João Pedro
Dias Ventuzelos

Aprendizagem Automática para Classificação de Distúrbios da Marcha Humana

Machine Learning Classification of Human Gait Disorders

João Pedro
Dias Ventuzelos

Aprendizagem Automática para Classificação de Distúrbios da Marcha Humana

Machine Learning Classification of Human Gait Disorders

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtenção do grau de Mestrado em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Doutor Filipe Miguel Teixeira Pereira da Silva, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor João Paulo Morais Ferreira, Professor Adjunto do Departamento de Engenharia Eletrotécnica do Instituto Superior de Engenharia de Coimbra

o júri / the jury

presidente / president

Professora Doutora Pétia Georgieva Georgieva
Professora Associada do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Professor Doutor Paulo Luis Serras Lobato Correia
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores do Instituto Superior Técnico (Arguente Principal)

Professor Doutor Filipe Miguel Teixeira Pereira da Silva
Professor Auxiliar do Departamento de de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (Orientador)

agradecimentos

Aos meus pais, por todo o amor, atenção, disponibilidade e carinho em toda a minha vida. Sempre me ajudaram em tudo e apoiaram. Foi devido a eles que consegui seguir os meus objetivos, sem que nunca me faltasse nada.

À minha irmã, pelo carinho e pela alegria que sempre me transmitiu, ajudando-me a nunca desanimar.

À minha namorada, Helena, pelo carinho, presença e força em todos os momentos. Foi sem dúvida um dos maiores pilares na minha caminhada universitária.

A todos os meus amigos, por todos os momentos proporcionados e pela amizade e companhia em todas as fases da minha vida.

Ao meu orientador e co-orientador, Prof. Filipe Silva e Prof. João Ferreira, por estarem sempre disponíveis e presentes para ajudar. Sempre foram muito prestáveis e graças a eles aprendi muito nesta última fase universitária.

Ao IrisLab da Universidade de Aveiro e ao Eurico Pedrosa, por terem disponibilizado os servidores e ajudado em qualquer problema, para que todo o trabalho pudesse ser feito da melhor maneira.

Palavras-Chave

Distúrbios da Marcha Humana, Forças de Reação no Solo, Classificação de Séries Temporais, Perceptron de Múltiplas Camadas, Redes Neuronais Convolucionais

Resumo

A análise computadorizada da marcha humana é normalmente usada por investigadores e médicos para detectar distúrbios, avaliar o progresso da terapia ou melhorar o desempenho atlético. Os avanços da tecnologia e dos instrumentos de medidas têm permitido a quantificação das características da marcha humana, como parâmetros cinemáticos e cinéticos, atividade eletromiográfica e consumo de energia. Em particular, a quantificação das forças de reação do solo (FRS) têm se revelado uma ferramenta importante no contexto da saúde. No entanto, a extração de características significativos e a sua interpretação a partir de grandes quantidades de dados ainda é uma tarefa desafiadora. Consequentemente, os métodos de aprendizagem automática estão a tornar-se populares para lidar com a alta dimensionalidade, dependências temporais, grande variabilidade e relações não lineares presentes nos dados de marcha humana. Esta dissertação tem como objetivo estudar a aplicação de técnicas de aprendizagem automática na classificação de distúrbios da marcha humana, utilizando o *dataset* anotado GaitRec. O *dataset* contém dados bilaterais 3D-FRS de indivíduos saudáveis, bem como de pacientes com lesões musculoesqueléticas no quadril, joelho, tornozelo e calcanhar. Este trabalho aborda o desenvolvimento de modelos de classificação capazes de diferenciar padrões de marcha normais vs. anormais (problema binário), bem como classificar distúrbios patológicos da marcha (problema multiclasse). O estudo está centrado na comparação entre os modelos clássicos totalmente conetados e as redes neurais convolucionais (CNNs). Adicionalmente, as séries temporais são pré-processadas e convertidas numa imagem bidimensional que é aplicada a uma rede convolucional 2D para explorar assimetrias nas FRS bilaterais. Os resultados obtidos mostram que a rede com múltiplas camadas totalmente conetadas supera em 1% a rede convolucional. O classificador binário alcançou uma precisão em torno de 99,0%, enquanto a precisão do modelo multiclasse é de cerca de 97,2%. Os resultados preliminares obtidos com a rede convolucional 2-D baseada em imagens são inferiores, o que pode indicar que são necessários esforços adicionais para tirar proveito dessa abordagem.

Keywords Human Gait Disorders, Ground Reaction Forces, Time Series Classification, Multilayer Perceptron, Convolutional Neural Networks

Abstract

Computerized human gait analysis is commonly used by researchers and physicians to detect disorders, evaluate therapy progress, or improve athletic performance. Advances in instrument and measurement technology has allowed the quantification of human gait characteristics, such as kinematic and kinetic parameters, electromyographic activity and energy consumption. In particular, the quantification of ground reaction forces (GRFs) has proved to be an important tool in the healthcare context. However, the extraction of meaningful features and their interpretation from the amount of complex data is still a challenging task. Consequently, machine learning methods are becoming popular to deal with the high-dimensionality, temporal dependencies, strong variability, and non-linear relationships present in human gait data. This dissertation aims to study the application of machine learning techniques for the classification of human gait disorders, using the annotated GaitRec dataset. The dataset contains bi-lateral 3D-GRF data from healthy individuals, as well from patients with musculoskeletal impairments at the hip, knee, ankle and calcaneus. This work addresses the custom development of classification models capable of differentiating normal vs. abnormal gait patterns (binary problem), as well as classifying pathological gait disorders (multi-class problem). The focus is on the comparison between classical fully-connected models and 1D convolutional neural networks (CNNs), in terms of prediction accuracy. Additionally, pre-processed time series are converted into a two-dimensional input image, which is applied to a 2D-CNN to explore asymmetries in bilateral GRFs. The results obtained show that the fully-connected model outperforms in 1% the 1D-CNN model. The binary classifier achieved a prediction accuracy around 99.0%, while the multi-class accuracy score is around 97.2%. The preliminary results achieved with the image-based 2-D CNN are much lower which may indicate that additional efforts will be needed to take advantage of this approach.

Contents

List of Figures

List of Tables

Acronyms

A Ankle

AI Artificial Intelligence

ANN Artificial Neural Network

API Application Programming Interface

BM Boltzmann Machine

C Calcaneus

CNN Convolutional Neural Network

COP Center of Pressure

DBN Deep Belief Networks

EMG Electromyography

FDN Feedforward Deep Networks

FN False Negatives

FP False Positives

GAN Generative Adversarial Networks

GD Gait Disorder

GRF Ground Reaction Force

H Hip

HC Healthy Control

K Knee

KNN K-Nearest Neighbor

LSTM Long-short Term Memory

ML Machine Learning

MLP Multilayer Perceptron

NWS Non-Wearable Sensors

PCA Principal Component Analysis

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

SOM Self-Organizing Map

SVM Support Vector Machine

Tanh Hyperbolic Tangent

TN True Negatives

TP True Positives

WS Wearable Sensors

ZRB Zero Rule Baseline

Chapter 1

Introduction

Machine Learning (ML) is a branch of Artificial Intelligence (AI) dedicated to studying how to grant computer programs the ability to automatically learn and improve from data [1] [2]. Keeping in mind the developments of recent years, this field of investigation promises to have a noticeable impact on many sectors of the society. In particular, artificial neural networks and deep learning models are inspired by the network of neurons in the human brain, aiming to imitate humans in how they think and learn [2].

Currently, machine learning is used in several areas such as image classification [3], healthcare [2], self-driving cars [4], fraud detection [5], web-searches (Google RankBrain), games [6], among many others. Focusing on the healthcare field, convolutional neural networks (CNNs) are being applied with success in medical imaging for different purposes, such as organ segmentation, lesion detection or tumor classification. One of the benefits of machine learning is that it can manipulate and optimize very complex data sets located in very complex systems [2]. Healthcare needs a lot of attention and control and is constantly dealing with many variables. While tracking numerous variables is challenging for humans, it is something that computers deal very well with.

The primary objective of the work developed in this dissertation is to explore novel computational algorithms that can be used to automate the analysis of human gait patterns. The perspective is that these emerging technologies can be transferred into clinical practices and patient benefits in the near future.

1.1 Motivation

There are numerous examples of research studies combining higher computer processing and learning methods applied in the healthcare sector in order to solve, assist or automate problems [7] [8] [9] [10] [11]. Most of them implement the same pipeline: they use the information gathered by professionals, process that information and, then, use computer vision or machine learning methods to generate helpful tools and solutions for the healthcare sector.

Computerized human gait analysis is commonly used by researchers and clinicians to detect disorders, evaluate therapy progress, or improve athletic performance. Advances in instrument and measurement technology has allowed the quantification of human gait characteristics, such as kinematic and kinetic parameters, electromyographic activity and energy consumption [12]. In particular, the quantification of ground reaction forces (GRFs) has

proved to be an important tool in the healthcare context. However, the extraction of meaningful features and their interpretation from the amount of complex data is still a challenging task. Consequently, machine learning methods are becoming popular to deal with the high-dimensionality, temporal dependencies, strong variability, and non-linear relationships present in human gait data [13].

The recently created GaitRec-dataset [14] was one of the main reasons that motivated this work. GaitRec is a ground reaction force dataset, built over the years, containing complete information about the data recording process and protocol from a total of 2295 patients. It includes completely annotated 75,732 bi-lateral trials of 3D GRFs measurements corresponding to healthy and impaired patients, as well other relevant patients data such as, for example, about the sex, age and body mass. Fully-connected artificial neural networks (ANNs) and support vector machines (SVM) are well-established for gait classification [15] [16]. In contrast, relatively little work has explored convolutional neural networks for time series classification of gait patterns. Bearing this in mind, the major motivation for this work is to compare the performance of custom developed classifiers based on artificial neural networks applied to the GaitRec dataset.

1.2 Objectives

This dissertation proposes a machine learning framework for human gait classification based on the GaitRec dataset [14]. The study carried out addresses the custom development of classification models capable of differentiating normal vs. abnormal gait patterns (binary problem), as well as classifying pathological gait disorders (multi-class problem). The main goal is to compare the performance of classical multilayer perceptron models against convolutional neural networks, in terms of prediction accuracy and model robustness. Additionally, pre-processed time series are converted into a two-dimensional input image, which is applied to a 2D-CNN to explore asymmetries in bilateral GRFs. The intention is to explore a way to encode time series into an image, aiming to take advantage of CNNs for learning features and identifying data structures.

The study was developed from the GaitRec dataset considering the data source and potential biases which may affect the generalization ability of the models. The data preparation stage played a preponderant role in the performance of the supervised learning models. For example, the use of balanced datasets, preventing over representation of data from one class, and the appropriate choice of sub-sets of the larger dataset will be considered.

1.3 Dissertation Outline

The remainder of the dissertation is organized as follows:

- Chapter 2 reviews related work and the main concepts with relevance for this study. It starts with the human gait characterization and proceeds to the several ways of acquire human gait data. Machine learning methods for time series classification are reviewed. Finally, a brief literature review is provided.

- Chapter 3 starts by introducing the hardware and software tools used throughout the work, and a detailed description of the GaitRec dataset. The overall framework of the

present study is considered here, as well a description of the experiments to be carried out in the following chapters.

- Chapter 4 is dedicated to the development and evaluation of neural network models to solve the binary classification problem, including the dataset preparation, parameter tuning, and discussion of the main results.
- Chapter 5 follows a similar structure to the previous chapter, but it addresses a multi-class classification problem.
- Chapter 6 presents the study concerning the novel representation of the time series by an image that will be the input data of a 2-D convolutional neural network. It considers both the binary classification and the multi-class classification problems.
- Chapter 7 concludes the dissertation by providing a final discussion of the most relevant results and suggesting several points of future work.

Chapter 2

Literature Review

This chapter provides a review of the literature around the topics and context of the dissertation. Section 2.1 provides the fundamental concepts of the human gait and the general characteristics of a healthy and impaired patterns. Section 2.2 presents the most common methods and equipment used to acquire human gait data. It considers wearable and non-wearable sensors, how they work and their benefits. Section 2.3 provides an introduction to machine learning concepts and architectures, with particular emphasis on the multilayer perceptron, convolutional neural networks and support vector machines. Section 2.4 presents a review of some related works using machine learning techniques for human gait classification based on Ground Reaction Force (GRF) measurements.

2.1 Human Gait Characterization

Human bipedal motion is classified as one of the greatest changes in evolution because it allowed the human being for free use of the hands [17]. Normally, at around one year old that ability is obtained and, in most of the cases, is preserved throughout life. It seems to be a basic, instinctive ability that we perform calmly every day, but it is a remarkably complex and unique motor behavior. It is one of the principal characteristics that mostly involves both motor ability and adaptability. Consists of three primary components: locomotion, balance, and capacity to adapt to the environment [17].

Human gait depends on a cooperation of several human systems, such as the nervous, musculoskeletal and cardiorespiratory. It is impacted by age, personality, mood and sociocultural factors [18]. Almost all of the muscles of the human body are required to walk, as well as different cortical and subcortical structures. The previous sentence explains the reason for the extended learning phase in infancy and the generally difficult re-learning phase after injury [19]. In short, it involves the combination of sensory information within the nervous system, leading in motor commands to control muscle contraction and subsequent joint movement.

In order to perform a healthy gait it is necessary strength, balance, sensation and coordination. The period between successive points at which the heel of the same foot strikes the ground is characterized as the gait cycle. It is divided into two phases, the stance phase and the swing phase [20]. The complete gait cycle contains several subdivisions as described in the literature articles [17][18][20][21].

The human gait is characterized by alternating movements of the lower extremities in a rhythmic motion that results in the forward progression of the body. Considering the gait

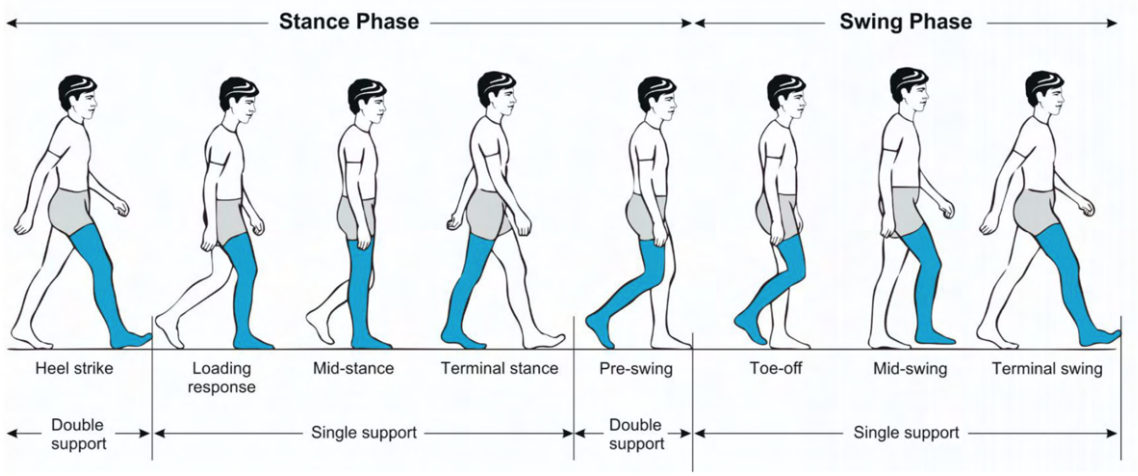


Figure 2.1: Motion phases in a healthy gait cycle (taken from [18]).

cycle depicted in Figure 2.1, we first have the heel strike, also known as initial contact. The sub-phase is included in the stance phase and marks the moment when the heel hits the ground. In addition, it marks the start of the joint loading response pattern. Moving next inside the stance phase, the loading response. The loading response phase is characterized by the flat foot floor contact. When the opposite foot is raised for the swing the loading phase ends and the mid stance begins. In this phase, to support the forward foot propulsion, the shank advances. Continuing on with the motion, when the body weight is aligned to the forefoot, the mid stance phase ends and we the terminal stance starts. The terminal stance is defined by the moment where the heel leaves the ground and it is extended until the opposite foot touches the ground.

Following the terminal stance sub-phase, to end the stance phase, the pre-swing part arrives. This sub-phase serves as a transition between the stance and swing phase, beginning with the initial contact of the opposite limb and ending with the toe leaving the ground. When the foot leaves the ground, the swing phase starts with the toe-off sub-phase (also known as initial swing). The initial swing causes a flexion in the knee and ankle and lasts until the swing foot is opposite to the stance foot. Following the initial swing, the mid swing starts. In this phase, the thigh hits its maximum advancement and remains until the hip and knee flexion postures become equal. To end the gait cycle and the swing phase, the terminal swing. This phase concludes the limb progression through knee extension. After the terminal swing, the foot returns to the stance phase and to the heel strike sub-phase [20]. There is a period in the cycle where both feet are in contact with the ground that is called double support. This period can be omitted (during running), enlarged (during cautious/senile gait, weakness, or disequilibrium) or asymmetric (during limping gait) [17].

The analysis of the parameters of the human gait requires the use of several sensors and data processing methods. Among the most important gait parameters are the stride length, walking speed, support time and ground reaction forces. An understanding of the gait cycle and the various phases of the gait cycle is required to to assess and treat patients with different conditions, pathologies or injuries affecting their ability to walk.

Most of motion disorders are effortlessly identified to the naked eye. Hip, knee, ankle or calcaneus are an example of some parts of the human body that are connected to gait disor-

ders. They are commonly manifested due to joint replacements, fractures, ligament ruptures along with others. A person with a gait disorder generally has one of the subsequent systems or functions damaged: locomotor function, balance, postural reflexes, sensory function and sensorimotor integration, motor control the musculoskeletal apparatus and cardiopulmonary functions [18]. The previous sentence means that in order to have a healthy gait it is mandatory to have flawless all these functions and systems.

At the same time, gait disorders are one of the most frequent problems that neurologic patients experience. Existent in more than half of all bedridden subjects admitted to a neurologic service, they conduct to a privation of personal freedom and have overwhelming consequences, being the most popular reduced mobility and falls, with subsequent reduction in the quality of life and lifetime[17]. A study performed stated that the predominance of gait disorders rises from 10% in people aged 60-69 years to more than 60% in subjects aged over 80 years [22]. Other gait impairments are associated closely with poor quality of life and increased chance of mortality [17]. Falls are considered the most frequent cause to serious injuries in the elderly [23]. Since gait is particularly susceptible to any taunt to the nervous system, its judgement should be performed conscientiously in routine clinical practice.

2.2 Data Acquisition for Human Gait

Actually, evaluating and examining gait characteristics has a large importance in several areas, such as sports or clinical field. It exposes essential information about various gait parameters and prior diagnosis of diverse disorders. If meticulously acquired, spatial and temporal parameters of gait grant valuable diagnostic and therapeutical data [24]. Studying gait generally includes the measurement of kinetic and gait parameters, kinematic analysis and Electromyography (EMG) [25].

Sensors, wearable or not, are an effective way of gathering information nowadays. They capture and measure information in order to next evaluate efficiently the different gait parameters. The accessories used for the analysis can be divided into three categories: accessories that depend on Wearable Sensors (WS), accessories that depend on Non-Wearable Sensors (NWS) and accessories that combine the previous two [12].

Wearable sensors involve the placement of measuring devices on several parts of the body, such as feet, waist, knees, among others. Inertial sensors, electromyography, extensometers, force sensors, goniometers, gyroscopic sensors, accelerometers, magnetometers are some examples of wearable sensors. As stated before, their objective is to collect indicators related to kinematics, kinetics, and EMG. The kinematics characterize the movements of the main joints and segments of the lower extremity in the human gait process. The kinetics examine the forces and moments resulted from the movement. For that is regularly demanded the orientation of all the leg segments gathered from gait kinematics. An electromyography reveals the electric reaction produced from the muscles activity during the motion process. Most of the gait methods based on wearable sensors have been considered and presented as an affordable and and less complicated way to manage, by the laboratories, the patients in their everyday activities [25].

Non wearable sensors systems depend upon the utilization of controlled accommodations. There is where the sensors are placed and gather information about gait while the patient walks on a marked walkaway. They can be divided into two, the sensors based on floor sensors and the ones based on computer vision methods [12]. The floor sensor systems depends on

sensors placed along the floor on the commonly named "force platforms". The gait data is obtained through pressure sensors and ground reaction force sensors that measure the force exerted by the subject's feet on the floor when walking. The computer vision systems compile information on the gait through optic sensors and capture precise measurements of the distinct parameters through digital image processing.

Figure 2.2 displays the human gait analysis approaches accordingly to Prakash, C. et al. [26]. The ones used in this work are the sensor based on force platforms. It is important to mention that, as observable in Figure 2.2, there are several methodologies and sensors that can be applied to classify human gait disorders which are different from the ones used in this work. A general presentation of them will be made in the next sub-section.

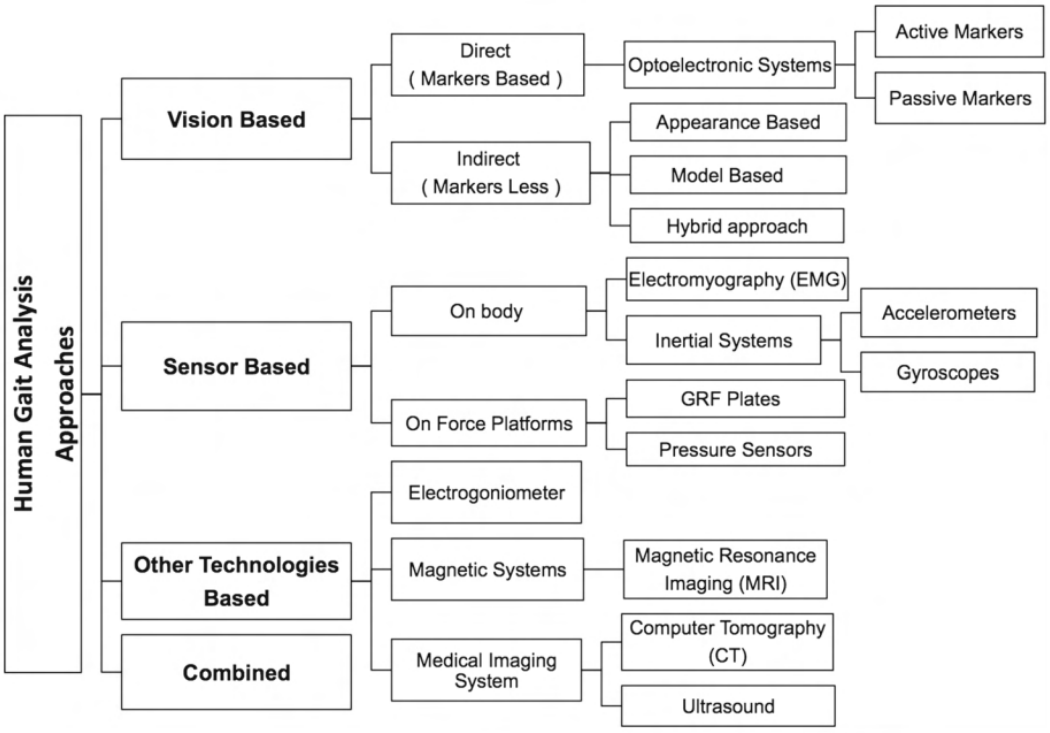


Figure 2.2: Types of human gait analysis (taken from [26]).

2.2.1 Wearable and Non-wearable Sensors

Goniometers are wearable sensors that measure angles of the lower human articulations, such as ankles, knees, among others. They determine the variation in the physical signal emanating from angular alterations. There are several types of goniometers, such as strain gauge-based, mechanical flexible, among others [27]. The first mentioned type, strain gauge-based, has a resistance that depends on how flexed the instrument is. The mechanical system obtains the angular variation by determining the longitudinal displacement of two parallel wires bent in the plane of rotation, i.e., manifested by measuring the knee joint during motion [27].

Nowadays, there are already electrogoniometers available in the market that quantify the flexibility. This type contains a potentiometer instead, introduced over the centre of rotation

Figure 2.3: Example of a goniometer (taken from [28]).

of the joint in observation [29]. When there is motion, the potentiometer returns an electrical output that can be analysed.

Ultrasonic sensors are wearable instruments that are used to study the propagation of waves in relation to an object. Knowing the speed of a signal, they can measure the time between the send and reception of a wave. The reception wave is a wave that is produced and reflected at the time that the sending one reaches the desired object. Knowing the time it is possible to know the distance between the two points.

Ultrasonic sensors are utilized to acquire short step and stride length and the separation distance between feet or between foot and floor [30]. Their range is in the interval of 1.7 cm and 450 cm [25]. Figure 2.4 represents an ultrasonic sensor system [31]. It comprises a micro-controller and ultrasonic sensors on separate boards. The sensor that transmits the information is attached to the heel of the patient.

Inertial sensors are one of the most generally used type of wearable sensors in gait analysis. They are electronic accessories used to determine angular velocity, acceleration, orientation, and gravitational forces for the considered body [12]. Inertial sensors are commonly formed by a junction of accelerometers and gyroscopes, but, occasionally, they also have magnetometers [25]. Initially, inertial sensors were used to study vibration and impact or movements at low velocities such as gait and running [25]. Actually, they evolved and have lots of different applications. Small, low-powered electromechanical sensors (MEMS technology) may be able to bridge the current gap between large laboratory systems and clinical systems, providing dynamic three-dimensional motion analysis.

Gyroscopes are active sensors that rely on a property which insinuates that all bodies that spin around an axis develop rotational inertia [12]. They are an angular velocity sensor

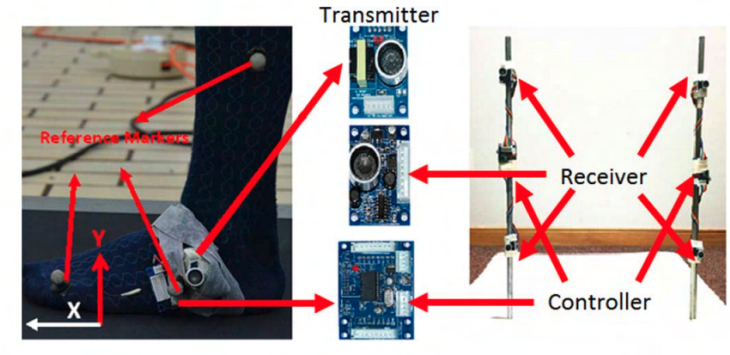


Figure 2.4: Example of an ultrasonic sensor system (taken from [31]).

included in numerous devices to measure motion and posture in subjects by gathering the angular rate [27]. Accelerometers are instruments that measure the acceleration forces that act on an object in relation to gravity, with the purpose of designate and supervise it's location and acceleration/velocity in space. Gyroscopes and accelerometers have always been commonly incorporated to produce a complete initial sensing system [32] [33]. At last, magnetometers are instruments used to measure the intensity, direction and orientation of magnetic fields.

Electromyography measures muscle contraction as an electrical reaction. This type of wearable sensor marked an important advance in gait analysis, since it allowed a better management of patients with neuromuscular disorders [25]. EMG is a proper technique to evaluates the walking performance by detecting the gait phase and interpreting variations in function.

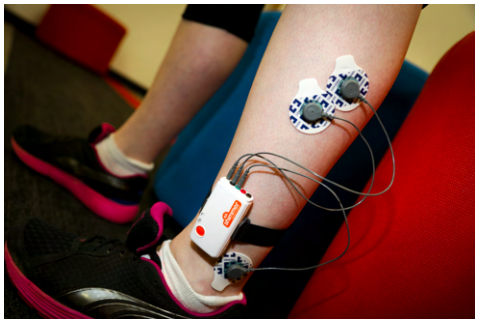


Figure 2.5: Example of a gait EMG system (taken from [34]).

Computer Vision / Image Processing are one of the most known methods used to categorize and analyse gait, inside the non wearable sensors. Computer vision has a lot of impact on the society because provides a majority of methods/alternatives that help to facilitate daily processes. The most common image processing system is composed by several cameras that are used to collect gait-related data with their lens. Methods like threshold filtering that transforms images into black and white, the pixel count that obtains the number of light or dark pixels, or background segmentation that discards the background of an image, are a few of the many existent ways to gather data to measure the gait variables [12].

One of the most relevant techniques used in gait are the ones based on depth measurement. In general they calculate and obtain a map of distances from a viewpoint [35]. Stereoscopic

vision is a method utilized to measure the depth of points in a scene. Firstly, it is necessary to obtain corresponding points in different images in order to obtain accurate measurements. For that it is necessary to calculate the similar triangles between the optical sensor, the light-emitter and the body, in order to create a model. Several images in multiple plane are required to be acquired to obtain a calibrated system.

