp which works well on online and non-stationary settings. The algorithm is formally represented by

$$w_{t+1} = w_t - \alpha \cdot \frac{\hat{m}_t + 1}{\sqrt{\hat{v}_t + 1} + \varepsilon} \tag{2.46}$$

$\alpha \rightarrow$ learning rate
$w \rightarrow$ weight
$m_{t+1} = \beta_1 \cdot m_t + (1 - \beta_1) \cdot g_t \rightarrow$ momentum estimation
$v_{t+1} = \beta_2 \cdot v_t + (1 - \beta_2) \cdot g_t^2 \rightarrow$ RMSProp momentum estimation
$g_t \rightarrow$ gradient
$\hat{m}_{t+1} = \frac{m_{t+1}}{(1-\beta_1^{t+1})} \rightarrow$ momentum bias correction
$\hat{v}_{t+1} = \frac{v_{t+1}}{(1-\beta_2^{t+1})} \rightarrow$ RMSProp momentum bias correction
$\beta_1 \rightarrow$ momentum moving average constant
$\beta_2 \rightarrow$ RMSProp moving average constant

2.6.5 Network Regularization

In order to prevent overfitting problems, regularization techniques emerged and are a very important part of the development of neural networks.

2.6.5.1 L2 and L1 Regularization

These regularization techniques place a penalty value in the calculation of the loss and prevent the network from making strong assumptions, *i.e.* prevents the networks weights from reaching high values.

L2 Regularization In the case of the L2 regularization, also called Ridge Regularization, we have

$$Loss = Loss + \frac{\lambda}{2N} \sum \|w\|^2 \tag{2.47}$$

$\lambda \rightarrow$ regularization parameter
$\|w\|^2 \rightarrow$ euclidean norm of the weight squared
$N \rightarrow$ number of parameters

This method reduces some parameters making them smaller, helping reduce the attempt to learn specific information such as noise.

L1 Regularization In case of the L1 regularization, also called Lasso Regularization, we have

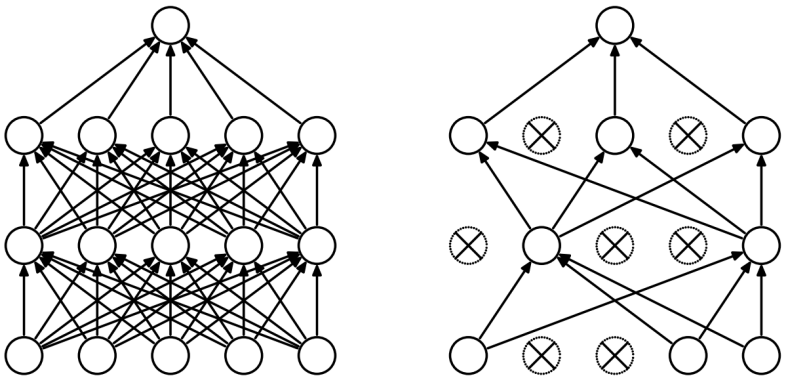
$$Loss = Loss + \frac{\lambda}{2N} \sum \|w\| \tag{2.48}$$

$\lambda \rightarrow$ regularization parameter
$\|w\| \rightarrow$ euclidean norm of the weight squared
$N \rightarrow$ number of parameters

This method reduces some parameters to zero, and because of this characteristic, it is used in implementations where we're looking to compress the neural network and make it more simple.

2.6.5.2 Dropout

Dropout is a regularization technique which randomly deactivates neurons temporarily from the network layers and their respective connections. Each neuron has a probability to drop, which can be defined by the user, and it is the same for all neurons in the neural network [39].



(a) Standard Neural Net (b) After applying dropout.

Figure 2.11: **(a)** Normal neural network. **(b)** Dropout regularization technique. Some neurons and their connections are temporarily removed. Retrieved from [39].

2.6.5.3 Early Stopping

Early stopping is a regularization method which stops the training of the network based on a criteria set by the user. Early stopping is used to stop the training whenever the validation loss stops decreasing. This shift in the rate of change of validation loss can represent the beginning of overfitting of the network. The user can also define a number of consecutive epochs where the criteria for early stopping needs to be met before stopping the training of the network in order to guarantee the networks stops when the training has reached its maximum.

2.6.6 Types of ANNs

The simplest artificial neural networks are composed of three layers, an input layer, one hidden layer and one output layer. An ANN can have a infinite number of hidden layers and there are many variations of neural networks where the behaviour of the neurons is modified, the connection between layers and neurons is different and these and other modifications change the overall function and application of the ANN. We will present some of the most relevant implementations.

2.6.6.1 Fully Connected Neural Network

As the name suggests, in this neural network all the neurons in one layer are connected to all the neurons in the next layers. The neurons do not connect with the preceding layers or with neurons from the same layer, keeping the information moving forward. In this implementation, all the neurons in one layer will have an impact in the final output of all the neurons in the next layer.

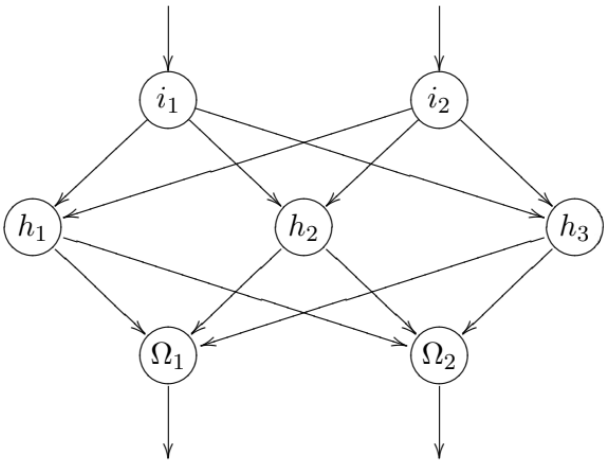


Figure 2.12: Fully connected neural network representation. Retrieved from [35].

2.6.6.2 Recurrent Neural Networks

"Recurrence is defined as the process of a neuron influencing itself by any means or by any connection"[35]. Traditional Recurrent Neural Networks are able to have this recurrence by feeding the previous output back to the new input. This allows RNNs to better learn temporal features and patterns of the data.

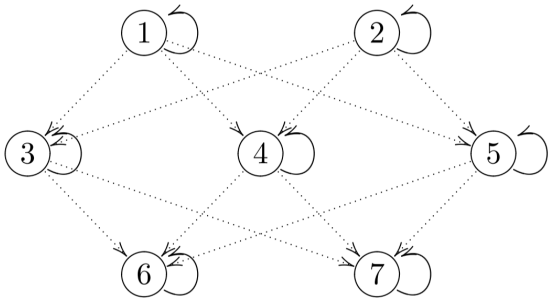


Figure 2.13: Recurrent Neural Network representation. Retrieved from [35].

2.6.6.3 Long Short Term Memory(LSTM) Neural Networks

LSTM Neural Networks were proposed by S. Hochreiter and J. Schmidhuber. [40] in order to solve exploding and vanishing gradient problems which common RNN faced during backpropagation. This was achieved through the implementation of memory cells and gates. The memory cells allow a longer term internal storage of previous data when compared to RNNs and the gates give the ability to control the flow of information, and select the data features to retain or dismiss in the calculation of the output. This neural network is able to receive and process sequences of data and learn long term dependencies.

2.6.6.4 Convolutional Neural Networks

In applications of neural networks such as the recognition of handwritten digits and the analysis of images, there are spatial features which are important in order to achieve the desired output of the network. Convolutional neural networks (CNNs) are one of the best approaches for these application due to the way they analyse the image data. In the analysis of an image, the CNN applies convolution to the data through filter/kernels with dimension height per breath per number of channels and the input for the convolution are the values of the pixels captured by the kernel. These kernels move throughout the whole image.

Convolution Operation In the convolution operation, the filter "overlaps" the data and value of the data in each cell is multiplied by the value of filter in the same cell. This multiplication occurs in to all the value of the data to which the filter is being applied. The result of the individual multiplications are then summed to give the result of the convolution operation.

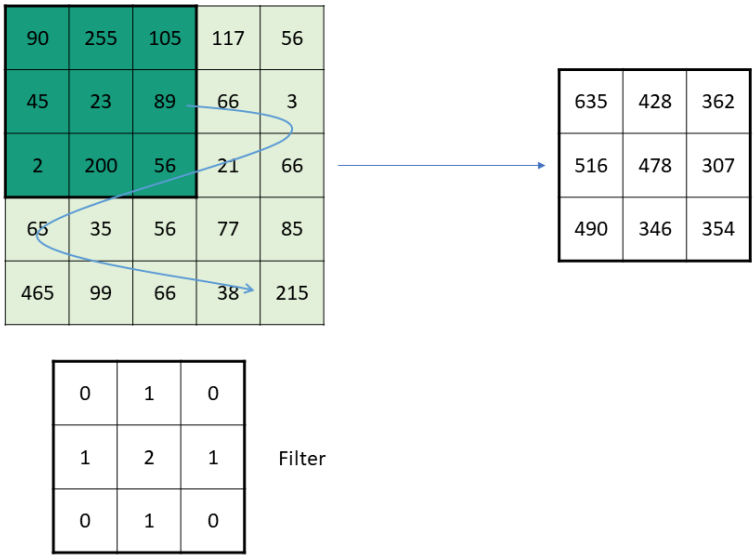


Figure 2.14: Convolution operation.

Both the size of the kernel and the steps/strides taken while moving through the image can be adjusted to improve results. This analysis of smaller parts of the image instead of the image as a whole allow for better analysis of important details and reduces the computational requirements. One important strength of CNNs is that in models with several CNNs, the first layers extract low levels features such as shapes and borders and subsequent layers extract higher level features such as faces or objects from an image.

Pooling Pooling operation is also a part of many convolutional neural network. This operation allows for the selection of the features of highest relevance resulting from the convolution operations and helps improve the computational efficiency of the network.

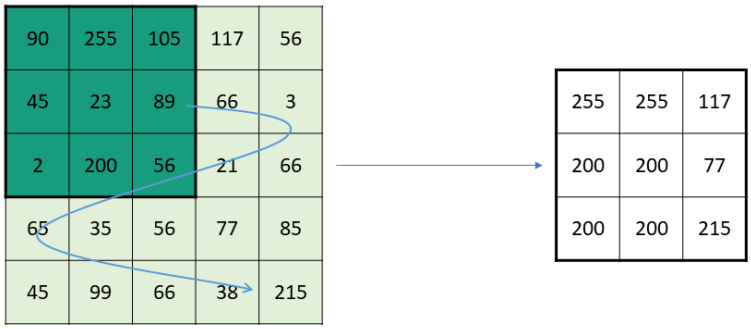


Figure 2.15: Max pooling operation. The kernel only selected the maximum values.

1 Dimensional Convolutional Neural Networks 1D Convolutional Neural Networks work in a similar fashion to traditional convolutional neural networks. Instead of applying convolution to 2 or 3 dimensional data such as images, it applies 1 dimensional data which is the case of the smartphone's accelerometer, gyroscope and magnetometer. The filter runs through the data, as exemplified in Figure 2.16, applying convolution. The filter can also run through the several axis of a sensor simultaneously.

