POLITECNICO DI TORINO

Master of Science's Degree in Biomedical Engineering

Politecnico
di Torino

Master of Science's Degree Thesis

A machine learning approach for spatio-temporal gait analysis based on a head-mounted inertial sensor

Supervisors

Prof. Andrea CEREATTI

Prof.ssa Gabriella BALESTRA

Prof.ssa Samanta ROSATI

Dott.ssa Francesca SALIS

Candidate

Paolo TASCA

December 2022

Abstract

Gait is fundamental for the person's mobility, as it is crucial for many activities in the workplace, domestic environment, and social life. In the last decades, several studies proved the relevance of instrumented gait analysis for clinical and wellness applications based on quantitative metrics (e.g., spatio-temporal parameters of gait, kinematics) to provide deeper insights about individual walking ability, especially when analyzing gait in free-living conditions, where motor performances can be assessed. In this sense, magneto-inertial measurement units (MIMUs) represent the most convenient solution in terms of ease of use and affordability. The most used locations include trunk/lower back and wrist and have been widely explored. Conversely, less attention has been given to other sites, such as the head, which offers the possibility of integrating the MIMU with a wide range of devices, such as virtual reality (VR) visors, earbuds or sensor fusion devices. However, the informative content related to gait in signals recorded by head MIMUs (H-MIMUs) is generally lower than the one retrieved by MIMUs positioned at other body sites. MIMUs allow the estimation of spatio-temporal parameters (STP), such as the stride length, the stride time and the stride speed. While temporal parameters can be directly derived from the inertial signals, the stride speed can hardly be inferred deterministically from MIMU signals; therefore, the need for machine learning (MaLe)-based methods that find a mapping between stride features in the inertial signals and the corresponding stride speed value. The aim of the present thesis is to design methods for the assessment of STP based on data recorded from a single H-MIMU. The study focuses on the development of MaLe models for the analysis of gait on healthy young (HY) subjects both in supervised and unsupervised conditions. Reference data were collected with the INertial module with Distance sensors and Pressure insoles (INDIP) multi-sensor system, including four MIMUs positioned on the lower back, head and feet, pressure insoles and distance sensors. The data used for constructing and testing the models have been recorded by the single H-MIMU and acquired indoor in standardized conditions on 11 HY subjects (6 males, 26 ± 3 years) while performing a set of tasks (straight and round walking at three speeds) according to a precise experimental protocol. Models were also tested on 2.5 hours free-living recordings from 3 HY subjects (2 males, 22 ± 1 years), to assess their generalization skill on unseen data acquired in unsupervised conditions. Two models have been optimized to assess STP from H-MIMU data. A first deep learning (DeLe) classification model determines the occurrence instants of the gait events (initial and final foot-ground contacts). At this point, gait events are used to define the strides. Strides are given as input to a MaLe model that provides an estimate of the stride speed for each input

stride. For gait events detection (GED), two sets of DeLe classification models were compared: temporal convolutional networks (TCNs) and long-short term memory (LSTM) networks. For the estimation of the stride speed, two sets of MaLe regression models were compared: gaussian process regression (GPR) and support vector machine (SVM). Performance of the head-based method were validated comparing the results with those provided by the INDIP system. For gait event detection, TCNs showed better results than LSTM, as they achieved lower mean absolute error (MAE) values both on the training set (step time MAE = 0,02 ± 0,02 s; stride time MAE: 0,01 ± 0,02 s) and on the standardized test set (step time MAE = 0,06 ± 0,03 s; stride time MAE: 0,02 ± 0,03 s). For the LSTM networks, performance on the free-living dataset were significantly lower, while TCNs maintained a sufficient level of accuracy (84 %). For stride speed estimation, GPR and SVM provided similar results in terms of predicted values of stride speed, both showing significantly limited MAE values both on the training set (GPR: 0.05 m/s; SVM: 0.07 m/s) and the standardized test set (GPR: 0.07 m/s; SVM: 0.06 m/s). However, the models for the estimation of the stride speed on free-living data attained much lower performance, as the achieved coefficient of correlation does not overpass 0.62. Such results suggest that a single H-MIMU can match the performance of other single and multi-sensor configurations in the estimation of temporal parameters both in supervised and unsupervised conditions. A single H-MIMU can also provide reliable values of the stride speed in supervised conditions; however, applicability to unsupervised walking remains an open issue, as well as the extension of the methods to gait of pathological subjects.

Acknowledgements

I would like to thank all the people who contributed with their advice, their collaboration and their support to the development of this thesis.
This project would not have been possible without the support of Prof. Andrea Cereatti, who supervised all the stages of the work and guided me in its general organization. His support, experience, inspiration, enthusiasm and kindness have been priceless to me.
I would like to express my gratitude to Dott.ssa Francesca Salis. Since the first day, she has always assisted me through the many challenges of the project and has always found some time to answer to my many doubts and questions. Thank you from the bottom of my heart.
I wish to thank Prof.ssa Gabriella Balestra and Prof.ssa Samanta Rosati, my co-supervisors, who have showed great availability and provided me with smart suggestions throughout the course of the thesis work.
I also would like to thank all the guys from the Departement of Electronics and Telecommunications of Politecnico di Torino: Francesca, Rachele, Roberto, Giulia, Alba, Sogand, Francesco, Elena and Marco. I am grateful for all the time spent with you, sharing lunches or delicious caffè Leccese at Mixto. You are all brilliant people and I am sure that you have a bright future ahead of you.
Thanks to my friends: Andrea, for his true and sincere friendship, Michele, for his example of resilience and for always inspiring me, Elisabetta, for her binding affection, Sofia, for being by my side since my earliest childhood, Giorgia, for her kindness and empathy, Giovanni, for his leadership and confidence, Federico, for his overwhelming joy and positive attitude, and many others.
I want to express my love and gratitude to Lavinia, who has been my cornerstone in the last four years. Her love, support, patience and empathy have been invaluable to me, and I could never imagine any of my past or future accomplishments without her by my side.
Last - but not least - thanks to my family - Maria, Mimmo and Fabrizio - who have always encouraged and supported me in every challenge that I have undertaken.

"You've been putting it up your whole life. You just didn't know it."
Anton Chigurh, No Country for Old Men

Table of Contents

List of Tables

List of Figures

Acronyms

% GC
: % of the gait cycle

ϵ-SVM
: epsilon-insensitive SVM

ADAM
: adaptive moment estimation

ADL
: activities of daily life

AI
: artificial intelligence

ANN
: artificial neural network

AP
: anterior-posterior

AR
: augmented reality

ARD
: automatic relevance determination

B-MIMU
: lower back MIMU

BCI

brain-computer interface

BLE

bluetooth low-energy

BP

back-propagation

BPTT

back-propagation through time

CEL

cross-entropy loss

CM

confusion matrix

CNN

convolutional neural network

COPD

chronic obstructive pulmonary disease

CP

cerebral palsy

DeLe

deep learning

DET

Dipartimento di Elettronica e Telecomunicazioni

DNN

deep neural network

DoP

direction of progress

dps

degrees per second

DS

double support

DT

decision tree

EM

electro-magnetic

EMG

electromyogram

EU

European Union

F-MIMU

foot MIMU

FC

final contact

FCL

fully connected layer

FCN

fully convolutional network

FDA

flexible discriminate analysis

FES

functional electrical stimulation

FFT

fast Fourier transform

FN

false negative

FP

false positive

FSR

full-scale range

FSW

fast straight walking

GAD

gait activity detection

GAsD

gait asymmetry detection

GDD

gait disorder detection

GED

gait events detection

GPR

gaussian process regression

GS

gold standard

GUI

graphic user interface

H-MIMU

head MIMU

HMM

hidden markov model

HO

heel off

HS

heel strike

HY

healthy young

IC

initial contact

IIR

infinite impulse response

IMU

inertial measurement unit

INDIP

INertial module with Distance sensors and Pressure insoles

IR

infra-red

KKT

Karush-Kuhn-Tucker

KNN

k-nearest neighbours

LDA

linear discriminate analysis

LSE

least square errors

LSTM

long-short term memory

MAE

mean absolute error

MaLe

machine learning

MAPE

mean absolute percentage error

MAV

mean absolute value

ME

mean error

MEMS

micro electro-mechanical systems

MIMU

magneto-inertial measurement unit

ML

medial-lateral

MOCAP

MOtion CAPture

MSE

mean squared error

NaN

not a number

NPV

negative predictive value

NSW

normal straight walking

ODR

output data rate

PCA

principal component analysis

PCB

printed circuit board

PPV

positive predictive value

QDA

quadratic discriminate analysis

ReLu

rectified linear unit

RF

random forest

RL

reinforcement learning

RMS

Root Mean Squared

RMSE

root mean squared error

RMSProp

root mean square propagation

RNN

recurrent neural network

RW

round walking

Seq2Seq

sequence-to-sequence

SlSW

slow straight walking

SMA

signal magnitude area

SMO

sequential minimal optmization

SS

single support

STP

spatio-temporal parameters

SVM

support vector machine

SVR

support vector regression

tanh

hyperbolic tangent

TCN

temporal convolutional network

TN

true negative

TNR

true negative rate

TO

toe off

ToF

time of flight

ToS

toe strike

TP

true positive

TPR

true positive rate

TRS

training set

TS

test set

V

vertical

VR

virtual reality

VS

validation set

WHO

World Health Organization

WS

walking speed

Glossary

Machine learning A set of supervised techniques of artificial intelligence that aim at modelling a phenomenon.

Overfitting A model is said to "overfit the training data" when the features described by it arise from noise or variance in the specific training data, rather than the underlying population from which the data were sampled. Overfitting typically yields to achieve lower accuracy on unseen data [1].

Spatio-temporal parameters Space or/and time-domain variables related to walking e.g. stride duration, stride length, step duration, etc...