Time-of-Flight systems, structure light or infrared thermography are some of another techniques used. The first, Time-of-Flight, depends on cameras that use a signal modulation. That signal measures the distance based on the phase-shift principle [36]. Structured Light is the projection of a light pattern under geometric calibration on an object whose shape is to be recovered. Three-dimensional information is acquired by examining the deformation of the projection of the pattern onto the scene with respect to the original projected pattern [12]. A known device that uses this is the Kinect sensor. Infrared thermography creates visual images based on surface temperatures [12]. The capacity to correctly obtain the infrared thermal intensity of the human body is possible due to the human body skin's properties.

Two very well known motion capture systems are the Optitrack [37] and Vicon [38]. They work with a variety of areas, from games to science. Figure 2.6 and 2.7 represent an example of a computer vision extraction method and an example of a computer vision system.

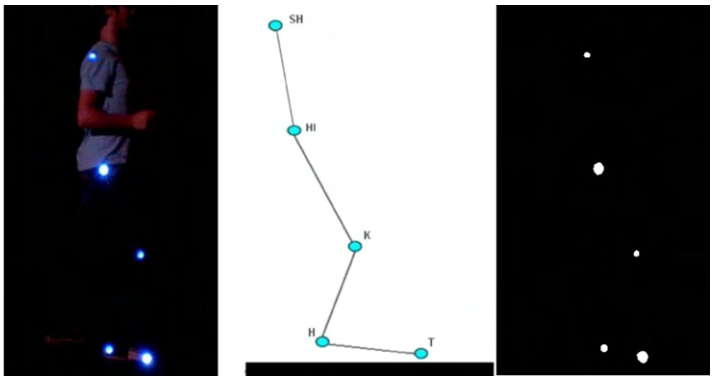


Figure 2.6: Example of a computer vision extraction method (adapted from [39]).

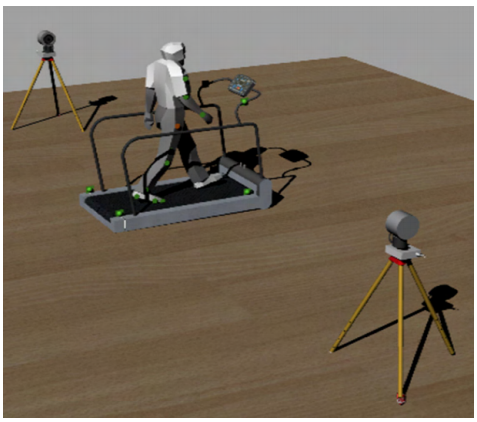


Figure 2.7: Example of a simple computer vision system (taken from [40]).

2.2.2 Force and Pressure Sensors

Force and Pressure Sensors can be associated to wearable or non-wearable sensors. Force sensors measure the ground reaction forces existent under the foot when moving. They send back a voltage that is proportional to the pressure obtained [25]. Pressure sensors are most of the times appropriate to measure the pressure distribution of the plantar foot and to detect step and gait phase [41] [42]. Generally, the most frequent wearable models are resistive, piezoelectric, capacitive and piezoresistive and the most frequent non-wearable ones are the ones present in the floor sensors. Ground reaction forces are defined by the force that the ground applies on a body that is touching it [43]. When the body is not in movement, the GRF coincides to its weight. Otherwise, the GRF increments thanks to the acceleration forces.

Figure 2.8: Example of a Tekscan FlexiForce pressure sensor (piezoresistive) (taken from [44]).

Considering the wearable ones, each is implemented depending on different factors, like range of pressure or sensitivity [25]. This type of sensors belong to both sensor types (wearable and non-wearable) because they are not only integrated to instrumented shoes or baropodometric insoles to measure the ground reaction forces but also are applied on the floor in "force platforms" or instrumented walkways [12].

When on the floor, gait is acquired by pressure or force sensors and moment transducers at the moment that the subject walks on them. There are two types of floor sensors, the force platforms and pressure measurement systems. They are different because, even if the force platforms quantify also the center of pressure, they don't measure instantaneously the force vector applied. In the other hand, despite pressure measurement systems are effective quantifying the pressure patterns existent under a foot over time, they cannot measure horizontal or shear elements of the applied forces [12].

Figure 2.9 represents an example of an instrumented shoe that aims to gather ground reaction forces. It is displayed the outside and inside picture of the shoe. Figure 2.10 represents an example of a floor sensor that uses pressure sensing technology to determine key variables for gait or balance [45]. The information gathered to build the database mentioned and used in this dissertation was obtained with this type of sensors, more properly gathered with the Kistler type 9281B12 force plates [46].

Figure 2.9: Example of an instrumented shoe prototype (adapted from [10]).

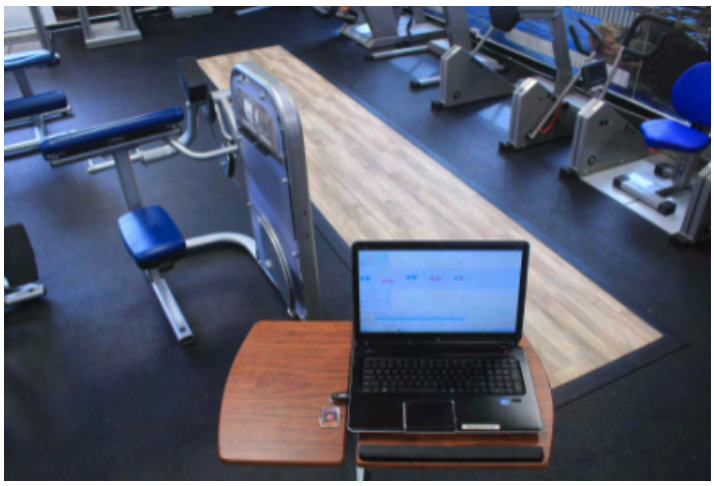


Figure 2.10: Example of a floor sensor (taken from [45]).

2.2.3 Wearable vs. Non-Wearable Sensors

Considering all existing systems and comparing them by area, some are better than others in some situations and worse in others. Therefore the previous sentence is also applied to the gait sensor systems. In the next figure is shown a comparison between wearable and non-wearable systems in regard to its advantages and disadvantages.

System	Advantages	Disadvantages
NWS	- Allows simultaneous analysis of multiple gait parameters captured from different approaches - Non restricted by power consumption - Some systems are totally non-intrusive in terms of attaching sensors to the body - Complex analysis systems allow more precision and have more measurement capacity - Better repeatability, reproducibility and less external factor interference due to controlled environment. - Measurement process controlled in real time by the specialist.	- Normal subject gait can be altered due to walking space restrictions required by the measurement system - Expensive equipment and tests - Impossible to monitor real life gait outside the instrumented environment
WS	- Transparent analysis and monitoring of gait during daily activities and on the long term - Cheaper systems -Allows the possibility of deployment in any place, not needing controlled environments - Increasing availability of varied miniaturized sensors - Wireless systems enhance usability - In clinical gait analysis, promotes autonomy and active role of patients	- Power consumption restrictions due to limited battery duration - Complex algorithms needed to estimate parameters from inertial sensors - Allows analysis of limited number of gait parameters - Susceptible to noise and interference of external factors not controlled by specialist

Figure 2.11: Comparison between wearable and non-wearable systems (taken from [12]).

2.3 Machine Learning for Time Series Classification

As mentioned before, ML is the study of how it is possible to grant computer programs the ability to automatically learn and improve from data. The purpose is to teach a computer, defined as a mere machine without self thinking, how to learn and make decisions just like a human [47]. ML algorithms can be classified into three types: supervised, unsupervised, and reinforcement [47]. Table 2.1 provides examples of learning techniques and their usage. Supervised learning algorithms contain a dataset where each sample has a corresponding label or target. Unsupervised learning algorithms also have a dataset with features, but without any corresponding label. The algorithm must learn from the given unlabeled data the important and useful properties that can be gathered. Between them, semisupervised learning algorithms deal with partially labeled data [47]. Reinforcement learning algorithms learn from interactions with the environment and with their own experience. Generally it helps to take decisions sequentially.

In the context of supervised methods, there is a distinction between shallow learning and deep learning [7]. Shallow learning rely on learning from data described by pre-defined features. Linear Regression, Decision Tree, Support Vector Machine (SVM), Random Forest, Naïve Bayes, and K-Nearest Neighbor (KNN) are examples of shallow learning. Deep learning is influenced by the biological neural networks' architecture and behavior. Relies on the notion of multi-layer Artificial Neural Network (ANN) with the principal objective of learning data representations automatically. Normally, the term 'deep' is related to the number of layers of the several existing network structures [7]. Some of the most known structures are the Deep Belief Networks (DBN), Feedforward Deep Networks (FDN), Boltzmann Machine (BM), Generative Adversarial Networks (GAN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Long-short Term Memory (LSTM).

Supervised Learning	**Unsupervised Learning**	**Reinforcement Learning**
Artificial Neural Networks - Classification and Regression	Self-Organizing Maps - Feature Detection and Grouping	Markov Decision Process - Control and Automation
Convolutional Neural Networks - Computer Vision	Boltzmann Machines - Recommendation Systems	Q-Learning - Autonomous driving (lane changing)
Recurrent Neural Networks - Time Series Analysis	Auto Encoders - Recommendation Systems	-

Table 2.1: Example of some machine learning types of algorithms.

Time series are common in many real-world applications ranging from health care, automated disease detection, human activity recognition, cyber-security, and finance. The increasing availability of temporal data has contributed to interest on new applications and algorithms based on time series. This section provides a brief description of machine learning techniques for time series classification problems, namely the multilayer perceptron, convolutional neural networks and support vector machines. Except the algorithms based on deep learning, all the others require feature engineering as a separate task before the classification is performed. This can result in loss of some relevant information and the increase of the development time. On the contrary, deep learning models already incorporate, internally, this kind of feature engineering.

2.3.1 Artificial Neural Networks (ANN)

Artificial Neural Networks are a machine learning method that aims to mimic the human neural networks in the learning process. The human brain contains neurons and axons. The neurons are what the brain uses to process information and the axons are what establishes the connection, transmitting signals between them (electric signals, synapses). Chemical substances are released from the synapses and enter the neurons dendrites, increasing or decreasing the electrical potential of the neurons cell body. The neuron activates or deactivates whenever the input is greater than a defined value.

The architecture of an artificial neural network is similar to the previous description. They are formed by interconnected neurons, known also as nodes, that interact with each other through axons, known as edges. When providing an input value to a neural network, it processes it and returns a response. The neuron is activated only if that value is higher than a given threshold.

Figure 2.12 represents the nodes, edges/weights and sum/activation function of a neural network. The nodes send information to the next layer of nodes through edges. Each edge has an associated weight that can be adjusted based on experience plus a different weight named as bias. The previous sentence describes the parameters of the network. If the sum of the connected edges satisfies a defined threshold, known as the activation function, it will

trigger a neuron at the next layer. However, if not, the activation will not be performed. All the weights are unique to make sure that the nodes do not all return the same outcome. Equation 2.1 provides the mathematical representation of the artificial neuron's operation, as defined previously:

$$y(x) = a(\sum_{i}^{n} x_i w_i + b) \tag{2.1}$$

where "x" represents the inputs, "w" the weights, "b" the bias function and finally "a" the activation function. If the result of the expression is bigger than 0 it is returned 1, otherwise 0 [48].

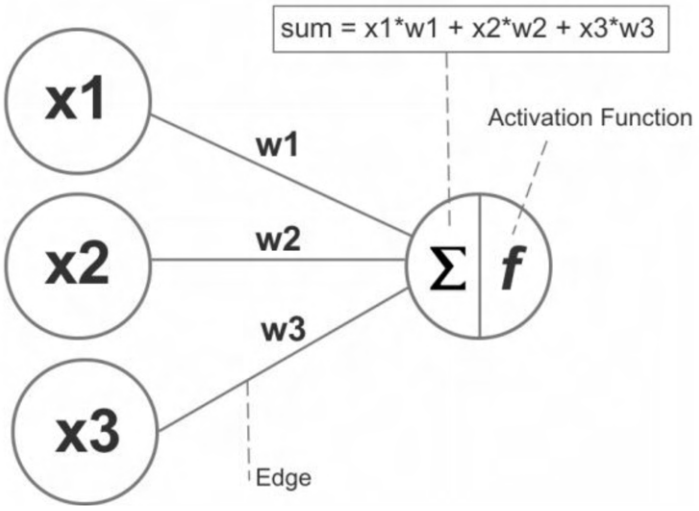


Figure 2.12: Illustrative example of nodes, edges/weights and sum/activation function (taken from [49]).

Neural networks, in a general way, are composed by three major layers, the input, the hidden and the output. The input layer gets all the data that contains all the features to be used, the hidden layer analyses and treats that same data and the output layer presents the results. The hidden layer can be composed by only one or in the other hand by multiple rows that will process the data. Actually, there are several ways to assemble the neural network nodes. Figure 2.13 represents an example of a feed-forward artificial neural network with 3 hidden layers.

Training a network means comparing the model's predicted output to the actual one. By measuring their difference it is obtained the cost value. The aim of training is to decrease that cost continuously till the prediction nearly matches the appropriate output. Adjusting the network's weights until the lowest possible cost value is achieved through the back-propagation algorithm. Instead of only operate from left to right, back-propagation works in reverse, from the output to the input layer. The process cycles through the training data several times (referred to as an *epoch*). Normally, it is advised to randomize the order of the presentation of the data each time [48]. The following subsections discuss various components of every neural architecture that are at the core of its performance, namely, activation functions, weight initialization and nonlinear optimizers.

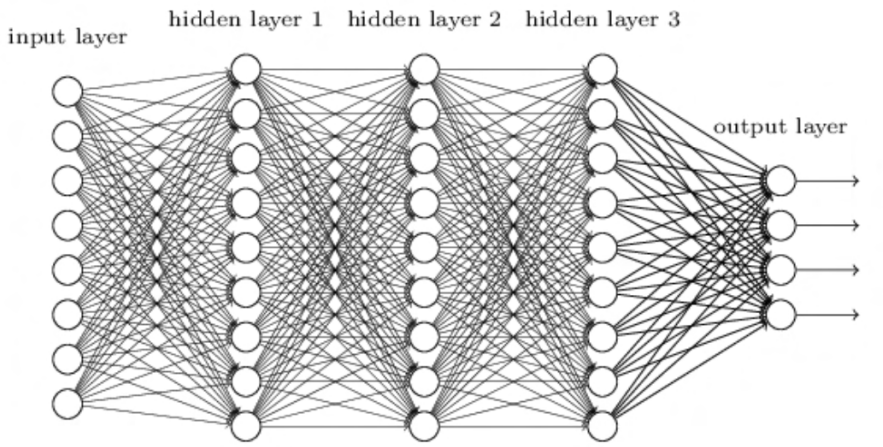


Figure 2.13: Example of an artificial neural network (taken from [50]).

Activation Functions

An activation function is meant to define, within a neural network, the activation of a neuron, depending on his input and weight. If activation functions did not exist, a layer were only be expressed by two linear operations, as represented in the following expression:

$$y(x) = (\sum_{i}^{n} x_i w_i + b) \tag{2.2}$$

where "y" represents the output, "x" the inputs, "w" the weights and "b" the bias function.

Therefore, it would only be able to learn linear transformations coming from the input data. In consequence, the hypothesis space of the layer would be the set of all possible linear transformations of the input data into a 16-dimensional space, as stated in [51]. The previous defined hypothesis space is very limited and would not benefit from multiple layers, considering that a deep stack of linear layers would still implement a linear operation. This means that, with the addiction of more layers, the hypothesis space would not increase. Thereby, with the purpose of achieve a valuable hypothesis space that could be favoured from deep representations, it is necessary an activation function. Table 2.2 describes some of the existent activation functions and their functions and way of working.

Weight Initialization

An important characteristic in this type of network is to set the weights correctly and the in the best possible way. An easy and simple way is to set them all to zero. The problem with this implementation is that every neuron will have identical activations and consequently gradients and therefore the same parameter update. The previous sentence emphasizes that is necessary to came up with a solution of how to distribute the initial weights.

The most known layer weight initializers are the random normal, random uniform, Glorot normal, Glorot uniform, He normal and He uniform. The normal ones generate a normal distribution and the uniform ones a uniform one. Depending on the problem, each one of them should be studied in order to encounter the one that fits the best.

Activation Function	Function	General Explanation
Sigmoid	$y(x) = \frac{1}{1+exp^{-x}}$	Provides a gradient in the range of 0 to 1. As a downside it is associated with computational problems such as the vanishing gradient.
Tanh (Hyperbolic Tangent)	$y(x) = \frac{e^x - e^{-x}}{e^z + e^{-z}}$	It is similar to the sigmoid function but its range is from -1 to 1.
ReLU (Rectified Linear Unit)	$y(x) = max(0, x)$	The most popular activation function when working with deep neural networks. It transforms the negatives values into 0 and maintains the positive ones with their respective value. It has better gradient propagation and faster convergence compared to other activation functions. It still can suffer from vanishing gradient problems.
Softmax	$y(x) = \frac{exp(z_i)}{\sum_j exp(z_j)}$	Commonly adopted as output layer in classification neural networks with several classes. It outputs the probability of each class.

Table 2.2: Most known activation functions.

Nonlinear Optimizers

As stated before, neural networks have weights that contain the information acquired from the exhibit to training data. In the beginning, all the weight matrices have random values, process known as random initialization. After the initialization it is then necessary to gradually adjust the previous weights, having into account a feedback signal. The previous gradual adjustment is also known as training. In general, optimizers define how neural networks learn. They find the values of parameters such that a loss function is at its lowest. It is important to mention that the optimizers have no perception of the loss, so they need to find the minimum point of its associated function without any knowledge. In machine learning, an algorithm needs to know if it is necessary to change the value of the weights and, if yes, how much.

Gradient descent consists in taking small steps iteratively until the correct weights are found and defined. A problem appears since the weights are only updated once after the observation of the entire dataset [52]. The gradient is typically large and data can only make large "jumps". For that reason, it may just approach its optimal value without being able to reach it. One solution to that problem is to update the parameters more frequently like in the case of the stochastic gradient descent. For every epoch, the gradient descent algorithm uses the function:

$$\theta_{i+1} = \theta_i - \alpha \cdot \nabla_\theta J(\theta) \tag{2.3}$$

, where "θ_{i+1}" represents the next position, "θ_i" the current position, "α" the waiting factor, "∇_θ" the gradient term and "$J(\theta)$" the loss function.

Stochastic Gradient Descent (SGD) updates the weights after observing each data point instead of the entire dataset [52]. Despite that, a problem emerges here too. The

gradient can make many "noisy" jumps that move away from the optimal values and it is influenced by every sample. It uses the same function as the gradient descent and applies it to every sample in every epoch [52]. To solve that it emerges the mini-batch gradient descent that only updates the parameters after a few samples.

Stochastic Gradient Descent + Momentum introduces another way to reduce the noise of stochastic gradient descent, including the concept of momentum. The parameters of a model may have the tendency to change in one direction. Generally, if examples follow a similar pattern with the momentum, the model can learn faster by paying little attention to the few examples that throw it off time to time [52]. A problem appears here too, since choosing to ignore samples simply because it isn't typical it may be a costly mistake. Adding an acceleration term solves that problem. For every sample in every epoch, it uses the functions:

$$\begin{aligned} \upsilon_i &= \gamma \cdot \upsilon_{i-1} + \eta \cdot \nabla_\theta J(\theta) \\ \theta &= \theta - \alpha \upsilon_i \end{aligned} \tag{2.4}$$

, where "υ_i" is the actual sequence, "γ" is an hyper-parameter that takes values from zero to one, "υ_{i-1}" is the previous sequence, "η" is a simplification for "$1-\gamma$", "∇_θ" the gradient, "$J(\theta)$" the loss function and "α" the learning rate.

Stochastic Gradient Descent + Momentum + Acceleration represents an evolution to the previous algorithm [52]. The model is training gaining momentum and the weights are becoming larger. It reaches a time where it finds an odd sample and, due to momentum, it deliberates very little of it but discarding it leads to a loss decrease that is not drastic [52]. This is the point where the model decelerates the weight updates, making them smaller again and allowing future samples to fine-tune the current model. For every sample in every epoch, it uses the functions:

$$\begin{aligned} \upsilon_i &= \gamma \cdot \upsilon_{i-1} + \alpha \cdot \nabla_\theta J(\theta - \gamma \cdot \upsilon_{i-1}) \\ \theta &= \theta - \upsilon_i \end{aligned} \tag{2.5}$$

, where "υ_i" is the actual sequence, "γ" is an hyper-parameter that takes values from zero to one, "υ_{i-1}" is the previous sequence, "∇_θ" the gradient, "$J(\theta)$" the loss function and "α" the learning rate. The learning rate, here, needs to be scaled by a "$1-\gamma$" factor due to the omission of that variable.

Adaptive learning rate optimizers are able to learn more along one direction than another. With an adaptive loss there are more degrees of freedom to increase the learning rate in one direction and decrease in another [52].

AdaGrad, AdaDelta, Adam and RMSProp are some examples of adaptive learning rate optimizers. Starting with adagrad, for every parameter θ_i in every epoch t:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii}} + \epsilon} \nabla_{\theta_{t,i}} J(\theta_{t,i}) \tag{2.6}$$

In the optimizer update, the $G_{t,ii}$ is the sum of squares of the gradients with respect to θ_i parameter until that point. The previous formula is $G_{t,ii} = G_{t-1,ii} + \nabla^2_{\theta_{t,i}} J(\theta_{t,i})$. The problem with this is that the G term is monotonically increasing over iterations so the learning will decay to a point where the parameter will no longer update and there is no learning [52]. The previous sentence means that $\frac{\eta}{\sqrt{G_{t,ii}}+\epsilon}$ will tend to 0. As the iterations go on, it learns slower and slower even though the optimal trajectory is quite clear.

To solve the previous problem appears Adadelta. In adadelta, for every parameter θ_i in every epoch t:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{E[G_{t,ii}] + \epsilon}} \nabla_{\theta_{t,i}} J(\theta_{t,i}) \tag{2.7}$$

It reduces the influence of past squared gradients by introducing a gamma weight to all of those gradients, reducing their effect by an exponential factor:

$$G_{t,ii} = \gamma G_{t-1,ii} + (1-\gamma) \nabla^2_{\theta_{t,i}} J(\theta_{t,i}) \tag{2.8}$$

The $\frac{\eta}{\sqrt{E[G_{t,ii}]+\epsilon}}$ does not "explode" which prevents the learning rate from tending to 0. With this, adadelta has learning rate updates for every single parameter [52].

Adding a momentum to improve adadelta, gives rise to Adam [53]. The only necessary change is to add the expected value of past gradients. This means that initially it starts slow but over time it gains speed, which is similar to momentum. In Adam [53], for every parameter θ_i in every epoch t:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{E[G_{t,ii}] + \epsilon}} \nabla_{\theta_{t,i}} J(\theta_{t,i}) \times E[g_{t,i}] \tag{2.9}$$

where,

$$G_{t,ii} = \gamma G_{t-1,ii} + (1-\gamma) \nabla^2_{\theta_{t,i}} J(\theta_{t,i})$$
$$E[g_{t,i}] = \beta E[g_{t-1,i}] + (1-\beta) \nabla_{\theta_{t,i}} J(\theta_{t,i})$$

The momentum part is the $E[g_{t,i}]$. Adam can take different size steps for different parameters. With momentum for every parameter it can also lead to faster convergence. If acceleration is added to Adam it originates Nadam.

RMSProp tunes the learning rate for each parameter in a similar way to AdaGrad but uses a moving average of gradients to make the optimization more suitable for optimizing non-convex cost functions [52].

2.3.2 Convolutional Neural Networks (CNN)

Convolutional neural networks (CNN) are a particular type of neural network that processes grid-like topology data [47]. They are commonly used for computational vision, the area that is concerned with the image and video processing. Some of its applications are autonomous cars, pedestrian detection, digit classification etc.

As mentioned before, convolutional neural networks are very efficient processing data with a grid-like topology, handling 1-D, 2-D and 3-D data [54]. They are able to gather, from large datasets, solid levels of abstraction and features by implementing to the input data a convolution operation. As stated in [55], convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

Generally, convolutional neural networks are composed of convolution layers, pooling layers and normalization layers [7]. All these layers have a group of filters and weights associated. The convolutional layers generate a feature map automatically from the input data. The pooling layers decrease the size of representation and conceive a more robust convolution layer output [56].

Frequently, two types of pooling layers are utilized, the max pooling and the average pooling. The previous layers, convolution and pooling, all have activation functions that, as stated, will decide when a neuron is activated or not [57].

Convolution Layers

Convolution is a mathematical operation which slips one function over another and determine the integral of their point-by-point multiplication [47]. In general, the convolutional layer is considered as the crucial building block of a CNN [47]. Figure 2.14 represents a CNN layer example with the receptive fields marked. It is observable that neurons in the layer 1 are not connected to each one of all the pixels in the input layer and that neurons in the layer 2 are also not connected to all of the layer 1. Instead, they are only connected to the receptive fields of each layer. That happens because, in that manner, the network is capable of detecting the small low-level features, in order to gather them into larger higher-level features in the consequent layer. The previous process occurs continuously and is frequent in images of the real-world.

The receptive field denotes a small image that represents the neuron's weights. The sets of weights represent the filters, or convolution kernels, and act as a mask in order to detect only what is desired. Everything else located in the neuron's receptive field will be ignored. A layer that has the same filter in all neurons outputs a feature map that emphasises the region of the image in where the filter is activated most. The filters are defined automatically during the training process. The convolutional layer learns the suitable filters for its task and the subsequent layers learn how to merge them towards more complex patterns.

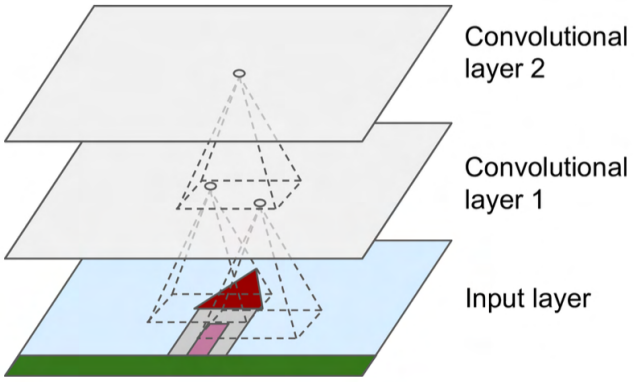


Figure 2.14: Convolutional layers (taken from [47]).

Pooling Layers

Pooling layers reduce the input image with the aim of having a better computer processing performance and a decrease number of parameters. Identically to the convolutional layers, each neuron only interacts with a restricted number of neurons in a small receptive field of the past layer. It is also necessary to define all of its parameters such as size, stride or the padding type in order to create a pooling layer that will gather the most important aspects of an image. A pooling neuron aggregates the inputs using an aggregation function and does not have weights.

Figure 2.15 represents one of the most frequent pooling types, the *max pooling layer*. It has a 2x2 pooling kernel with stride equal to 2 and no padding. In the *max pooling layer*, the max input value in each receptive field is gathered and proceeds to the next layer. The remaining values are dropped.

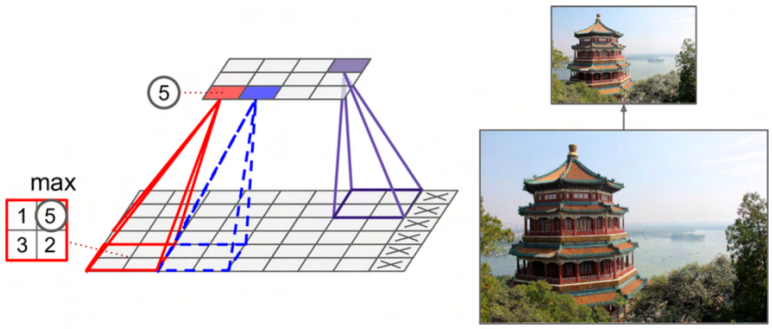


Figure 2.15: Example of a max pooling layer (taken from [47]).