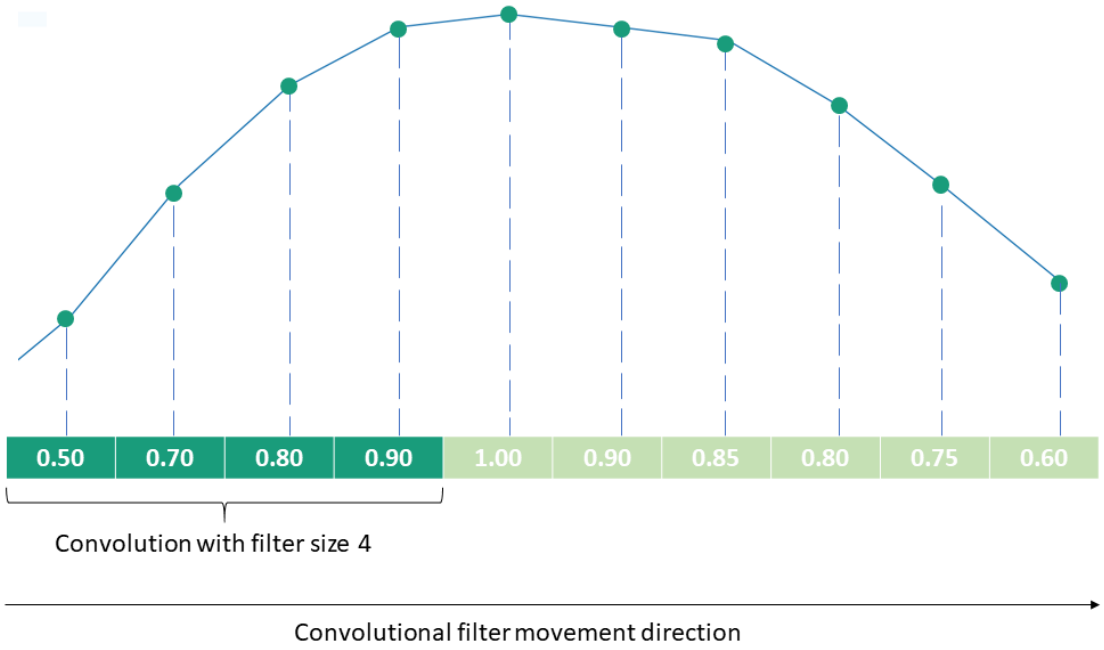


Figure 2.16: 1D Convolution. The kernel with size 4 moves along the data and performs convolution.

Chapter 3

Data Acquisition

The goal of this thesis is to develop an algorithm capable of performing step detection with high accuracy in an unconstrained manner. For the training and testing of the algorithm, a data set which comprises of several of different situations and movements a normal individual encounters in day to day activities is essential, and that was taken into account in the development of the acquisition protocol. Two different data sets were recorded, an indoor data set with a predefined course and set of actions and a free living data set.

3.1 Sensors Acquired

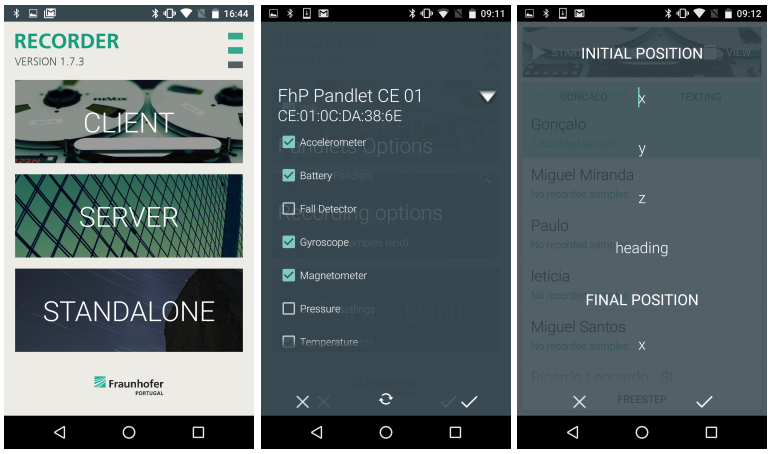
For data collection for both training and testing of the algorithm, the accelerometer, gyroscope and magnetometer data where recorded using a set of two Pandlet Recorders and a Nexus 5 smartphone.

Fraunhofer AICOS Pandlets (Figure 3.1) are small sensing devices with several sensors incorporated, including a tri-axial IMU with an accelerometer, a gyroscope and a magnetometer. Due to its small size, these devices are easily attached to and individual for data collection. The Pandlets also come with an embedded Bluetooth module, allowing the wireless connection with external devices. These devices allow the recording of inertial data at 100 Hz [41].

Figure 3.1: Fraunhofer AICOS Pandlet

Nexus 5 smartphone also comes with tri-axial accelerometer, gyroscope and magnetometer sensors which record data with a 200 Hz sampling frequency[42].

The data was collected through a mobile application developed by Fraunhofer AICOS, the Recorder App. This app allows the collection of data from the smartphones many sensors and also from external sensors simultaneously, through a Bluetooth connection.



(a) (b) (c)

Figure 3.2: (3.2a) Fraunhofer AICOS Recorder App. (3.2b) Connection to Pandlets and sensor selection. (3.2c) Final acquisitions annotations screen.

3.2 Indoor Data Acquisition

Using the Pandlets and the Nexus 5 smartphone, several data acquisition sessions were conducted with multiple volunteers. For the Indoor Data Set a total of 7 volunteers and for the Free Living Data Set a total of 11 volunteers, with ages between 22 and 29 years old and without prior gait related disabilities, performed the acquisition protocols.

3.2.1 Sensor Placement

The Pandlets were used to record data to obtain the Ground Truth of the steps while the smartphone recorded the users movement in several positions and movement speeds found in the day to day routine. The Pandlets were placed over the volunteer's ankles in order to obtain the best signal possible for step detection. This Pandlet placement allows for the detection of clear peaks from the accelerometer signal and also for the clear detection of the cyclical nature of the swing and stance phases of gait through the gyroscope.

The smartphone placements were:

- **Texting** - smartphone held in front of the upper abdomen/lower chest area in a texting position
- **Calling** - smartphone held close to one of the ears of the user similarly to when answering a phone call
- **Handheld** - smartphone held in one of the hands with arm straight and beside the trunk
- **Jacket Front Pocket** - smartphone stored in one of the front pockets of the jacket
- **Front Pocket** - smartphone placed in one of front pocket of jeans
- **Back Pocket** - smartphone placed in the back pocket of jeans

3.2.2 Recorded Activities

In the selection process of the movements to analyse using the inertial sensors of the smartphone and Pandlets, both step related movements and false step/false positive related movements were taken into account for the protocol. The final list of user movements analysed were:

- **Walking** - walking in a natural pace
- **Running** - running in a natural/jogging pace
- **Walking while dragging feet** - walking while purposely dragging the feet on the floor to recreate the walking of people with some gait impairment or the walking of elderly individuals
- **Walking downstairs** - normal pace walking downstairs
- **Walking upstairs** - normal pace walking upstairs
- **Jumping** - jumping without moving from the current position. The purpose of the movement is to prevent the detection as step. Jumps have high inertial signal amplitude and as a consequence can be easily misdetected as steps.

- **False Steps** - This movement has the purpose of recreating the small movements we perform while standing in a conversation, or other situations where we perform small movements with our feet without leaving the same place.

3.2.3 Indoor Route

This indoor route was planned and performed in Fraunhofer AICOS Lisbon Office and it took in consideration all the movements and smartphone placements previously mentioned. In Annex I, the collection protocol, a representation of the trajectory followed for data collection as well as some stop positions where activities were performed, is presented. In the stop positions, the volunteers also stopped their movement for 5 seconds in order to have data while standing still and to make the beginning and end of different activities easily detectable in a visual representation of the data. In Figure 3.3, there is a representation of the signal acquired using the Pandlet sensors and the different activities are represent with a corresponding color.

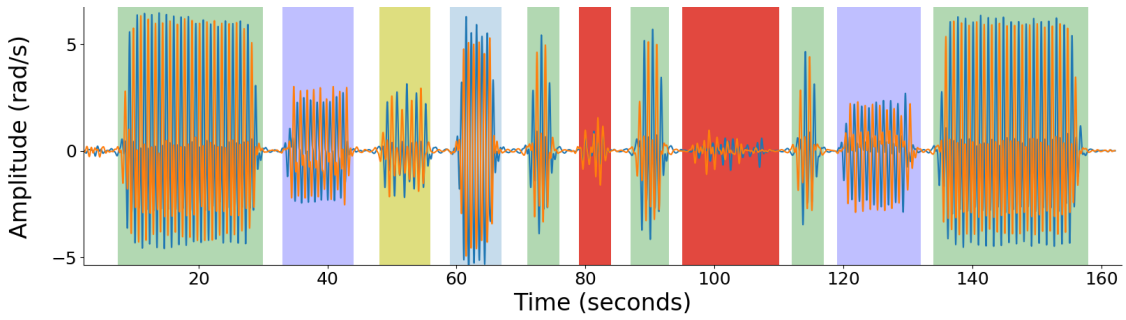


Figure 3.3: Signal segmentation by activity. From left to right, Green - walking; light purple - downstairs and upstairs, respectively; yellow - dragging Feet sequence; light blue - running; red - jumps and false steps, respectively.

3.3 Free Living Data Acquisition

Since we cannot predict every single movement or smartphone placement we encounter in our daily lives, the Free Living Data Set does not have a predefined trajectory or fixed smartphone placements. The objective of this data set was to collect data in real day to day life situation and as such, the volunteers had the Pandlets placed in their ankles for ground truth data collection and were only asked not to keep the smartphone in the same position for more than 10 minutes at a time in order to collect as many position and uses as possible, so when training the algorithm with this data, it would be able to learn as much as possible from daily life situations. Among the collected data were activities such as walking up and downstairs, sitting and having lunch and walking to and from a nearby coffee shop.

3.4 Final Data Set

Unfortunately, a few problems with the data collection procedure were encountered which resulted in a reduction of the data usable for the acquisition of the Ground Truth and the training and testing of the algorithm.

3.4.1 Pandlet Disconnections

In a few acquisitions, the Recorder App lost connection with one or both Pandlets mid acquisitions and this resulted in acquisitions with data of only one Pandlet or data from Pandlets which stopped early than end of the acquisition protocol. In the Indoor data set, some of these acquisition were retried and the data was collected. In the Free Living data set some attempts to salvage data from acquisitions were made, using only the data until the moment where one of the Pandlets disconnected. However, in some situations, the data would be cut too short and these acquisitions had to be discarded.

3.4.2 Temporal Issues

A problem with the acquisition of data from the Pandlets was found when analysing the timestamps of the data, post acquisitions. The Pandlets acquire the data with a sampling frequency of 100 Hz, which means every timestamp should be separated by a 0.01s interval. In several acquisitions, time intervals superior to 0.01s were found, meaning there was missing data. The longer the data acquisition the more of these "jumps in time" we encountered.

Another temporal issue encountered was that in most of the acquisitions of the Free Living Dataset, after a few minutes, the signals of the Pandlets would start to desynchronize in an unpredictable manner. This desynchronization resulted in a general failure of the Ground Truth algorithm for the detection of the steps, and this data, if fed to the neural network for training, would result in learning of false behaviours.