Stereophotogrammetry Stereophotogrammetric devices (also called optoelectronic devices) are marked-based systems devised to track with accuracy the trajectories of body segments on which the markers are placed, through the use of IR light and triangulation techniques.

Supervised techniques Methods of AI based on the training of a model through a labeled dataset i.e. a dataset of predictors with their responses. Unsupervised techniques are instead trained on non-labeled datasets i.e. a dataset of only predictors. Supervised techniques are applied in several tasks such as classification and segmentation, while unsupervised techniques are mainly used for clustering.

Chapter 1

Introduction

1.1 Motivation and general introduction

Today, instrumented assessment of gait plays a significant role within the clinical routine, since the quantitative analysis of gait can provide a number of bio-markers and parameters that support the evaluation of one's level of mobility and can assist clinicians in the shaping of rehabilitation programs [2]. In addition, being able to analyze and recognize the patterns of physiological/pathological gait can help to shape new strategies for the control in real-time of actuator systems for functional electrical stimulation (FES), bio-feedback and treatment of neurological disorders [3]. Out of the pathological framework, quantitative analysis of gait can also provide insights related to sport or wellness endeavors, monitor the physical activity of employees in the workplace and integrate with virtual reality (VR) or augmented reality (AR) systems for several purposes [4][5].
In the last decades, stereophotogrammetry has emerged as the gold standard (GS) for gait analysis, as it offers the chance to collect reliable kinematic and spatio-temporal parameters [6]. However, stereophotogrammetry is intrinsically constrained to the laboratory settings [7]; as it consists in a set of infra-red (IR) cameras and markers that require controlled light conditions and calibration procedures to work properly.
Today, magneto-inertial measurement units (MIMUs) represent a valid alternative to stereophotogrammetry [2]. A MIMU typically holds three tri-axial inertial sensors (an accelerometer, a gyroscope and a magnetometer) that are able to track the position and orientation of the sensor with respect to an internal framework. As a consequence, besides being compact, inexpensive and having low power consumption, they can be successfully used as wearable devices in real-life conditions out of the laboratory settings [7]. In the context of the analysis of human walking, MIMUs allow for measurement of both the kinematic variables and spatio-temporal

parameters of gait - even if generally their accuracy does not match the one of stereophotogrammetry.
In literature, several MIMU-based wearable solutions have been proposed during the years [8][9]. The most reported configurations include one or more MIMUs positioned at the level of the feet, shanks or pelvis; however, other body sites are recently starting to be explored, such as the head [10]. Head MIMUs (H-MIMUs) have peculiar characteristics in terms of morphology of the recorded signals with respect to other sensor configurations, since accelerations and angular velocities at the head during gait are generally more attenuated than other - lower - body sites [11]. Nevertheless, the deployment of H-MIMUs for industrial, health and user applications is expected to grow in the next years, as they are particularly suitable to be integrated with a plethora of devices such as smart glasses, earbuds or AR/VR systems [12].
Unfortunately, signals recorded by MIMUs - as well as the clinical indicators that are derived by them - typically show large variability according to age, gender, health conditions and individual traits [7]. In addition, methods that are developed to deal with gait data acquired in laboratory settings often fail to achieve sufficient accuracy when validated outside, as gait data recorded in laboratory settings typically do not reflect one's way of walking in free-living conditions [13]. In the last decades, with the advent of wearable sensors and the technological advancements in the fields of data transmission and storage, an increase in the number of publications reporting the use of machine learning (MaLe) and deep learning (DeLe) for the purposes of gait analysis has occurred [14]. MaLe and DeLe models are completely data-driven; therefore, they are free from most of the assumptions on the signal's nature that prevent "conventional methods"[1] from reaching very high performance. MaLe and DeLe algorithms are able to grasp the complexity of large heterogeneous amounts of data and use the information contained within it to learn how to infer one or more responses from multiple predictors [14]. Theoretically, by feeding a model with a sufficiently large dataset, the full variability of the target data distribution can be accounted; therefore, the heterogeneity of the data distribution may be addressed - even across different pathological and/or healthy populations. MaLe and DeLe architectures and training algorithms should be designed properly for allowing the trained models to recognize gait patterns and predict the values of the desired output, such as spatio-temporal parameters. In addition, validation during training is crucial to prevent overfitting.
Given such complexity and such large number of degrees of freedom in the design of the algorithms, the validity of clinically suitable methods based on MIMUs for

[1]Such expression refers to methods based on the analysis of the morphology of the signals in the time or frequency domains or on bio-mechanical models of the human gait [14].

estimating spatio-temporal parameters still represents an open question.

1.2 Challenges and objectives

The goal of the present thesis concerns the training and validation of MaLe and DeLe methods for the estimation of spatio-temporal parameters (STP) in both laboratory and real-world settings. The algorithms are meant to predict the values of five different STPs from data recorded by a single H-MIMU (Figure 1.1).

In the following, the most challenging aspects of the thesis work are highlighted:

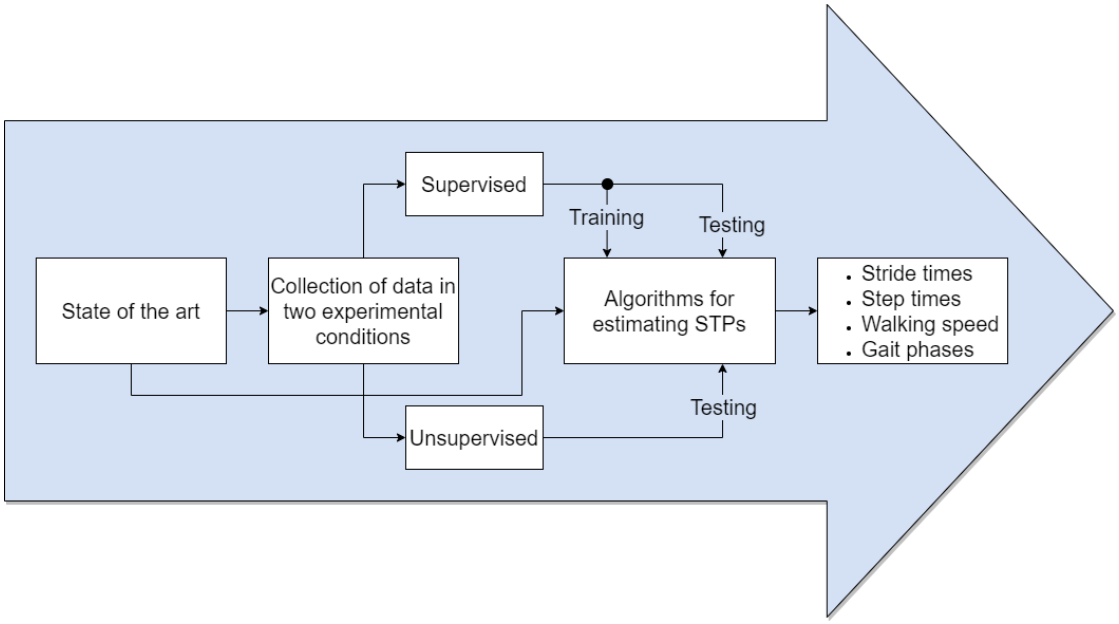


Figure 1.1: General pipeline of the thesis work. The first part of the work consisted in the literature review of MIMU-based methods for the estimation of STP. Then, two experimental protocols have been outlined and used to perform a set of acquisitions in supervised and unsupervised walking conditions. Eventually, the algorithms for the estimation of four spatio-temporal parameters have been developed and tested.

- **H-MIMU signals**: Signals at the head show strong attenuation and artifacts related to the head's nods and gestures that occur during gait [11]. In addition, as visually inspected, the gait informative content of signals at the head is significantly lower with respect to lower body sites such as shanks or feet. Such difficulty is mainly due to the stabilization action performed by the neck muscles on the upper trunk districts.

- **Heterogeneity of data**: Recorded data are acquired in different experimental conditions (laboratory, free-living) and refer to both straight and round walking at different speeds and on different grounds.
- **Data processing**: MaLe and DeLe algorithms need to be fed with data that have specific input and output formats. In addition, raw data need to be pre-processed and post-processed to clean outliers, reduce noise, detect and correct artifacts and validate the results.
- **MaLe & DeLe**: Besides training the learnable weights of the MaLe and DeLe models, the architecture and training parameters should be optimized. For that, stages of fine hyperparameters tuning and cross-validation are generally required.

At the time of the present study, the body of literature regarding the estimation of STP from H-MIMU data is relatively small. Then, the choice of investigating the performance of innovative methods that exploit data completely acquired by a single H-MIMU is driven by the wish of exploring an aspect of gait analysis that is still uncharted. Actually, thanks to its proximity to the brain and to the sight and otolith organs, the head could be employed as an important body hub for the integration of different devices and technologies such as brain-computer interfaces (BCIs), AR/VR systems and many others. In addition, recent findings suggest that upper body acceleration during gait can help to discriminate between gait patterns of Parkinson and non-Parkinson subjects and between Parkinson fallers and non-fallers [11]. For all these reasons, the use of the head as a body site for positioning multi-modal sensors - MIMUs included - is believed to become more widespread in the future.
The heterogeneity of the training data represents at the same time a strength and a challenge for the task of estimating spatio-temporal parameters: on one side, a heterogeneous dataset is more representative of the effective data distribution; however, it can yield the algorithm to underfit the training data, especially if the algorithm is based on the morphological features of the signal in the time or frequency domain. Instead, heterogeneity and variability of the signals can be more easily addressed by MaLe and DeLe techniques, as they are able to learn complex non-linear relationships between predictor signals and use them on new predictors to infer the value of their response. The use of MaLe for gait analysis has been already reported by some authors; however, its deployment is still limited due to the difficulty of evaluating the performance of the developed models. The challenge for the present thesis consists in developing robust methods based on MaLe and DeLe techniques that are able to predict STP in unsupervised walking conditions. Models are trained with different dataset combinations - in order to prove their robustness - and tested on both the training laboratory data and the unseen test data acquired in unsupervised conditions, in order to assess the extent of their generalization capability.