A different pooling that can be considered is the average pooling: it takes all the values from the pooling kernel and makes an average value from them. While the max pooling chooses from all values the maximum one to pass to the next layer, the average pooling performs an average of them all and it sends to the next layer the average value. A general CNN is composed by an input layer that receives an image, convolutional layers, where each one of them is commonly proceeded by a Rectified Linear Unit (ReLU) layer, a pooling layer, a fully connected layer and a final output layer (see Figure 2.16). In the convolutional/pooling processes, the image is reduced in size while the algorithm obtains more and more feature maps. After that, a feedforward neural network with various fully connected layers will process what comes from the previous layers, producing a prediction at the final layer.

The standard Convolution Neural Network is the 2D CNN. As stated previously, is commonly used on image data [58]. In this type of CNN the kernel moves on the data along two dimensions. CNN acquire spatial features from data using a kernel, feature that other networks cannot do. However, 1D and 3D CNNs find application with time-series data and 3D volumetric data [58], respectively. A 1D convolutional layer works similarly as a 2-D one, where a layer slides multiple kernels over a sequence, generating a 1-D feature map for each kernel. After that, the kernels will be able to recognize a single sequential pattern. Activity recognition, sensory data, audio and text data are examples of their applications since the data is represented by a time-series. The 3-D convolutional neural networks operates in three dimensions since the kernel moves along them. Like 2D CNN it is commonly used on image data, but in this case on 3-D volumetric data [58]. Medical data and video are some of its many usages. Table 2.3 summarizes the operation and typical application of each of the convolutional models.

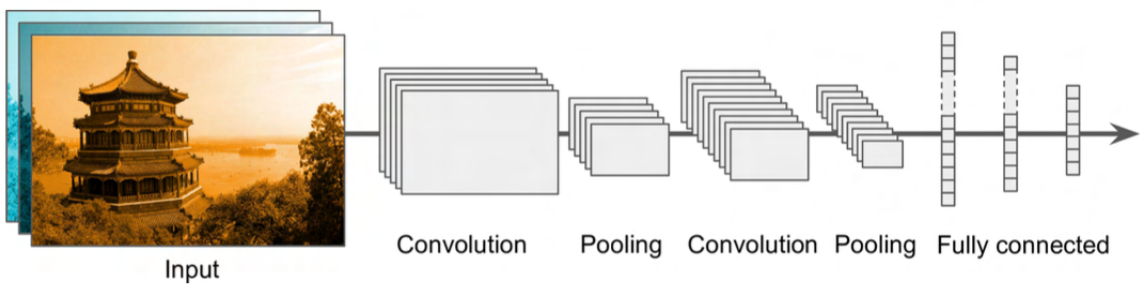


Figure 2.16: Common convolutional neural network architecture (taken from [47]).

	1-D CNN	2-D CNN	3-D CNN
Kernel	Moves in 1 dimension	Moves in 2 dimensions	Moves in 3 dimensions
Input and Output Data	2 Dimensional	3 Dimensional	4 Dimensional
Usages	Time-Series data	Image data	3-D Image data

Table 2.3: Different types of CNNs.

2.3.3 Support Vector Machines (SVMs)

Support Vector Machines (SVM) are machine learning models commonly used to perform classification and regression [47]. In addiction to the two previous examples, they can also perform outlier detection. Further, fully-connected ANNs and support vector machines are well-established for gait classification [15] [16]. SVM looks at the extremes of a given dataset and chooses a decision boundary, also known as a hyperplane, close to the extreme points in the dataset. If an algorithm has an unoptimized decision boundary it could result in greater misclassifications on new data, so for that reason, it is important to have the best decision boundary.

Support vectors are data points that the margin pushes up against all points that are close to the opposing class. Therefore, the algorithm indicates that only support vectors are significant whereas other training examples can be ignored. If the classes are linear separable, that is, they can be separated through a straight line, they represent a Linear SVM. If the classes are not linear, in other words, it is not possible to separate the classes with a single line, it is necessary to implement a Non-linear SVM. Thereby, it is required to use a function to transform the data into an higher dimensional space. One of the existent problems that this SVM has, is that transforming the data into an higher dimensional space, is very computational expensive. There are some kernel tricks to reduce these costs. A function that gathers as input vectors in the original space and give back the dot product of the vectors in the feature space is denominated a kernel function. Using that it is possible to implement the dot product between two vectors in order to map every point into a high dimensional space via some transformation. The most known kernel types are the polynomial kernel, the radial basis function kernel and the sigmoid kernel. The aim of the Non-linear SVM is to transform a non linear space into a linear one. Choosing the best kernel is a non-trivial task and may depend on a specific task at hand, no matter which kernel is chosen. It is necessary to tune the kernel parameters to get good performance from a classifier. A popular technique for that is the k-fold cross-validation.

2.4 An Overview of Gait Disorders Classification

As stated before, it is difficult to analyze and interpret the data produced from recordings during clinical gait analysis due to their high-dimensionality, temporal dependencies, variability, non-linear relationships and correlations within the data [13]. Over the last years, various approaches based on machine learning were proposed and published, in order to help clinicians in fix these problems [7]. In the human gait context, there are also a relevant literature applying different methodologies [10] [11]. They are useful to assist in the process of identification and categorization of specific gait patterns. The most common machine learn-

Advantages	Disadvantages
Effective in high dimensional spaces	Do not provide probability estimates
Effective in cases where the number of dimensions is greater than the number of samples (it is memory efficient)	Poor performance when the number of features is bigger than the number of samples
Different kernel functions for various decision functions	
Possibility to add kernel functions together to achieve even more complex hyperplanes	

Table 2.4: Advantages and disadvantages of using SVMs.

ing methods applied to solve gait problems are neural networks, support vector machines, nearest neighbor classifier and different clustering solutions. The way the data is organized and represented affects heavily on how the previous methods behave [59]. Table 2.5 presents some studies about machine learning classification of human gait disorders using GRFs.

One of the most common way that clinicians use to follow a patient in his recuperation is the combination of simple visual inspection or 2-D video records with ground reaction forces [61] [66]. There are lots of studies based only on the vertical ground reaction force for classification purposes for a better and precise analysis of the gait pattern. The reason for that is that all the datasets were very small, with less than 50 people. Most of the classification attempts were aiming on the distinction between particular diseases instead of drawing a distinction between functional gait disorders [66]. With this, to classify properly a gait disease it is necessary a large, informative and organized dataset, in order to make a robust and reliable algorithm that can be applied in real-world scenarios. In this work, are used three-dimensional ground reaction forces of the affected and unaffected side as input. In the next paragraphs are presented some articles that approach the gait classification. They mention the articles that are present at the table 2.5.

Lozano-Ortiz et al. [60] studied the human gait classification after lower limb fracture using ANN and Principal Component Analysis (PCA). Their study contained 51 subjects, 38 with normal pattern and 13 with abnormal (binary). The database had information with GRFs. Their classifier compared the Artificial Neural Network algorithm versus the Self-Organizing Map (SOM). The ANN obtained an accuracy of 92% and the SOM 96%.

Alaqtash et al. [61] studied the automated classification of pathological gait patterns from healthy walking. Their study comprised 20 participants, 12 healthy, 4 with cerebral palsy and 4 with multiple sclerosis. The database has 19 features based on amplitude and temporal parameters of GRFs. Their classifier compared the KNN and ANN algorithms. As result, the KNN obtained an accuracy of 85% and the ANN 80%.

Slijepcevic et al. [62] studied the effects of different principal component analysis based representations on ground reaction force measurements for gait classification tasks. Their study intended to discover what was the best practice for the previous stated problem. The dataset used comprised 440 patients, 279 with gait disorders and 161 healthy. The patients were classified into four categories, namely calcaneus, ankle, knee, and hip. The four categories had 82, 62, 69, 66 patients respectively. The dataset had information of the bilateral GRF and center of pressure. All the data was pre-processed earlier and the classifier chosen was a

Reference	Study Goal	Classifier
Lozano-Ortiz et al. [60]	Human Gait Classification after Lower Limb Fracture using Artificial Neural Networks and Principal Component Analysis	ANN and SOM
Alaqtash et al. [61]	Automatic classification of pathological gait patterns from healthy walking	KNN and ANN
Slijepcevic et al. [62]	Ground Reaction force measurements for gait classification tasks	SVM
Slijepcevic et al. [63]	Automatic classification of functional gait disorders	SVM
Slijepcevic et al. [64]	Optimal combination of input signals and derived representations for automatic gait classification, based on GRF	SVM
Slijepcevic et al. [65]	Automatic classification and explanation of machine learning predictions in clinical gait analysis	CNN, SVM and MLP

Table 2.5: Studies about the classification of gait disorders with machine learning and GRFs.

support vector machine. Two classification were proposed, the first healthy vs. gait disorders (binary) and the second healthy vs. all four categories. Figure 2.17 shows the results obtained for the two different classifications. The first one had an acceptable result but the second one, due to its complexity, resulted in a lower accuracy.

Slijepcevic et al. [63] studied the automatic classification of functional gait disorders. The study was based on ground reaction force measurements. The study comprised two objectives, the examination of the suitability of the GRF parameterization techniques for the study of gait disorders and the creation of a baseline for the automated classification of gait disorders. The dataset used was the same used on [62], with 440 patients divided into four classes. All the data was pre-processed and the classifier was also an SVM algorithm with two classification problems (Healthy Control (HC) vs. Gait Disorder (GD) and HC vs. four GD classes). Figure 2.18 demonstrates the results obtained.

Slijepcevic et al. [64] studied what was the optimal combination of input signals and derived representations for gait classification based on GRFs. The study intended to answer the question of which input signals, derived representations, and combinations were the most effective for gait classification. The dataset used comprised 728 patients, 546 with gait disorders and 182 healthy. The patients with gait disorders were divided into three categories, calcaneus, knee and hip with 182 patients each. All the data was pre-processed like the two

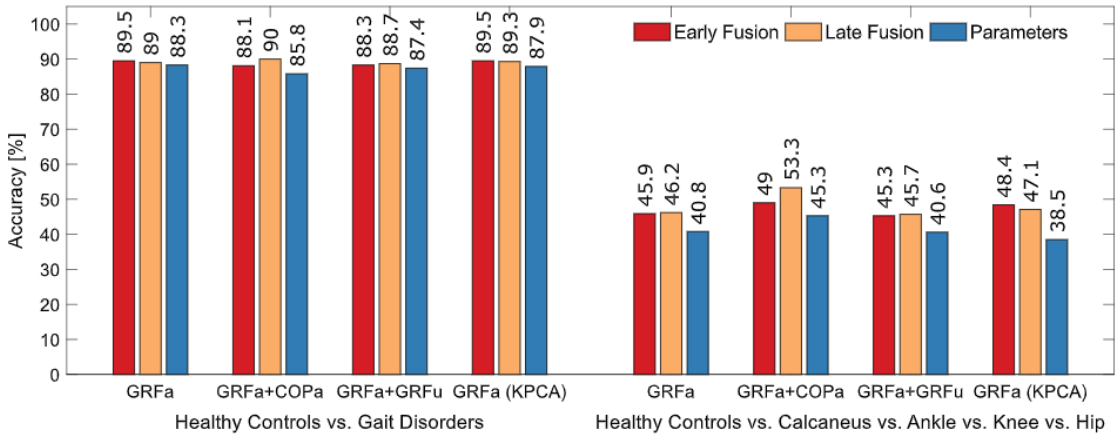


Figure 2.17: Overview of the prediction accuracy (SVM) - 2017 Slijepcevic et al. article (taken from [62]).

Parameterization	Norm.	Dim.	$N/C/A/K/H$ (RB: 31.8%)		N/GD (RB: 87.1%)	
			linear SVM	RBF SVM	linear SVM	RBF SVM
GRF Parameters (DPs and TDPs)	z-score	52	15.0 (46.8)	8.8 (40.6)	2.4 (89.5)	−0.8 (86.3)
GRF Parameters (DPs and TDPs)	min-max	52	14.3 (46.1)	9.5 (41.3)	1.6 (88.7)	−3.8 (83.3)
PCA on F_V, F_{AP}, F_{ML}	z-score	30	19.8 (51.6)	15.4 (47.2)	2.4 (89.5)	**2.0 (89.1)**
PCA on F_V, F_{AP}, F_{ML}, COP_{AP}, COP_{ML}	z-score	39	**22.5 (54.3)**	**19.4 (51.2)**	**3.7 (90.8)**	1.9 (89.0)
PCA on z-standardized GRF parameters	z-score	28	13.8 (45.6)	8.8 (40.6)	2.6 (89.7)	−0.6 (86.5)
PCA on min-max normalized GRF parameters	z-score	28	13.5 (45.3)	7.9 (39.7)	2.8 (89.9)	0.1 (87.2)

Note that the random baseline (RB) is stated next to the task name and that the values in the table represent the deviation from the random baseline (RB) and the corresponding absolute accuracy in brackets.

Figure 2.18: Overview of the prediction accuracy (SVM) - 2018 Slijepcevic et al. article (taken from [63]).

previous studies and the chosen algorithm was also the SVM. The study had only one task, the classification of healthy vs. the three gait disorders. The best result was obtained on the GRF + COP (center of pressure) with an accuracy of 67.8%.

Djordje Slijepcevic et al. [65] approach three machine learning methods, the Support Vector Machine, Multilayer Perceptron (MLP) and Convolutional Neural Networks. They were all compared in terms of prediction accuracy and learned input relevance patterns. Since this work will be focused on the ANN and CNN, these two will be the ones that will be focused. This is the most recent article and also the one that focus also the GaitRec dataset (only a subset). Thus, it will be the one that this work will be most compared to [65].

The MLP contained three consecutive fully connected layers with ReLU as activation function and, after them, there was a SoftMax activation function in the output layer. The size of the hidden layers and output was 768 and c, respectively. The value of the output is c because it can comprise different numbers of the target classes, depending on the study. The CNN comprised three consecutive convolutional layers, with a <filter size>-<stride>-<output channel>configuration of 8-2-24, 8-2-24 and 6-3-48. In addition, they all had ReLU as neuron activation function. The previous configuration results in a 48x48 feature map that is subsequently unrolled into a 2304-dimensional vector. After that, the data reaches a fully-

connected layer, that leads to the model output. The fully-connected layer has a SoftMax activation function in the output layer, that acts as a multi-class predictor in relation to the c target classes. The two methods stated before were trained via standard error back-propagation with stochastic gradient descent and a mean absolute loss function.

The training process incorporated $3*10^4$ iteration of mini batches of five randomly selected training samples and an initial learning rate of $5 * 10^{-3}$. It is important to refer that the learning rate was gradually decreased after every 10^{-4}-th training iteration by a factor of 0.2 and subsequently to $5 * 10^{-4}$ by a factor of 0.5. All the model weights were initialized with random values related to a normal distribution from 0 to $m^{-1/2}$, with m equal to the number of inputs of each neuron layer. The CNN receives an input of 1x606-dimensional vector and, because of that, the convolution operations are 1-dimensional, moving only over the time.

An additional information that was concluded and has some relevance is that there were negligible differences between 1-D and 2-D CNNs. All the accuracies were reported over a stratified ten-fold cross-validation, where eight partitions of data were used for training, one for validation and one for testing. The results were reported as mean with standard deviation. It was also calculated the Zero Rule Baseline (ZRB) for each classification, that is a theoretical accuracy obtained assigning class labels according to the prior probabilities of the classes. The results were presented also with min-max normalization and with no-normalization.

Figure 2.19 represents an overview of the prediction accuracy of the related article. As observable, Six classification approaches were made, such as healthy vs gait disorder in general (binary classification), healthy vs hip, healthy vs knee, healthy vs ankle, healthy vs knee vs ankle and finally healthy vs hip vs knee vs ankle. Moving on to the results obtained, it was concluded that the mean prediction was superior in relation to the ZRB. Finally, Table 2.6 presents the best results obtained in the previous presented gait classification articles.

Reference	**Binary Performance**	**Multi-class Performance**
Lozano-Ortiz et al. [60]	ANN: 92% SOM: 96%	-
Alaqtash et al. [61]	KNN: 85% ANN: 80%	-
Slijepcevic et al. [62]	SVM: 90%	SVM: 53.3%
Slijepcevic et al. [63]	SVM: 90.8%	SVM: 54.3%
Slijepcevic et al. [64]	-	SVM: 67.8%
Slijepcevic et al. [65]	CNN Best: 97% Mean: 89% SVM Best: 95% Mean: 89% ANN Best: 97% Mean: 90%	CNN Best: 72% Mean: 53% SVM Best: 62% Mean: 52% ANN Best: 63% Mean: 50%

Table 2.6: Best performance of the previously presented human gait disorder articles.

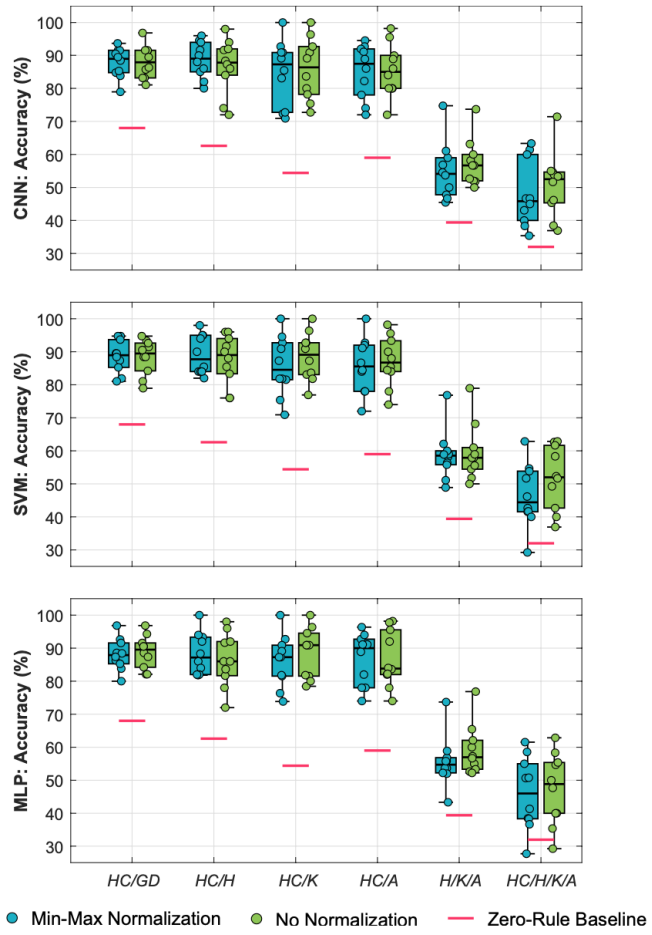


Figure 2.19: Overview of the prediction accuracy (CNN, SVM, MLP) - 2020 Slijepcevic et al. article (taken from [65]).

Chapter 3

Materials and Methods

As stated previously, this dissertation addresses the application of supervised machine learning techniques for the classification of human gait disorders using the annotated GaitRec dataset. The dataset contains bi-lateral 3D-GRF data from healthy individuals, as well from patients with musculoskeletal impairments at the hip, knee, ankle and calcaneus. This chapter aims to clarify the objectives of this dissertation and its delimiting boundaries, being followed by a description of the methodological approaches used to face the key challenges of the work. Section 3.1 describes the computational resources used to perform the research. Section 3.2 provides a detailed description of the GaitRec dataset used throughout the work. Section 3.3 presents the overall framework of the study, including the architectures selected for comparison, as well as the model development strategy and the performance measures to be considered. Section 3.4 summarizes the set of scenarios and experiments to be evaluated.

3.1 Experimental Setup

Training the neural network models on the GaitRec dataset can be very resource and time-consuming in terms of processor and memory. Accordingly, most of the experiments performed in this work were conducted in a high-performance dedicated server available in the IRIS-LAB (IEETA). The server contains three GPU, where only two of them were provided for this work, and is composed by the following components:

- CPU: 2x Intel Xeon 2.1 GHz
- GPU: 2x NVIDIA GEFORCE RTX 3080
- RAM: 64 GB

The host computer runs on a JupyterLab server. To manage the train and test environment are available several frameworks such as pip, virtualenv, or anaconda. The server has installed a Miniconda version due to the fact that has an ease use, fits the general purposes of the majority of the users in terms of packages and occupies less memory in the CPU. Anaconda and Miniconda differ only in the pre-installed packages, in where Miniconda contains a smaller number of packages compared to Anaconda. The necessary libraries were installed in the provided user account. Commonly, to interact with NVIDIA GPUs for parallel computing it is used an Application Programming Interface (API) called CUDA. The server has installed the 11.2.2 CUDA version. In addition, in case of working with deep neural networks,

NVIDIA provides a library called cuDNN that allows high-level frameworks to benefit of the high computing power of GPUs. The version 8.1.0 is utilized in the server. All the code was written in Python 3.7, which was also already available. The following packages were used to create the files of the work:

- Tensorflow 2.5.0 (Utilized as a backend of Keras framework, integrated in tf.keras)
- NumPy 1.19.5
- Pandas 1.2.5
- scikit-learn 0.24.2
- Jupyter 1.0.0
- matplotlib 3.0.2
- Microsoft Excel
- Matlab R2021a

3.2 GaitRec Dataset

As mentioned previously, one of the standard tools that the clinicians use to evaluate and study in detail the human locomotion are the ground reaction forces. Remembering, ground reaction force is the force applied by the ground on a body in contact with it. One disadvantage of the GRF application is that the results obtained are very complex and challenging to interpret. Machine Learning methods, such as neural networks, are a promising way to assist clinicians in the diagnose and classification of gait patterns [15]. In order to obtain reliable results, it is necessary a large, organized and informative quantity of data to train the neural network model.

The GaitRec dataset, managed by an Austrian rehabilitation center, contains anonymized GRF measurements from 2295 patients, where 2085 of them represent subjects with different musculoskeletal impairments and the remaining 211 represent the healthy controls. It provides 75,732 bi-lateral trials distributed across five categories, as depicted in Figure 3.1:

- Healthy Controls (HC) - represents the top layer class of the dataset containing the data from healthy people. In this case, healthy subjects are the ones free of pain and complaints at the lower extremity and spine and did not have any orthotics or orthopedic insoles. People with history of surgery or trauma at the spine or lower extremities were excluded.
- Gait Disorders (GD) - represents the top layer class containing the data from patients with musculoskeletal impairments at the hip, knee, ankle and calcaneus. The data was gathered from a rehabilitation center and comprises an entire rehabilitation process.
- Hip (H) – the hip class contains several injuries such as fractures of the pelvis and thigh along with luxation of the hip joint, coxarthrosis, and total hip replacement.
- Knee (K) - the knee class contains patients after patella, femur or tibia fractures, ruptures of the cruciate or collateral ligaments or the meniscus, and total knee replacements.

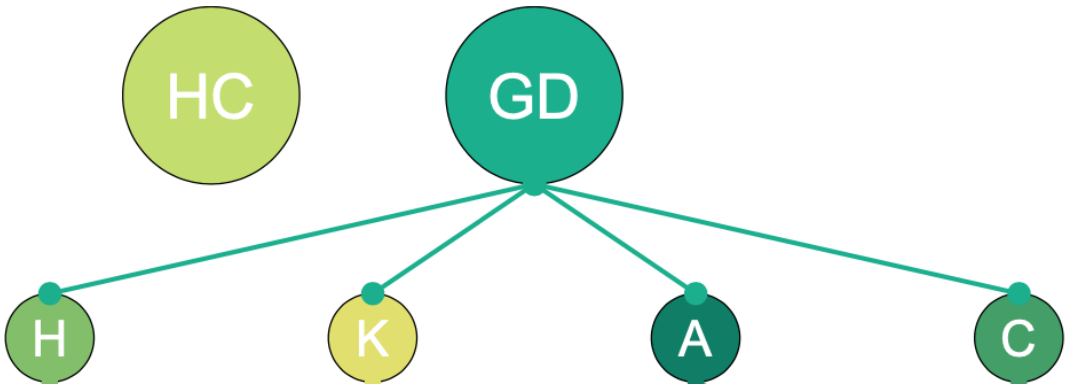


Figure 3.1: Hierarchical class structure of the GaitRec dataset relevant to this study: Healthy Controls (HC), Gait Disorders (GD), Hip (H), Knee (K), Ankle (A), and Calcaneus (C) (adapted from [14]).

- Ankle (A) - the ankle class comprises patients after fractures of the malleoli, talus, tibia, or lower leg, and ruptures of ligaments or the achilles tendon.
- Calcaneus (C) - the calcaneus class includes patients after calcaneus fractures or ankle fusion surgery.

Although there are additional lower-level layers in the original dataset, this work only considers the HC vs. GD (binary classification) and HC vs. H vs. A vs. C vs. K (multi-class classification). The multi-level hierarchical categorization permits grouping the data into a dataset with four classes associated to gait disorders and one healthy controls class. The dataset was manually labeled by a professional and experienced therapist [14]. Figure 3.2 contains an overview of the data present in the GaitRec database, including the classes, the number of patients in each, their mean age and body mass, sex and the number of bi-lateral trials per class.

All subjects follow the same recording protocol based on two centrally embedded force plates (Kistler, Type 9281B12 [46]), being submitted to a 10 m walkway. During a session, they usually walked until ten valid records were obtained. Subjects with a gait perturbation walked in a unassisted way at a self-selected speed. The healthy subjects walked either barefoot or with their normal shoes at three different speeds, such as slow, fast and self-selected. The patients either walked barefoot, with their orthopedic or normal shoes, and with or without orthopedic insoles.

Class	N	Age (yrs.) Mean (SD)	Body mass (kg) Mean (SD)	Sex (m/f)	Bi-lateral Trials
Healthy C.	211	34.7 (13.9)	73.9 (15.6)	104/107	7,755
Hip	450	42.6 (12.8)	82.4 (15.6)	373/77	12,748
Knee	625	41.6 (12.0)	84.3 (18.6)	426/199	19,873
Ankle	627	41.6 (11.4)	87.0 (18.0)	498/129	21,386
Calcaneus	382	43.5 (10.4)	84.0 (14.5)	339/43	13,970
Total	**2,295**	**41.5 (12.1)**	**83.6 (17.3)**	**1,740/555**	**75,732**

Figure 3.2: GaitRec database overview (taken from [14]).

The GRF measurements included the vertical, the anterior-posterior and the medio-lateral force components. In addition, the Center of Pressure (COP) was also recorded. The data available in the GRF files contains separate files for the left and right foot and for each one of the three GRF forces, as can be seen in Figure 3.3. All the data used was the processed one (center of pressure data was not used). The next sentences describe the authors' data processing work [14]. They started by converting the three analog GRF signals to digital signals using a sampling rate of 2000 Hz and a 12-bit analog-digital converter (DT3010, Data Translation Incorporation) with a signal input range of $\pm 10V$. The GRF was registered in the local force plate coordinate system (reaction-orientated) and to facilitate, the orientation of all the medio-lateral and anterior-posterior signals were uniformed, in order to always express the medial and anterior forces as positive values.

Given the internal standards of the rehabilitation centre, raw signals were only available down-sampled to 250 Hz. Noise and signal peaks were avoided at the beginning and end of the signals, by applying a threshold of 25 N to all force data. At this point, the data is referred as unprocessed (raw) GRF signals. In addition, the authors have generated processed "ready to use" data (the data used in this dissertation). The processed force signals were filtered using a 2nd order low-pass Butterworth filter with a cut-off frequency of 20 Hz to reduce noise and were time-normalized to 100% stance (i.e., 101 points).

The dataset contains twenty files with GRF data (raw and processed). In addition to the GRF left/right foot information exists also a file with all the additional information about the subjects and their trials. Figure 3.3 shows the contents of the .csv files and its detailed description. It is possible to observe that the associated file contains an "*" that is a placeholder for "left" and "right" (foot). In order to better understand the available data, several graphs were made available with the information of the 3D GRFs of the affected side. Figure 3.4 illustrates all the five classes, along the mean and standard deviation. More information about the data processing work is available in [14].

Variables	Associated file	Format	Dimension	Unit	Description
Vertical GRF	GRF_F_V-RAW_*.csv	double	1 × n	Newton	Raw vertical ground reaction force
Anterior-posterior GRF	GRF_F_AP-RAW_*.csv	double	1 × n	Newton	Raw breaking and propulsive shear force
Medio-lateral GRF	GRF_F_ML_RAW_*.csv	double	1 × n	Newton	Raw medio-lateral shear force
COP anterior-posterior	GRF_COP_AP_RAW_*.csv	double	1 × n	Centimeter	Raw COP coordinate in walking direction
COP medio-lateral	GRF_COP_ML_RAW_*.csv	double	1 × n	Centimeter	Raw COP coordinate in medio-lateral direction
Vertical GRF	GRF-F_V_PRO_*.csv	double	1 × n	Multiple of body weight	Post-processed vertical ground reaction force
Anterior-posterior GRF	GRF_F_AP_PRO_*.csv	double	1 × n	Multiple of body weight	Post-processed breaking and propulsive shear force
Medio-lateral GRF	GRF-F_ML_PRO_*.csv	double	1 × n	Multiple of body weight	Post-processed medio-lateral shear force
COP anterior-posterior	GRF_COP_AP_PRO_*.csv	double	1 × n	% stance	Post-processed COP coordinate in walking direction
COP medio-lateral	GRF_COP_ML_PRO_*.csv	double	1 × n	% stance	Post-processed COP coordinate in medio-lateral direction

Figure 3.3: GaitRec .csv file description (taken from [14]).