3.4.2.1 Signal Analysis and Correction Attempts

Analysis of Acquisitions The first step towards the correction of the data set was to check the data for time jumps superior to 1 second. Acquisitions with time jumps superior to 1 second potentially had steps missing from the data and were related with data sequences which demonstrated desynchronization of the signals. These acquisitions were removed from the data sets.

Time Synchronicity Check-Up The remaining acquisitions were visually analysed in order to check for a desynchronization of the Pandlets signal in time and relative to each other and to the smartphone. This desynchronization was also checked through the results of the ground truth algorithm, since a change in the synchronicity of the data would, in some cases, result in a reduction of the detected steps of up to half of the real

number of steps, since the signals of the Pandlets would shift into phase. The real life equivalent would be for a person to be walking with both feet moving synchronously in the same direction, which is not feasible or true.

Time Interpolation The next step was to try to correct the data in time. For this we interpolated the data with respect to time axis. The timestamp list would now move perfectly in intervals of 0.01 seconds. This correction was applied to all data since most of the collected data exhibited at least a few small time jumps of 2 or 3 milliseconds.

Post Interpolation Synchronicity Check-up After the interpolation in time, the data synchronicity was then again visually analysed to see if it had solved the problem. Unfortunately, in the Free Living Data, since these acquisitions sometimes we're almost up to an hour long, this was not enough to solve the problem.

Free Living Data Removal For the remaining data of the Free Living Data Set, a more in depth analysis of the desynchronization of the signal was made in order to check for data that could be salvaged through the removal of data which clear was not synchronized. Some data was salvaged through this process but the overall data set was quite reduced with this entire process.

3.4.3 Heading Estimation Issues

Due to some constrains in the performance of the heading estimation algorithm employed for the rotation of the smartphone's reference frame to the pedestrian's reference frame, the Hand smartphone placements had to be removed from the final indoor data set, since the algorithm demonstrated a higher error in heading estimate and this would affect the final data in the pedestrian's frame of reference.

3.4.4 Employed Data for Training and Testing

In the end, and despite the attempts to preserve as much of the collected data as possible for training and testing of the network, the final data set end up being reduced by a large amount. The final data set consists of only indoor data collected. The Free Living Data Set presented too many irregularities to be used successfully to train the neural network and as such it was discarded. The final composition of the data set is as follows:

Training Data:

- Texting - 3 Acquisition
- Right Pocket - 3 Acquisitions
- Back Pocket - 3 Acquisitions
- Jacket - 3 Acquisitions

Testing Data:

- Texting - 1 Acquisition
- Right Pocket - 1 Acquisition
- Back Pocket - 1 Acquisition
- Jacket - 1 Acquisition

Chapter 4

Proposed Algorithm

In this chapter, the novel algorithm for step detection using unconstrained smartphone will be explained in detail. The final algorithm aims to provide a step detection solution for unconstrained use of the smartphone which doesn't require parameter tuning or activity recognition. The diagram of the overall algorithm is represented in image 4.1.

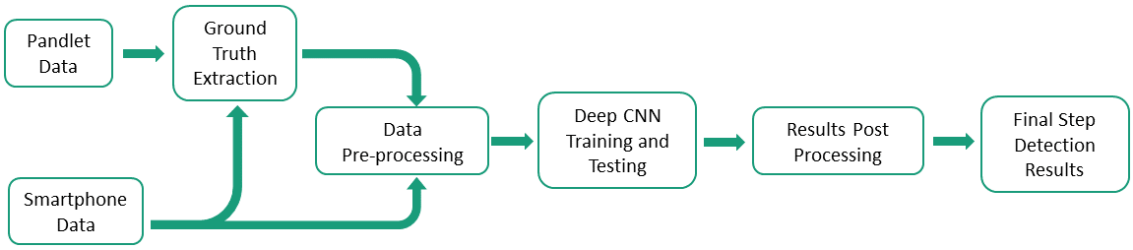


Figure 4.1: Step detection for unconstrained smartphones algorithm diagram.

As mentioned in the previous chapter, the data from the Pandlets attached to the volunteer's ankles was used to obtain the ground truth of the steps. The ground truth will be employed later as the labels for the neural network training and testing data. The smartphone's sensors are used as input data for the deep CNN network. Before being fed to the network, the data must be transformed and reshaped into a format the network can best learn from and as such, a pre-processing step before the training phase was necessary. After training, a testing phase is performed. The model only needs data already collected to detect steps so a real-time implementation is possible. The results of the model go through a post-processing stage in order to have the final and most correct list of detected steps.

4.1 Ground Truth Extraction

For the detection of steps from the pandlets data for a ground truth, we have 2 sensors which can be used, the accelerometer and the gyroscope. The data extracted from the accelerometer exhibits the cyclic nature of the human gait and also is characterized by sharp peaks in the moment of contact of the foot with the ground (heel strike) at the end of the swing phase.

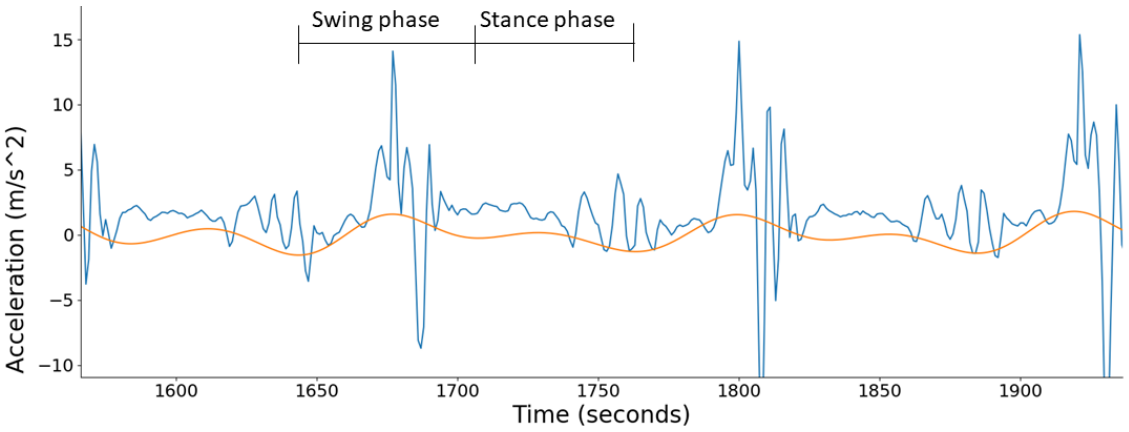


Figure 4.2: In blue - unfiltered accelerometer signal from one of the Pandlets. In orange - filtered accelerometer signal from one of the Pandlets.

The gyroscope signal also demonstrates the cyclical nature of the human gait but through the angular velocity of the leg. For this sensor, the peaks occur mid swing phase, when the angular velocity of the leg is maximum.

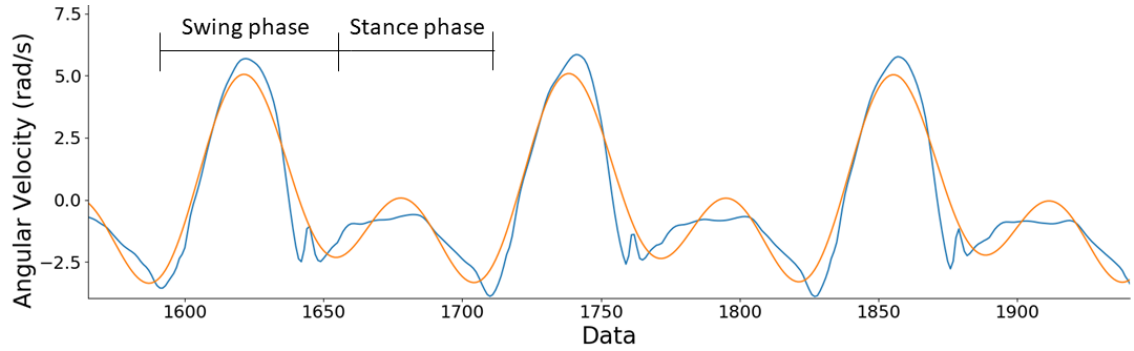


Figure 4.3: In blue - unfiltered gyroscope signal from one of the Pandlets. In orange - filtered gyroscope signal from one of the Pandlets.

Through the analysis of the signals collected from the accelerometer and the gyroscope of the pandlets, both signals exhibited similar capacity for the detection of steps for the ground truth, when filtered with a bandpass filter with cutoff frequencies of 0.6 Hz and 2Hz, the average frequency of the human locomotion[19]. In the end, for the ground truth extraction we selected the gyroscope signal.

4.1.1 Final Ground Truth Algorithm

In Figure 4.4, a diagram representation of the different stages of the ground truth information retrieval is displayed. This diagram is composed of 2 big sections, the individual pandlet ground truth retrieval and the dual pandlet ground truth retrieval. In the single pandlet ground truth retrieval section, peak detection is performed on the filtered gyroscope data from each pandlet and peak filtering thresholds are applied individually to the signal of each pandlet. In the dual pandlet section, the steps retrieved from each pandlet individually are joined and another threshold is applied to this list of steps in order to obtain the final step list. More details about each individual step of the ground truth algorithm are explained in the following subsections.

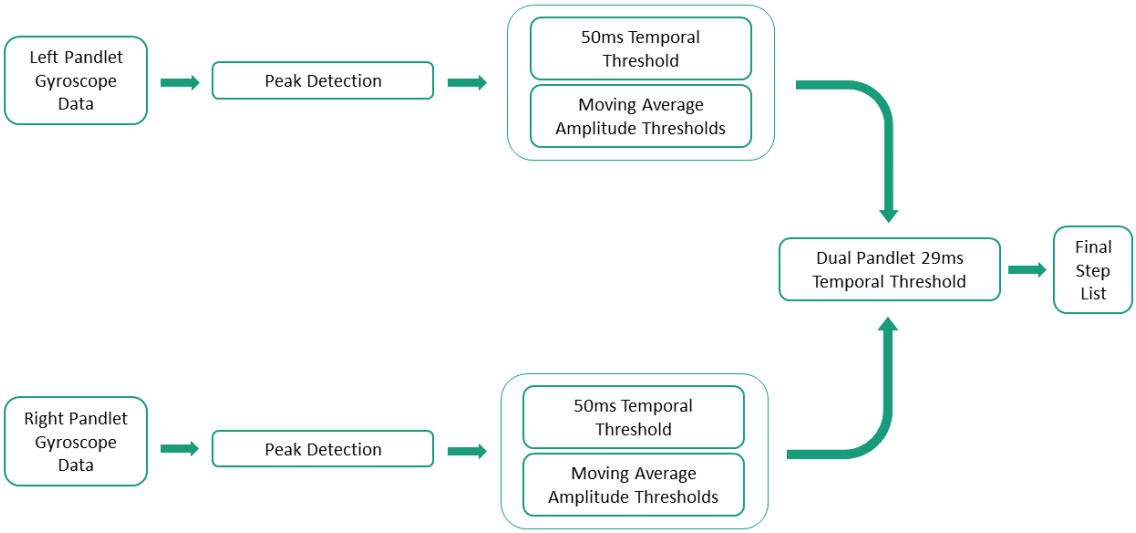


Figure 4.4: Final ground truth algorithm representation.