1.3 Thesis outline

The thesis is organized as follows:

Chapter 1 (current chapter) gives an introduction of the topic of the thesis and explains the thesis rationale and general objectives. In the end, the general outline of the thesis is described.

Chapter 2 describes the features of human gait of healthy subjects and provides the definition of the basic terminology related to the gait cycle. A general overview of gait analysis and the most widespread techniques is provided, together with a detailed description of each of the most common STPs. Eventually, the state of the art of artificial intelligence (AI) applied to gait analysis is presented.

Chapter 3 provides a short summary about the general functioning of inertial sensors. Then, it focuses on the description of the experimental setup employed for acquiring data.

Chapter 4 outlines the experimental protocols observed for the acquisitions.

Chapter 5 describes the construction of the datasets, the data pre-processing steps and the architectures of the developed MaLe and DeLe models. Then, training, validation and testing of the models is described in detail.

Chapter 6 displays the results obtained with the implemented models.

Chapter 7 critically discusses the results presented in Chapter 7.

Chapter 8 sums up the main achievements of the thesis and provides an outlook for future research and perspectives.

Appendices A, B, C, D provide detailed information about the experimental setup and protocol, the Matlab codes used for defining the architectures of the models and an extensive background to the MaLe and DeLe techniques employed in the present work.

Chapter 2

Gait

The present Chapter provides the reader with the basic notions about gait and gait analysis, in order to set a terminology and introduce the concepts that will be discussed in the next Chapters.

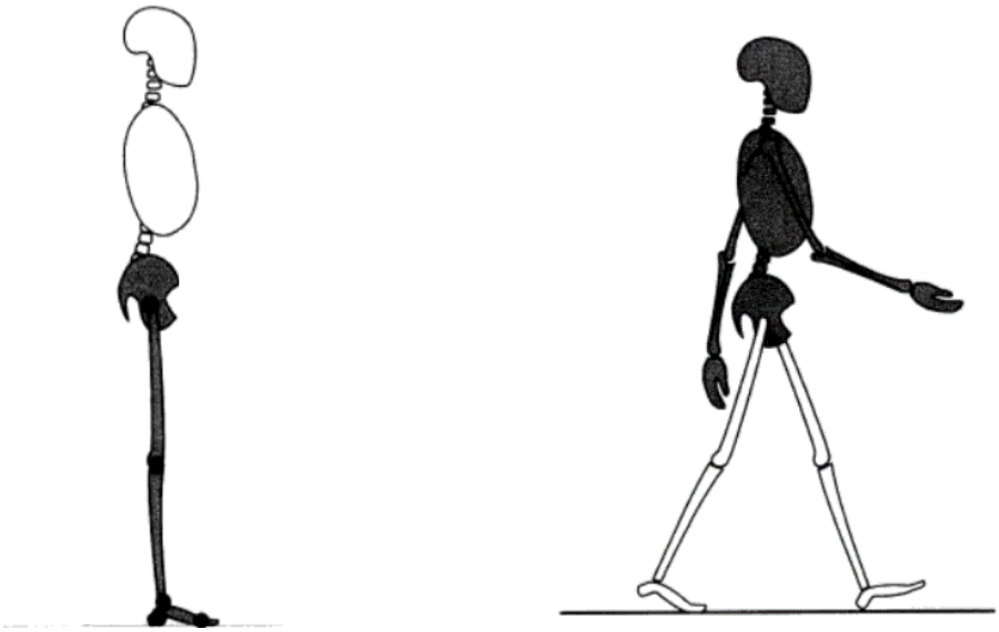
2.1 The gait cycle

Walking is the main form of locomotion for healthy adult human beings [13]. In general, every subject's way and speed of walking are calibrated to the one's need of minimizing the energetic cost of walking [15]; however, gait of healthy subjects shows recurrent patterns that can help to analyze it.

2.1.1 Basic notions

According to Perry [16], during gait, the human body can be resolved into two functionally separated units (Figure 2.1).

- The **locomotive unit** is constituted by the pelvis and the lower limbs. Such unit includes 11 joints: the timing and the amplitudes of their motion is controlled by a total amount of 57 muscles. Bone segments of the lower limbs (pelvis, thigh, shank, foot and fingers) support alternatively the body and drive its forward progression by acting as levers. After the unloading of one limb, this rapidly swings forward to provide again support to the body weight. The locomotive unit fulfils four main functions: a) propulsion; b) vertical stability; c) absorption of impact at the onset of each step and d) conservation of energy to reduce the energy cost for muscles. Each of these functions involves a complex series of interactions between the body mass and the segments of the lower limbs, that integrate to form a single tri-dimensional motion framework.



(a) The locomotive unit. **(b)** The passenger unit.

Figure 2.1: The two functional units of gait according to Perry [16].

- The **passenger unit** - sometimes also referred to as HAT[1] - consists in arms, head and trunk, and it represents around 70% of the body weight. The passenger unit is in charge of the postural integrity: actually, mechanics of the healthy gait is so efficient that the functional tasks of the passenger unit are almost negligible. Still, the alignment of the passenger unit above to the locomotive unit represents the main determinant of the muscle activity of the lower limbs. During healthy gait, muscles of the trunk and the neck maintain the postural alignment neutral, while the swing of the upper limbs - which involves both passive and active mobilization - seems to affect poorly the mechanics of gait.

The locomotive unit is in charge of supporting the above passenger unit. However, the passenger unit has a much more considerable mass with respect to the locomotive unit; then, the mass ratio between such units is one of the main determinants of postural stability during gait [16].
Recent studies have demonstrated that stabilization of the head and the upper trunk is fundamental for maintaining postural control during gait [11]. As a consequence, accelerations at the head result in a more attenuated and smoothed waveform

[1]Head, Arms, Trunk [17].

than signals recorded at the lower limbs. The reason for such stabilization may be addressed to the need for keeping the vision and vestibular systems stable, as they are crucial for navigation and planning of adaptive motor strategies [11].

2.1.2 Taxonomy of gait

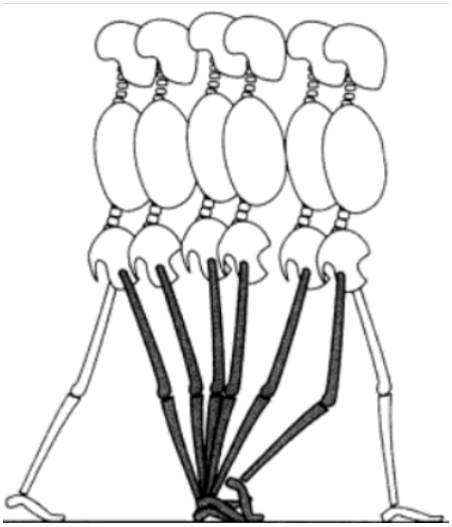
Given its complex and multi-interactional nature, gait can be studied and analyzed according to three possible approaches [16]:

- the first one divides the gait cycle according to the reciprocal feet-to-floor contacts;
- the second one considers the time and distance features of the stride;
- the third one determines the events within the gait cycle and names the intervals delimited by them as the functional phases of gait.

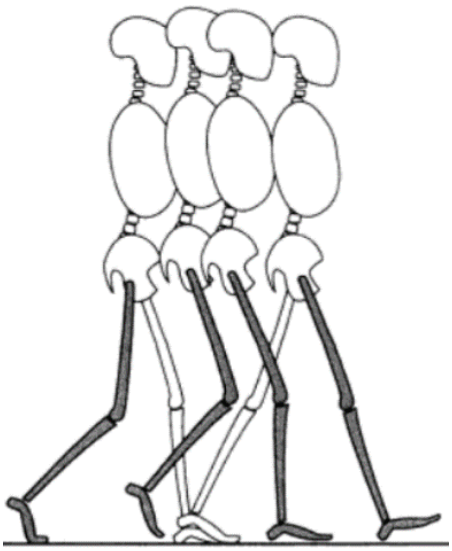
In the first approach, one limb supports the body while the other advances to the next support stage; then, the limbs switch role and the body weight is transferred from one limb to the other while both feet are in touch with the ground. Such series of events is repeated by every limb alternatively until the destination is reached. A single sequence of such functions for one limb is commonly referred to as *gait cycle*. As a matter of fact, due to its cyclic nature, every event of gait could be considered as the start or the end of the cycle; however, since the time step at which the foot-to-ground contact occurs is the easiest to detect, it is usually accepted as the beginning of the gait cycle. Such event is commonly referred to as heel strike (HS) or initial contact (IC)[2]. The gait cycle can be segmented into two *periods* or *phases* - *stance* and *swing* [19] - which define the functional partition of the activity of one lower limb during gait. The stance starts with the IC and is defined as the period of foot-to-ground contact. The swing starts with the toe off (TO) and refers to the period during the one the limb is detached from the ground to allow progression (Figure 2.2).

Within the stance, three intervals can be recognized (Figure 2.3): two stages of bi-lateral contact of the feet with the ground at the onset and at the end of the cycle (*double support*) and one stage of mono-lateral contact of only one foot with the ground in the central part of the stance (*single support*). It is important to stress that, during the double support stages, the body weight is not equally distributed among the limbs.

[2]In the gait of healthy subjects, the foot initial contact and final contact can respectively be referred to as *heel strike* and *toe off*; however, such terminology fails when dealing with gait of elderly people or pathological subjects (e.g. subjects with clubfoot or Huntington disease), whose contact and detachment points may differ from the heel or the toes [18].



(a) During stance, the limb provides a support basis that moves forward over the foot.



(b) During swing, the limb advances towards a useful configuration for the next load acceptance.

(c) Feet orientation during stance (black) and swing (white).

Figure 2.2: The two main phases of gait according to Perry [16].