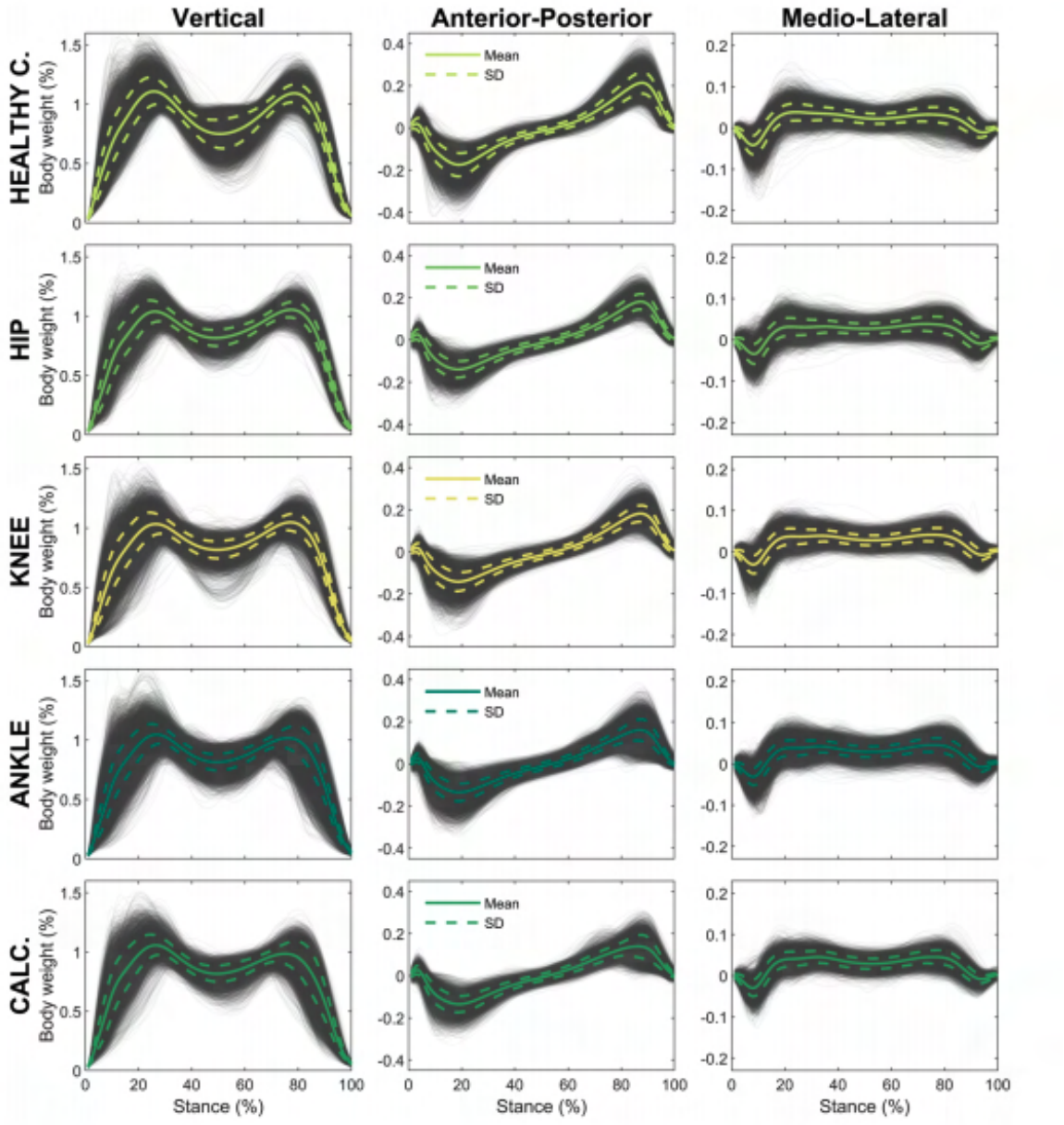


Figure 3.4: GaitRec data visualization (plot) (taken from [14]).

The values that made up the graphics are the ones that are submitted to the neural networks. A concatenation of the 1-D left and right foot GRF (medio-lateral, anterior-posterior and vertical) represent the input information to all 1-D models. At this point, we have a 1×606 vector that represents both feet, each 101 samples for each GRF's component. Additional metadata information is also concatenated to the previous one, including sex, age and walking speed. Therefore, the input data for each subject's trial is a 1×609 vector. Figure 3.5 describes what are the fields contained in that file, more precisely the categories/variables of each subject/trial, the format of the information, the units and the description. It is important to refer that all the data is anonymous.

Categories/Variables	Format	Unit	Description
Identifiers			
SUBJECT_ID	integer	—	Unique identifier of a subject
SESSION_ID	integer	—	Unique identifier of a session
Labels			
CLASS_LABEL	string	—	Annotated class labels
CLASS_LABEL_DETAILED	string	—	Annotated class labels for subclasses
Subject Metadata			
SEX	binary	—	female = 0, male = 1
AGE	integer	years	Age at recording date
HEIGHT	integer	centimeter	Body height in centimeters
BODY_WEIGHT	double	$\frac{kg\ m}{s^2}$	Body weight in Newton
BODY_MASS	double	kg	Body mass
SHOE_SIZE	double	EU	Shoe size in the Continental European System
AFFECTED_SIDE	integer	—	left = 0, right = 1, both = 2
Trial Metadata			
SHOD_CONDITION	integer	—	barefoot & socks = 0, normal shoe = 1, orthopedic shoe = 2
ORTHOPEDIC_INSOLE	binary	—	without insole = 0, with insole = 1
SPEED	integer	—	slow = 1, self-selected = 2, fast = 3 walking speed
READMISSION	integer	—	indicates the number of re-admission = 0 … n
SESSION_TYPE	integer	—	initial measurement = 1, control measurement = 2, initial measurement after readmission = 3
SESSION_DATE	string	—	date of recording session in the format "DD-MM-YYYY"
Train-Test Split Information			
TRAIN	binary	—	is part (=1) or is not part (=0) of TRAIN
TRAIN_BALANCED	binary	—	is part (=1) or is not part (=0) of TRAIN_BALANCED
TEST	binary	—	is part (=1) or is not part (=0) of TEST

Figure 3.5: GaitRec metadata file description (taken from [14]).

3.3 Overall Framework of the Study

The integration of machine learning techniques for analysis of gait data has proved to be a promising solution to deal with the difficult interpretation of ground reaction forces [60] [61] [64]. Fully-connected ANNs and support vector machines are well-established for gait classification [15] [16]. The main goals of this study are two-fold: First, to compare the performance of fully connected vs. convolutional neural networks in terms of prediction accuracy and model robustness. Second, following recent work with the GaitRec dataset [64] [65], to address two classification problems using neural networks. On the one hand, a binary classification problem focused on classifying healthy (normal) against pathological (impaired) gait is addressed. On the other hand, the second problem concerns gait classification across five classes of disorders affecting the hip, knee, ankle, and calcaneus, according the GaitRec dataset. The multiclass classification problem still has a considerable margin for improvement as shown by the work carried out in [65].

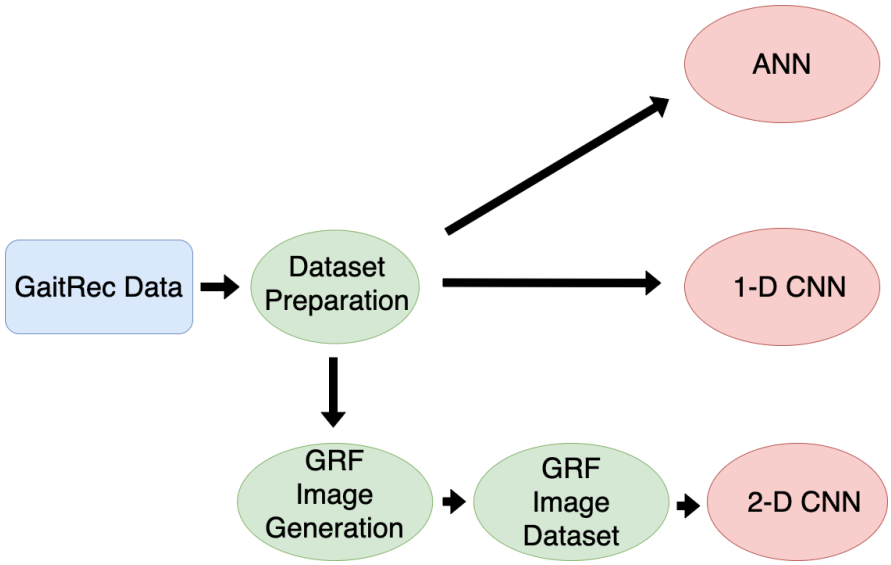


Figure 3.6: Overall framework of the work.

The study was developed from the GaitRec dataset considering the data source and potential biases which may affect the generalization ability of the models. The data preparation stage played a preponderant role in the performance of the supervised learning models. For example, unbalanced datasets (over representation of data from one class) and use of a sub-set of the larger dataset were considered to improve the prediction accuracy. In line with this, the following subsections aim to describe the architectures chosen in the study, as well as the model development strategy and the metrics used to assess their performance.

3.3.1 Selected Architectures

Ground reaction force records, saved in time series form, can be used to find out various gait disorders. Distinguishing patterns of a normal gait from the one with a disorder, and recognizing the disorder, is a Time Series Classification problem. This study proposes to compare the performance of multilayer feedforward neural networks against convolutional neural networks, when dealing with time series data of GRFs during walking (see Figure 3.6). The application of CNNs considers both univariate and multivariate time series. The univariate time series is an ordered set of real values, while the M-dimensional multivariate time series consists of M different univariate time series with the same length.

The first building block for time series classification is the multi-layer perceptron. This class of feedforward neural networks consists of several layers of nodes: one input layer, one or more hidden layers, and one output layer (see Figure 3.7). This is a fully connected model where every node is connected to all the nodes of the previous layer and of the next layer. Given a dataset consisting of a collection of pairs (input, class), it can be fitted to compute the probability of any new input to belong to each possible class. To do this, first of all we need to represent the pairs (input, class) in the dataset in a more suitable way. First, every object must be flattened and then represented with a vector, that will be the input vector for training (i.e., taking the whole multivariate time series as input). Second, every class in the dataset must be represented with its one-hot label vector.

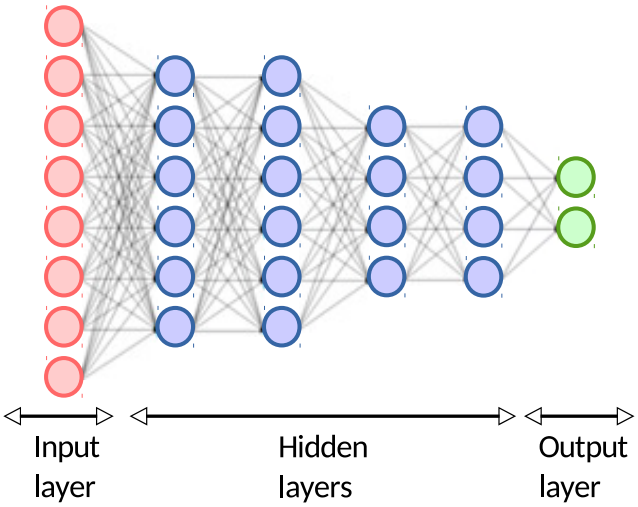


Figure 3.7: Schematic diagram of the MLP neural network.

A natural question addressed in the study was related to the need (or not) to proceed with a reduction of dimensionality on the input vector to reach good classification results. Given the dimension of input vector, the alternative would be to extract the relevant features of the input time series and use them as input of a classification algorithm.

The second approach to be considered is to take advantage of deep learning algorithms since the relevant features are learned during the training and not handcrafted. The main goal is to evaluate how a custom CNN model can improve the accuracy of the results, when dealing with uni- and multivariate time series at the input of the network (see Figure 3.8). CNN models are particularly efficient to capture the spatial and temporal patterns. Additionally, pre-processed time series are converted into a two-dimensional input image, which is applied to a 2D-CNN to explore asymmetries in bilateral GRFs. The intention is to explore a way to encode time series into an image and, consequently, to take advantage of the CNNs to learning features and identifying structure in the data.

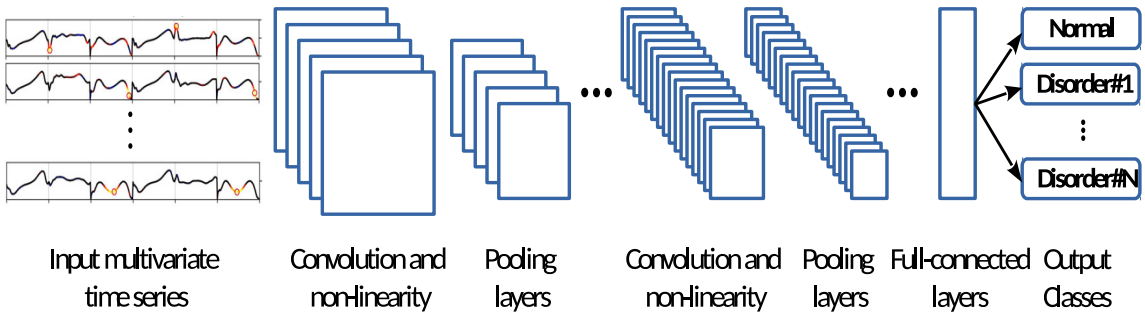


Figure 3.8: Schematic diagram of the CNN model.

3.3.2 Model Development

The model development process consists of two-phases. The first phase comprises training and validation, while the second phase is the test of the trained model with an unseen dataset. Figure 3.9 describes the process that will be applied to this work, following the training, validation and test approach. The objective is to divide the whole dataset into three subsets. The training set will be used to train the model, the validation set will provide an unbiased evaluation of the model fit on the training set and to tune the model parameters and, finally, the test set is applied to evaluate the model. The test set contains data that was never seen by the model (including its hyperparameters) in order to get an unbiased final estimate. The training and the test set are assumed to be obtained from the same probability distribution.

Two factors determine the performance of the neural network model [55]. First, the ability to make the training error small. Second, the ability to perform well on previously unseen inputs (generatization), i.e., to make the gap between training and test error small. These two factors are related to the underfitting and overfitting challenges that may occur when developing a custom model from scratch. Underfitting takes place when the model is not able to achieve a reasonable low error on the training set. In other words, the model is too basic to learn the hidden structure of the data. On the other hand, overfitting happens when there is a large gap between the training and test error, performing well on train set but not on test set [47]. Using a mini-batch gradient descent algorithm, that splits the training set into small batches, the overfitting will be detected by comparing the accuracy in the training and validation sets after each epoch.

Model development and learning algorithms have several hyper-parameters whose choice affects the behaviour of the learning process. The validation set of examples, that the training algorithm does not observe, will be used to parameter tuning. The batch size specifies the number of training samples that pass through one network information flow, more properly one forward and backward. It is defined by the Equation (3.1).

$$FlowperEpoch = \frac{NrSamples}{BatchSize} \tag{3.1}$$

where one flow is one forward and backward pass. An epoch represents the ending of a cycle, when all training examples passed forward and backward across the network.

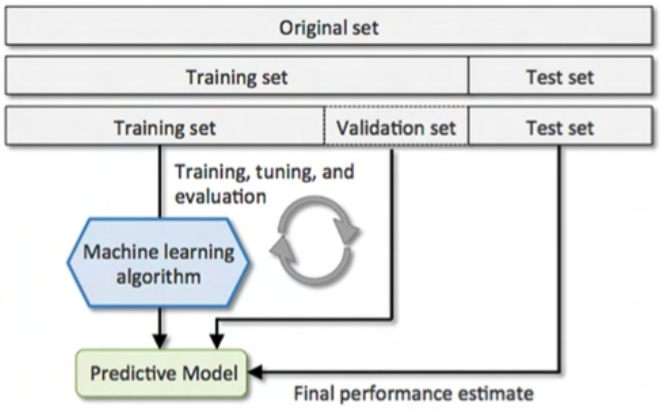


Figure 3.9: Model development workflow (taken from [67]).

3.3.3 Performance Measures

The metrics chosen to evaluate the performance of the neural models are the accuracy and the confusion matrix. The prediction accuracy is the proportion of samples correctly classified [68]. Some examples are the binary accuracy, the categorical accuracy, among others. Another alternative is by measuring the error rate. They differ only at the output, where the error rate presents the incorrect outputs and the accuracy the correct ones [68].

The confusion matrix is also a common measure [69]: it is a table that contains the number of correct and incorrect classified predictions for each class. The table comprises True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) classification incidences. In a binary problem, one class is the negative and the other is the positive. The true negatives and positives represent the correctly classified samples. The false negatives and positives, on the other hand, represent the samples of one class that were mistakenly classified as another class [68]. Is is very useful since it allows easily to identify the decision confusions, thus concluding on the quality of the model and data involved [7]. Figure 3.10 displays the binary confusion matrix table division. A confusion matrix can be applied also in a multi-class problems. The values can be presented with or without normalization.

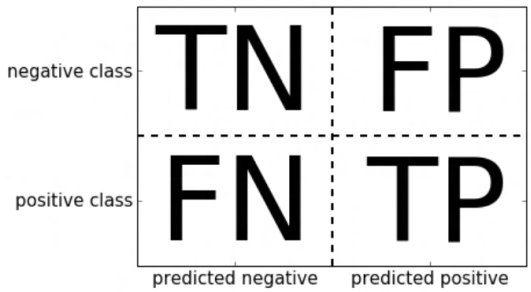


Figure 3.10: Binary confusion matrix (taken from [68]).

3.4 Description of the Experiments

According to previous ideas, the three scenarios to be implemented and the selected architectures for processing the GRFs are summarized as follows:

Binary classification (Chapter 4)

Chapter 4 addresses the binary classification problem. As it is a binary problem, its objective is only to distinguish an healthy subject from one with a gait disorder (unspecified). It is intended to compare the performance of a multilayer perceptron and a convolutional neural network to solving the same problem.

Multi-class classification (Chapter 5)

Chapter 5 addresses the multi-class classification problem. The objective is to distinguish healthy subjects from those with an hip, knee, ankle, and calcaneus gait disorder. The models to be studied are the same as the previous point.

Binary and multi-class classification with a 2-D CNN (Chapter 6)

Chapter 6 studies the viability of the classification of the binary and the multi-class gait disorders using a 2-D convolutional network. The image to encode from the GRF measurements is a plot where in the x-axis are the GRF of the left foot and in the y-axis the ones from the right foot. It is important to refer that this is just one way of representing the image. The global objective of the chapter is to analyse the performance of the 2-D convolutional neural networks on both classification problems.

Chapter 4

Neural Network Binary Classification

This chapter aims to compare the performance of two neural networks when dealing with the binary classification of human gait disorders: multi-layer perceptron and convolutional neural network. It covers all the steps performed during the implementation of the binary classification, starting with the dataset preparation, moving to the parameter tuning and concluding with the final evaluation results. The dataset used in this work is the GaitRec, currently, the most complete GRFs record [14].

4.1 Dataset Preparation

The GaitRec dataset contains information about the subjects and their GRF measurements. The authors [14] provide information about the dataset and assists anyone who needs to use it for research purposes. It contains all the dataset background/summary, the methods like the data recording and test protocol or the dataset and annotation, the data records, the technical validation, the usage notes and finally the references, author contributions, acknowledgments, competing interests and additional information.

The metadata file already contains a train-test split of the dataset labeled as TRAIN and TEST. It also has a balanced training set labeled as TRAIN_BALANCED due to the fact that the training set contains an unbalanced number of classes that can affect negatively the optimization of the machine learning models [14]. The previous is formed by at least five trials for each body side per session and only by data from initial assessments (first session of the subject) [14]. The TEST subset was not balanced.

The unbalanced set contains a train-test split of 70%-30% with 52745 samples of training and 22987 samples of test and the balanced one a split of 22%-78% with 6308 samples of training and 22987 samples of test. Figure 4.1 shows the division of the GaitRec dataset proposed by the article authors.

4.1.1 Dataset balance

The first problem was the binary classification that, like the name implies, only contains two classes (Healthy Controls and Gait Disorders). For this case, it is necessary to transform the Knee, Ankle, Calcaneus and Hip classes into only one class, the Gait Disorders.

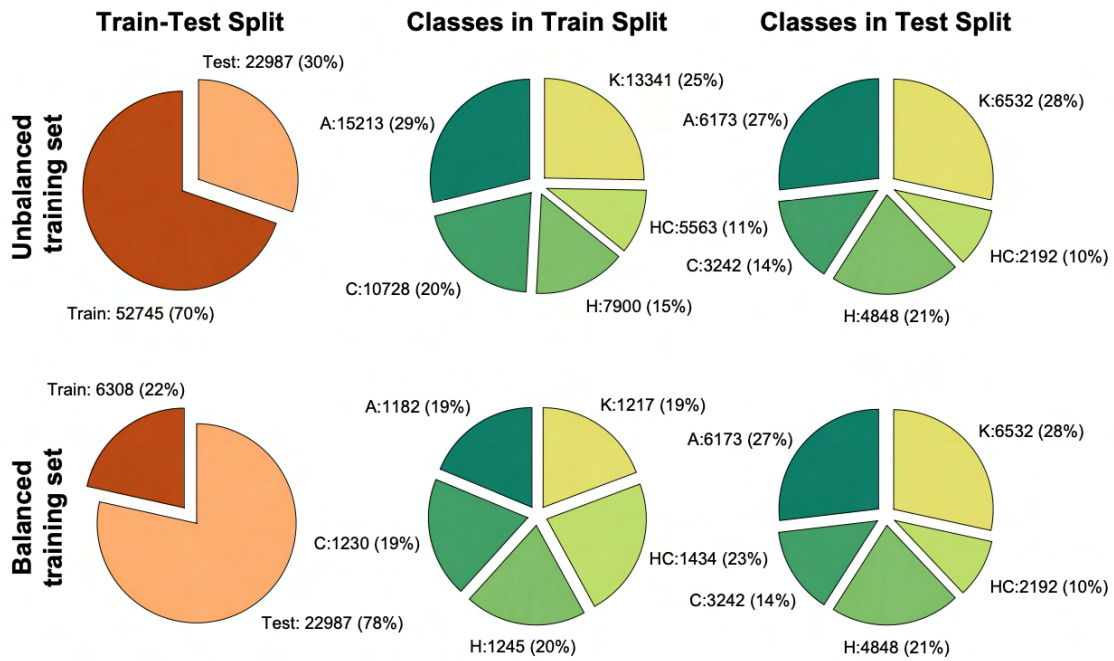


Figure 4.1: GaitRec multi-class (Healthy Controls (HC), Hip (H), Knee (K), Ankle (A), and Calcaneus (C)) split (taken from [14]).

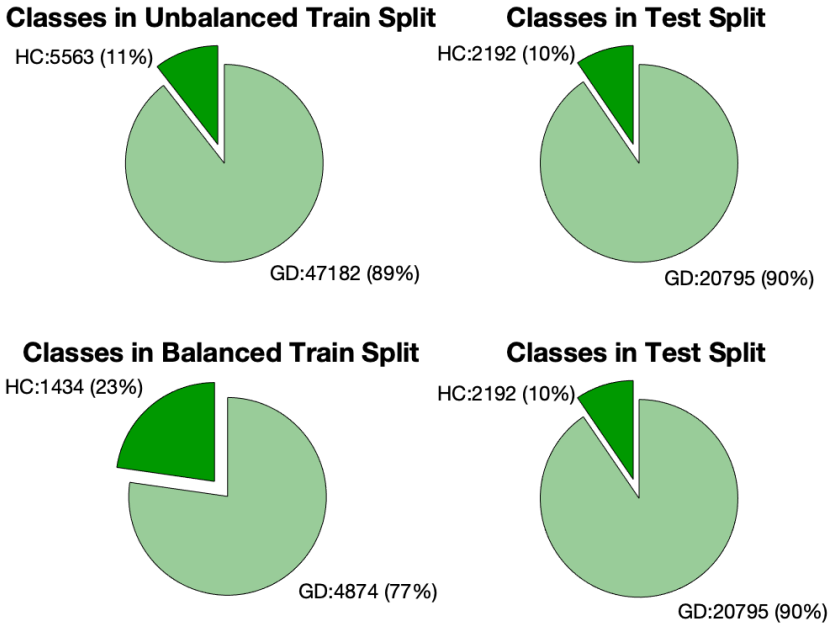


Figure 4.2: GaitRec Binary Split.

Figure 4.2 represents the result of the transformation of the original dataset into a binary one. As it is possible to observe there is no balance in both training and testing sets. In machine learning it is very important to have a balanced dataset since the class discrepancy might affect negatively the performance and optimization of the machine learning methods. In addition, it is also crucial to prepare the dataset in advance. For that, the first approach for a possible final training dataset was to gather the Healthy Controls from the unbalanced training set (5563 trials) and join them with the Gait Disorders from the balanced training set (4874 trials). In this way it is granted that the binary class contains practically the same number of trials for each existent disorders and nearly the same number of healthy controls. The same number of trials within the disorder class ensures that in the future the algorithm learns all the possibilities in the same way, making it as general as possible, thus covering all the possibilities. The test dataset was cut randomly to 5000 trials, maintaining the 2192 trials from the healthy controls and reducing the gait disorders to 2808 trials.

After some tests it was added a validation set, created with some trials of the training one. This new dataset contained 2609 trials (25% of training set).

4.1.2 Selection of other relevant inputs

Again, after some tests, it was concluded that the metadata file contained information that could be very important to the learning process and so it was necessary to make changes to the train-validation-test sets. The four most important fields to consider were sex, age, speed and session type. As it is known, men and women walk differently, the age influences the walk, the speed affects the GRF measures and the best session to compare all the subjects is the initial measurement. In the metadata file, the sex field represents women (female) with 0 and men (male) with 1. The age field is represented in years and expresses the age of the subject at the recording date. The speed field contains three types of walking speed, in which slow is 1, self-selected is 2 and fast is 3. Lastly, the session type has three different classifications, in where initial measurement is 1, control measurement is 2 and initial measurement after readmission is 3. With this, a new training, validation and test set were created with all the relevant information to this problem.

For this second approach, the training and test sets that were already available were not used. All the information was processed first with all the data together (75732 trials) and only then divided randomly into train-validation-test sets. As mentioned previously, there are 3 different speed categories in the metadata file, slow, self-selected and fast. The approached problem involves comparing a healthy subject that can walk perfectly at many different speeds (in this context 1,2 and 3) with a subject with a gait disorder that, at the most, only can walk at a slow/self-selected speed (1 and 2). With this, using the fast walking speed makes no sense in the context of the problem, in which its objective is to classify human gait disorders. In addiction, it is also important to consider the different session types. In this context, the control measurement and the readmission could mess up the classifications and so, were also excluded. The sex and the age will be later explained in more detail.

4.1.3 Final dataset

At this point, where in the beginning there were 75732 trials, only 20130 remain. Of these 20130, 5021 are from healthy controls and 15109 are from gait disorders (4545 from knee disorders, 4365 from ankle disorders, 2867 from calcaneus disorders and 3332 from hip

disorders). To obtain a good performance, both classes need to have approximately the same number of trials. With this, 5020 trials from the gait disorders were chosen in order to equalize the number of healthy controls. It is important to refer that these 5020 gait disorder trials contain 1255 samples for each previous classes (knee, ankle, calcaneus and hip) and were chosen randomly. To remember, as stated before, the same number of trials within the disorder class ensures that the algorithm learns all the possibilities in the same way, making it as general as possible, thus covering all the possibilities.

Now that the dataset information is adapted to the problem it is necessary to divide, as stated previously, in train, validation and test sets. After some discussion, a division of 60% for training, 20% for validation and 20% for test was reached containing 6024, 2008 and 2009 trials for each respectively. The division was performed with sklearn train_test_split. Figure 4.3 and 4.4 represent the class distribution for each set of the new binary dataset.