4.1.1.1 Peak Detection

For the analysis of the ground truth signals and extraction of steps, the Scipy [43] and nova.instrumentation [44] python libraries were used. The gyroscope signal was first filtered using a bandpass filter with cutoff frequencies of 0.6Hz and 2Hz and then the find_peaks function was employed for peak detection. The find_peaks function detects all the peaks in the signal and has the possibility of establishing time and amplitude thresholds[43]. After the detection of the peaks of the signal, peaks which do not represent steps need to be filtered out.

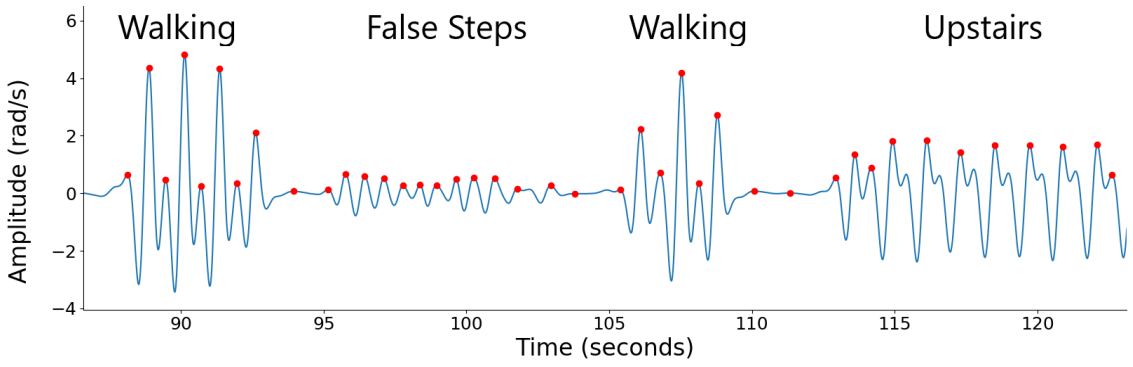
4.1.1.2 Single Pandlet Peak Filtering

The first stage of peak filtering involves establishing thresholds to the peaks detected in the gyroscope signal of each Pandlet individually. A temporal threshold of 0.50s was established in order to remove false positives, taking into account the average human locomotion frequencies. In order to filter all the peaks that do not represent steps, for

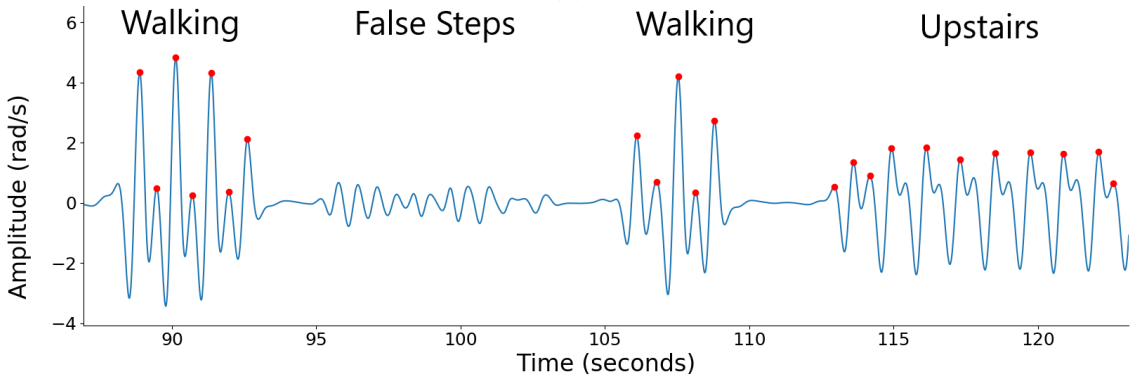
some cases a more situation specific approach was necessary. Two moving averages were employed in order to remove low amplitude peaks. The results can be visualized in Figure 4.5.

1 Second Moving Average Amplitude Threshold The first was a moving average with a time window of 1 second was applied to the gyroscope signal of the pandlet being analysed. The peaks with amplitude below 0.2 rad/s both in the original signal and in the result of the moving average were removed. This moving average resulted in the removal of most peaks resulting from noise.

3 Seconds Moving Average Amplitude Threshold The second was a moving average with a time window of 3 seconds applied to the z axis of the smartphone's accelerometer signal. The smartphone's accelerometer signal underwent sensor fusion through a complementary filter and reoriented to the earth's reference system. This moving average was employed for the removal of the peaks from the false steps sequence. This sequence demonstrates a low signal amplitude both in the accelerometer and gyroscope data and the 3 seconds moving average allows us to make sure only peaks in the false step sequence are removed. This moving average removed peaks which demonstrated an amplitude below 1.5 rad/s and which had a value below 22% of the max of amplitude in the moving average signal.



(a)

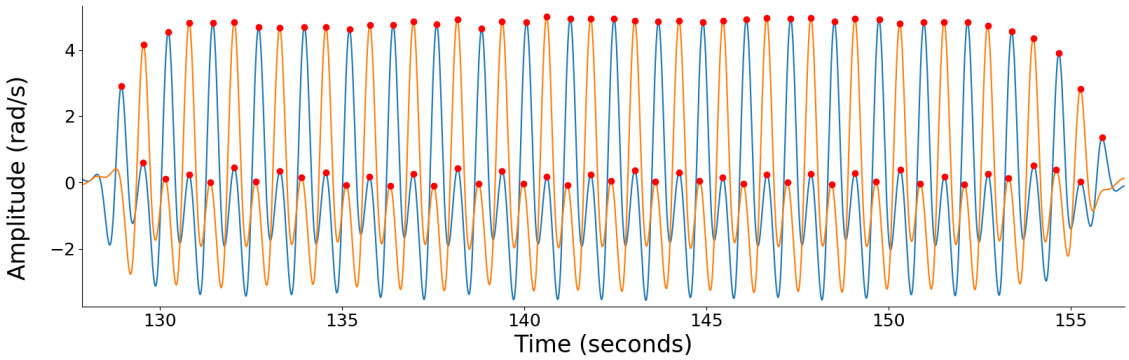


(b)

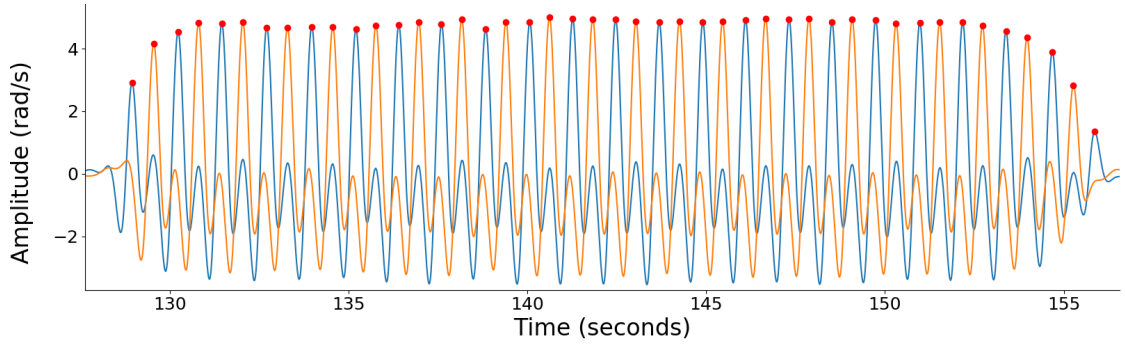
Figure 4.5: Moving averages application visual representation. (4.5a) Unfiltered peaks with only a 0.5 seconds threshold, detected on the gyroscope signal from one of the Pandlets. (4.5b) Filtered peaks using a 0.5 second threshold and moving average amplitude thresholds on the gyroscope signal from one of the Pandlets.

4.1.1.3 Dual Pandlet Peak Filtering

The filtered gyroscope signal exhibits, in most of the acquisition, a behaviour represented by a big peak followed by a small peak. The bigger peak represents the angular velocity of the swing phase of the step while the smaller peak represents the angular velocity of the stance phase. In a visual representation of the gyroscope signals of both Pandelts, we see they synchronize with a 180º phase difference so that when one leg is reaching maximum angular velocity in the swing phase, the other is reaching maximum angular velocity in the stance phase. When joining the peaks of both signals, an overlap of peaks occurs due this behaviour. In order to remove these overlapping peaks, a temporal threshold of 0.29s was placed and the lowest amplitude peaks were removed. The results can be visualized in Figure 4.6.



(a)



(b)

Figure 4.6: Representation of the dual Pandlet filtering procedure. The signals represented in blue and orange originated from the Pandlets gyroscope sensors.(4.6a) Before Dual Pandlet Peak Filtering. (4.6b) Result of Dual Pandlet Peak Filtering.

4.1.1.4 Automatic Step Filtering Attempts

Removal of Jumps Jumps present a very high amplitude both in accelerometer and gyroscope signals and the volunteer jumped using both legs which resulted in a synchronization of the signal of both Pandlets. We sought to use this synchronization to remove the peaks detected due to jumps through the comparison of the amplitude values around the jumps peaks. If the difference between the signals amplitude was below a threshold of 0.4 rad/s for 50 data points around the peak, the algorithm would considered it a peak resulting from a jump and would remove it. Unfortunately the algorithm was unable to remove the all the jumps correctly since in some cases the signal synchronization was not optimal.

4.1.2 Manual Corrections

Unfortunately, after the peak filtering, a few situations remained where the wrong peaks were removed and some of the false positive peaks remained. In order to obtain 100% correct ground truth data, a few manual corrections were made.

Dragging Feet Correction For some acquisitions, the false steps and the dragging feet situations exhibited similar behaviours which resulted in the removal of some peaks of

the Dragging Feet scenario. As a consequence, some steps had to be manually reinserted.

Jumps Correction The peaks detected in the signal resulting from jumps had to be manually removed due to the failure of the automatic approach.

False Steps Correction Most of the false steps were removed by the algorithm and in some rare cases a few false steps had to be manually removed.

4.2 Deep Convolutional Neural Network

For unconstrained step detection and without hyperparameter tuning or activity recognition we opted to employ deep neural networks. The implemented network, as well as other considered approaches are explained in this section. Before beginning the training of the network, the input data needs to be prepared in order to have the most efficient learning process. After pre-processing the data, it is fed to the network for training and testing. In the end, the results suffer post processing in order to obtain the most accurate step list.

4.2.1 Data Pre-processing

The pre-processing of the data is employed through several steps. First, sensor fusion is applied to the smartphone data and the reference frame of the data is also modified in order to have the most informative data. We also need to shape the data into a format the model can read and learn from in the most effective manner. Two data formats were attempted, one where the network received an input with 128 samples and returned an output with shape 1 and another where the network received an input of 256 samples and returned an output of 128 samples, having in the end selected the latter for network. In the case of the Free Living Data Set, a data set balancing strategy was also employed.

4.2.1.1 Sensor Fusion

In order for the network to learn in the best possible manner from the data, the most appropriate frame of reference for the smartphone data is necessary. For sensor fusion and reference frame translation a complementary filter was applied to the smartphone data, similar to 2.9. The chosen data frame of reference was the Pedestrian's Frame of Reference for providing more information about the movement of the smartphone user, thus portraying better the situations encountered in day to day locomotion.

The translation for the chosen frame of reference was not a direct one. First, we translated from the frame of reference of the smartphone to the earths frame of reference and then we translated the data to the pedestrians frame of reference using the heading information.

The first step of the translation was accomplished through the application of a complementary filter, similar to the on 2.9 and through quaternion rotation. Using the data from the accelerometer to establish the vertical direction for the z axis and the magnetometer information to establish the direction of the magnetic north for the y axis, and the inner product of these two axis to establish the x axis, a rotation matrix was establish which was converted into quaternions for application with the complementary filter.