The timing of the alternation of stance and swing, single and double support can vary according to age, pathology or walking speed [20][21]; however, adult healthy subjects tend to have stance and swing distributed respectively as 60 and 40 % of the gait cycle (% GC). A single support stage takes around 10 % GC, while a double support stage takes around 40 % GC. The time duration of the double support stage is inversely proportional to the walking speed: eventually, when the speed is sufficient, the double support stage does not occur and gait evolves to running.
The gait cycle - also referred to as *stride* - can be considered as the sequence of two *steps* (Figure 2.4). A stride begins with the initial contact of one foot and ends with the next initial contact of the same foot. Hence, if the stride starts with a left step, it will be ended by a right step [22].

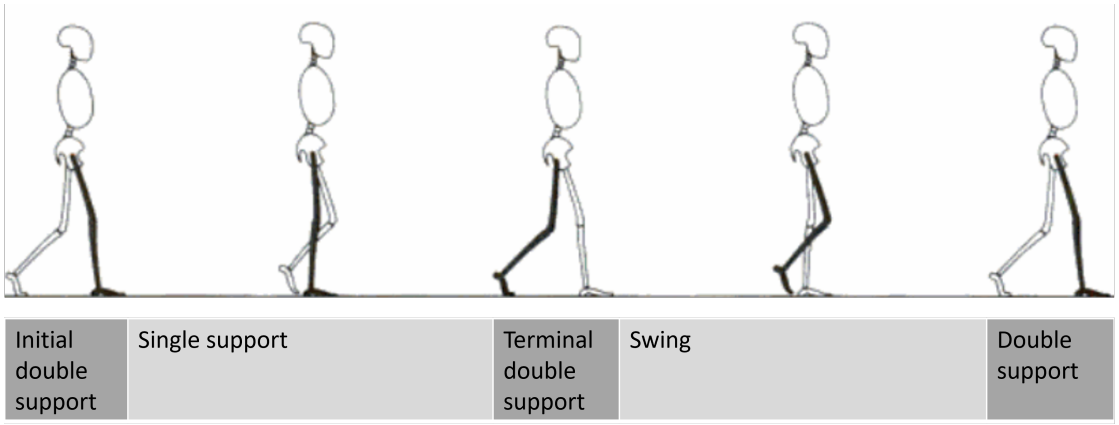


Figure 2.3: Overview of the single support and double support stages [16]. The dark bands refer to double support, while the light bands refer to single support. The stance period includes an initial double support, a single support and a terminal double support before the swing. The last dark band identify the onset of the next cycle.

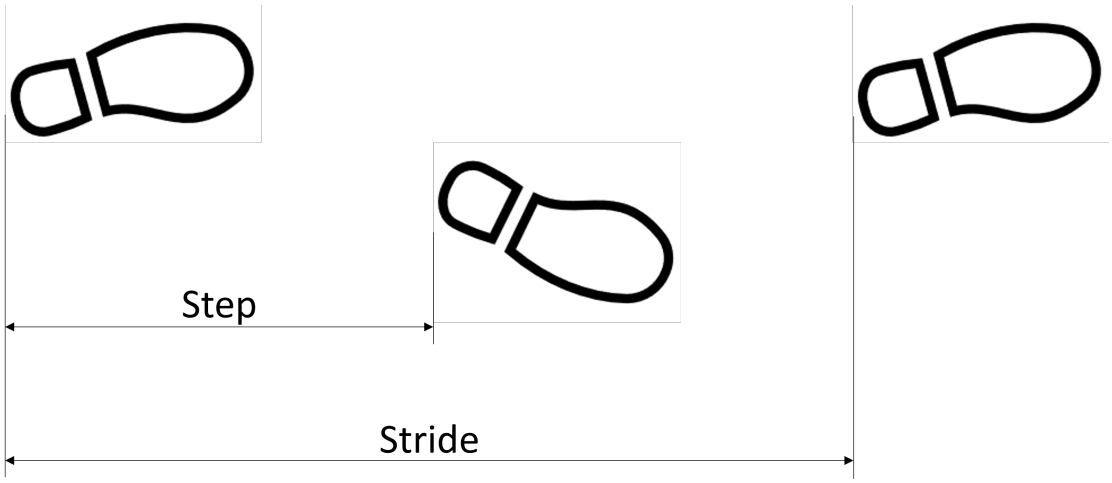


Figure 2.4: Relationship between step and stride.

2.1.3 Phases and sub-phases of gait

According to Perry [16], the gait cycle can be hierarchically divided into *periods*, *tasks* and *sub-phases* (Figure 2.5). Each sub-phase represents a precise motor schedule oriented to the accomplishment of a specific functional need. Eventually, the synergy of the 8 sub-phases allows the achievement of three main functional tasks - weight acceptance, single support and limb progression.
A stance phase consists of five sub-phases:

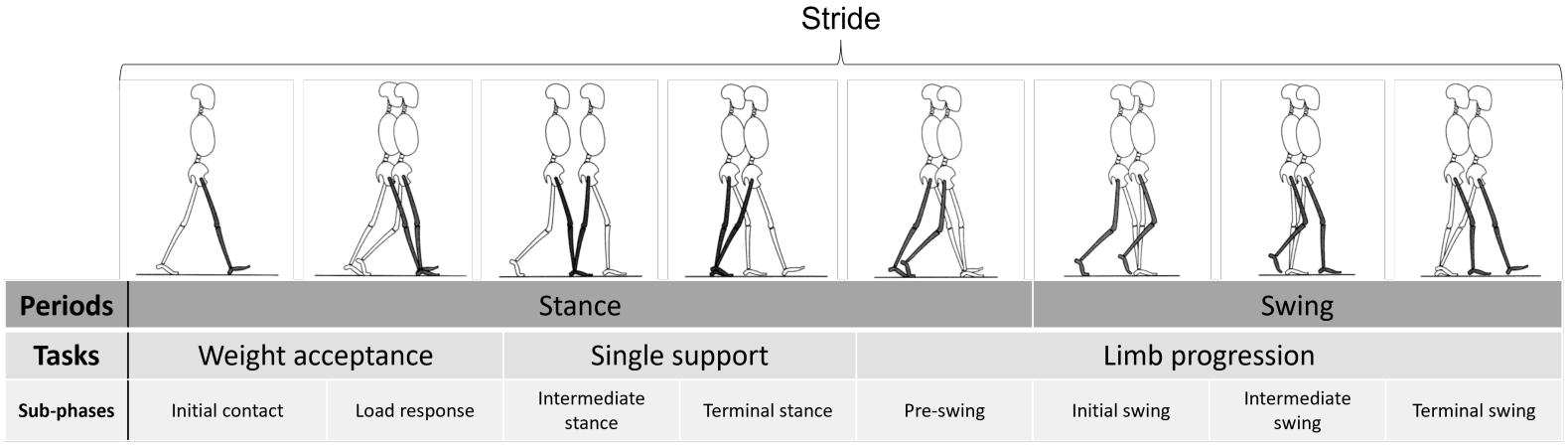


Figure 2.5: Functional sub-phases of gait according to Perry [16].

- **Initial contact**: 0 - 2 % GC.
  The initial contact is the first stage of the weight acceptance. During this phase, the ground reaction force shifts behind the ankle joint and forces a clockwise rotation of the foot.

- **Loading response**: 2 - 10 % GC.
  During the loading response, the knee performs a flexion to initiate the loading acceptance. During this phase, the first rocker occurs. The heel hits the ground, the foot rotates around it, and the ankle joint axis rotates towards the flat foot position. Contraction of plantar-flexion muscles coordinates this motion [23].

- **Mid stance**: 10 - 30 % GC.
  During the mid-stance, the second rocker occurs. The tibia rotates around the ankle joint and goes up and over the talus. The intrinsic muscles of the foot and tibialis posterior activate to keep a medial longitudinal trajectory [23].

- **Terminal stance**: 30 - 50 % GC.
  During the terminal stance, the third and last rocker occurs, while the performs a plantar-flexion over a fixed forefoot (at the metatarsophalangeal joints) [23].

- **Pre-swing**: 50 - 60 % GC.
  During the pre-swing phase, the body load is transferred from one foot to the other. In this interval, both feet touch the ground, hence, the pre-swing phase is also referred to as *double support* interval - in contrast to the remaining *single support* interval.

The swing phase consists in three sub-phases:

- **Initial swing**: 60 - 73 % GC.
  The initial swing initiates the limb forward progression.

- **Mid-swing**: 73 - 87 % GC.
  During the mid-swing, the knee reaches its maximum flexion, which is functional to raise the foot and avoid stumbling.

- **Terminal swing**: 87 - 100 % GC.
  Pelvis flexion continues due to the leg's inertia with poor muscular contribution.

The analysis of the sub-phases of gait allows to determine the functional meaning of the various motions at the joints; moreover, it provides also a tool for correlating the activities of the single joints to the functional schemes of each limb [16].

2.2 Gait analysis

2.2.1 Milestones

Walking has always been crucial for the lives of the human beings. Through the centuries, the cyclic gesture called *gait* has captivated the minds of scholars, scientists and philosophers, who dedicated much effort in the attempt of grasping the functioning of its underlying mechanism [24].
The first comment on the analysis of walking can be tracked to Aristotle (384–322 BCE) [25]; however, the world had to wait until the Renaissance for the spreading of a real interest towards gait analysis. During this period, the Italian mathematician Girolamo Cardan (1501 - 1576) - the first one to use complex numbers - studied the properties of three-dimensional angles, while the French philosopher and scientist Rene Descartes (1596 - 1650) introduced an orthogonal co-ordinate system for describing the position of objects in space [26]. The first experiment in gait analysis was performed by one of the pupils of Galileo Galilei, Giovanni Borelli (1608–1679). He placed two poles at unspecified distance one from the other and tried to walk towards them maintaining one pole in front of the other. He noticed that near pole always seemed to move to the left and right with respect to the far pole, hence, he found that the head moves in the medial-lateral (ML) direction during walking [27]. Later in the 19$^{\text{th}}$ century, the Weber brothers[3] - using only a stop watch, measuring tape and a telescope - observed that cadence and step length depend reliably on the walking speed [28]. Moreover, they also tried to figure out the pose of the limbs at 14 different instants in the gait cycle and were the first to draw illustrations showing frames of the limb segments orientation at these different instants (Figure 2.6).