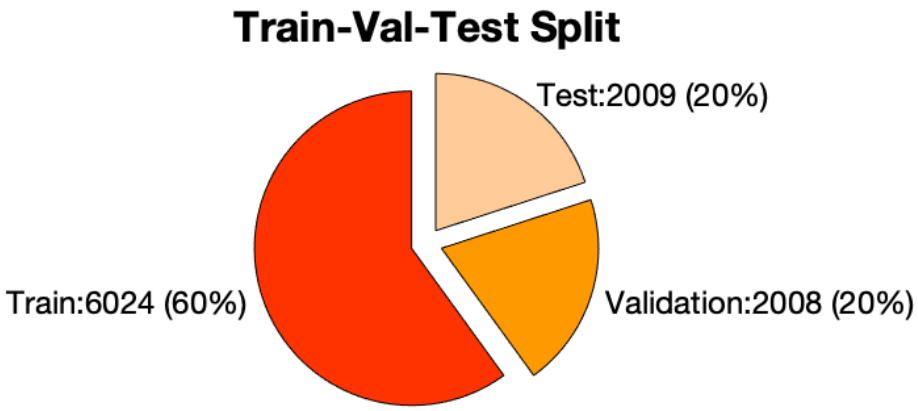


Figure 4.3: GaitRec Balanced Binary Train-Val-Test Split.

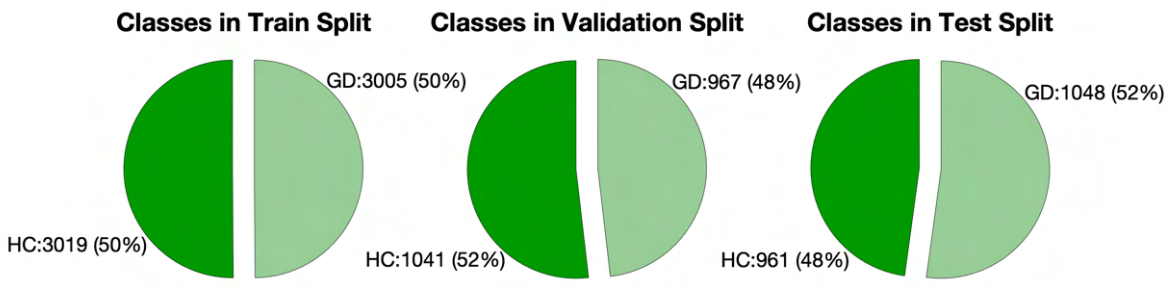


Figure 4.4: GaitRec Balanced Binary Class Split.

All the data was processed with Python3. Later in this work and with the purpose of building a neural network, the information of each train-validation-test trial will be handled

as predictors and classes. In this case, predictors can be seen as the "hints" provided to the model so it can establish what class variable it needs to be assigned to each trial. The classes are labels/targets of the data that categorize the predictors information. In a classification problem, the predictors contain information that points to one of the classes.

4.2 Parameter Tuning

One of the major difficulties of machine learning is that learning algorithms require to set parameters before the usage of the models. These parameters are for example the loss function, the optimizer, the activation function, etc. They have lots of options, each of which has its applicability depending on the problem. One way to find the best parameters for a model is to perform the tuning. The tuning chooses, amongst several possibilities, the most favourable parameters that allow to finish a learning task in the best possible manner.

Both neural networks receive as input a 1x609-dimensional input vector that represents the horizontal concatenation of the left and right foot ground reaction forces and the important previously stated metadata information (sex, speed and age). The medio-lateral, anterior-posterior and vertical components of the GRFs contain, in its respective file, time-normalized vectors of 101 points. The order of the concatenation is left foot first and then right foot. So, more specifically, the input information is left medio-lateral, left anterior-posterior, left vertical, right medio-lateral, right anterior-posterior, right vertical, sex, age, speed. These represent the input predictors. The classes are two, healthy controls or gait disorders, since this is a binary problem. All the experiments realized in the next subsections were performed with a train and validation set. The metric was binary accuracy for this binary classification. All graphics are presented to corroborate the tuning result/accuracy.

4.2.1 ANN

The authors of the dataset performed some experiments with a mini dataset before creating GaitRec [65]. They built an ANN (Multilayer Perceptron) with three consecutive fully connected layers that lead to a final output layer. The size of each hidden layer was 768 whereas the size of the output layer was 1, the number of classes (0 for healthy and 1 for gait). The output activation was a sigmoid due to the fact that the problem was a binary one. Additionally, the batch size and the epoch number were 512 and 750, respectively. The starting point and a good practice is to mimic their neural network most important parameters. In addition, some other parameters are added to perform the tuning.

Starting with the tuning process, it was decided that the parameters submitted to the tuning should be tuned in an "automatic" way with sklearn GridSearchCV, writing all the possibilities and letting the computer define what are the best parameters. The activation function, the kernel initializer, the optimizer and the loss function were the first parameters that were tuned. The terms compared for the activation function were ReLU vs. Hyperbolic Tangent (Tanh). For the kernel initializer, the Random vs. He vs. Glorot (Normal and Uniform), for the optimizer the Adam vs. SGD and for the loss function the binary crossentropy vs. hinge vs. poisson. The results of the tuning, in other words the best parameters, were ReLU for the activation, Adam for the optimizer, binary crossentropy for the loss function and random uniform to the kernel initializer. It is important to consider that the value of the neurons, layers, batch size, epochs, learning rate and dropout will be studied in the subsequent sub sections. The study will be more "manual" since the training and validation of

the model will be repeated several times to find the best parameters.

Neurons and Dense Layers

The neurons in a neural network aim to mimic those in human ones. They can be seen as nodes through which data flows. A layer is made up of nodes. It normally receives a weighted input, transforms it with a set of functions and then send these values as output to the next layer. Creating a viable neural network involves using the right amount of layers and neurons. It is important to study the influence of each layer and the number of neurons in order to have a proper result in the end. The objective was to compare the number of neurons and layers. Starting with 2 layers, the objective was to compare the number of neurons (305, 384, 450, 600, 768, 880, 1000, 1200, 1500). After that, add another layer on and on and compare the same number of neurons till reach the max accuracy and low loss. The value of the learning rate, decay, epochs and batch size was a fixed value amongst all the tests. Table 4.1 presents the results of the performed tuning and their best parameters/scores.

Dense Layers	**Best Result**
2	Neurons: 1200 Acc: 95.9%
3	Neurons: 1500 Acc: 96.8%
4	Neurons: 1200 Acc: 97.43%
5	Neurons: 1500 Acc: 97.6%
6	Neurons: 1000 Acc: 97.4%

Table 4.1: ANN Binary tuning best parameters/score and layers.

As it is possible to observe, the best accuracy was obtained with 5 layers with 1500 neurons each. Another tuning was made, increasing even further the number to 1700 and subsequently 2000. 1700 neurons obtained an accuracy of 97.7%, this being the best value obtained.

Batch Size

Four batch sizes were compared, more precisely 128, 256, 512 and 1024. After the tuning process, the best batch size was 512 with 97.7% accuracy. Figure 4.5 represents the binary comparison (train and validation) of the ANN batch size. The size 512, the best one, is represented by the black colour. Blue, green and red represent 128, 256 and 1024, respectively. Note that this is a binary problem with a good amount of data so the discrepancy is not very watchable.

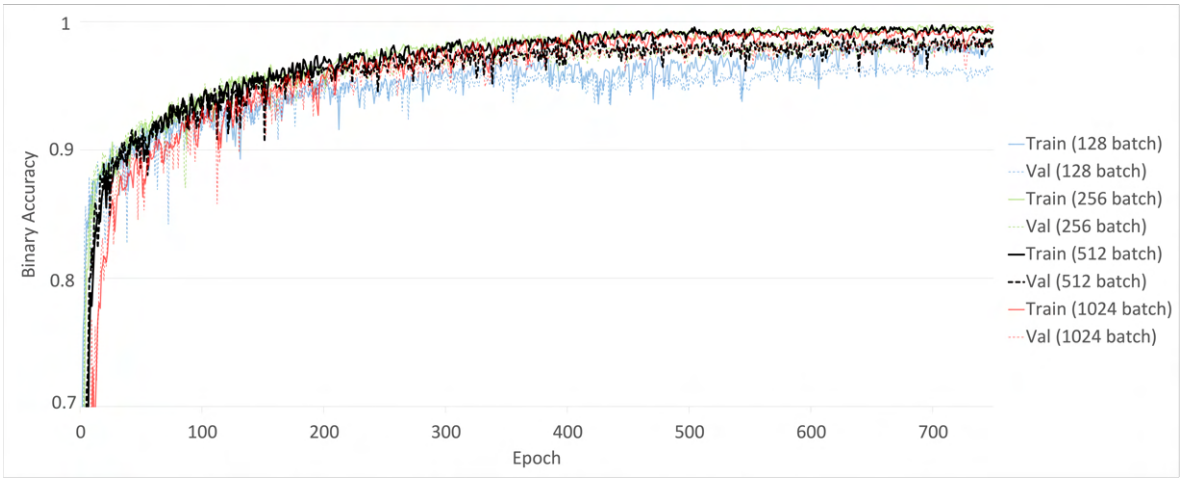


Figure 4.5: Binary ANN batch size comparison.

Number of Epochs

In this case, the number of epochs represents how many times each trial is examined by the network. Four epoch sizes were set side by side, more properly 500, 750, 1000, 1250. The best epoch number was 750 with an accuracy of 97.7%. Figure 4.6 represents the binary comparison (train and validation) of the ANN epoch size. The size 750, the best one, is represented by the black colour. Red, blue and green represent 500, 1000 and 1250, respectively.

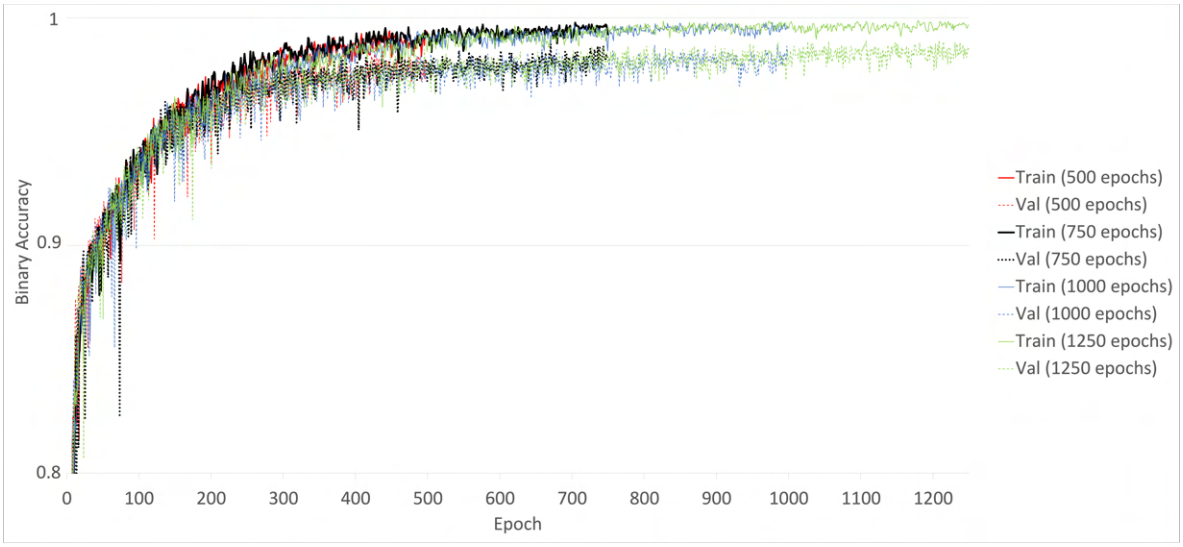


Figure 4.6: Binary ANN epoch size comparison.

Learning Rate

The learning rate is one of the most important parameters to optimize. It handles the step size for model weight updates regarding the loss function. It has a small positive value and controls how fast the model is adapted to the problem. Different values for the learning

rate and decay were submitted to the tuning process, 0.00005, 0.0001, 0.0005, 0.001, 0.005. The best learning rate was 0.0005 and the decay 0.0001 with an accuracy of 98.4%. Figure 4.7 and 4.8 represent the binary comparison (train and validation) of the ANN learning rate. The 0.0005 learning rate, the best one, is represented by the black colour. Red, green and blue represent 0.0001, 0.001 and 0.005, respectively.

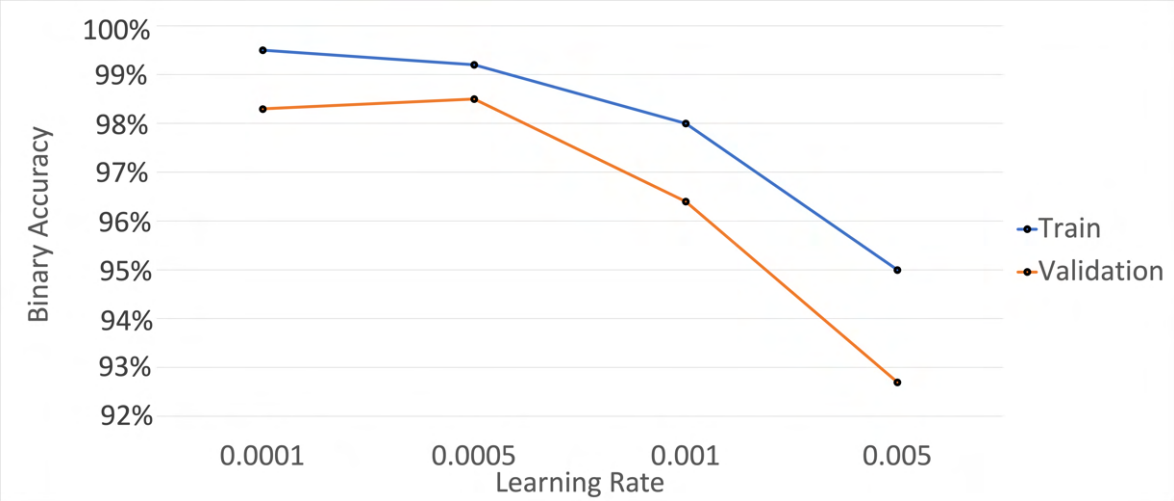


Figure 4.7: Binary ANN learning rate comparison.

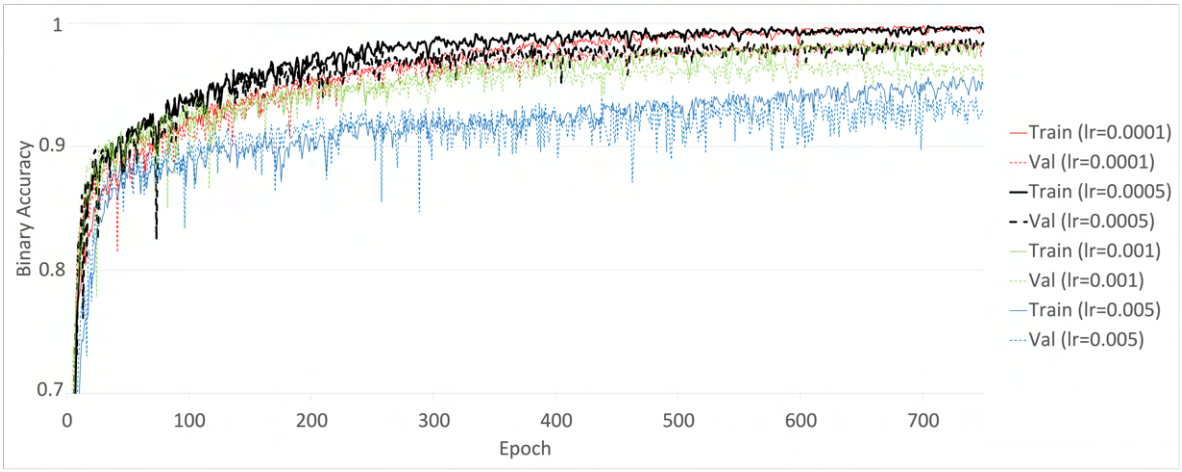


Figure 4.8: Binary ANN learning rate accuracy comparison.

Dropout

The dropout, like the name implies, drops out some random neurons in a neural network during an iteration in the training phase. The dropout plays a major role in preventing a model from overfitting. In this case it is essential to analyze if an addition of dropout will increase the performance of the model, and if yes analyse the one that most benefits the model. The dropouts were added after each dense layer. Five values for the dropout were used in

the tuning process, 0 (no dropout), 0.1 (10%), 0.2 (20%), 0.3 (30%), 0.4 (40%). The best dropout was 0.1 (10%) with an accuracy of 98.5%. After that, another tuning was performed with 0.05, 0.1 and 0.15 and the best one was 0.15 (15%) with 98.8%. Figure 4.9 represents the binary comparison (train and validation) of the ANN dropout. The 10% dropout, the best one, is represented by the black colour. Red, yellow, green and blue represent 0%, 20%, 30% and 40%, respectively.

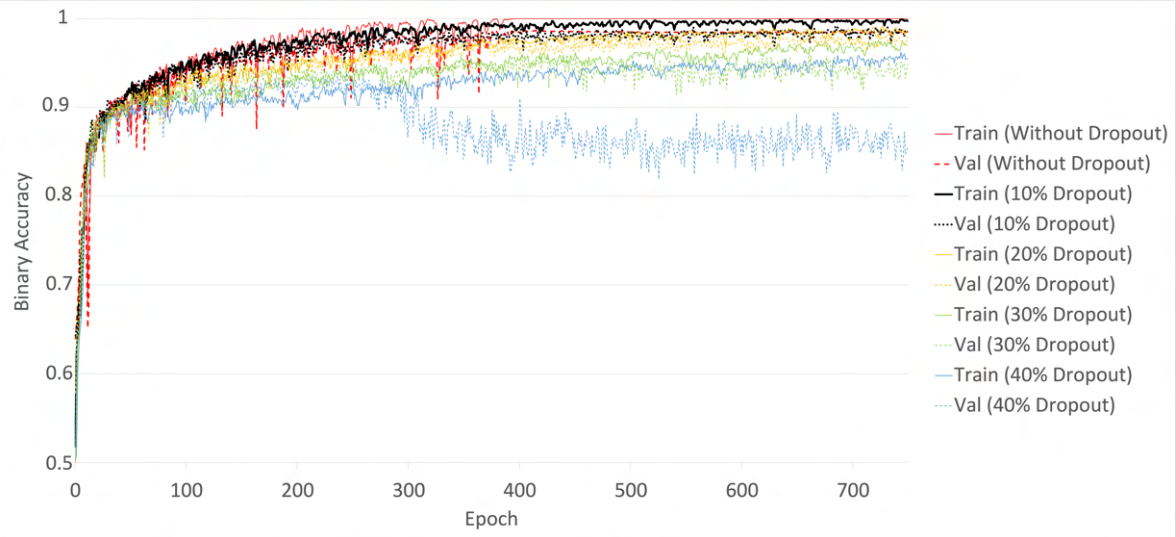


Figure 4.9: Binary ANN dropout comparison.

With this, the ANN parameter tuning is concluded. The best parameters were ReLU for the activation, Adam for the optimizer, binary crossentropy for the loss function, random uniform for the kernel initializer, 5 dense layers with 1700 neurons each, 512 for the batch size, 750 epochs, 0.0005 for the learning rate and 0.15 for the dropout.

4.2.2 CNN

Just like on ANN, the starting point was to mimic the authors neural network parameters and merge them with other ones. The CNN (1-Dimension CNN) had three consecutive convolutional layers that led to a final output layer [65]. The convolutional layers had as filter size for the first, second and third convolution layer 8, 8 and 6, respectively, as stride 2, 2 and 3 and for the output channel 24, 24, 48. Between the convolution layers there was always a 1-D max pooling layer with a pool size of 2. After the third convolution, the resulting information was flattened. It was subsequently fed into a fully connected layer with only a binary output layer. In the same way as the ANN, the output activation was a sigmoid.

The batch size and the epoch number were 512 and 750, respectively. As stated above, it is important to consider that the value of the batch size, layers, epochs, learning rate and dropout will be studied in the subsequent sub sections like the ANN approach. The results of the tuning, in other words the best parameters, were ReLU for the activation, Adam for the optimizer and binary crossentropy for the loss function.

Number of Convolutional and Dense Layers

In order to create a viable neural network, it is necessary to use the right amount of layers and neurons. Additionally, it is important to study the influence of each layer and the number of neurons in order to have a proper result in the end. Starting with 2 convolutional layers several values for the filter were submitted to the tuning, 24, 32, 48, 64, 128 and 256. After that, the process was to increase on and on by 1 the number of convolutional layers and to do the tuning again. Table 4.2 presents the results of the performed tuning and their best parameters/scores. Filters represent the the number of output filters in the convolution.

Conv Layers	**Best Result**
2	Filters: 64 - 256 Acc: 89.8%
3	Filters: 64 - 128 - 256 Acc: 92.92%
4	Filters: 64 - 256 - 128 - 256 Acc: 92.8%

Table 4.2: CNN Binary tuning best parameters/score and layers.

As it is possible to observe, the best choice is 3 convolutional layers with an accuracy of 92.92%. The next step is to assess whether or not it is worth putting a fully connected network after the flattening process. Table 4.3 displays the results of the performed tuning with 3 convolutional layers with the best one being 4 dense layers with 768 neurons.

Dense Layers	**Best Result**
1	Neurons: 1700 Acc: 93.94%
2	Neurons: 1500 Acc: 94.25%
3	Neurons: 1000 Acc: 94.44%
4	Neurons: 768 Acc: 94.6%
5	Neurons: 1000 Acc: 94.49%

Table 4.3: CNN Binary tuning best parameters/score with dense layers.

Batch Size

Four different values for the batch size were used in the tuning process, 128, 256, 512 and 1024. The best batch size was 256 with 95.43%. 512 also presented notable results.

Figure 4.10 represents the binary comparison (train and validation) of the CNN batch size. The size 256, the best one, is represented by the black colour. Red, green and blue represent 128, 512 and 1024, respectively.

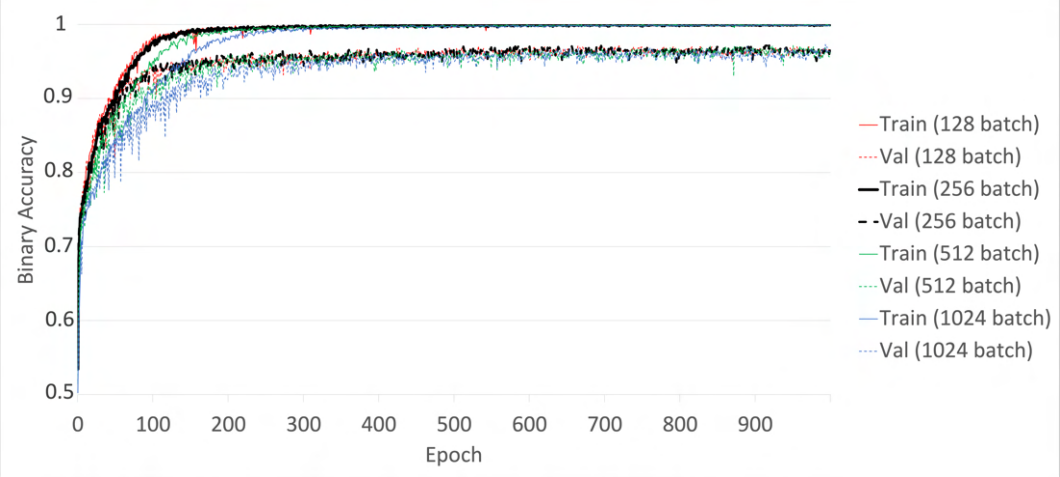


Figure 4.10: Binary CNN batch size comparison.

Number of Epochs

The number of epochs used in the tuning process were 750, 1000, 1250 and 1500. The best number of epochs was 1000 with 95.43%.

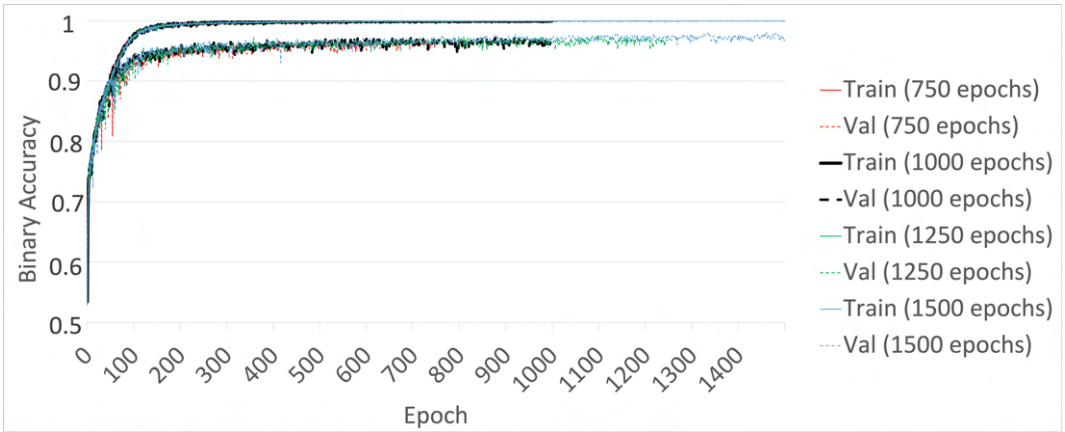


Figure 4.11: Binary CNN epoch size comparison.

Figure 4.11 represents the binary comparison (train and validation) of the CNN epoch size. The size 1000, the best one, is represented by the black colour. Red, green and blue represent 750, 1250 and 1500, respectively.

Learning Rate

Different values for the learning rate and decay were submitted to the tuning process, 0.001, 0.0001 and 0.0005. The best ones were 0.0005 for the learning rate and 0.0001 for the decay. The best accuracy was 96.42%.

Figure 4.12 and 4.13 represents the binary comparison (train and validation) of the CNN learning rate. The 0.0005 learning rate, the best one, is represented by the black colour. Red and blue represent 0.0001 and 0.001, respectively.

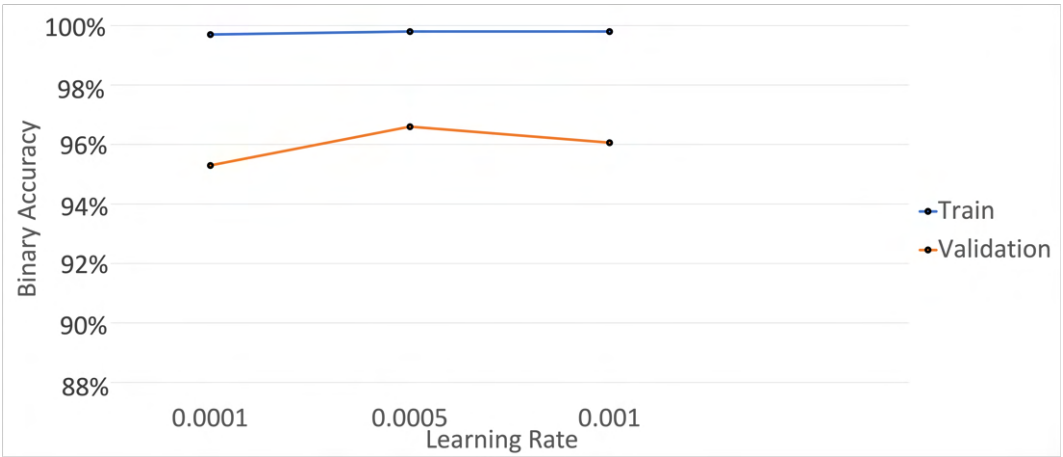


Figure 4.12: Binary CNN learning rate comparison.