For the second step, the actual translation to the pedestrian's reference frame, the heading was calculated using an algorithm developed in Fraunhofer AICOS by Leonardo *et al.*[45]. This algorithm uses only the data from the smartphones accelerometer in the Earths coordinate system. In this proposal, first a vector $\overrightarrow{w}$ which points in the direction of the movement of the pedestrian is obtained and can be formally represented as

$$\overrightarrow{w} = \frac{d\,\widetilde{acc}_{z_E}}{dt}\overrightarrow{\widetilde{acc}}_{y_E} - \frac{d\,\overrightarrow{\widetilde{acc}}_{y_E}}{dt}\widetilde{acc}_{z_E} \tag{4.1}$$

$\widetilde{acc}_{z_E} \rightarrow$ filtered acceleration magnitude in the z axis direction of the earth's reference frame.

$\overrightarrow{\widetilde{acc}}_{y_E} \rightarrow$ filtered acceleration vector in the y axis direction in the earth's reference frame.

Then the vector is filtered using a bandpass filter for the frequencies of human locomotion (between 0.6Hz and 2Hz[19]) estimation of the user's heading is performed through the following expression

$$\theta = \tan^{-1}\left(\frac{\overrightarrow{\widetilde{w_E}} \cdot \overrightarrow{y_E}}{\overrightarrow{\widetilde{w_E}} \cdot \overrightarrow{x_E}}\right) \tag{4.2}$$

Using the heading information, the earths reference frame is translated into the pedestrian's reference frame through the following expressions

$$a_{x_P} = a_{x_E} * \sin(\theta) - a_{y_E} * \cos(\theta) \tag{4.3}$$

$$a_{y_P} = a_{x_E} * \cos(\theta) + a_{y_E} * \sin(\theta) \tag{4.4}$$

and the z axis is the same in both reference systems.

4.2.1.2 Ground Truth Binarization

The timestamp list of the steps retrieved from the Ground Truth Data needs to be converted into labels the network can read and use to compare the results of the step predictions with. For that, we convert the list of timestamps to a binary list, where the zeros represent non-step situations and the ones represent the steps. This binary list needs to have the same length of the data. If we limit a step to a very specific point in time of

0.01s, the sampling period, we end up with a very unbalanced data set which could affect negatively the training procedure of our model. In order to prevent that, we extended the steps timestamps to an interval of 0.10 seconds around the timestamp of the peak. The resulting signal is depicted in Figure 4.7. This idea is inspired in a technique used for trigger word detection in voice recognition deep learning systems[46]. This way, the network can better learn the patterns around of the signal of a step and detect them in a larger variety of situations.

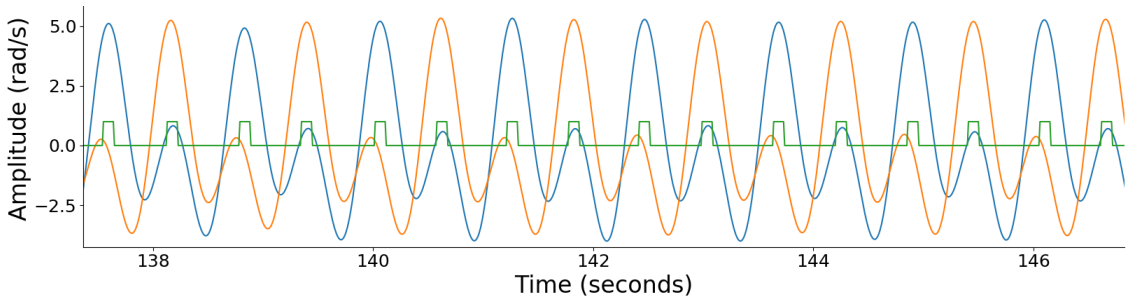


Figure 4.7: Ground Truth Binarization. In blue and orange - Pandlets gyroscope data. In green - steps array after binarization procedure.

4.2.1.3 Data Set Balancing - Free Living Data Set

In Free Living, the majority of the volunteers spent a large portion of acquisition time standing still, since the acquisition were performed in a office working environment. Attempts were made to reduce this immobility data percentage, by recording data in lunch breaks but it still proved insufficient to balance the data. This unbalanced data set often resulted in training attempts where the network started to learn to detect inactivity and not moments of movement/steps. So additional measures were taken.

Zero Activity Sections Removal The removal of periods of inactivity was how we attempted to balance the Free Living Data Set. This portion of the pre-processing algorithm searched for intervals in the data where the activity exhibited in the tri-axial smartphones signal was close to zero, representing a period of time were the smartphone was not in motion. The norm of the accelerometer signal was used for this endeavor. Every interval of 256 data points whose acceleration norm was below the threshold of 0.5 m/s^2 was removed from the data set.

4.2.1.4 Training, Testing and Validation Data Sets

From the data acquired only the triaxial acceleration data from the smartphone was used for training and testing. Several combinations of data from the accelerometer and gyroscope were experimented with. Data sets using only accelerometer or only gyroscope data,

a combination of data from several axis from both and even all data from the accelerometer and gyroscope were applied and, in the end, data from only the accelerometer yielded the best results.

For the creation of the data sets:

- The data from one volunteer was used to create the Testing data set;
- 10% of the remaining data was randomly selected to create the Validation data set;
- The remaining data was used to create the Training data set.

4.2.2 Deep Learning Network Model

Throughout the development of the final model, several model constitutions were analysed using different types of neural networks and extensive parameter tuning occurred in order to reach an optimized solution. In this subsection, an explanation of the employed model is presented in further detail.

The final model for unconstrained step detection is schematically represented in Figure 4.8. The model contains 5 1D Convolutional layers with ReLU activation functions and 1 1D Convolutional layer with sigmoid activation function as output layer so the output is a probability value. This probability value will allow for better interpretation of the results and to see where the network considers that a step occurs.

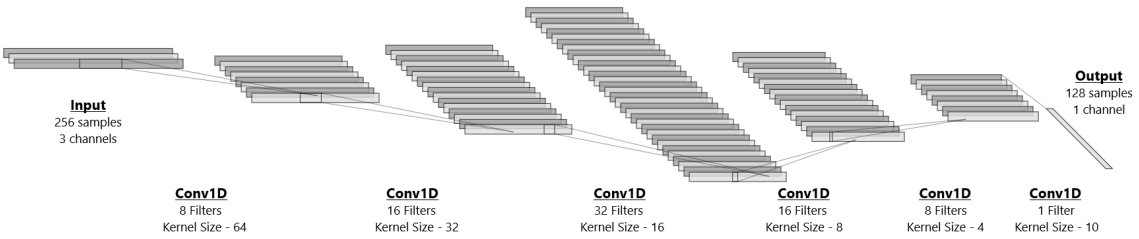


Figure 4.8: Final model for step detection.

Data Shape The data was fed to the neural network in time intervals of 256 samples from 3 channels, the x, y and z axis. The model trained using batches of 32 samples in length. The model presented an output with a shape of 128 samples.

Optimizer Adam optimizer was used for our model with the parameters sugested by the authors ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$) since it typically demonstrates faster convergence speeds among state of the art optimizers[38].

Loss Function The loss function was calculated using Binary Crossentropy.

Learning Rate The final model learning rate is 0.0001 with a learning rate decay value of 0.0001.

Regularization Early Stopping was applied in order to prevent overfitting with a patience parameter of 10 epochs.

4.3 Post processing

The proposed neural network for unconstrained step detection has as output an array with probability values of a step occurring. This array shows where in the test data the model exists a high or low probability that a step is happening. It still isn't a list or a counting of the steps taken and as such, post processing is necessary. The post-processing part of the algorithm utilizes two main tools, the find_peaks class of the Scipy Python library and a vector W which quantifies the movement of the user.

4.3.1 Peak Detection

The find_peaks detected the peaks in the output array of neural network. Using its built in peak filtering capabilities, a peak temporal threshold of 0.5 seconds was applied.

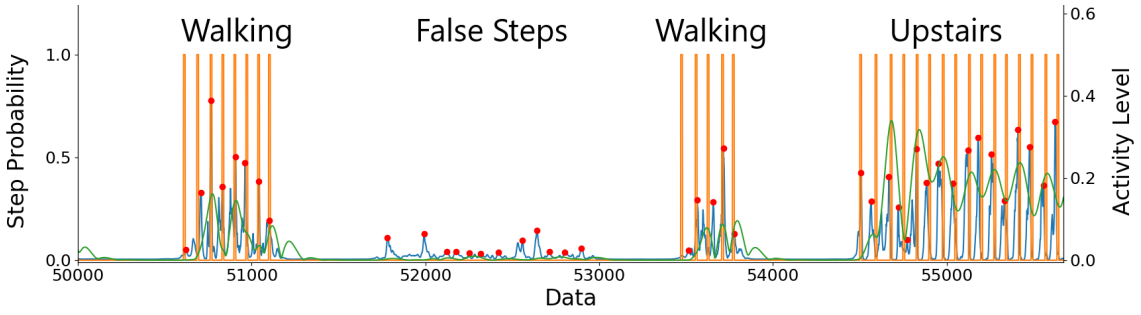
4.3.2 Movement Vector

The vector $\overrightarrow{w}$ described in Equation 4.1 points in the direction of the user's movement. Another characteristic of this vector is that its magnitude can represent the level of movement of the user, being close to zero when standing still, and increasing as the pedestrian's walks faster.

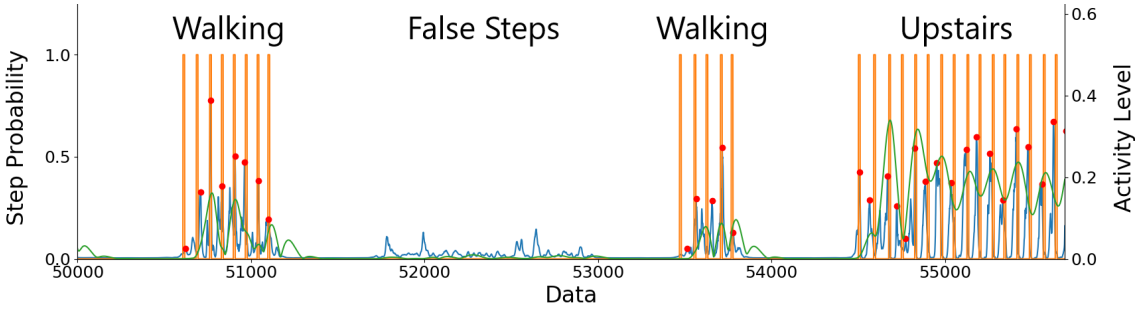
The magnitude of this vector was calculated in order to remove peaks in levels of low activity, such as in the false steps situation, using the norm of the vector

$$motion\ level = \sqrt{(w_x)^2 + (w_y)^2} \tag{4.5}$$

Having a value for the motion level, then a threshold was established. Every peak detected in the results whose motion level, in 2,5 seconds each direction in time, presented no values of activity above 0.02 was removed. This implementation demonstrated high success in situations such as the False Steps sequence, as demonstrated in Figure 4.9.