After that, relevant steps forward in gait analysis were promoted by Marey (1830–1904) [29], Muybridge (1830–1904) [30], Fischer (1861–1917) [31], Bernstein (1896–1966), Amar (1879–1935) [32] and Inman [33][34]; who contributed both with experiments and theorizing and laid the foundations for modern gait analysis.
After the Second World War, electromyogram (EMG) represented the election technology for performing clinical gait analysis, due to its ease of recording and low dimensionality with respect to 3-D motion data. Nevertheless, in the 1970s, the two American surgeons Jacquelin Perry and David Sutherland - both pupils of Inman and two of the most eminent names in the field of gait analysis- recognized that EMG

[3]Ernst Heinrich Weber (1795–1878) and Eduard Friedrich Willhelm Weber (1806–1871), brothers of Willhelm Eduard Weber (1804–1891), Professor of Physics at Gottingen and still remembered in the eponymous SI unit of magnetic flux.

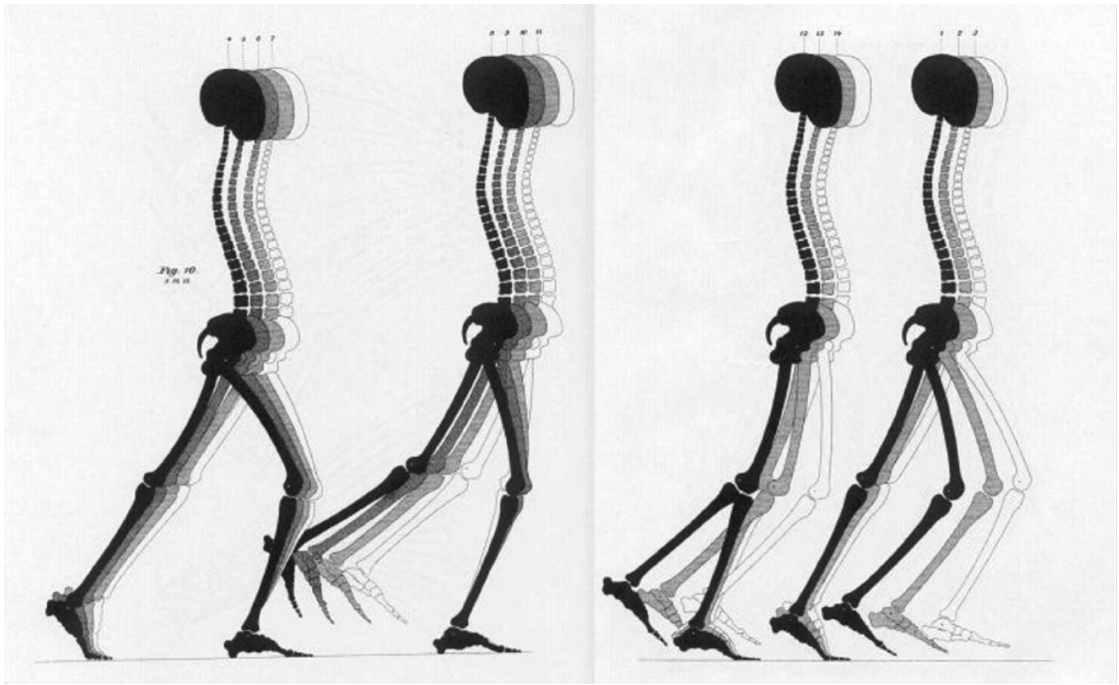


Figure 2.6: Configuration of the trunk and lower limbs at 14 instants during the gait cycle [28].

was not enough. Perry developed instrumented methods for measuring temporal-spatial parameters [35][36], while Sutherland investigated ways of obtaining three-dimensional information from cine film [37]. More recently, the engineer Ed Chao paved the way for the modern use of three-dimensional joint angles [38]. Up to the 1970s, instrumented gait analysis represented a mere research tool and was limited to few subjects. Experimental setups were lumbering and complex to use, and the amount of time required for processing prevented data from being presented in a clinically-appealing format. Gait analysis really became a commons with the advent of the computer era, which allowed for faster processing of data [24].

2.2.2 General overview

The World Health Organization (WHO) highlighted the importance of the aspects of the motor function such as activity level and participation in the 2001 International Classification of Functioning Disability and Health [39]. Gait function is considered a major determinant for independent motor function; then, the extent of walking during the day is an indicator for the degree of physical activity [13]. Gait and balance can be affected by a wide range of diseases, such as heart failure, chronic obstructive pulmonary disease (COPD), stroke, Parkinson, cerebral palsy (CP)[8],

osteoarthritis, osteoporosis, diabetes, low vision, etc... so that the treatment of gait-related disturbances involves several clinical disciplines [40]. Moreover, the gait disorders have proven to be related to mild cognitive impairment and Alzheimer's syndrome [41]. Then, the evaluation of gait function is crucial for monitoring the recovery or decline of the motor function and to study physical activity patterns on the long run [13].
Gait analysis is the instrumented measurement of variables related to the human walking[4]. The goal of gait analysis is the quantification of factors that control the functionality of lower limbs. This is functional to detect gait anomalies, identify postural instabilities and assess clinical interventions and rehabilitation plans [14]. Modern gait analysis can assist the clinical evaluation of the gait of both healthy and pathological individuals, providing the clinician with quantitative values of relevant gait-related clinical variables.
In the last years, gait analysis has emerged as an effective tool to the early detection of neurodegenerative diseases or to monitor their progression [2]. Early diagnosis of motor and neurodegenerative diseases can anticipate loss of mobility [43] and limit healthcare-related costs, which represent a significant issue for developing countries [44].
Out of the pathological framework, gait analysis can be employed also for wellness, sport training optimization, gaming and entertainment.
Today, gait analysis mainly regards 4 areas of science: kinematics, kinetics, EMG and engineering mathematics [45].

- **Kinematics**: Kinematics is the measurement of movement [45] i.e. the limbs position, orientation and relative angles. Today, the instrumentation to measure kinematics includes timing systems, velocity-measuring systems, accelerometers, micro electro-mechanical systems (MEMS) inertial sensors, optical and IR imaging systems and MOtion CAPture (MOCAP) systems [46].

- **Kinetics**: Kinetics aims at measuring the forces acting between the foot and the ground, which are recorded by an instrumented surface known as a *force platform* [45]. By measuring forces, other dynamic quantities can be derived from the integration of force data with kinematic data, such as power, energy and torques [46][45]. Modern tools for directly measuring kinetic bio-mechanical variables include force platforms, strain gauges and pressure-sensing devices [46].

[4]Today, the expression "gait assessment" is used similarly to "gait analysis"; however, the former refers to the whole process of examining a patient's gait and making suggestions for treatment, while the latter should be reserved for the technical side of the gait assessment [42].

- **EMG**: Muscle forces produced during walking may be indirectly estimated using EMG. EMG records the electrical activity of a contracting muscle through surface electrodes placed on the skin over a superficial muscle or micro-electrodes that penetrate the epymisium and record electrical activity of more profound muscles. EMG data denote whether a muscle is contracting or not; moreover, by processing the EMG signal, the relative strength of the contraction may be estimated [46].
- **Engineering mathematics**: The transmission of forces and moments at different joints and the conservation of energy during walking have been object of study and mathematical modeling since the 1930s [46]. Nowadays, such branch of gait analysis concerns the use of *inverse dynamics* to compute joint moments and powers from the limb motion and/or ground reaction force.

2.2.3 Spatio-temporal parameters

Typically, a distinction is made between local and global gait-related variables.

- **Global variables**:
  - Spatio-temporal parameters
  - Average and maximum speeds of the center of mass;
  - Consumed metabolic energy;
  - Walking endurance time interval;
  - Average rate of fall[5].
- **Local variables**
  - **Of muscles**: e.g. s-EMG.
  - **Of joints and muscle groups**: e.g. Joint angles, joint torques and generated/absorbed power.

Global variables are useful for planning interventions or assessing drugs efficacy, while local variables are typically employed to create complex models of a specific joint for orthopedic purposes or for designing prosthetic implants.
In the framework of global gait-related variables, STP play a crucial role in the quantitative evaluation of the motor function both in research and clinical settings [9]. STP describe in a quantitative way the main events of gait, and therefore reflect the ability of the subject to achieve the general requirements of gait i.e. the

[5]The rate to which a person tends to fall is especially relevant for elderly people.

weight acceptance, the single limb support and the forward progression of the limb during the swing phase [47].
The most widely used temporal gait parameters include stride and step time and cadence; in addition, spatial gait parameters can be defined from the distance traveled between two consecutive gait events [2]. STP can be estimated indirectly from spatial and temporal parameters according to various definitions (Table 2.1). The above-mentioned STP can be calculated according to several definitions, depending on the specific need and application. For instance, the walking speed (WS) may be defined either as the ratio between the stride length and the stride time or as the ratio between the traveled distance and the amount of time employed to walk that distance. STP describe complementary aspects of gait; then - if combined - they can provide a solid overview of the physio-pathological condition of one's gait. Eventually, STP can be employed to diagnose pathological gait and assess functional outcome after treatments [47]. Distinctive relevance is covered by the WS, as it correlates with functional ability, balance confidence and has the potential to predict future health status and functional decline [49].