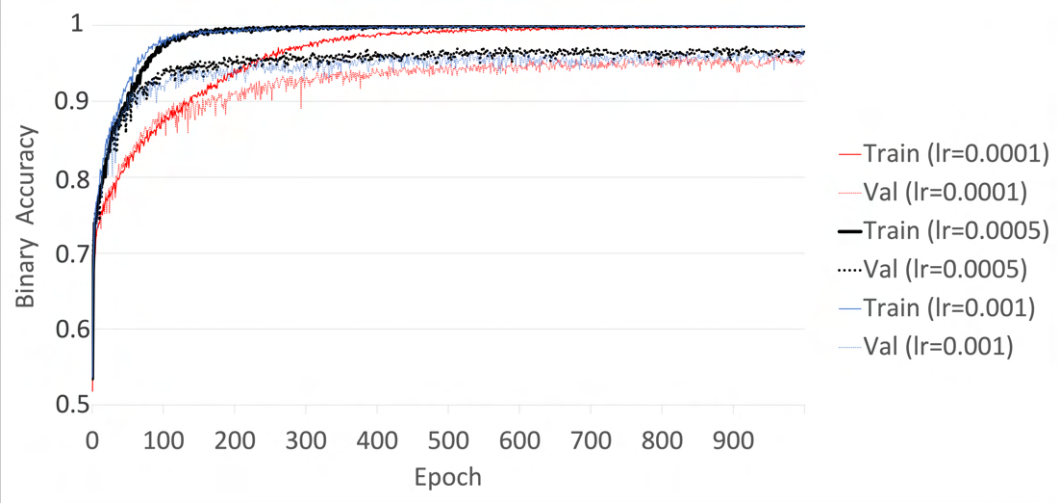


Figure 4.13: Binary CNN learning rate accuracy comparison.

Dropout

Different values for the dropout were used, 0 (no dropout), 0.1 (10%), 0.2 (20%), 0.3 (30%), 0.4 (40%). The best dropout value was 0.2 (20%) with an accuracy of 97,36%.

Figure 4.14 represents the binary comparison (train and validation) of the CNN dropout.

The 20% dropout, the best one, is represented by the black colour. Red, blue, green and yellow represent 0%, 10%, 30% and 40%, respectively.

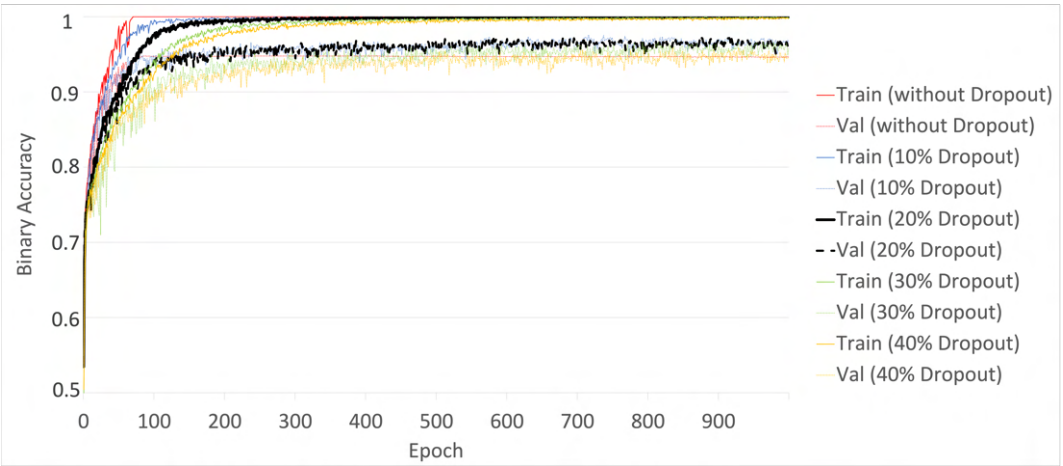


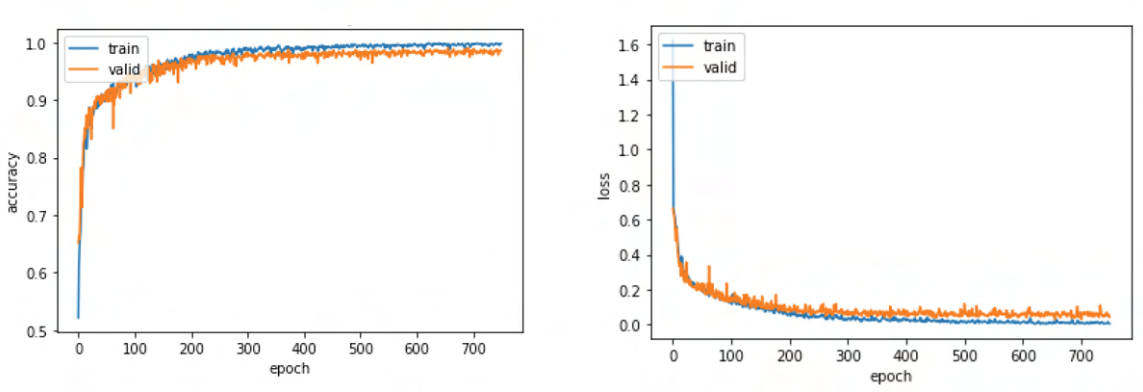
Figure 4.14: Binary CNN dropout comparison.

With this last step, the parameter tuning of the 1-D convolutional model is concluded. The best model parameters were three convolutional layers followed by four dense layers with 768 neurons in each dense and the ReLU activation function. The best learning parameters are the Adam optimizer, binary crossentropy loss function, a batch size of 256 and 1000 epochs, a learning rate of 0.0005 and 20% dropout.

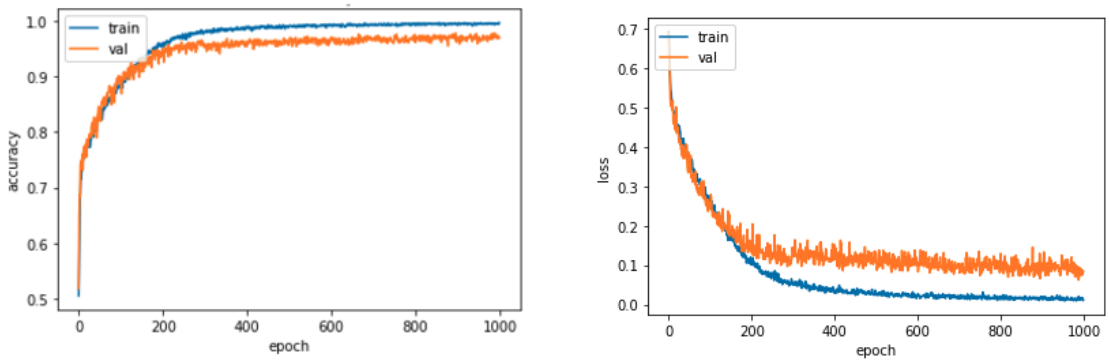
4.3 Results

After the tuning process for finding the best parameters, this section provides the results of comparing the performance of the feedforward neural network against the 1D-CNN. Figure 4.15 represents the learning curves over time. The model is evaluated on the training dataset and by an hold out validation dataset during the iterative learning process. The analysis of these curves indicates that the training and validation datasets are suitably representative and the models did not overfit. Anyway, the convolutional model shows a greater gap between the training and the validation learning curves, which may indicate less generalization ability.

The prediction on the test dataset is evaluated using two metrics: the accuracy and the confusion matrix. In addition to the results obtained on the test dataset, a cross-validation process is performed in order to evaluate the estimators predictive power and how well they generalize. At the end, the performance results are represented as a function of a mean value and a standard deviation. The cross-validation approach allows to evaluate the estimators for predictive power and how well they generalize across different test (validation) sets. The cross-validation was performed with the KerasClassifier, StratifiedKFold and CrossValScore. The CrossValScore returns the result of the cross-validation: when all the k-folds are concluded, the mean of them all and the standard deviation of the quality measure are performed. A high mean and low standard deviation means that the modelling technique is doing well. The cross-validation is in charge of dividing randomly the dataset. In this study, it was chosen a stratified 10-fold cross-validation to evaluate the predictive power and the generalization ability of the neural models.



(a) ANN model accuracy and loss.



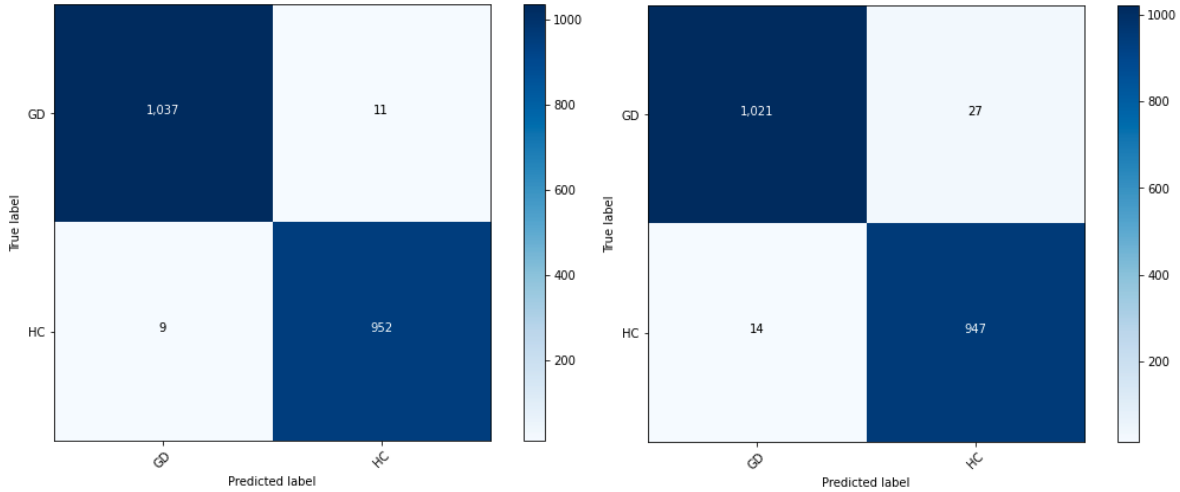
(b) 1-D CNN model accuracy and loss.

Figure 4.15: Binary ANN and 1-D CNN model accuracy and loss.

Table 4.4 compares the two models in study, the multilayer perceptron (ANN model) and the convolutional network (1D-CNN model). The ANN model obtained a cross-validation accuracy of 98.42%. This result shows that the model generalizes well, showing a good predictive ability. The model obtained an accuracy of 99% on the test dataset, only failing to classify correctly 20 trials of a total of 2009 (1%). After cross validation, the mean is 0.9842 (98.42%) with a standard deviation of 0.0053. Further, this model proved to be the one with the best results in all the experiments performed.

Neural Network	**Train**	**Val**	**Test**	**Conf.Matrix Correct/Wrong**	**10-fold Cross-val**
ANN	99.7%	98.8%	99%	GD - 1037 / 11 (98.95%) HC - 952 / 9 (99.06%)	98.42%
1-D CNN	99.68%	97.21%	97.96%	GD - 1021 / 27 (97.42%) HC - 947 / 14 (98.54%)	97.17%

Table 4.4: Results comparison of the ANN vs. 1D-CNN models.



(a) ANN Confusion Matrix.

(b) 1-D CNN Confusion Matrix.

Figure 4.16: Binary ANN and 1-D CNN confusion-matrices.

The application of 1-D CNN models to the binary human gait classification demonstrated to be also very effective with a cross-validation accuracy of 97.92%. The model achieved an accuracy of 97.96% on the test set, only failing to classify correctly 41 trials of a total of 2009 (2.04%). In this case, the mean of all results is 0.9717 (97.17%) with a standard deviation of 0.0091. Figure 4.16 depicts the confusion matrices generated for the ANN and the models. On one hand, the ANN model was able to correctly identify 1037 (98.95%) gait disorders out of 1048 and 952 (99.06%) healthy controls out of 961. It only failed to classify 20 trials out of 2009. On the other hand, CNN model was able to identify 1021 (97.42%) gait disorders out of 1048 and 947 (98.54%) healthy controls out of 961. It only failed to classify 41 trials out of 2009.

Chapter 5

Neural Network Multi-class Classification

This chapter discusses the comparison and performance of two neural networks, artificial and convolutional, when dealing with the multi-class classification of human gait disorders. It covers all the steps performed in the multi-class classification, starting with the dataset preparation, moving to the parameter tuning and ending with the final results. As stated previously, 1-D convolutions are compared to ANN in order to ensure a proper comparability. The dataset used in this work is the GaitRec, currently the most complete GRFs dataset [14].

5.1 Dataset Preparation

The dataset preparation for the multi-class problem is different from the binary one. Figure 5.1 represents the GaitRec dataset overview. As its possible to observe, there is not a balanced number of trials for each class. Taking into account the binary classification case, it is possible to foresee that the division given by the authors of the article it is not adapted for all problems, being important to manipulate the dataset according to the specific needs. With this in mind, a new division of train-validation-test set needs to be done.

Class	N	Age (yrs.) Mean (SD)	Body mass (kg) Mean (SD)	Sex (m/f)	Bi-lateral Trials
Healthy C.	211	34.7 (13.9)	73.9 (15.6)	104/107	7,755
Hip	450	42.6 (12.8)	82.4 (15.6)	373/77	12,748
Knee	625	41.6 (12.0)	84.3 (18.6)	426/199	19,873
Ankle	627	41.6 (11.4)	87.0 (18.0)	498/129	21,386
Calcaneus	382	43.5 (10.4)	84.0 (14.5)	339/43	13,970
Total	**2,295**	**41.5 (12.1)**	**83.6 (17.3)**	**1,740/555**	**75,732**

Figure 5.1: GaitRec Database Overview (Remember) (taken from [14]).

5.1.1 Selection of relevant inputs

Similarly to the binary dataset preparation, the metadata file contains information that is relevant and helps to make a proper dataset division. The most relevant fields to consider are

sex, age, speed and session type. As explained before, men and women walk differently, the age influences the walk, the speed affects the GRF measures and the best session to compare all the subjects is the initial measurement. To manipulate the dataset only the speed and session type were factors of exclusion, more properly trials with fast speed and trials that were not initial measurement. With this, a new training, validation and test set were created with all the information.

Identically the binary dataset preparation, all the information was processed first with all the data together (75732 trials) and only then divided randomly into train-validation-test sets. There are 3 different speed categories in the metadata file, slow, self-selected and fast. The multi-class problem implicates comparing a healthy subject that can walk perfectly at many different speeds with a subject with a gait disorder that, at the most, only can walk at a slow/self-selected speed. With this, using the fast walking speed makes no sense in the context of the problem, in which its objective is to classify human gait disorders. Additionally, it is also necessary to consider the different session types. In this context, the control measurement and the readmission could mess up the classifications and so, were also excluded. The sex and the age are also relevant and are included in the predictors set, in the subsequent subsections.

5.1.2 Final dataset

At this point only 20130 trials are left. Of these 20130, 5021 are from healthy controls, 4545 from knee disorders, 4365 from ankle disorders, 2867 from calcaneus disorders and 3332 from hip disorders. To achieve a proper result, all classes need to have approximately the same number of trials to ensure that the algorithm learns all the possibilities in the same way, making it as general as possible, thus covering all the possibilities. With this, it was decided to reduce healthy, knee, ankle and hip classes to 2867 trials, the number of trials of the calcaneus, the smallest class. It is relevant to refer that the trials were chosen randomly. Reducing the number of trials made the number 20130 pass to 14335. Now that the dataset information is adapted to the problem it is necessary to divide in train, validation and test sets. Figure 5.2 represents the train-validation-test division. A division of 70% for training,

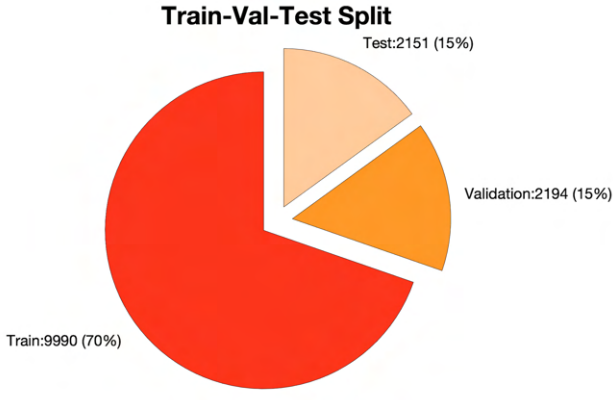


Figure 5.2: GaitRec balanced multi-class train-val-test split.

15% for validation and 15% for test was reached containing 9990, 2194 and 2151 trials for each respectively. The division was performed with sklearn train_test_split. Figure 5.3 represents the class distribution for each set of the new multi-class dataset. All the data was processed

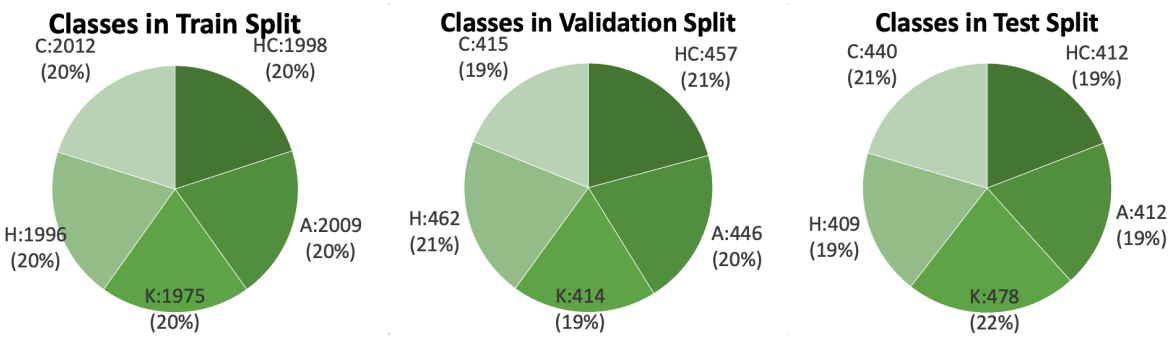


Figure 5.3: GaitRec balanced multi-class class split.

with Python3. In the next sections and with the purpose of building a neural network, the information of each train-validation-test trial will be handled as predictors and classes. In this case and to remember, predictors can be seen as the "hints" provided to the model so it can establish what class variable it needs to be assigned to each trial. The classes are labels/targets of the data that categorize the predictors information. In a classification problem, the predictors contain information that points to one of the classes.

5.2 Parameter Tuning

The tuning process is one way to find the best parameters for a model. The tuning chooses, among several possibilities, the most favourable parameters that allow to finish a learning task in the best possible manner.

The input for both neural networks is a 1x609-dimensional vector that represents the horizontal concatenation of the left and right foot ground reaction forces and the relevant metadata information (e.g., sex, speed and age). The order of the concatenation is left foot first and then right foot. So, to elucidate better, the input information is left medio-lateral, left anterior-posterior, left vertical, right medio-lateral, right anterior-posterior, right vertical, sex, age, speed. These represent the input predictors. The classes are five, healthy controls, ankle disorder, knee disorder, hip disorder, calcaneus disorder and are represented in the model by 3, 0, 4, 2 and 1, respectively.

All the experiments realized in the next subsections were performed with the train and validation sets. The metric was accuracy for this multi-class classification. All graphics are presented to corroborate the tuning result.

5.2.1 ANN

The starting ANN (Multilayer Perceptron), similar to the binary problem, contains three consecutive fully connected layers that lead to a final output layer. The size of each hidden layer was 768 whereas the size of the output layer was 5, the number of classes (healthy, ankle disorder, knee disorder, hip disorder, calcaneus disorder). The output activation was softmax due to the fact that the problem is a multi-class one. Additionally, the size of the batch size and epochs was random and was 512 and 750, respectively. It is important to consider that the value of the neurons, layers, batch size, epochs, learning rate and dropout will be studied in the subsequent sub sections. The parameters submitted to the first tuning, like in

the binary model, were the activation function, the kernel initializer, the optimizer and the loss function. For the activation, the compared functions were ReLU vs. Tanh, for the kernel initializer the Random vs. He vs. Glorot (Normal and Uniform), for the optimizer the Adam vs. SGD and for the loss function the sparse categorical crossentropy vs. hinge vs. poisson.

The results of the tuning, in other words the best parameters, were Tanh for the activation, Adam for the optimizer, sparse categorical crossentropy for the loss function and random normal to the kernel initializer.

Neurons and Dense Layers

Creating a viable neural network involves using the right amount of layers and neurons. It is important to study the influence of each layer and the number of neurons in order to have a proper result in the end. The objective is to compare the number of neurons and layers. Starting with 2 layers, the objective is to compare the number of neurons (384, 450, 600, 768, 880, 1000, 1200, 1500). After that, add another layer on and on and compare the same number of neurons till reach the max accuracy and low loss. The value of the learning rate, decay, epochs and batch size is a fixed value (default for the learning rate and decay) and 750 and 512 for the other 2 parameters.

Table 5.1 presents the results of the performed tuning and their best parameters/scores.

Dense Layers	**Best Result**
2	Neurons: 1000 Acc: 93.71%
3	Neurons: 768 Acc: 94.33%
4	Neurons: 1000 Acc: 93.52%
5	Neurons: 768 Acc: 92.8%

Table 5.1: ANN Multi-class tuning best parameters/score and layers.

As it is possible to observe, the best accuracy was obtained with 3 layers, each with 768 neurons, with an accuracy of 94.33%.

Batch Size

Four batch sizes were compared, more precisely 128, 256, 512 and 1024. The best batch size was 1024 with an accuracy of 95.2%. After that was tested the 1024 vs. 2048 and the 1024 obtained the best accuracy.

Figure 5.4 represents the multi-class comparison (train and validation) of the ANN batch size. The 1024 batch, the best one, is represented by the black colour. Red, green and blue represent 512, 256 and 128, respectively.

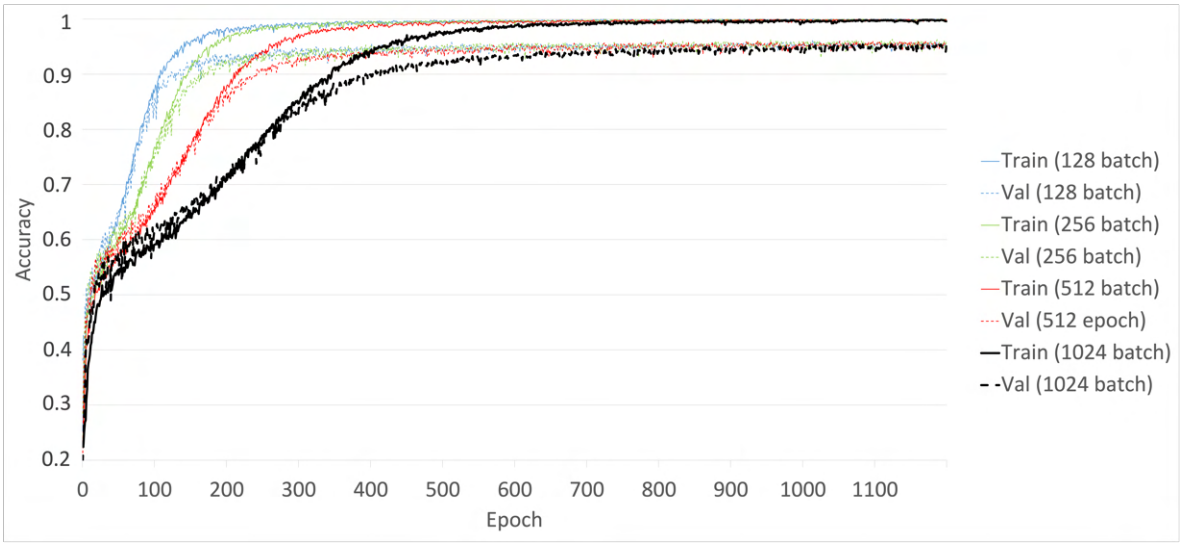


Figure 5.4: Multi-class ANN batch size comparison.

Number of Epochs

Four epoch sizes were set side by side, more properly 750, 1000, 1250 and 1500. The best epoch number was 1200 with an accuracy of 95.2%.

Figure 5.5 represents the multi-class comparison (train and validation) of the ANN epoch size. The 1250 epoch size, the best one, is represented by the black colour. Blue, red and green represent 750, 1000 and 1500, respectively.

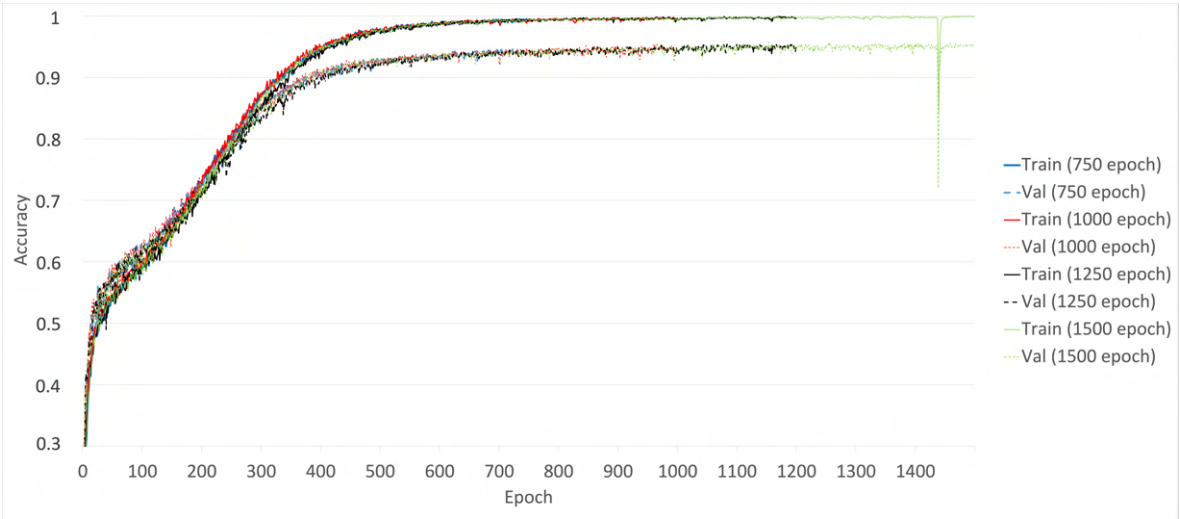


Figure 5.5: Multi-class ANN epoch size comparison.

Learning Rate

The learning rate is one of the most important parameters to optimize. It handles the step size for model weight updates regarding the loss function. It has a small positive value

and controls how fastly the model is adapted to the problem. Different values for the learning rate and decay were submitted to the tuning process, 0.0001, 0.0005 and 0.001. The best learning rate was 0.0005 and the decay 0.0001 with an accuracy of 95,8%.

Figure 5.7 and Figure 5.6 represent the multi-class comparison (train and validation) of the ANN learning rate. The 0.0005, the best one, is represented by the black colour. Red, blue and green represent 0.0001, 0.001 and 0.005, respectively.

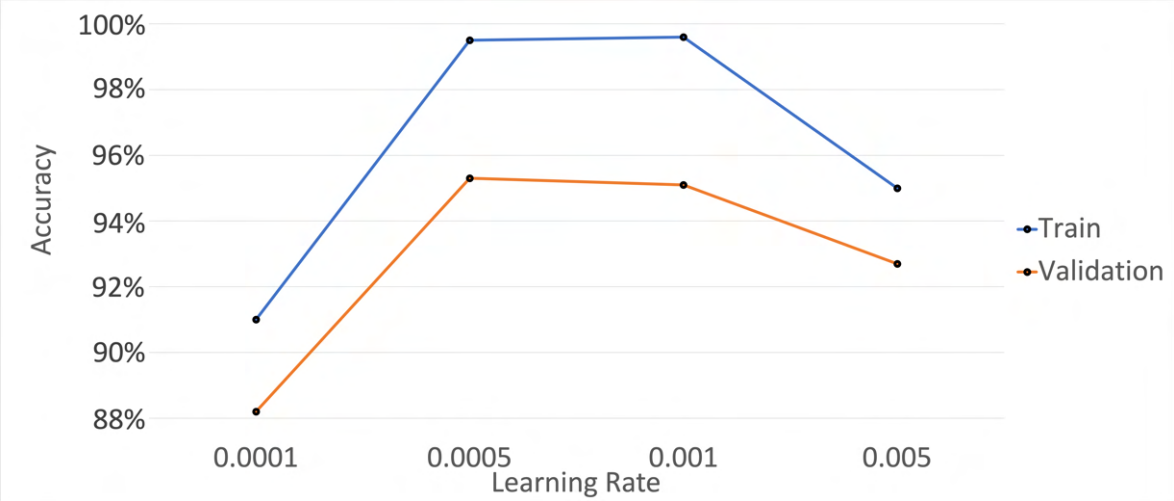


Figure 5.6: Multi-class ANN learning rate comparison.