(a)



(b)

Figure 4.9: Pedestrian's Movement Peak Filtering. In blue - plot of the model results. In orange - plot of the binary ground truth array. In green - plot of the activity level magnitude. (4.9a) Before peak filtering using the pedestrians movement vector. (4.9b) After Peak Filtering using the pedestrians movement vector.

Chapter 5

Results

The testing results of the proposed algorithm for unconstrained step detection will be presented in this chapter. The data was collected in accordance to the data collection protocol, however, as seen, the data set was subjected to modification afterwards which will be explained in detail. The algorithm was tested for the overall testing data set, for each smartphone placement and for each activity performed by the volunteers. The system was tested in an offline scenario.

5.1 Metrics

In order to properly evaluate the results of the step detection algorithm, the validation metrics presented in 2.5.5 were analysed in order to apply the most relevant ones. Regarding the classification of the results into True Positives, False Positives, True Negatives and False Negatives, the line of reasoning applied was:

- **True Positive** - If a peak in the Deep Learning result **finds** a Ground Truth peak in a 0.4 seconds neighbourhood in each direction in time, then it would be classified as a True Positive result.

- **False Positive** - If a peak in the Deep Learning result **does not find** a Ground Truth peak in a 0.4 seconds neighbourhood in each direction in time, then it would be classified as a False Positive result.

- **False Negative** - If a Ground Truth peak does not find any Deep Learning peak in a 0.4 second neighbourhood in each direction in time, then it would be classified as a False Negative result. For this label, a verification of the peak used to classify a ground truth peak as not False Negative was also put in place. If two consecutive

ground truth peaks used the same deep learning detected step to classify itself as not a False Negative, then one of the peaks is a False Negative.

A conclusion was reached that the True Negative classification was not relevant to the results, because defining a point of the results as a True Negative does not yield relevant information about the step detection algorithm's performance. For this reason, we did not look to classify the results also with the True Negative label. As a consequence, the specificity metric was also not applied in the evaluation of the performance of the algorithm.

The accuracy metric was also not applied to the results because it might provide incorrect assumptions about the results of the algorithm. The accuracy metric, as defined in 2.11, is the number of True Positives and True Negatives divided by the total data set length. Since this metric also took into account the number of True Negatives, we also did not applied to the study of the performance of the proposed model.

The metrics applied to the analysis of the results of the step detection are Recall, Precision, and F-score.

5.2 Algorithm Results

The training of the Deep CNN model was performed through 48 epochs, after which the network stopped training due to Early Stopping. The Early Stopping regularization was set with a patience parameter of 10 epochs. The network reach a minimum loss of 0.2437 and a validation loss of 0.2541. The progress of the Loss curves can be visualized in Figure 5.1.

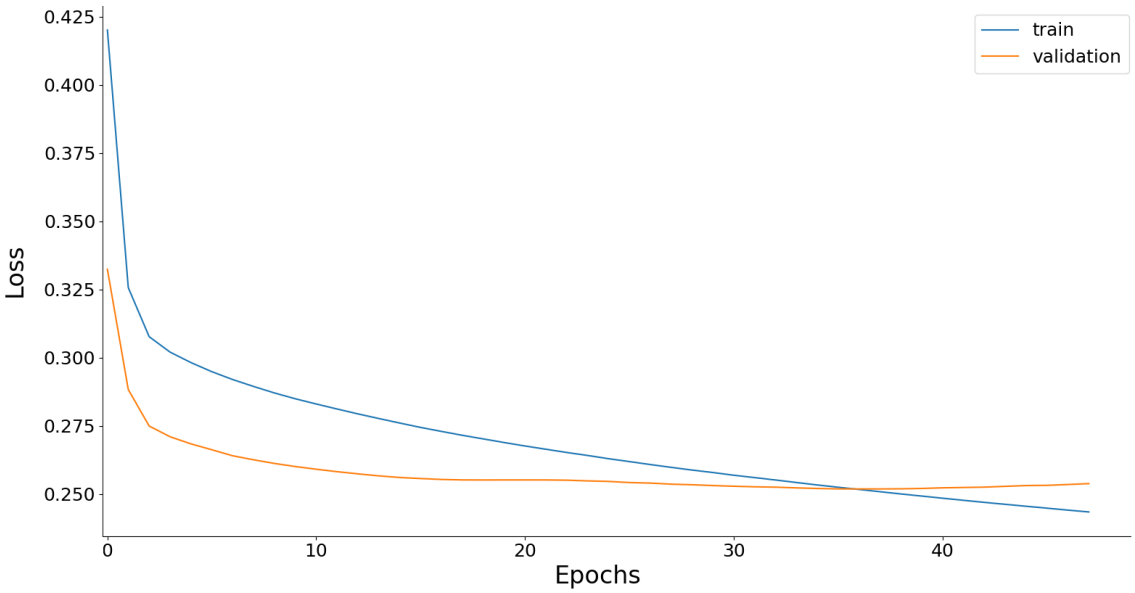


Figure 5.1: Loss and Validation Loss progression curves.

The results on the testing data set are presented on 3 different levels, from the most specific, the user movement types results, moving to the smartphone placement results and ending with the overall training data set results. The results are also compared to the results achieved by the solutions proposed in [17] and [14] when applied to our data set. More in-depth result about the performance for each movement type in each smartphone placement can be found in Appendix A.

The overall results for each of the analysed user movement types can be found in the Table 5.1. The algorithm shows good performance in general in walking, downstairs and upstairs scenarios and struggles in the running and dragging feet scenarios. In these scenarios, the algorithm under counts steps which results in the low Recall but high Precision.

	Recall(%)	Precision(%)	F-score(%)	GT Steps	Model Steps
Walking	94.16	92.01	93.07	427	438
Downstairs	93.51	98.63	96.00	76	73
Dragging Feet	71.43	100.00	83.33	70	50
Running	54.39	93.94	68.89	57	33
Upstairs	88.61	98.59	93.33	74	71

Table 5.1: Algorithm results for each type of user movement.

The results of the performance of the algorithm for each of the smartphone placements can be found in Table 5.2. The algorithm shows good performance in all of the smartphone placements, with a minimum Recall value of 87.08% in the Jacket Placement and a minimum Precision value in the Front Pocket placement of 85.64%. The algorithm was able to reach a maximum recall of 94.48% in the Back Pocket position and a maximum precision in the Texting position of 89.60%.

	Recall(%)	Precision(%)	F-score(%)	GT Steps	Model Steps
Texting	87.08	89.60	88.32	175	173
Front Pocket	90.12	85.64	87.82	171	181
Back Pocket	94.48	88.14	91.20	179	194
Jacket	87.78	89.27	88.52	179	177

Table 5.2: Algorithm results for each smartphone placement.

The overall performance of the algorithm for the entire training data set can be found in Table 5.3.

	Recall(%)	Precision(%)	F-score(%)	GT Steps	Model Steps
Testing Data	89.87	87.90	88.87	704	727

Table 5.3: Algorithm results for the entire testing data set.

In Table 5.4, the results for the proposed algorithm and the proposed solutions presented by Lee *et al.*[17] and Edel *et al.*[14] applied to the collected data set are presented. All three algorithms perform well in the detection of steps which is exhibited by the recall values. The solution proposed by Lee et al. presents the highest recall value of 92.49%. However, as predicted, these solutions don't manage to maintain a good performance in the precision metric due to false positives which where not taken in consideration in the construction of their solutions. Due to this behaviour, the solution proposed in this thesis manages to exhibit a more stable and accurate behaviour for real life situations, outperforming the analysed state of the art solutions. More detailed information about the performance of the different algorithms for different smartphone placements and activities can be found in Appendix A.

	Recall (%)	**Precision (%)**	**F-Score (%)**
Proposed Model	87.87	87.90	88.87
Lee et al.	98.24	35.47	52.12
Edel et al.	87.37	59.39	70.71

Table 5.4: Results of the proposed and analysed steps detection solutions.

	Ground Truth	**Proposed Model**	**Lee et al.**	**Edel et al.**
Total Steps	704	727	2041	1118

Table 5.5: Total detected steps by the analysed solutions.

5.3 Discussion

The results of the algorithm prove that a deep CNN network is a viable approach for unconstrained step detection. They also show that the step detection performance is not affected by changes in the smartphone placement, exhibiting all considered placements similar good results.

Most of state of the analysed art step detection algorithms were validated using an accuracy with respect to the total amount of steps. This method of validation gives way to the possibility of false positives contributing to an exaggerated accuracy value of the algorithm. In order to analyse thoroughly the performance of the proposed method, an analysis of the detected steps with respect to the ground truth temporal positions was implemented. This way, precise and reliable performance metrics were assured. Despite the reduced amount of data available, the algorithm was able to perform well with an overall recall of 89.87%, an overall precision of 87.90% and a F-score of 88.87%. When compared with other state of the art implementations, such as the ones proposed by Lee *et al.*[17] and Edel *et al.*[14] which were applied to the collected data set, the proposed solution of this thesis exhibits a more stable and robust to false step situations which provides higher performance levels to the algorithm. The study by Steinmetzer *et al.*[3]

used a similar validation methodology that was analysed was the one which achieved a recall of 99.2%, a precision of 90.1% and a F-score of 94.4% using CNNs and a foot mounted insole with sensors. Our proposed algorithm exhibits comparable performance, using smartphone data and a more extensive list of sensor placements and activities.

The final model also presented little to no difference in performance when the various smartphone placement results were analysed. The model also showed good performance in different activities performed by the user such as walking, downstairs and upstairs scenarios, while its performance was sub-par in the dragging feet and running activities. This could be due to these activities showing the most variability in the smartphone sensor's data, since in the dragging feet situation, the volunteers are performing an unnatural activity, and as such they end up varying in the way they drag their feet more from volunteer to volunteer, and in the running situation the signals experience higher amplitude and frequency changes. The poor results could also be due to the low amount of data existent for training in these situations in comparison with the remaining activities. The walking activity is most prevalent one, which is in accordance to what happens in day to day situations, having a larger amount of data and the downstairs and upstairs situation end up having similarities which could be contributing to a more successful training for these cases. The algorithm also proved successful in the prevention of false step detection, a situation which is not tested in any of the analysed state of the art solutions.

Chapter 6

Conclusion and Future Considerations

In this chapter, a summary of the developed work as well as its achievements and improvements points is presented.

6.1 Conclusion

The continuous development of smartphones has lead to an increase of the sensors and the data collection capabilities they provide. Step detection is at the core of many applications such as indoor location, virtual reality, health and activity monitoring and some of these require high levels of precision.

In this work, a novel step detection solution using unconstrained smartphones was proposed using deep Convolution Neural Networks. Most state of the art solutions still provide a limited approach in the way the steps are recorded and do not account for possible situations where the algorithms might fail. As such, solutions which can perform step detection without being affected by outside interference resulting from the different kinds of movements of the user or smartphone placements are needed.

Using the inertial data from a smartphone and convolutional neural networks, a model was built which could detect steps with good results. To train the network, data from the smartphone tri-axial accelerometer was used together with the ground truth data collected from Fraunhofer AICOS Pandlets inertial sensors placed in the volunteers ankles. In the data collection procedure, a large variety of movements and smartphone locations were included, as well free living data, in an attempt to give the algorithm the opportunity to learn from situations as close as possible to the ones encountered in daily living. Unfortunately, due to failure of the collection devices and issues with the collected data, the final data set ended up fairly limited in amount and variety of data.