2.2.4 Instrumented measurements of spatio-temporal parameters

Modern analysis of gait strongly relies on instrumented tools to extract objective and reliable measures of locomotion patterns and their variability [2]. Such measures can contribute to the investigation of gait illnesses and support the design of targeted rehabilitation programs [50][51]. Regarding STP, they can be estimated instrumentally through a wide plethora of devices [9]:

- **Force platforms**
- **Instrumented mats**
- **Foot-switches**
- **Pressure insoles**
- **Opto-electronic systems**
- **MIMUs**

Force platforms

Force platforms - otherwise known as *force plates* - measure reaction forces in three directions - vertical, anterior-posterior and medial-lateral - and determine the direction and point of application of the resultant reaction force [48]. Force

Domain	Parameter	Unit	Description
Time	Stride time	$[s]$	Time interval between two consecutive ICs of the same foot.
	Step time	$[s]$	Time interval between a IC of one foot and the IC of the other.
	Cadence	$[steps/min]$ or $[strides/min]$	Number of steps or strides per minute.
	Stride frequency	$[Hz]$ or $[stride/s]$	Number of strides per second.
	Step frequency	$[Hz]$ or $[steps/s]$	Number of steps per second.
	Stance time	$[s]$	Time interval between IC and the next final contact (FC) of the same foot.
	Stance duration	% GC	The stance time as percentage of the gait cycle.
	Load time	$[s]$	Time interval between IC and toe strike (ToS) of the same foot.
	Load duration	% GC	Load time as percentage of the gait cycle.
	Foot flat time	$[s]$	Time interval between ToS and FC of the same foot.
	Foot flat duration	% GC	Foot flat time as percentage of the gait cycle.
	Push time	$[s]$	Time interval between heel off (HO) and FC of the same foot.
	Push duration	% GC	Push time as percentage of the gait cycle.
	Swing time	$[s]$	Time interval between FC and IC of the same foot.
	Swing duration	% GC	Swing time as percentage of the gait cycle.
	Double support time	$[s]$	Time interval between an IC of one foot and the next FC of the other.
	Double support duration	% GC	Double support time as percentage of the gait cycle.
	Single support time	$[s]$	Time interval of support on one foot (corresponding to the swing time of the other).
	Single support duration	% GC	Single support time as percentage of the gait cycle.
	Limp index	–	The ratio of the stance times of the two feet.
	Turn time	$[s]$	Time interval of a turn during walking.
Space	Stride length	$[m]$	The distance of two consecutive ICs of the same foot.
	Normalized stride length	$[m/m]$	Ratio between stride length and height.
	Step length	$[m]$	The distance of the IC of one foot and the IC of the other.
	Normalized step length	$[m/m]$	Ratio between step length and height.
	Step width	$[m]$	The distance of feet along the ML direction.
Velocity	Walking speed	$[m/s]$	The average instantaneous speed of the gait cycle.
	Normalized walking speed	$[Hz]$	Ratio between walking speed and height.
	Arm swing velocity	$[m/s]$	Speed of the arm during forward swing.

Table 2.1: Starting from gait events, several useful STP can be defined [48] in the time, space and velocity domains. STP values expressed as % GC represent the value for that parameter expressed as percentage of the time duration of the gait cycle.

platforms, which are typically rectangular-shaped and sized between 40 cm and 60 cm (Figure 2.7), can also provide accurate gait temporal parameters such as foot ICs and FCs [7].

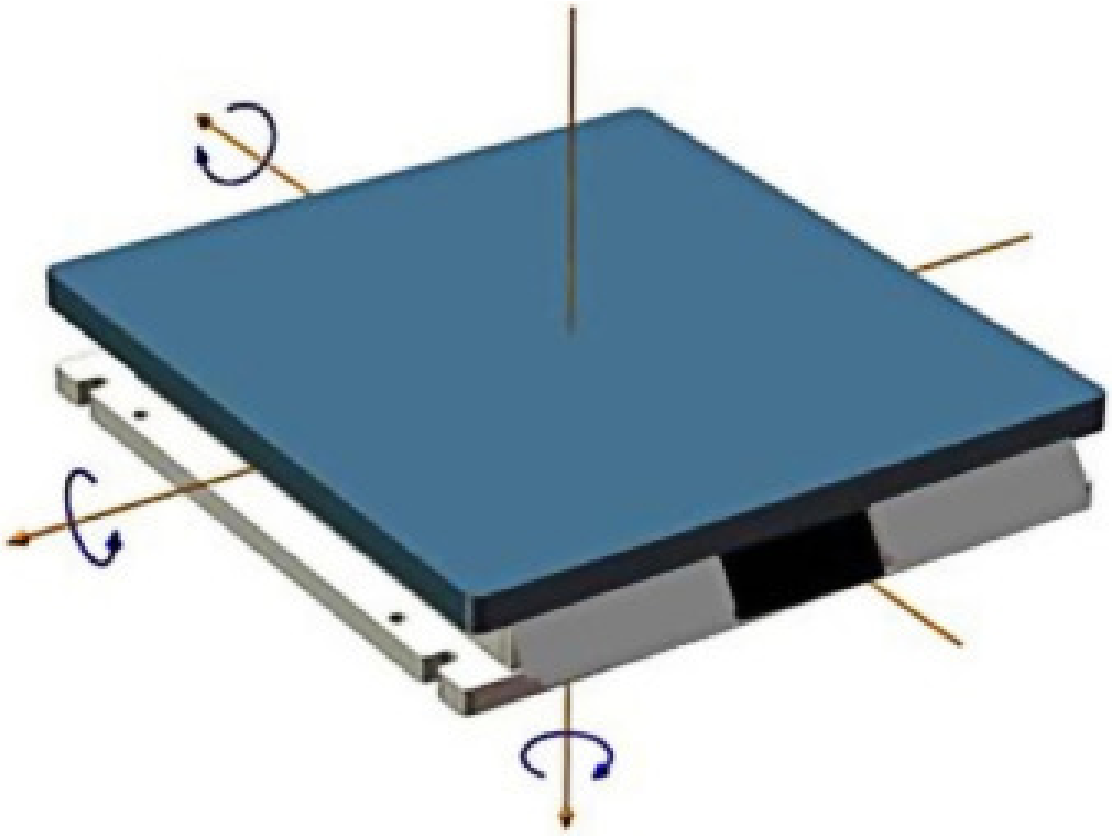


Figure 2.7: Illustration of a force plate [52]. The lines denote the directions of measurable forces and torques.

Instrumented mats

Instrumented mats - also called *instrumented walkways* - are platforms that embed a number of pressure sensors (Figure 2.8). Then, instead of measuring resultant forces, instrumented mats allow the measurement of the force distributed on a certain area i.e. pressure [48]. Force platforms and instrumented mats share the same strengths i.e. the ease of use and the possibility of measuring spatial parameters as well as the temporal ones. However, they both are expensive, require much space and can hardly be employed outside of laboratory settings [7].

Figure 2.8: The GAITRITE system, one of the most widely used instrumented walkways [53].

Foot-switches

Foot switches are basically force sensing resistors placed under the feet (Figure 2.9). They are wearable and cheap; however, they take longer subject preparation and provide temporal parameters only [48].

Pressure insoles

Pressure insoles extend the concept of foot switches by using a higher number of pressure sensors (typically more than 10). By increasing the number of sensors, pressure insoles allow for a more accurate depiction of the foot-to-ground contact. Sensors are typically embedded in a flexible plastic substrate to adhere to the foot shape (Figure 2.10). Pressure insoles provide more detailed information on the foot-to-ground contact with respect to foot switches, as they can keep track of the pressures exerted by the foot on the ground during stance instead of just determining the HS and TO.

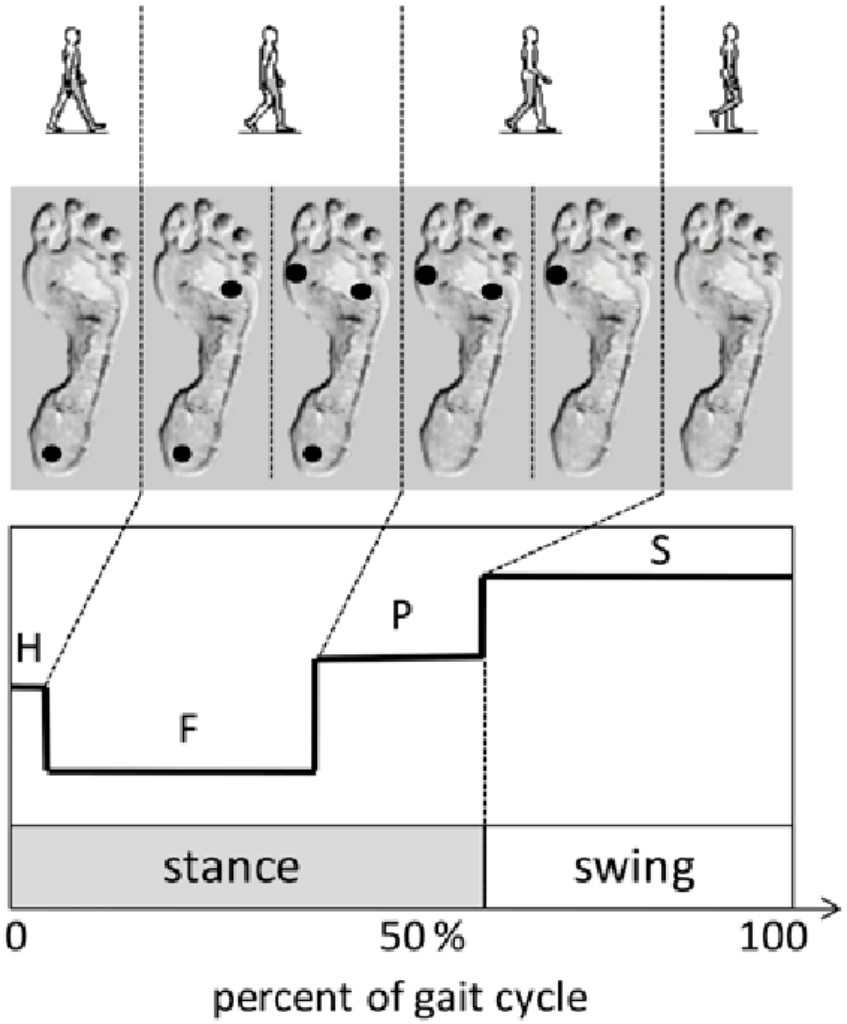


Figure 2.9: Signals originated by a combination of three foot-switches [54].