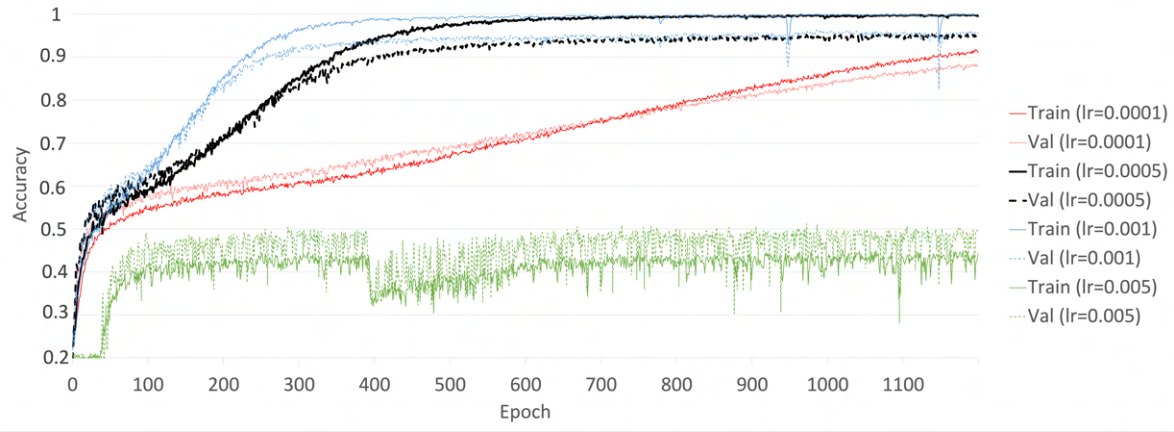


Figure 5.7: Multi-class ANN learning rate accuracy comparison.

Dropout

The dropout, like the name implies, drops out some random neurons in a neural network during an iteration in the training phase. The dropout plays a major role in preventing a model from overfitting. In this case it is essential to analyze if an addition of dropout will increase the performance of the model, and if yes analyse the one that most benefits the model. The dropouts were added after each dense layer. Five values for the dropout were

used in the tuning process, 0 (no dropout), 0.1 (10%), 0.2 (20%), 0.3 (30%), 0.4 (40%). The best dropout was 0.2 (20%) with an accuracy of 96.4%. In addition, 0.3 (30%) also presented decent results.

Figure 5.8 represents the multi-class comparison (train and validation) of the ANN dropout. The 0.2 (20%), the best one, is represented by the black colour. Red, blue, green and orange represent 0 (without dropout), 0.1 (10%), 0.3 (30%) and 0.4 (40%), respectively.

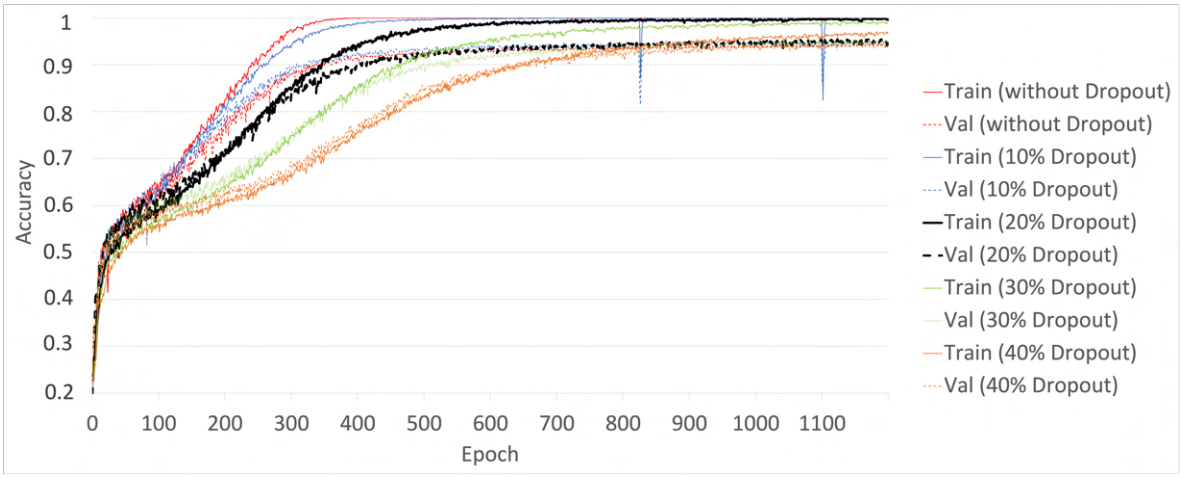


Figure 5.8: Multi-class ANN dropout comparison.

After that, more tests were made and it was concluded that the best choice was to insert a 0.25 (25%) dropout only in the first two layers, with an accuracy of 97%.

With this, the ANN parameter tuning is concluded. The best parameters were Tanh for the activation, Adam for the optimizer, sparse categorical crossentropy for the loss function, random normal for the kernel initializer, 3 dense layers with 768 neurons each, 1024 for the batch size, 1250 epochs, 0.0005 for the learning rate and 0.25 for the dropout.

5.2.2 CNN

To start, it was also used as reference the binary CNN model. The output activation was a softmax due to the fact that the problem was a multi-class one. The batch size and epochs value was random and was 512 and 750, respectively. The parameters submitted to the tuning were the activation function, the optimizer and the loss function. Specifically, the parameters were the same as the multi ANN ones. The results of the tuning, the best parameters, were also ReLU for the activation, Adam for the optimizer and sparse categorical crossentropy for the loss function.

Number of Convolutional and Dense Layers

Like in the ANN, in order to create a viable neural network it is necessary to use the right amount of layers and neurons. Additionally, it is important to study the influence of each layer and the number of neurons in order to have a proper result in the end.

Starting with 3 convolutional layers several values for the filter were submitted to the tuning, 24, 32, 48, 64, 128 and 256. After that, the process was to increase on and on by 1 the number of convolutional layers and to do the tuning again. Table 5.2 presents the results

of the performed tuning and their best parameters/scores. Filters represent the the number of output filters in the convolution.

Conv Layers	**Best Result**
3	Filters: 64 - 256 - 512 Acc: 76.5%
4	Filters: 64 - 64 - 128 - 256 Acc: 84.24%
5	Filters: 64 - 64 - 128 - 256 - 512 Acc: 65.8%

Table 5.2: CNN Multi-class tuning best parameters/score and layers.

As it is possible to observe, the best choice is 4 convolutional layers with an accuracy of 84.24%. The next step is to assess whether or not it is worth putting a fully connected network after the flattening process.

Dense Layers	**Best Result**
1	Neurons: 768 Acc: 85.9%
2	Neurons: 1250 Acc: 86%
3	Neurons: 1250 Acc: 89.3%
4	Neurons: 1250 Acc: 90.58%
5	Neurons: 1000 Acc: 91.34%
6	Neurons: 1250 Acc: 90.59%

Table 5.3: CNN Multi-class tuning best parameters/score with dense layers.

Table 5.3 displays the results of the performed tuning with 4 convolutional layers. The best accuracy was obtained with more 5 dense layers, each with 1000 neurons, with 91.34%.

Batch Size

Five different values for the batch size were used in the tuning process, 128, 256, 512, 1024 and 2048. The best batch size was 1024 with an accuracy of 93.23%.

Figure 5.9 represents the multi-class comparison (train and validation) of the CNN batch size. The 1024 batch, the best one, is represented by the black colour. Orange, blue, red, green represent 128, 256, 512 and 2048, respectively.

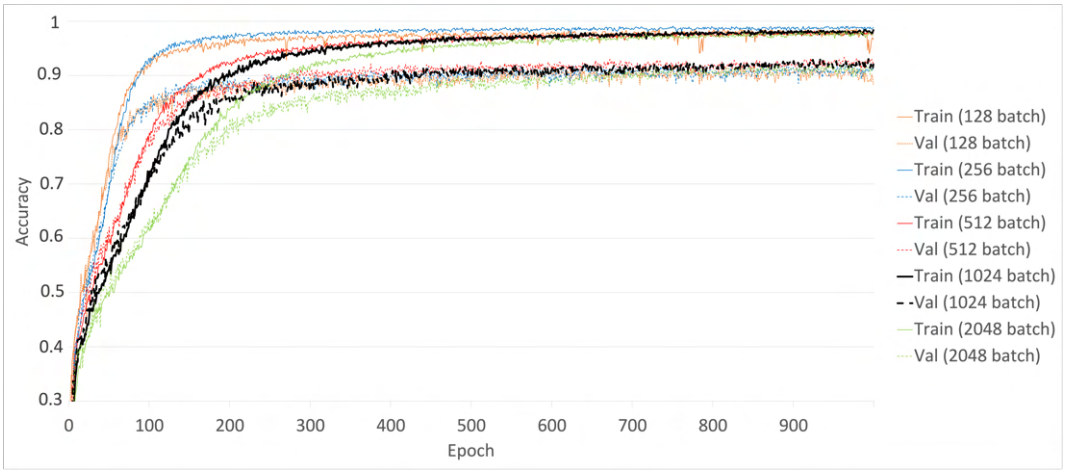


Figure 5.9: Multi-class CNN batch size comparison.

Number of Epochs

The number of epochs used in the tuning process were 750, 1000, 1250 and 1500. The best epoch number was 1250 with an accuracy of 93.23%. 1500 has also presented decent results.

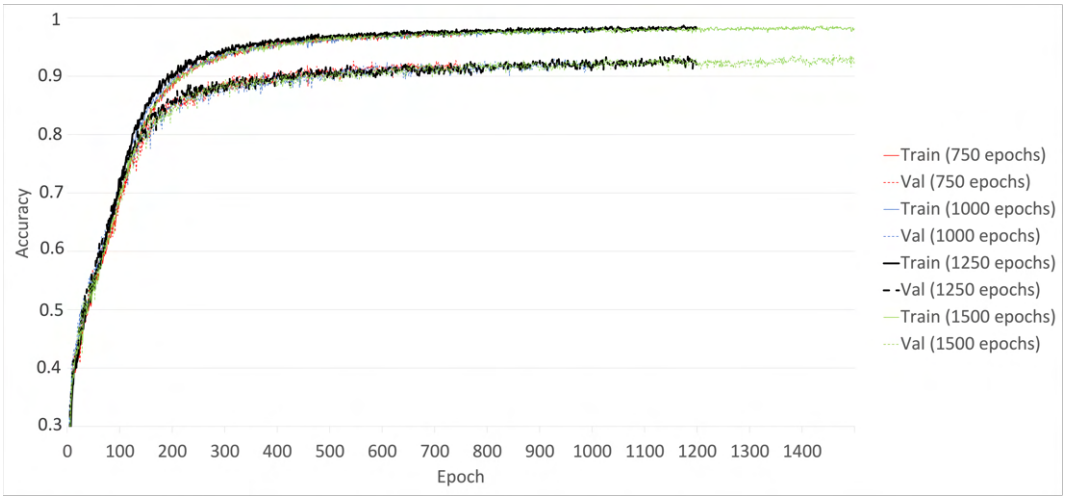


Figure 5.10: Multi-class CNN epoch size comparison.

Figure 5.10 represents the multi-class comparison (train and validation) of the CNN epoch

size. The 1250 epoch value, the best one, is represented by the black colour. Red, blue and green represent 750, 1000 and 1500, respectively.

Learning Rate

Four different values for the learning rate and decay were submitted to the tuning process, 0.001, 0.005, 0.0001, 0.0005. The best learning rate was 0.001 and the decay 0.0001 with an accuracy of 93.8%.

Figure 5.12 and Figure 5.11 represent the multi-class comparison (train and validation) of the CNN learning rate. The 0.001, the best one, is represented by the black colour. Blue, red and green represent 0.0001, 0.0005 and 0.005, respectively.

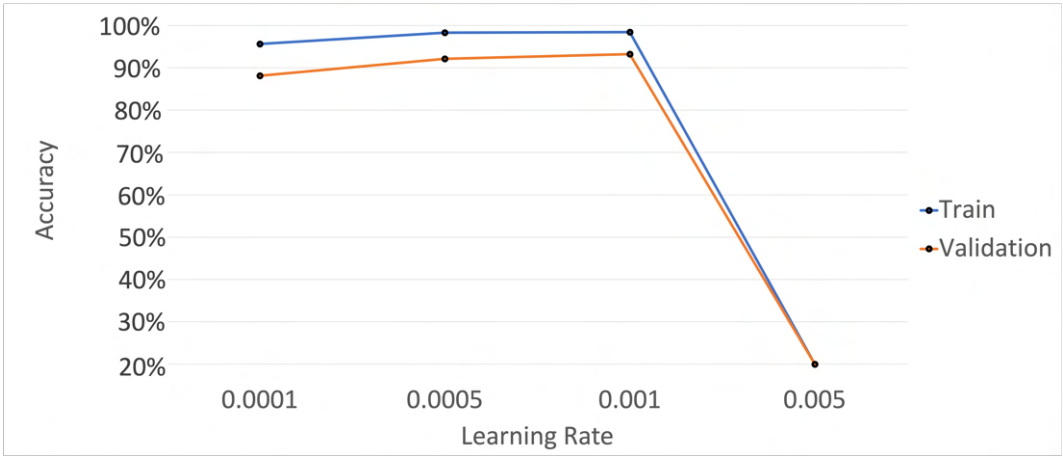


Figure 5.11: Multi-class CNN learning rate comparison.

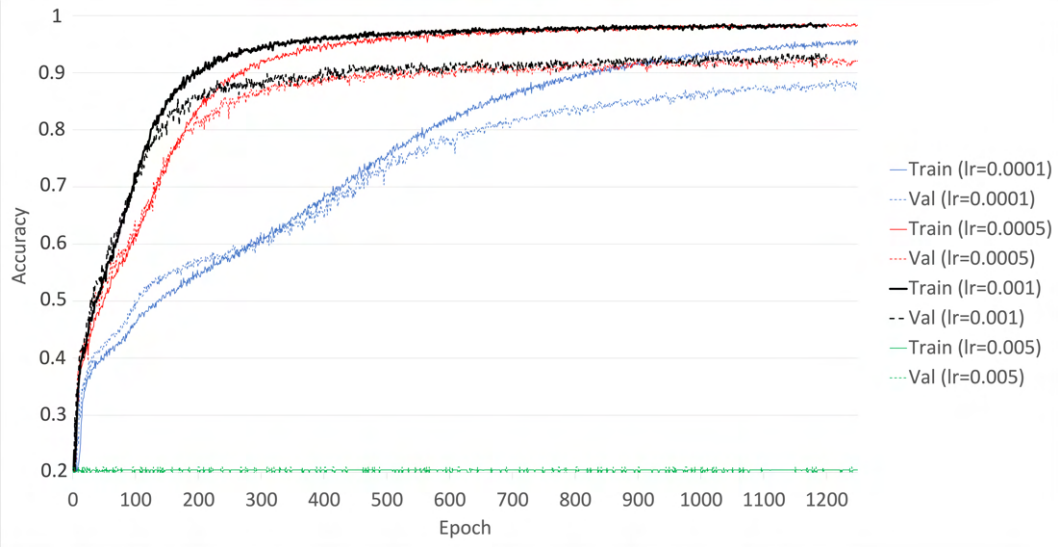


Figure 5.12: Multi-class CNN learning rate accuracy comparison.

Dropout

Different values for the dropout were used, 0 (no dropout), 0.1 (10%), 0.2 (20%), 0.3 (30%), 0.4 (40%). The best dropout value was 0.3 (30%) with an accuracy of 95.6%.

Figure 5.13 represents the multi-class comparison (train and validation) of the CNN dropout. The 0.3 (30%), the best one, is represented by the black colour. Red, blue, orange and green represent 0 (without dropout), 0.1 (10%), 0.2 (20%) and 0.4 (40%), respectively.

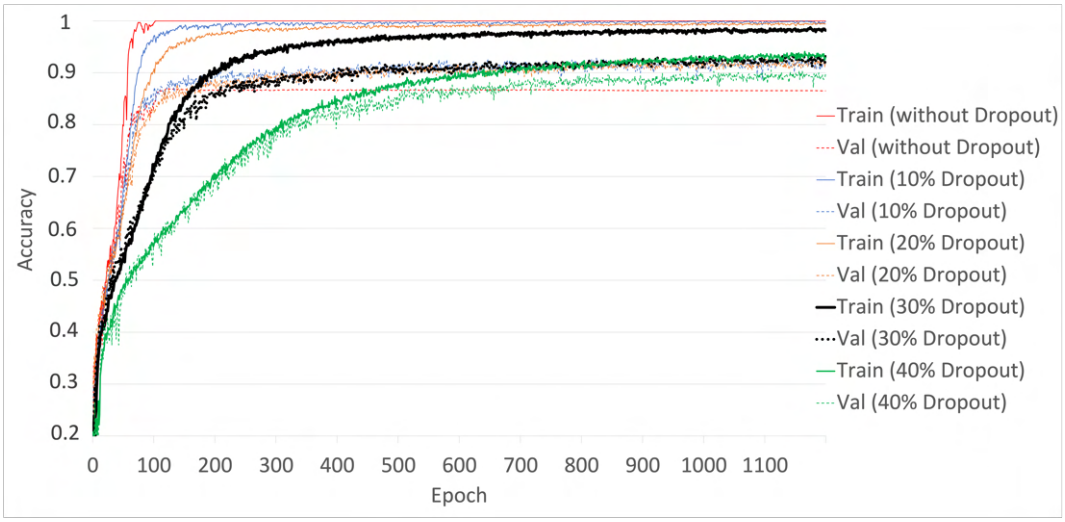


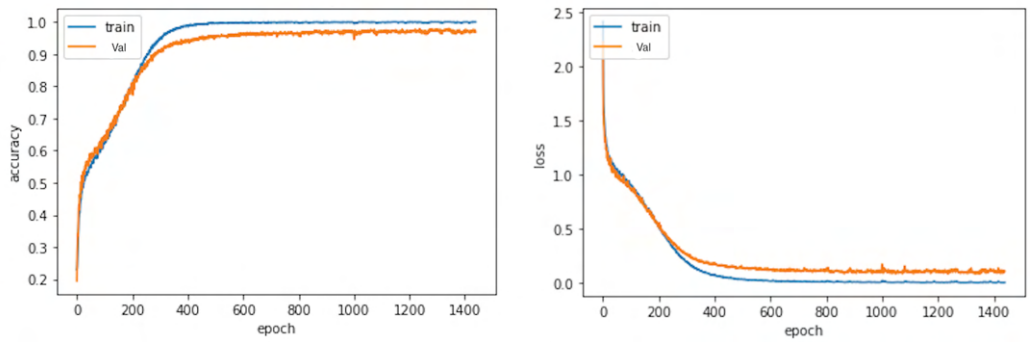
Figure 5.13: Multi-class CNN dropout comparison.

After that more tests were performed in order to find where the dropout was more valuable. With a dropout in the second and fourth convolutional layer and also with one in every dense layer was obtained an accuracy of 95.9%. With this, the 1-D CNN parameter tuning is concluded. The best parameters were ReLU for the activation, Adam for the optimizer, sparse categorical crossentropy for the loss function, 4 convolutional layers followed by 5 dense layers with 1000 neurons in each dense, 1024 for the batch size, 1250 epochs, 0.001 for the learning rate and 0.3 for the dropout.

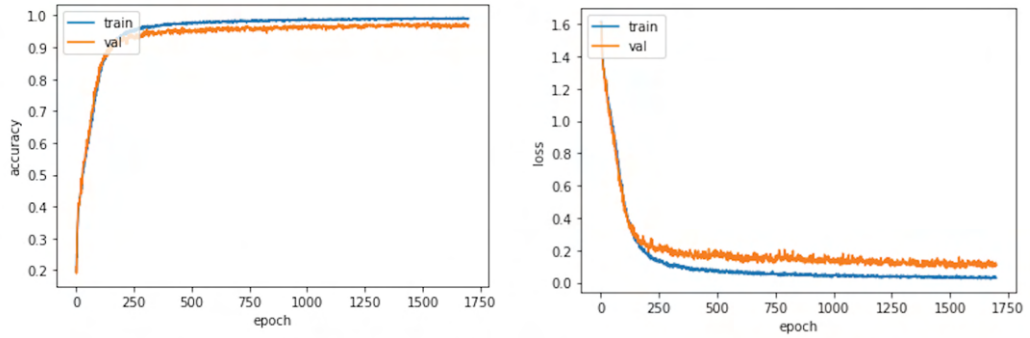
5.3 Results

As in the binary classifier, the results are concentrated on the analysis of learning curves and the evaluation of the prediction accuracy on the test dataset and the confusion matrices. Figure 5.14 represents the learning curves, including the evolution of the model accuracy and the loss function over several iterations. Once again, the fully-connected ANN model shows a good behavior, being more likely the model generalizes correctly from the test data.

The results achieved on the test dataset are depicted in the form of prediction accuracy (see Table 5.4) and confusion matrices (see Figure 5.15). The ANN model achieves a slightly higher value of test accuracy (97.16% against 96.23% for the CNN). At the same time, the confusion matrices provide a better idea of the predictions failures and successes for each class.

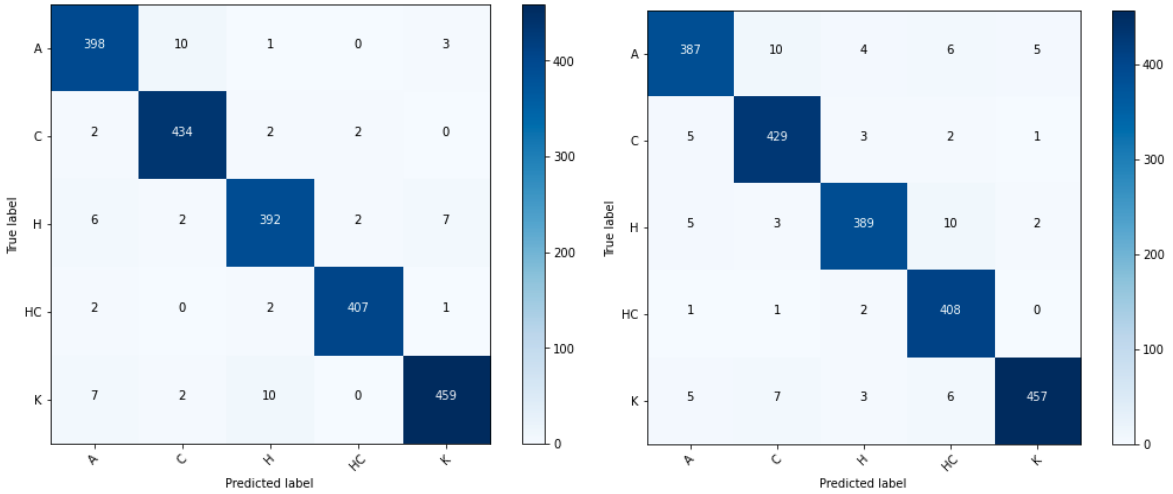


(a) ANN model accuracy and loss.



(b) 1-D CNN model accuracy and loss.

Figure 5.14: Multi-class ANN and 1-D CNN model accuracy/loss.



(a) ANN confusion matrix.

(b) 1-D CNN confusion matrix.

Figure 5.15: Multi-class ANN and 1D-CNN confusion matrices.

Neural Network	**Train**	**Val**	**Test**	**Conf.Matrix Correct/Wrong**	**10-fold Cross-val**
ANN	99.9%	96.8%	97.16%	A - 398 / 14 (96.6%) C - 434 / 6 (98.6%) H - 392 / 17 (95.8%) HC - 407 / 5 (98.8%) K - 459 / 19 (96%)	97.6%
1-D CNN	99.05%	96.3%	96.23%	A - 387 / 25 (93.9%) C - 429 / 11 (97.5%) H - 389 / 20 (95.1%) HC - 408 / 4 (99%) K - 457 / 21 (95.6%)	96.31%

Table 5.4: Multi-class ANN vs. 1-D CNN comparison.

On one hand, the ANN model succeeded in identifying 2090 test classes out of 2151 (97.16%). More precisely, 398 of 412 ankle disorders (96.6%), 434 of 440 calcaneus disorders (98.6%), 392 of 409 hip disorders (95.8%), 407 of 412 healthy controls (98.8%) and 459 of 478 knee disorders (96%). On the other hand, the 1D-CNN model was able to identify 2070 test classes out of 2151 (96.23%): 387 of 412 ankle disorders (93.9%), 429 of 440 calcaneus disorders (97.5%), 389 of 409 hip disorders (95.1%), 408 of 412 healthy controls (99%) and 457 of 478 knee disorders (95.6%). The results obtained with the cross-validation process confirm that the ANN network produces better average results than the 1D-CNN. More concretely, the ANN attains a mean accuracy of 0.9760 (97.60%) with standard deviation of 0.0049, while the 1D-CNN performance stands at 0.9631 (96.31%) for a deviation of 0.0051.

Chapter 6

Image-Based Convolutional Neural Network Classification

As a new approach to the problem, the creation of 2-D images emerged. This appeared as a new proposal for the CNN approach and it was not known if it would give good results. The goal was to transform the 1-D GRF into an image. Due to the fact that in the early stages of the problem the results were not so good, two solutions were proposed based on their frequent use in similar problems. One was to generate an image with a 2-D line plot where on the x-axis were the GRF of the left foot and on the y-axis those of the right foot. The other one consisted also on an image with a 2-D line plot where on the x-axis were the GRF of the left foot and right foot concatenated and on the y-axis the derivative of the x-axis. The two represent valid solutions and after some discussion the first one was the chosen one. The second one was not discarded because it also represents a possibility in a future work. Again, the binary and the multi-class classification will be performed using the final dataset of Chapter 4 and 5, respectively.

6.1 Binary Classification

This section approaches the binary classification of human gait disorders using 2-D convolutional neural network. The input of the network will be an image, created with the final GRF dataset of the Chapter 4. It covers all the steps performed in the binary classification, starting with the dataset preparation, moving to the parameter tuning and ending with the final results. The next subsection describes the dataset preparation and how the images were generated.

6.1.1 Dataset Preparation

A Matlab script was created to process the data. The dataset with all the train-validation-test data from the 1-D binary section was applied to this one. All left and right foot GRF were analysed (anterior-posterior, medio-lateral and vertical) in order to find the min and the max for each and to define the overall limits separately.

The values defined were -0.4 and 0.4 for the anterior-posterior, -0.15 and 0.15 for the medio-lateral and 0.015 and 1.45 for the vertical. Then, all the data were normalized between -1 and 1. Consequently, a color for each GRF was chosen to differentiate them. The colors

were black for the vertical, green for the anterior-posterior and red for the medio-lateral. The size of all images is 224x224. Figure 6.1 represents an example of a generated healthy image (left) and disorder one (right). It is important to refer that the information about the sex, age and speed is not present in the 2-D CNN.

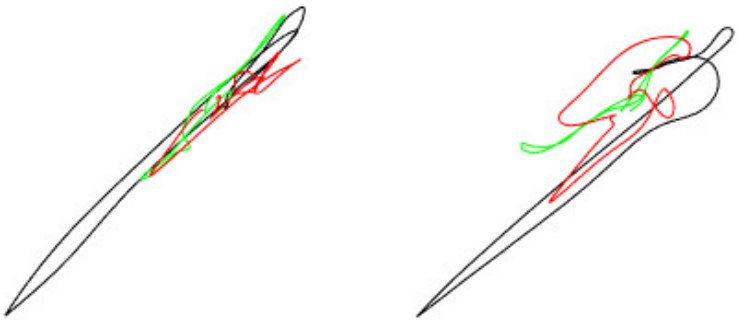


Figure 6.1: Binary 2-D generated image (Healthy vs. Gait Disorder)

Again, the colors of the Figure 6.1 are black for the vertical GRF, green for the anterior-posterior GRF and red for the medio-lateral GRF.

6.1.2 Parameter Tuning

One way to find the best parameters for a model is to perform the tuning. The tuning chooses, amongst several possibilities, the most favourable parameters that allow to finish a learning task in the best possible manner. The input vector will be the images generated with the left and right GRF. The classes are two, healthy control and gait disorder. The metric is binary accuracy since we are dealing with a binary problem. Just like on CNN 1-D, the starting point was to mimic the authors neural network parameters and merge them with other ones. The CNN contained three consecutive convolutional layers that lead to a final output layer. The convolutional layers had as filter size for the first, second and third convolution layer 8, 8 and 6, respectively, as stride 2, 2 and 3 and for the output channel 24, 24, 48. Between the convolution layers there was always a 1-D max pooling layer with a pool size of 2.

After the third convolution, the resulting information was flattened. It was subsequently fed into a fully connected layer with only a binary output layer. The output activation was a sigmoid. With this and to start, the architecture of the 2-D Binary CNN will be the same. The size of the batch size and epochs was random and was 32 and 500, respectively. As stated above, it is important to consider that the value of the batch size, layers, epochs, learning rate and dropout will be studied in the subsequent sub sections. The results of the tuning, in other words the best parameters, were ReLU for the activation, Adam for the optimizer and binary cross-entropy for the loss function.