For the validation of the algorithm, metrics which took into account not only the total

number of detect steps but also the position in time where they occurred were implemented in order to obtain the most correct and rigorous results of the model. The metrics implemented for the study were Recall, Precision and F-score. The solutions proposed by Lee *et al.*[17] and Edel *et al.*[14] were applied to the data set for performance comparison. The proposed solution was able to outperform these step detection algorithms demonstrating similar good performance in the detection of steps and outperforming in the robustness against the detection of false step situations. The model was also compared with the solution proposed by Steinmetzer *et al.*[3], which evaluated their model with similar metrics and which exhibited a recall of 99.2%, a precision of 90.1% and a F-score of 94.4% using CNNs and a foot mounted insole with sensors. Our solution was able to exhibit comparable performance, with a recall of 89.87%, a precision of 87.90% and a F-score of 88.87%, while using data collected from a smartphone and taking into account various smartphone placements and user walking and non-walking activities.

The final algorithm shows that convolutional neural networks are a viable approach to unconstrained step detection using smartphone sensors. The method shows potential and further development is necessary. With a more extensive and elaborate data set, with free living situations, the proposed convolutional neural networks model is expected to be able to provide an accurate step detection solution with high performance and very low error in smartphone unconstrained use, allowing applications with real-time step detection at its core to thrive and improve their performance.

6.2 Future Work

The development of the model can still be continued further on and there are still various improvements which can be made.

Firstly, the data set used for the training of the model should be extended in size and complexity in order to contain the most amount of day to day scenarios for a more complete learning process. More step situations, such as, for instance, walking backwards and sideways, and false step situations, such as vibrations suffered by the smartphone when placed on a surface, should be included in the data set. Various walking and running speed can also be introduced in the data set and the influence of different surfaces and shoes in the step detection procedure can also be studied. The inclusion of these variables in the data set could contribute to a more robust step detection system. Free living situations, which were originally planned for this work, should also be included in the training process of the network. The effects of gait impairments in the step detection procedure should also be studied and is another step towards a more complete system.

The ground truth algorithm can also receive improvements. In an ideal situation, where a large and complex data set for training of the network is available, manual corrections become unfeasible and automatic ground truth information extraction becomes essential.

The current model was only tested and analysed in an offline manner. In order for real-time step detection to be provided, adaptations in the manner the model analyses new data need to be implemented and the algorithm would need to be optimized for smartphone use.

This system can be further developed to extract more information about the locomotion of the smartphone user such as step length and step frequency. This data could provide important insight about the movement of the user, such his indoor location through PDR techniques or his physical activity. Development towards a more in-depth analysis of the human gait through the extraction of gait parameters from the detected steps, in real-time and in free living situations, could assist medical professionals in the analysis of the gait of patients in a more realistic way, and further help the diagnosis of gait impairments. This last approach could include the addition of RNNs and/or LSTM neural networks, which have the ability to learn data dependencies in time, to detect abnormalities in the gait data of a patient.

Finally, after improvements have been made to the model, further testing would have to be made in order to prove its efficiency and applicability in many different applications with step detection at its core.

Bibliography

[1] F Proessl, C. Swanson, T Rudroff, B. Fling, and B. Tracy. "Good Agreement Between Smart Device And Inertial Sensor-Based Gait Parameters During A 6-Minute Walk." In: *Gait & posture* (2018).

[2] S. Chen, J. Lach, B. Lo, and G.-Z. Yang. "Toward pervasive gait analysis with wearable sensors: A systematic review." In: *IEEE journal of biomedical and health informatics* 20.6 (2016), pp. 1521–1537.

[3] T. Steinmetzer, I. Bönninger, M. Reckhardt, F. Reinhardt, D. Erk, and C. M. Travieso. "Comparison of algorithms and classifiers for stride detection using wearables." In: *Neural Computing and Applications* (2019), pp. 1–12.

[4] O. Dehzangi, M. Taherisadr, and R. ChangalVala. "IMU-based gait recognition using convolutional neural networks and multi-sensor fusion." In: *Sensors* 17.12 (2017), p. 2735.

[5] A. S. Alharthi, S. U. Yunas, and K. B. Ozanyan. "Deep learning for monitoring of human gait: a review." In: *IEEE Sensors Journal* 19.21 (2019), pp. 9575–9591.

[6] A. Rampp, J. Barth, S. Schülein, K.-G. Gaßmann, J. Klucken, and B. M. Eskofier. "Inertial sensor-based stride parameter calculation from gait sequences in geriatric patients." In: *IEEE transactions on biomedical engineering* 62.4 (2014), pp. 1089–1097.

[7] A. Poulose, O. S. Eyobu, and D. S. Han. "An indoor position-estimation algorithm using smartphone IMU sensor data." In: *IEEE Access* 7 (2019), pp. 11165–11177.

[8] H. Zhang, W. Yuan, Q. Shen, T. Li, and H. Chang. "A handheld inertial pedestrian navigation system with accurate step modes and device poses recognition." In: *IEEE Sensors Journal* 15.3 (2014), pp. 1421–1429.

[9] G. Rodríguez, F. E. Casado, R. Iglesias, C. V. Regueiro, and A. Nieto. "Robust step counting for inertial navigation with mobile phones." In: *Sensors* 18.9 (2018), p. 3157.

[10] J. Kupke, T. Willemsen, F. Keller, and H. Sternberg. "Development of a step counter based on artificial neural networks." In: *Journal of Location Based Services* 10.3 (2016), pp. 161–177.

[11] E Saadatzadeh, A Chehreghan, and R Ali Abbaspour. "PEDESTRIAN DEAD RECKONING USING SMARTPHONES SENSORS: AN EFFICIENT INDOOR POSITIONING SYSTEM IN COMPLEX BUILDINGS OF SMART CITIES." In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* (2019).

[12] M. Khedr and N. El-Sheimy. "A smartphone step counter using IMU and magnetometer for navigation and health monitoring applications." In: *Sensors* 17.11 (2017), p. 2573.

[13] S. Y. Park, S. J. Heo, and C. G. Park. "Accelerometer-based smartphone step detection using machine learning technique." In: *Electrical Engineering Congress (iEECON), 2017 International*. IEEE. 2017, pp. 1–4.

[14] M. Edel and E. Köppe. "An advanced method for pedestrian dead reckoning using BLSTM-RNNs." In: *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE. 2015, pp. 1–6.

[15] I. Ashraf, S. Hur, and Y. Park. "Application of Deep Convolutional Neural Networks and Smartphone Sensors for Indoor Localization." In: *Applied Sciences* 9.11 (2019), p. 2337.

[16] N. Lee, S. Ahn, and D. Han. "AMID: Accurate magnetic indoor localization using deep learning." In: *Sensors* 18.5 (2018), p. 1598.

[17] H.-h. Lee, S. Choi, and M.-j. Lee. "Step detection robust against the dynamics of smartphones." In: *Sensors* 15.10 (2015), pp. 27230–27250.

[18] S. Y. Cho and C. G. Park. "MEMS based pedestrian navigation system." In: *The Journal of Navigation* 59.1 (2006), pp. 135–153.

[19] X. Kang, B. Huang, and G. Qi. "A Novel Walking Detection and Step Counting Algorithm Using 7Unconstrained Smartphones." In: *Sensors* 18.1 (2018), p. 297.

[20] M. Susi, V. Renaudin, and G. Lachapelle. "Motion mode recognition and step detection algorithms for mobile phone users." In: *Sensors* 13.2 (2013), pp. 1539–1562.

[21] Y. Fu, Y. Lei, T. Wang, W. J. Curran, T. Liu, and X. Yang. "Deep Learning in Medical Image Registration: A Review." In: *arXiv preprint arXiv:1912.12318* (2019).

[22] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis. "Deep Learning-based Vehicle Behaviour Prediction For Autonomous Driving Applications: A Review." In: *arXiv preprint arXiv:1912.11676* (2019).

[23] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu. "Financial Time Series Forecasting with Deep Learning: A Systematic Literature Review: 2005-2019." In: *arXiv preprint arXiv:1911.13288* (2019).

[24] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu. "Object detection with deep learning: A review." In: *IEEE transactions on neural networks and learning systems* 30.11 (2019), pp. 3212–3232.

[25] P. Subramaniam and M. J. Kaur. "Review of Security in Mobile Edge Computing with Deep Learning." In: *2019 Advances in Science and Engineering Technology International Conferences (ASET)*. IEEE. 2019, pp. 1–5.

[26] S. Hong, Y. Zhou, J. Shang, C. Xiao, and J. Sun. "Opportunities and Challenges in Deep Learning Methods on Electrocardiogram Data: A Systematic Review." In: *arXiv preprint arXiv:2001.01550* (2019).

[27] V. Passaro, A. Cuccovillo, L. Vaiani, M. De Carlo, and C. E. Campanella. "Gyroscope technology and applications: A review in the industrial perspective." In: *Sensors* 17.10 (2017), p. 2284.

[28] J. Korvink and O. Paul. *MEMS: A practical guide of design, analysis, and applications*. Springer Science & Business Media, 2010.

[29] J. J. Carollo and D. J. Matthews. "Chapter 16: The Assessment of Human Gait, Motion, and Motor Function." In: *Pediatric Rehabilitation: Principles & Practice*. Demos Medical Publishing, LLC., 2009, pp. 461 –491. ISBN: 9781933864372.

[30] A. Kharb, V. Saini, Y. Jain, and S. Dhiman. "A review of gait cycle and its parameters." In: *IJCEM International Journal of Computational Engineering & Management* 13 (2011), pp. 78–83.

[31] J. B. Kuipers et al. *Quaternions and rotation sequences*. Vol. 66. Princeton university press Princeton, 1999.

[32] T. Islam, M. S. Islam, M. Shajid-Ul-Mahmud, and M. Hossam-E-Haider. "Comparison of complementary and Kalman filter based data fusion for attitude heading reference system." In: *AIP Conference Proceedings*. Vol. 1919. 1. AIP Publishing LLC. 2017, p. 020002.

[33] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[34] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[35] D. Kriesel. *A Brief Introduction to Neural Networks*. 2007. URL: availableathttp://www.dkriesel.com.

[36] B. Xu, N. Wang, T. Chen, and M. Li. "Empirical evaluation of rectified activations in convolutional network." In: *arXiv preprint arXiv:1505.00853* (2015).

[37] J. Duchi, E. Hazan, and Y. Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.Jul (2011), pp. 2121–2159.

[38] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014).

[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[40] S. Hochreiter and J. Schmidhuber. "Long short-term memory." In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[41] F. AICOS. *A day with Pandlets*.

[42] *Nexus Tech Specs*. https://support.google.com/nexus/answer/6102470?hl=en. Accessed: 2010-03-06.

[43] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey,
bibinitperiodI. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." In: *Nature Methods* 17 (2020), pp. 261–272. DOI: https://doi.org/10.1038/s41592-019-0686-2.

[44] H. Gamboa. *nova.instrumentation Python library*. URL: https://github.com/hgamboa/novainstrumentation.

[45] R. Leonardo, G. Rodrigues, M. Barandas, P. Alves, R. Santos, and H. Gamboa. "Determination of the Walking Direction of a Pedestrian from Acceleration Data." In: *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, pp. 1–6.

[46] A. Ng. *Trigger Word Detection*.