Opto-electronic systems

Opto-electronic systems - also referred to as optical MOCAP systems or stereophotogrammetry - enable the digital tracking of the human motion in tri-dimensional space (Figure 2.11). The subject's motion is recorded by a set of visible or IR light cameras; then, the output - in the shape of 2D digital images - is processed to obtain 3D trajectories of markers placed on the subject's body [48]. Optical motion tracking systems are meant to measure 3D kinematics; however, they can provide also STP. Due to their accuracy and reliability, stereophotogrammetry is considered as the GS for gait analysis; nevertheless, optical motion tracking

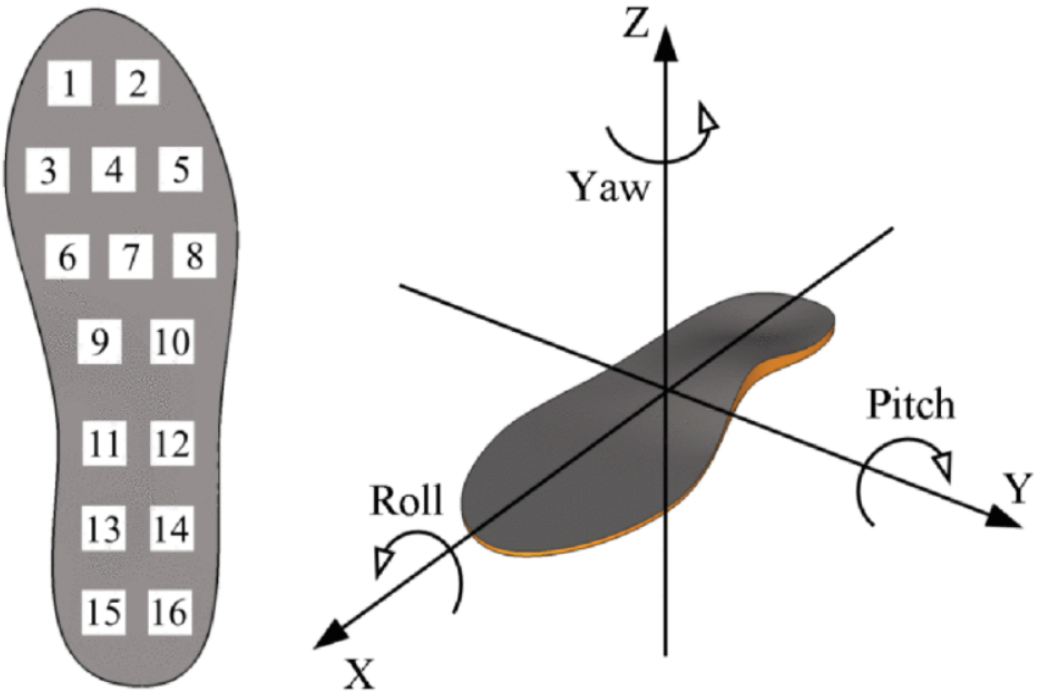


Figure 2.10: Illustration of a pressure sensing insole [55].

systems are significantly expensive, can be used only in laboratory settings and have a limited field of view [7]. As a consequence, results of studies relying on stereophotogrammetry or instrumented mats for the collection of data often do not reflect real-life gait [13].

MIMU

The above-mentioned technologies usually fail when gait analysis must be conducted in real-life conditions outside of the laboratory settings, since they are generally obtrusive and far from being easy-to-use [22]. Magneto–inertial sensing is an emerging technology with a growing number of potential applications in the analysis of the human motion [7]. MIMUs - sensing units hosting tri-axial gyroscope, accelerometer and magnetometer - are self-contained systems; then, their mode of operation does not depend on the experimental settings. Although the deployment of accelerometers for human motion analysis has been suggested since the 1970s, their usage for purposes related to gait analysis has become widespread only recently, with several studies reporting their use on trunk, thigh, shank or foot [22]. Inertial sensors are extensively employed in the consumer electronics market, hence,

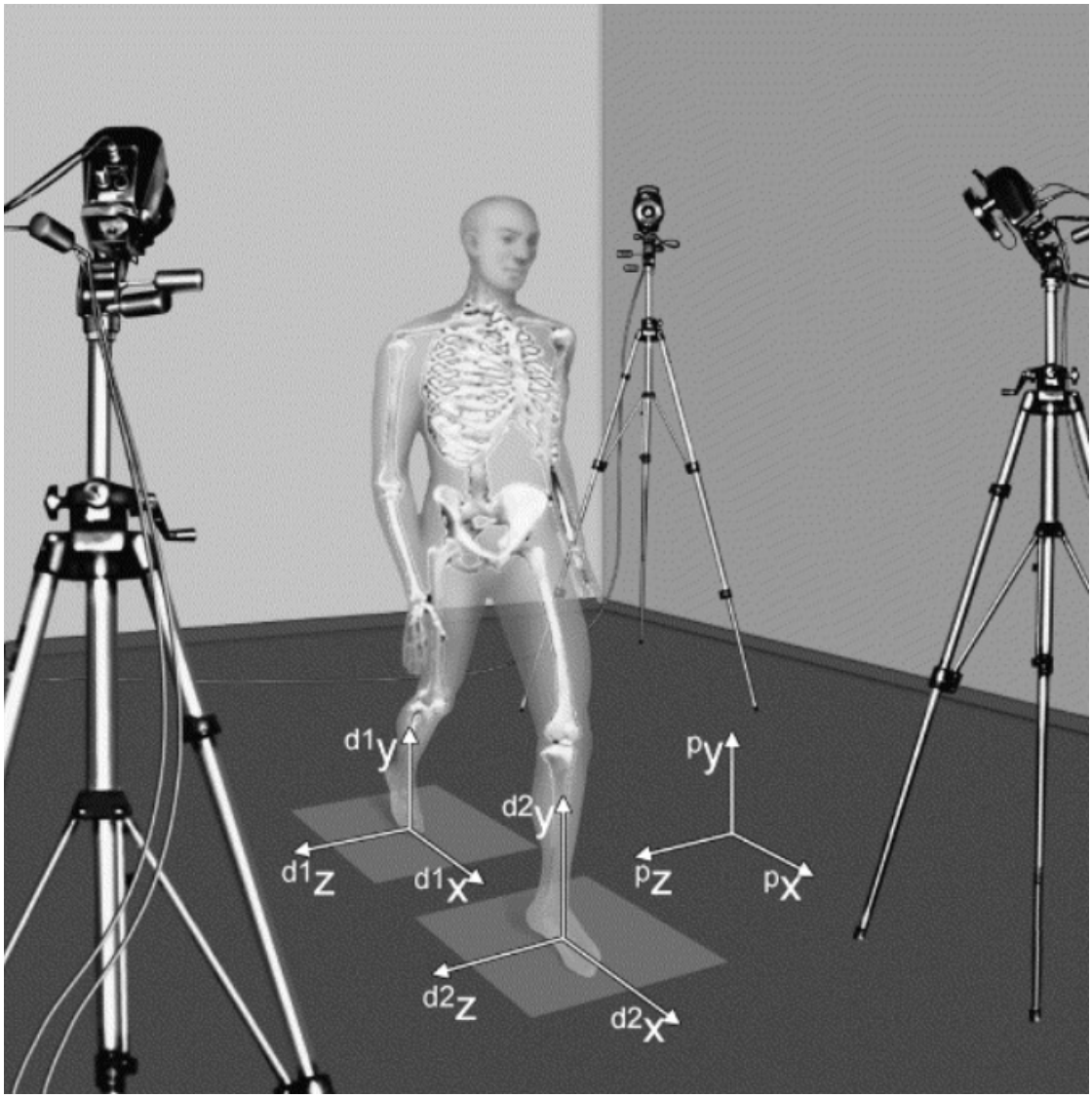


Figure 2.11: Illustration of an opto-electronic system [6].

their performance is constantly upgraded while their price decreases; moreover, the development of MEMS of the last years has paved the way for a wide range of applications based on MIMUs in the field of the analysis of the human motion [7]. An alternative to MIMUs is represented by inertial measurement units (IMUs) (Figure 2.12), which host only an accelerometer and a gyroscope [7].
Body worn accelerometer devices are widespread within the scope of step and stride counting applications, since they can be used to derive mobility-related metrics (e.g. total number of steps per day, duration and cadence of locomotion periods).

Nevertheless, the robust estimation of such parameters in real-life conditions is not trivial, given the influence of environment (e.g. surface type/slope/stairs, indoor vs outdoor etc.) and the variability in movement impairments [8].

Generally, inertial sensors are less accurate than laboratory-based technologies

Figure 2.12: A typical full-body configuration of IMUs including IMUs positioned on the knees, waist and chest (Notch Pioneer Kit, 2021, Notch Interfaces Inc. All rights reserved).

for the measurements of STP. However, their popularity is due to portability and unrestricted applicability to almost every environment [14].

2.2.5 Estimating spatio-temporal parameters from inertial sensors

Until 2003, not many studies had highlighted the relationship between STP and inertial signals i.e. accelerations and angular velocities [22].