Number of Convolutional and Dense Layers

In order to create a viable neural network it is necessary to use the right amount of layers and neurons. Additionally, it is important to study the influence of each layer and the number

of neurons in order to have a proper result in the end.

Starting with 4 convolutional layers several values for the filter were submitted to the tuning, 48, 64, 128 and 256. After that, the process was to increase on and on by 1 the number of convolutional layers and to do the tuning again. Table 6.1 presents the results of the performed tuning and their best parameters/scores. Filters represent the the number of output filters in the convolution.

Conv Layers	**Best Result**
4	Filters: 64 - 64 - 128 - 256 Acc: 82.11%
5	Filters: 64 - 64 - 128 - 128 - 256 Acc: 83.01%
6	Filters: 64 - 64 - 128 - 128 - 256 - 512 Acc: 81.17%

Table 6.1: CNN 2D Binary tuning best parameters/score and layers.

As it is possible to observe, the best choice is 5 convolutional layers with a mean accuracy of 83.01%. The next step is to assess whether or not it is worth putting a fully connected network after the flattening process. Table 6.2 displays the results of the performed tuning with 5 convolutional layers. The best accuracy was obtained with two more dense layers, with an accuracy of 83.4%.

Dense Layers	**Best Result**
1	Neurons: 768 Acc: 82.96%
2	Neurons: 768 Acc: 83.4%
3	Neurons: 768 Acc: 82.11%

Table 6.2: CNN 2D Binary tuning best parameters/score with dense layers.

Batch Size, Number of Epochs, Learning Rate and Dropout

Three different values (8, 16 and 32) for the batch size were used in the tuning process, the best batch size being 16. The number of epochs used in the tuning process were 500, 750 and 1000, resulting in 750 epochs. Four different values for the learning rate and decay were submitted to the tuning process: 0.001, 0.005, 0.0001 and 0.0005. The best value after

tuning was 0.0001. The best percentage of dropout was 20%, when considering a range from no dropout to 40% dropout.

The optimization of the model performance on the validation set led to five convolutional layers followed by two dense layers, with 768 neurons in each dense. At the same time, the best parameters were ReLU for the activation function, Adam for the optimizer, binary crossentropy for the loss function. A batch size of 16 for 750 epochs, a learning of 0.0001 and 20% dropout were the results after parameter tuning.

6.2 Multi-class Classification

This section approaches the multi-class classification of human gait disorders using 2-D convolutional neural network. The input of the network will be an image, created with the final GRF dataset of the Chapter 5. It covers all the steps performed in the multi-class classification, starting with the dataset preparation, moving to the parameter tuning and ending with the final results. The next subsection describes the dataset preparation and how the images were generated.

6.2.1 Dataset Preparation

The dataset used for this study was generated with the 14332 trials that resulted from the filtered 1-D task. The chosen train-validation-test division was 60%-20%-20% and, like so, contains 8600 train trials, 2865 validation trials and 2867 test trials. The train set contains 1752 healthy trials, 1705 ankle disorders, 1717 knee disorders, 1711 hip disorders and 1716 calcaneus disorders. The validation set contains 545 healthy trials, 569 ankle disorders, 559 knee disorders, 591 hip disorders and 603 calcaneus disorders. The test set contains 570 healthy trials, 593 ankle disorders, 591 knee disorders, 565 hip disorders and 548 calcaneus disorders.

A Matlab script was created to process the data. All left and right foot GRF were analyzed (anterior-posterior, medio-lateral and vertical) in order to find the min and the max for each and to define the overall limits separately. The values defined were -0.4 and 0.4 for the anterior-posterior, -0.15 and 0.15 for the medio-lateral and 0.015 and 1.5 for the vertical. Then, all the data were normalized between -1 and 1. Consequently, a color for each GRF was chosen to differentiate them. The colors were black for the vertical GRF, green for the anterior-posterior GRF and red for the medio-lateral GRF. The size of all images is 224x224. Figure 6.2 represents an example of a generated image. It is important to refer that the information about the sex, age and speed is not present in the 2-D CNN.

6.2.2 Parameter Tuning

The tuning process is one way to find the best parameters for a model. The tuning chooses, among several possibilities, the most favourable parameters that allow to finish a learning task in the best possible manner. To elucidate better, the input information was left and right medio-lateral, anterior-posterior and vertical GRF as an image. These represent the input predictors. The classes were five, healthy controls, ankle disorder, knee disorder, hip disorder, calcaneus disorder. The metric was accuracy for this multi-class classification.

To start, it was also used as reference the binary CNN model. The output activation was a softmax due to the fact that the problem was a multi-class one. The batch size and

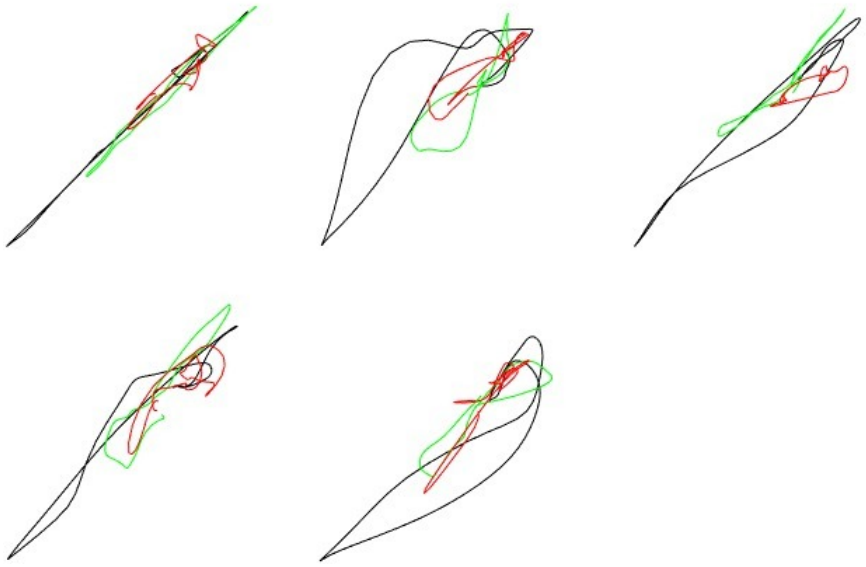


Figure 6.2: Multi-class 2-D generated images (Healthy, Ankle, Hip, Knee, Calcaneus).

epochs value was random and was 512 and 750, respectively. The parameters submitted to the tuning were the activation function, the optimizer and the loss function. Specifically, the parameters are the same as the CNN ones. The results of the tuning, more specifically the best parameters, were ReLU for the activation, Adam for the optimizer, sparse categorical crossentropy for the loss function.

Number of Convolutional and Dense Layers

In order to create a viable neural network it is necessary to use the right amount of layers and neurons. Additionally, it is important to study the influence of each layer and the number of neurons in order to have a proper result in the end. Starting with 3 convolutional layers several values for the filter were submitted to the tuning, 32, 48, 64, 128 and 256. After that, the process was to increase on and on by 1 the number of convolutional layers and to do the tuning again. Table 6.3 presents the results of the performed tuning and their best parameters/scores. Filters represent the the number of output filters in the convolution.

As it is possible to observe, the best choice is 4 convolutional layers with a mean accuracy of 53%. The next step is to assess whether or not it is worth putting a fully connected network after the flattening process. Table 6.4 displays the results of the performed tuning with 4 convolutional layers. The best accuracy, 54.4%, was obtained with only one dense layer. It is watchable that by increasing the number of layers the model performance decreases.

Batch Size, Number of Epochs, Learning Rate and Dropout

The previous results are very unsatisfactory and the tuning process has been simplified at this point. The results described below were obtained with the following set of parameter values: 16 batch size, 0.0001 learning rate, and 10% dropout.

Conv Layers	Best Result
3	Filters: 48 - 64 - 128: Acc: 48%
4	Filters: 48 - 64 - 128 - 256 Acc: 53%
5	Filters: 32 - 48 - 64 - 128 - 256 Acc: 50%

Table 6.3: CNN 2D Multi-class tuning best parameters/score and layers.

Dense Layers	Best Result
1	Neurons: 256 Acc: 54.4%
2	Neurons: 128 Acc: 52%
3	Neurons: 256 Acc: 40%

Table 6.4: CNN 2D Multi-class tuning best parameters/score with dense layers.

6.3 Results

Table 6.5 displays the preliminary results of the image-based CNN model. The overall performance is 85.00% for the binary classification problem and 55.25% for the multi-class model. The binary classifier has a much lower performance than previous models, although there still seems to be some space for improvement. However, the multi-class classifier shows unsatisfactory results and additional efforts will be needed in other to understand how to take advantage of these approach.

Type	Neural Network	Train	Val	Test
Binary	2-D CNN	92.5%	85.71%	85%
Multi	2-D CNN	75.69%	56.82%	55.25%

Table 6.5: Binary and multi-class 2-D CNN comparison.

Chapter 7

Conclusions

7.1 Final Discussion

The main focus of this dissertation was the application of supervised machine learning techniques for the classification of human gait disorders using the annotated GaitRec dataset. The study addressed two classification problems solved using neural networks. First, a binary classification problem associated to classifying normal against impaired gait. Second, the problem of gait classification across five classes of disorders affecting the hip, knee, ankle, and calcaneus. The main goal was to compare the performance of feedforward multilayer perceptrons against convolutional neural networks in terms of prediction accuracy.

Table 7.1 summarizes the results obtained by the binary and multi-class models in terms of prediction accuracy. For the binary classification, the MLP yielded an accuracy of 99.00%, the 1D-CNN 97.96% and the image-based CNN 85.00%. The multi-class classifiers follow the same trend, with an accuracy of 97.16% for the MLP network, 96.23% for the 1D convolutional network and 55.25% for the image-based convolutional network. The results of the study show that neural network models are reliable in classifying both healthy vs. pathological gait and different gait disorders. The MLP model shows consistently slightly higher prediction accuracy on the same test dataset. However, this result must be analyzed in perspective taking into account the many optimizations that could have been made. The preliminary results obtained with image-based 2D-CNN are much lower and even unsatisfactory in the case of the multi-class problem. Additional efforts will be needed in other to understand how to take advantage of these approach.

Type	**Neural Network**	**Test Accuracy**
Binary	**ANN**	**99.00%**
Binary	1-D CNN	97.96%
Binary	2-D CNN	85.00%
Multi-class	**ANN**	**97.16%**
Multi-class	1-D CNN	96.23%
Multi-class	2-D CNN	55.25%

Table 7.1: Comparison of results of the binary and multiclass classification using a MLP model, a 1D-CNN and an image-based 2D-CNN.

Reference	Binary Performance	Multi-class Performance
Lozano-Ortiz et al. [60]	ANN: 92% SOM: 96%	-
Alaqtash et al. [61]	KNN: 85% ANN: 80%	-
Slijepcevic et al. [62]	SVM: 90%	SVM: 53.3%
Slijepcevic et al. [63]	SVM: 90.8%	SVM: 54.3%
Slijepcevic et al. [64]	-	SVM: 67.8%
Slijepcevic et al. [65]	CNN Best: 97% Mean: 89% SVM Best: 95% Mean: 89% ANN Best: 97% Mean: 90%	CNN Best: 72% Mean: 53% SVM Best: 62% Mean: 52% ANN Best: 63% Mean: 50%

Table 7.2: Best binary and multi-class classification accuraccies of human gait disorders.

Table 7.2 presents the best results obtained in related works using GRFs, including recent studies of the authors of the GaitRec dataset. These works report as the best results values around the 97% for the binary classification and 72% for the multi-class problem. The proposed framework outperforms the state-of-the-art, achieving a classification accuracy of 99.00% for the binary problem and 97.16% for the multi-class problem. The study was developed from the GaitRec dataset considering potential biases which may affect the generalization ability of the models. In line with this, the data preparation stage played a preponderant role in the performance of the learning models. In particular, the use of a sub-set appropriate to the problem in question was decisive to improve the prediction accuracy.

7.2 Future Work

Although the results are promising, many different adaptations, tests, and experiments can be performed as future work. The following points summarize some aspects related to the improvements of the work and others related to new directions of research:

- While training the different models on the GaitRec dataset produced promising results in the test set, this is a proof-of-concept stage that requires validation with clinical data. The experiments with real data are usually very time consuming, and it was not possible to achieve this within the scope of this work.
- The results obtained when converting the GRF time series into an 2D image suggest the need for further study that should explore alternative ways of encoding the data. For example, a coordinate plane with axes being the values of the derivative of the GRF variables as a function of the GRFs. The generation of QR codes for each trial of the GRF dataset is an alternative that could be investigated.

- It could be interesting to explore recently introduced deep architectures such as a CNN called Inception Time or Echo State Networks. In the former, the convolutional layers and the pooling layers are replaced with inception modules. The later architecture are able to reduce the problems of Recurrent Neural Networks by eliminating the need to compute the gradient for the hidden layers, reducing the training time and avoiding the vanishing gradient problem.
- The proposed algorithms ignore additional information beyond the input data. Recent methodologies are used to understand deep structures (model interpretation), i.e., explaining the classifier's decisions by measuring the contribution of each input variable to the overall prediction.

References

[1] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[2] Stefano A. Bini. Artificial intelligence, machine learning, deep learning, and cognitive computing: What do these terms mean and how will they impact health care? *The Journal of Arthroplasty*, 33(8):2358–2361, August 2018. DOI: 10.1016/j.arth.2018.02.067.

[3] Pin Wang, En Fan, and Peng Wang. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, 141:61–67, January 2021. DOI: 10.1016/j.patrec.2020.07.042.

[4] Jianqiang Wang, Heye Huang, Keqiang Li, and Jun Li. Towards the unified principles for level 5 autonomous vehicles. *Engineering*, January 2021. DOI: 10.1016/j.eng.2020.10.018.

[5] N Kousika, G Vishali, S Sunandhana, and M Arvind Vijay. Machine learning based fraud analysis and detection system. *Journal of Physics: Conference Series*, 1916(1):012115, May 2021. DOI: 10.1088/1742-6596/1916/1/012115.

[6] Yilei Zeng, Aayush Shah, Jameson Thai, and Michael Zyda. Applied machine learning for games: A graduate school course, 2020. arXiv: 2012.01148.

[7] Abdullah S. Alharthi, Syed U. Yunas, and Krikor B. Ozanyan. Deep learning for monitoring of human gait: A review. *IEEE Sensors Journal*, 19(21):9575–9591, November 2019. DOI: 10.1109/jsen.2019.2928777.

[8] Jianqiao Tian, Glenn Smith, Han Guo, Boya Liu, Zehua Pan, Zijie Wang, Shuangyu Xiong, and Ruogu Fang. Modular machine learning for alzheimer's disease classification from retinal vasculature. 11(1), January 2021. DOI: 10.1038/s41598-020-80312-2.

[9] Ebru Aydındag Bayrak, Pınar Kırcı, and Tolga Ensari. Comparison of machine learning methods for breast cancer diagnosis. In *2019 Scientific Meeting on Electrical-Electronics Biomedical Engineering and Computer Science (EBBT)*, pages 1–3, 2019. DOI: 10.1109/EBBT.2019.8741990.

[10] João P. Santos, João P. Ferreira, Manuel Crisóstomo, and A. Paulo Coimbra. Instrumented shoes for 3d GRF analysis and characterization of human gait. In *Bioinformatics and Biomedical Engineering*, pages 51–62. Springer International Publishing, 2019. DOI: 10.1007/978-3-030-17935-9_6.

[11] Tanmay Tulsidas Verlekar, Paulo Lobato Correia, and Luis Ducla Soares. Using transfer learning for classification of gait pathologies. IEEE, December 2018. DOI: 10.1109/bibm.2018.8621302.

[12] Alvaro Muro de-la Herran, Begonya Garcia-Zapirain, and Amaia Mendez-Zorrilla. Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications. *Sensors*, 14(2):3362–3394, February 2014. DOI: 10.3390/s140203362.

[13] Tom Chau. A review of analytical techniques for gait data. part 1: fuzzy, statistical and fractal methods. *Gait & Posture*, 13(1):49–66, February 2001. DOI: 10.1016/s0966-6362(00)00094-1.

[14] Brian Horsak, Djordje Slijepcevic, Anna-Maria Raberger, Caterine Schwab, Marianne Worisch, and Matthias Zeppelzauer. GaitRec, a large-scale ground reaction force dataset of healthy and impaired gait. *Scientific Data*, 7(1), May 2020. DOI: 10.1038/s41597-020-0481-z.

[15] Joana Figueiredo, Cristina P. Santos, and Juan C. Moreno. Automatic recognition of gait patterns in human motor disorders using machine learning: A review. *Medical Engineering & Physics*, 53:1–12, March 2018. DOI: 10.1016/j.medengphy.2017.12.006.

[16] Angkoon Phinyomark, Giovanni Petri, Esther Ibáñez-Marcelo, Sean T. Osis, and Reed Ferber. Analysis of big data in gait biomechanics: Current trends and future directions. *Journal of Medical and Biological Engineering*, 38(2):244–260, July 2017. DOI: 10.1007/s40846-017-0297-2.

[17] Alfonso Fasano and Bastiaan R. Bloem. Gait disorders. *CONTINUUM: Lifelong Learning in Neurology*, 19:1344–1382, October 2013. DOI: 10.1212/01.con.0000436159.33447.69.

[18] Walter Pirker and Regina Katzenschlager. Gait disorders in adults and the elderly. *Wiener klinische Wochenschrift*, 129(3-4):81–95, October 2016. DOI: 10.1007/s00508-016-1096-4.

[19] Éric Watelain. Human gait: From clinical gait analysis to diagnosis assistance. *Movement & Sport Sciences*, n° 98(4):3, 2017. DOI: 10.3917/sm.098.0003.

[20] Abdul Saboor, Triin Kask, Alar Kuusik, Muhammad Mahtab Alam, Yannick Le Moullec, Imran Khan Niazi, Ahmed Zoha, and Rizwan Ahmad. Latest research trends in gait analysis using wearable sensors and machine learning: A systematic review. *IEEE Access*, 8:167830–167864, 2020. DOI: 10.1109/access.2020.3022818.

[21] Ziad O. Abu-Faraj, Gerald F. Harris, Peter A. Smith, and Sahar Hassani. *Human gait and Clinical Movement Analysis*, pages 1–34. American Cancer Society, 2015. DOI: 10.1002/047134608X.W6606.pub2.

[22] Philipp Mahlknecht, Stefan Kiechl, Bastiaan R. Bloem, Johann Willeit, Christoph Scherfler, Arno Gasperi, Gregorio Rungger, Werner Poewe, and Klaus Seppi. Prevalence and burden of gait disorders in elderly men and women aged 60–97 years: A population-based study. *PLoS ONE*, 8(7):e69627, July 2013. DOI: 10.1371/journal.pone.0069627.

[23] L Sudarsky. Gait disorders: prevalence, morbidity, and etiology. *Adv. Neurol.*, 87:111–117, 2001. PMID: 11347214.

[24] R. K. Begg, R. Wytch, and R. E. Major. Instrumentation used in clinical gait studies: A review. *Journal of Medical Engineering & Technology*, 13(6):290–295, January 1989. DOI: 10.3109/03091908909016204.

[25] D. TarniŢă. Wearable sensors used for human gait analysis. *Rom J Morphol Embryol*, 57(2):373–382, 2016. PMID: 27516008.

[26] Chandra Prakash, Rajesh Kumar, and Namita Mittal. Recent developments in human gait research: parameters, approaches, applications, machine learning techniques, datasets and challenges. *Artificial Intelligence Review*, 49(1):1–40, September 2016. DOI: 10.1007/s10462-016-9514-6.

[27] Weijun Tao, Tao Liu, Rencheng Zheng, and Hutian Feng. Gait analysis using wearable sensors. *Sensors*, 12(2):2255–2283, February 2012. DOI: 10.3390/s120202255.

[28] Scottmark Communications. Biometrics goniometers and torsiometers. `http://www.nexgenergo.com/ergonomics/biosensors.html`.

[29] Michael Kent. *The Oxford dictionary of sports science & medicine*. Oxford University Press, Oxford New York, 2006. ISBN: 9780191727788.

[30] Sangram Redkar. A review on wearable inertial tracking based human gait analysis and control strategies of lower-limb exoskeletons. *International Robotics & Automation Journal*, 3(7), December 2017. DOI: 10.15406/iratj.2017.03.00080.

[31] Yongbin Qi, Cheong Boon Soh, Erry Gunawan, Kay-Soon Low, and Rijil Thomas. Assessment of foot trajectory for human gait phase detection using wireless ultrasonic sensor network. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 24(1):88–97, January 2016. DOI: 10.1109/tnsre.2015.2409123.

[32] R.I. Spain, R.J. St. George, A. Salarian, M. Mancini, J.M. Wagner, F.B. Horak, and D. Bourdette. Body-worn motion sensors detect balance and gait deficits in people with multiple sclerosis who have normal walking speed. *Gait & Posture*, 35(4):573–578, April 2012. DOI: 10.1016/j.gaitpost.2011.11.026.

[33] A.L. Adkin, B.R. Bloem, and J.H.J. Allum. Trunk sway measurements during stance and gait tasks in parkinson's disease. *Gait & Posture*, 22(3):240–249, November 2005. DOI: 10.1016/j.gaitpost.2004.09.009.

[34] Shimmer. Shimmer3 emg unit. `https://www.shimmersensing.com/products/shimmer3-emg-sensor`.

[35] Ramesh Jain. *Machine vision*. McGraw-Hill, New York, 1995. ISBN: 0-07-032018-7.

[36] Andreas Kolb, Erhardt Barth, Reinhard Koch, and Rasmus Larsen. Time-of-Flight Sensors in Computer Graphics. In M. Pauly and G. Greiner, editors, *Eurographics 2009 - State of the Art Reports*. The Eurographics Association, 2009. DOI: 10.2312/egst.20091064.

[37] Optitrack. Motion capture systems. `https://optitrack.com/`.

[38] Vicon. Award winning motion capture systems, Sep 2021. `https://www.vicon.com/`.

[39] J.P. Ferreira, M.M. Crisostomo, and A.P. Coimbra. Human gait acquisition and characterization. *IEEE Transactions on Instrumentation and Measurement*, 58(9):2979–2988, September 2009. DOI: 10.1109/tim.2009.2016801.

[40] João P. Ferreira, Alexandra Vieira, Paulo Ferreira, Manuel Crisóstomo, and A. Paulo Coimbra. Human knee joint walking pattern generation using computational intelligence techniques. *Neural Computing and Applications*, 30(6):1701–1713, March 2018. DOI: 10.1007/s00521-018-3458-5.

[41] Filippo Casamassima, Alberto Ferrari, Bojan Milosevic, Pieter Ginis, Elisabetta Farella, and Laura Rocchi. A wearable system for gait training in subjects with parkinson's disease. *Sensors*, 14(4):6229–6246, March 2014. DOI: 10.3390/s140406229.

[42] Gang Ge, Wei Huang, Jinjun Shao, and Xiaochen Dong. Recent progress of flexible and wearable strain sensors for human-motion monitoring. *Journal of Semiconductors*, 39(1):011012, January 2018. DOI: 10.1088/1674-4926/39/1/011012.

[43] Kistler. Ground reaction force (grf). `https://www.kistler.com/en/glossary/term/ground-reaction-force-grf/`.

[44] Tekscan. Pressure mapping, force measurement, & tactile sensors. `https://www.tekscan.com/`.

[45] ProtoKinetics. Improve patient outcomes with protokinetics zeno walkway and pkmas software. `https://www.protokinetics.com/improve-patient-outcomes-with-protokinetics-zeno-walkway-and-pkmas-software/`.

[46] Kistler. Sistemas de medição e sensores. `https://www.kistler.com/pt/`.

[47] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, Inc, Sebastopol, CA, 2019. ISBN: 9781492032649.

[48] Eugene Charniak. *Introduction to deep learning.* The MIT Press, Cambridge, Massachusetts, 2018. ISBN: 9780262039512.

[49] Oliver Theobald. *Machine learning for absolute beginners : a plain English introduction.* The author, United States, 2017. ISBN: 9781549617218.

[50] Michael A. Nielsen. Neural networks and deep learning, 2018. `http://neuralnetworksanddeeplearning.com/`.

[51] Francois Chollet. *Deep Learning with Python.* Manning Publications, dec 2017. ISBN: 9781617294433.

[52] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016. arXiv: 1609.04747.

[53] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. arXiv: 1412.6980.

[54] Nikhil Ketkar. *Deep learning with Python : a hands-on introduction.* Apress, United States, 2017. ISBN: 978-1-4842-2766-4.

[55] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[56] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, may 2015. DOI: 10.1038/nature14539.

[57] Ningning Yi, Chunfang Li, Xin Feng, and Minyong Shi. Research and improvement of convolutional neural network. In *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*. IEEE, June 2018. DOI: 10.1109/icis.2018.8466474.

[58] Shiva Verma. Understanding 1d and 3d convolution neural network: Keras, Jul 2020. https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610.

[59] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, August 2013. DOI: 10.1109/tpami.2013.50.

[60] C A Lozano-Ortiz, A M S Muniz, and J Nadal. Human gait classification after lower limb fracture using artificial neural networks and principal component analysis. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*. IEEE, August 2010. DOI: 10.1109/iembs.2010.5626715.

[61] Murad Alaqtash, Thompson Sarkodie-Gyan, Huiying Yu, Olac Fuentes, Richard Brower, and Amr Abdelgawad. Automatic classification of pathological gait patterns using ground reaction forces and machine learning algorithms. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 453–457, 2011. DOI: 10.1109/IEMBS.2011.6090063.

[62] Djordje Slijepcevic, Brian Horsak, Caterine Schwab, Anna-Maria Gorgas, Michael Schüller, Arnold Baca, Christian Breiteneder, and Matthias Zeppelzauer. Ground reaction force measurements for gait classification tasks: Effects of different PCA-based representations. *Gait & Posture*, 57:4–5, September 2017. DOI: 10.1016/j.gaitpost.2017.07.009.

[63] Djordje Slijepcevic, Matthias Zeppelzauer, Anna-Maria Gorgas, Caterine Schwab, Michael Schuller, Arnold Baca, Christian Breiteneder, and Brian Horsak. Automatic classification of functional gait disorders. *IEEE Journal of Biomedical and Health Informatics*, 22(5):1653–1661, September 2018. DOI: 10.1109/jbhi.2017.2785682.

[64] D. Slijepcevic, M. Zeppelzauer, C. Schwab, A.-M. Raberger, B. Dumphart, A. Baca, C. Breiteneder, and B. Horsak. P 011—towards an optimal combination of input signals and derived representations for gait classification based on ground reaction force measurements. *Gait & Posture*, 65:249–250, September 2018. DOI: 10.1016/j.gaitpost.2018.06.155.

[65] Djordje Slijepcevic, Fabian Horst, Sebastian Lapuschkin, Anna-Maria Raberger, Matthias Zeppelzauer, Wojciech Samek, Christian Breiteneder, Wolfgang I. Schöllhorn, and Brian Horsak. On the explanation of machine learning predictions in clinical gait analysis, 2019. arXiv: 1912.07737.

[66] D. Slijepcevic, M. Zeppelzauer, A. M. Gorgas, C. Schwab, M. Schüller, A. Baca, C. Breiteneder, and B. Horsak. Automatic classification of functional gait disorders. *IEEE Journal of Biomedical and Health Informatics*, 22(5):1653–1661, 2018. DOI: 10.1109/JBHI.2017.2785682.

[67] Sagar Patel. Data science essentials: Why train-validation-test data?, Sep 2018. `https://medium.datadriveninvestor.com/data-science-essentials-why-train-validation-test-data-b7f7d472dc1f`.

[68] Andreas ller. *Introduction to machine learning with Python : a guide for data scientists*. O'Reilly Media, Inc, Sebastopol, CA, 2017. ISBN: 9781449369415.

[69] Salla Ruuska, Wilhelmiina Hämäläinen, Sari Kajava, Mikaela Mughal, Pekka Matilainen, and Jaakko Mononen. Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle. *Behavioural Processes*, 148:56–62, March 2018. DOI: 10.1016/j.beproc.2018.01.004.

[70] George Fisher (grfiv4). Plot a confusion matrix, Apr 2017. `https://www.kaggle.com/grfiv4/plot-a-confusion-matrix`.