Appendix A

Additional Deep Learning Results

	Recall	Precision	F-Score	GT Total Steps	DL Total Steps
Walking	93.52%	95.28%	94.39%	108	106
Downstairs	94.12%	100.00%	96.97%	17	16
Dragging Feet	64.71%	100.00%	78.57%	17	11
Running	57.14%	100.00%	72.73%	14	8
Upstairs	77.27%	100.00%	87.18%	19	17

Table A.1: Results for each activity with the Texting smartphone placement.

	Recall	Precision	F-Score	GT Total Steps	DL Total Steps
Walking	95.15%	89.09%	92.02%	102	110
Downstairs	94.74%	94.74%	94.74%	19	19
Dragging Feet	70.59%	100.00%	82.76%	17	12
Running	53.33%	100.00%	69.57%	15	8
Upstairs	94.44%	100.00%	97.14%	18	17

Table A.2: Results for each activity with the Right Pocket smartphone placement.

	Recall	Precision	F-Score	GT Total Steps	DL Total Steps
Walking	96.33%	91.30%	93.75%	109	115
Downstairs	95.24%	100.00%	97.56%	20	20
Dragging Feet	88.89%	100.00%	94.12%	18	16
Running	57.14%	88.89%	69.57%	14	9
Upstairs	100.00%	95.00%	97.44%	18	20

Table A.3: Results for each activity with the Back Pocket smartphone placement.

	Recall	Precision	F-Score	GT Total Steps	DL Total Steps
Walking	91.67%	92.52%	92.09%	108	107
Downstairs	90.00%	100.00%	94.74%	20	18
Dragging Feet	61.11%	100.00%	75.86%	18	11
Running	50.00%	87.50%	63.64%	14	8
Upstairs	85.00%	100.00%	91.89%	19	17

Table A.4: Results for each activity with the Jacket smartphone placement.

	Proposed Model	Lee *et al.*	Edel *et al.*
Walking	94.16%	100.00%	90.79%
Downstairs	93.51%	92.68%	81.36%
Dragging Feet	71.43%	97.37%	68.52%
Running	54.39%	88.57%	28.27%
Upstairs	88.61%	96.34%	80.88%

Table A.5: Recall values of the proposed model and the models proposed by Lee *et al.* and Edel *et al.* for the different activities analysed.

	Proposed Model	Lee *et al.*	Edel *et al.*
Walking	92.01%	45.06%	61.92%
Downstairs	98.63%	71.70%	60.76%
Dragging Feet	100.00%	51.75%	52.86%
Running	93.91%	82.67%	24.39%
Upstairs	98.59%	63.20%	64.71%

Table A.6: Precision values of the proposed model and the models proposed by Lee *et al.* and Edel *et al.* for the different activities analysed.

	Proposed Model	Lee *et al.*	Edel *et al.*
Walking	93.07%	62.13%	73.63%
Downstairs	96.00%	80.85%	69.57%
Dragging Feet	83.33%	67.58%	59.68%
Running	68.89%	85.67%	26.32%
Upstairs	93.33%	76.33%	71.90%

Table A.7: F-Score values of the proposed model and the models proposed by Lee *et al.* and Edel *et al.* for the different activities analysed.

	Proposed Model	Lee *et al.*	Edel *et al.*
Texting	87.08%	97.77%	88.83%
Right Pocket	90.12%	98.27%	84.18%
Back Pocket	94.48%	99.03%	87.82%
Jacket	87.78%	97.75%	88.83%

Table A.8: Recall values of the proposed model and the models proposed by Lee *et al.* and Edel *et al.* for the different smartphone placements analysed.

	Proposed Model	Lee *et al.*	Edel *et al.*
Texting	89.08%	39.41%	63.86%
Right Pocket	85.64%	34.14%	52.72%
Back Pocket	88.18%	29.67%	54.75%
Jacket	89.27%	42.86%	70.17%

Table A.9: Precision values of the proposed model and the models proposed by Lee *et al.* and Edel *et al.* for the different smartphone placements analysed.

	Proposed Model	Lee *et al.*	Edel *et al.*
Texting	88.32%	56.18%	74.30%
Right Pocket	87.82%	50.67%	64.83%
Back Pocket	91.20%	45.66%	67.45%
Jacket	88.52%	59.59%	78.40%

Table A.10: F-Score values of the proposed model and the models proposed by Lee *et al.* and Edel *et al.* for the different smartphone placements analysed.

ANNEX I

Indoor Data Set Acquisition Protocol

4. Step Detection Dataset

4.1. Introduction

The activities of the step detection dataset are intended for step detection under different types of user movement and smartphone placement. The types of movement in this data collection protocol are:

- Normal walking;
- Going down the stairs;
- Walking dragging your feet on the floor;
- Running;
- Jumping;
- Moving your feet while standing still (false steps).

The dataset includes samples collected with the Inertial Measurement Unit (IMU) of the smartphones placed in the various positions:

- Front pockets;
- Back pockets;
- Phone call;
- Texting;
- In hand;
- Purse or backpack;
- Jackets inside pocket and outside pocket.

And samples collected with the IMU of pandlets placed in the subject ankles.

The dataset for activity monitoring can be assessed through the path:
S:\Research and Development\Master Thesis\Theses\2019\Msc FreeStep - Gonçalo Rodrigues\Data.

4.2. Dataset collection Procedures

The movements of the subject described above will all be done in each sample. The type of smartphone placement will remain constant throughout each sample. One example of sample collection is the collection of data of all the movements above while the smartphone is in the volunteers' front pocket. The order and length of each activity will be described further down in the protocol section.

The volunteers should use trousers with two front pockets and two back pockets.

- Smartphone positions:
    - One of the above.

- External sensors:
    - 1 pandlet on the left ankle;
    - 1 pandlet on the right ankle.

4.3. Required Information

It is require to record individual information of the subject: name, age, weight, height, gender, leg height and shoe size. It is also required to record the subjects' footwear and type floor.

The sensors that should be recorded are:

- Accelerometer
- Gyroscope
- Magnetometer

No further recording options are required.

4.4. Protocol

The volunteers should do all of the activities during the entire sample. Each activity will occur in the location specified in the Fraunhofer Lisbon floor plants placed in the end of the document. A description of the several activities that are included in the dataset are described below:

4.4.1. Walking

The volunteer should walk in slow or normal speed while using the smartphones and external sensors in the required positions.

4.4.2. Walking down/up the stairs

The volunteer should walk down/up the stairs with the smartphone and external sensors located in the required positions.

4.4.3. Walking while dragging your feet

The volunteer should walk while dragging the feet on the floor with the smartphone and external sensors in the required positions.

4.4.4. Running

The volunteer should run at the desired speed along the drawn path with the smartphone and external sensors in the required positions.

4.4.5. Jumping

The volunteer should perform three jumps while standing still with the smartphone and external sensors in the required positions.

4.4.6. False Steps

The volunteer should move its feet while standing still with the smartphone and external sensors in the required positions.

The crosses (✖) present in the floor plants below represent stopping points. The volunteer will stop at the marked location for 5 seconds before starting the next activity.

The blue arrows (⟶) indicate walking activity.

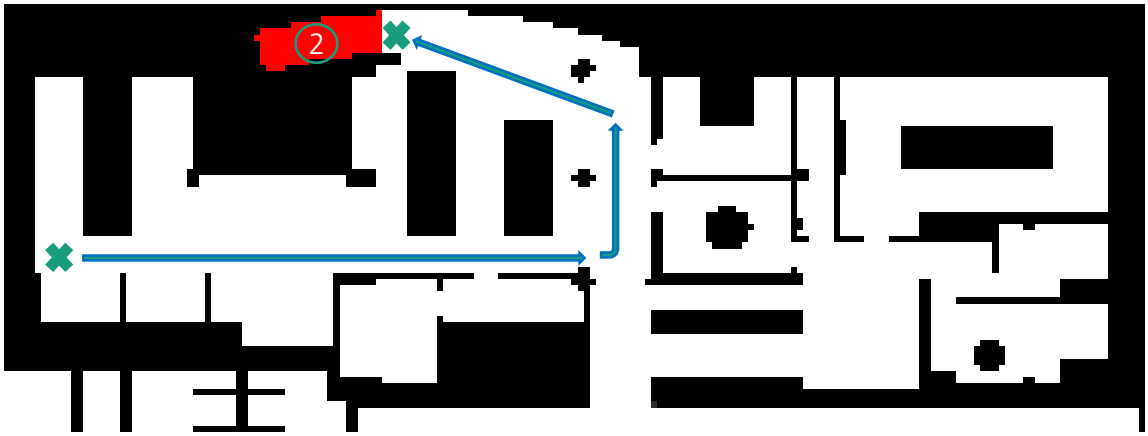


Figure 1 – Fraunhofer AICOS Lisbon first floor plant with the course and movements the volunteer has to perform

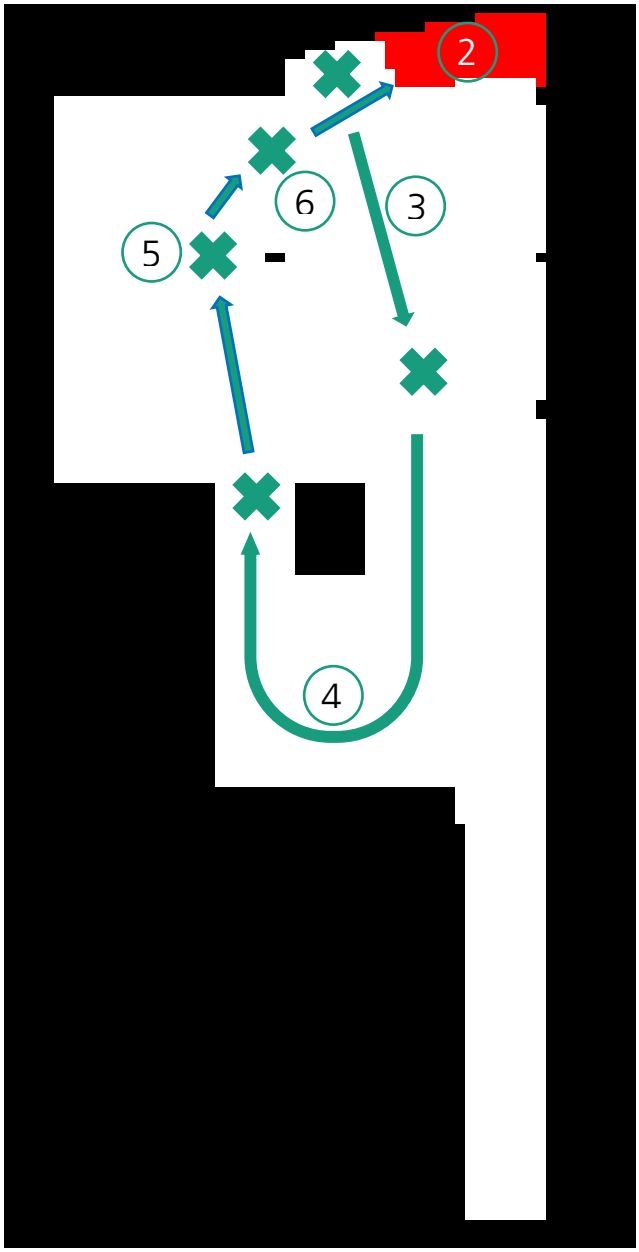


Figure 2 – Fraunhofer AICOS Lisbon basement plant with the course and movements the volunteer has to perform