A general trend in gait analysis is to approach sequentially to the challenges of the estimation of temporal and spatial parameters of gait, given the intrinsic difference of the domains that they refer to. Typically, gait events and temporal parameters

are first estimated; then, gait events are exploited to estimate displacements and eventually spatial parameters. At this point, temporal and spatial parameters are combined to estimate gait parameters depending on both time and space (e.g. the walking speed).
With today's availability of storing and processing large amount of data and the recent progress in the field of deep supervised computational models, the processing steps needed to go from signals to gait events to temporal and spatial parameters may be bypassed and unexplored relationships between inertial signals and STP may be directly inferred.
In general, methods for estimating the temporal parameters of gait from inertial signals can be divided into three main categories [56]:

- **Peak-detection methods**: Methods based on thresholds or peaks identification.
- **Time-frequency analysis**: Methods based on the analysis of the signals in the time-frequency domain or wavelet decomposition.
- **Machine learning**: Gait analysis involves a large number of interdependent parameters that are difficult to interpret due to a vast amount of data and their inter-relations. To simplify evaluation, the integration of MaLe with biomechanics is a promising solution.

Methods based on MaLe span over a wide plethora of algorithms such as hidden markov models (HMMs), support vector machines (SVMs); decision trees (DTs); artificial neural networks (ANNs), ensemble classifiers and others [14]. Most recently, thanks to the high computational power that modern computers have at their disposal, deep learning artificial neural networks have been successfully applied to the identification of gait events, which is typically the first step for the estimation of temporal parameters of gait [57].
Similarly, several methods for estimating spatial parameters from inertial signals have been reported in literature. In general, such methods can be divided into three main categories [56]:

- **Direct integration**: Direct integration methods - as their name says - are based on the direct integration of the linear acceleration signal along the direction of progress (DoP). Linear acceleration is the second derivative of linear displacement, hence, the acceleration signal must be integrated two times to get displacement of the sensor. Although direct integration may seem a trivial and easy procedure, the acceleration signal is affected by errors that are enlarged every time that the signal is integrated. Then, countermeasures based on signal assumptions are typically adopted to try and reduce such errors (e.g., zero velocity update, direct-reverse integration, etc...).

- **Human gait models and abstraction models**: Human gait can be modeled with proper mathematical expressions that reflect its underlying mechanism. A common model for the human gait is the inverse pendulum, though other more sophisticated models can be thought - for instance, the double inverse pendulum. Such models rely on the relationships that exist between the anthropometric features and gait parameters. Gait models can be personalized to fit to one's gait by introducing suitable parameters; however, the higher the amount of parameters of the model, the lower its generalizability.

- **Machine learning**: In the last years, MaLe has emerged as a valid solution to the problem of estimating spatial parameters. MaLe consists in a set of techniques that aim at extracting knowledge from data.

In the last decades, many studies have undertaken the non-trivial task of estimating STP with MIMUs. A widespread methodology consists in exploiting multi-sensor configurations i.e., multiple MIMUs positioned at different body sites. Panebianco et al. evaluated the influence of sensor positioning, target variable and computational approach on the performance of 17 algorithms for the estimation of spatio-temporal parameters with different sensor configurations including lower back MIMUs (B-MIMUs), shank-MIMUs and foot MIMUs (F-MIMUs) [9]. Bertoli et al. estimated STP of Parkinson, mildly cognitively impaired and healthy older adults with bilateral shank-MIMUs [2]. Salarian et al. developed an ambulatory method for the estimation of STP using gyroscopes attached to wrists, shanks and thighs [58]. Although multi-sensor configurations have proved to be able to provide accurate and reliable estimates of STP, wearable solutions based on multiple sensors often require much time for preparation and lack in usability and comfort [59]. As a consequence, many studies shifted towards the direction of single-sensor systems. Galperin et al. evaluated motor severity in patients with falls and Parkinson's disease in free-living conditions through a single accelerometer positioned on the lower back [60]. Jarchi et al. proposed a method for detecting gait events of subjects that underwent anterior cruciate ligament reconstruction with a single H-MIMU positioned on the ear [61]. Ionescu et al. detected cadence of children with cerebral palsy in free-living conditions using a single trunk-fixed accelerometer [8]. Soltani et al. developed algorithms for walking speed estimation using a single B-MIMU [62]. Sijobert et al. implemented and validated an algorithm for the estimation of the stride length of Parkinson adults using a single shank-MIMU [51]. Trojaniello et al. evaluated accuracy, sensitivity and robustness of five different methods for the estimation of temporal parameters using a single B-MIMU [63]. While conventional methods developed *ad hoc* for multi-MIMU configurations typically attain remarkable performance, results achieved through a single-MIMU approach are generally less accurate.

2.3 Role of AI in gait studies

Clinicians typically employ gait analysis for diagnosis and therapy selection but have been tested by highlighted complexities such as volume and non-unique connections between the variables of one's gait [14].
Actually, human biomechanics of gait encompasses a large amount of interrelated parameters that are tough to interpret due to a vast amount of data and their inter-dependencies. Sometimes, clinicians need to take into account at the same time symptoms of the patient, therapy programs, potential side effects, comorbidity, previous medical records, and many other elements. To facilitate their evaluation, the incorporation of AI with biomechanics is a promising solution [14].
As a matter of fact, gait analysis could benefit from AI methods, since these are able to handle high dimensional, time-varying and complex data [43][64]. AI techniques are capable of building models that learn automatically from databases, make accurate predictions, and act "intelligently" [65].
As a subcategory of AI, MaLe methods have been massively employed in various fields related to healthcare such as medical diagnosis [66][67][68], pattern recognition [69][70], image processing [71], classification [72][73][74], prediction analysis [75][76] and monitoring [72][77]; therefore, they can be easily adapted to support gait studies [14].
In gait analysis, MaLe techniques have been applied to several purposes, such as the detection of gait disorders [74][78], the prognosis of early intervention for fall-related risks due to disabilities or aging [65][68][79], the definition of motor recovery tasks [80][81] and the scheduling of rehabilitation or therapeutic interventions [82].
MaLe models trained to interpret complex relationships can reduce time to diagnosis, improve patient monitoring, and assist clinicians in the selection of treatment. On the other side, such methods generally require computing facilities at the operation site [14]. MaLe methods operate by analyzing data or images to determine existing patterns. Moreover, MaLe allows for continuous learning, thus enhancing the diagnostic outcomes [14].
MaLe techniques need relatively large databases: the more the training data, the higher the performance of the trained model. However, in the initial stage of its deployment, gait analysis was confined to laboratory settings, relying mainly on video-based systems [14]. At that time, the implementation of MaLe methods was limited by a lack of technological advent in data processing and unmet data collection and storage capabilities. With the technological advent of vision and sensor capture systems for gait analysis, an accumulation of a large amount of interdependent parameters occurred, thus the need of developing methods to process such great amount of temporal and highly complex data [14].
Despite statistical tools are employed extensively in gait analysis, they lack predictive power to generalise on unseen data [14]. Hence, the application of MaLe to

biomedical gait analysis can represent a valid solution.
Classification and prediction tasks can be approached through a variety of supervised and unsupervised techniques of MaLe in the context of vision-based [79][82] and sensor-based [83][84] systems. In addition, reinforcement learning (RL)[6] can be used to obtain better performance of tracking control in rehabilitation strategies [85].
One of the main detractors of MaLe is represented by the issue of assessing to what extent the trained model is able to perform the task on an unknown dataset i.e. its *generalization* capabilities [14]. Another relevant aspect of the design of a MaLe algorithm is represented by the feature extraction/selection phases [74][81], that have proved to increase computational efficiency by reducing computing intervals [14].

2.3.1 Overview of AI

With the expression AI, one usually refers to the ensemble of methods and techniques intended to work in synergy with the human beings to strengthen their decisions or enhance their performance. However, the meaning of the term "AI" has evolved through the decades, so that its current meaning significantly differs from the original one.
The term AI was adopted for the first time by a group of researchers in the title proposal of a 2-months workshop occurred in Dartmouth (Hanover, USA) in 1955 (Summer Research Project on Artificial Intelligence) [86]. In their vision, developing AI meant dedicating efforts to the implementation of *intelligent machines*, eventually capable of replacing the human intelligence. Soon, such purpose turned out to be unrealistic, due to technological constraints, leading to the so-called "First Winter of the AI" in the 1970s. Therefore, researchers tried to reshape the concept of AI towards the development of *expert systems*, that were supposed to extract knowledge from human experts and replicate it to assist humans in solving real-world problems. Still, feeding computers with knowledge was not a trivial challenge, due to inconsistency of the human thought and to the problem of codifying human knowledge into a form that was digestible for machines. Such struggle strongly limited the enforceability of expert systems to real applications, leading eventually to the Second Winter of the AI in the 1990s. After this failure, the idea of AI was again redesigned. In the last years of the 20^{th} century, a novel approach based on the extraction of knowledge from data turned out to be not only feasible but also effective: the era of MaLe had begun.

[6] In RL, the model is dynamically fed with training data, thus its performance increases with time.

MaLe

Typically, a distinction is made between AI and MaLe. In general, AI is an umbrella term for a wide set of methods and techniques, of which MaLe can be considered a subset.
The goal of MaLe is training computational models that are able to perform a certain task by learning from data.
In gait analysis, MaLe serves to model a biomechanical system $T(x)$ by determining the relationship between input data $f(x)$ and outputs $y(x)$, despite $f(x)$ being affected by noise $n(t)$, which requires pre-processing of the input data [14]. The input data are raw multidimensional arrays of size $V \times U$, where V corresponds to the number of observations and U represents the data features such as kinematics, kinetics or neuromuscular signals [14]. The model output consists in the classification of gait events, activities or disorders. Typically, evaluation of MaLe models adopts an iterative approach where the input dataset is divided into a training set (TRS), test set (TS) and validation set (VS). The model is trained using a TRS and validated on the VS to prevent overfitting and perform hyperparameters tuning. Finally, the performance is evaluated on an unseen TS i.e. a dataset not used in the training phase. When desired accuracy is achieved, the process ends, otherwise the model hyper-parameters are re-adjusted and the model is re-trained until a suitable value of accuracy is attained.
As a matter of fact, MaLe appears to be a problem of *induction*, as the goal is to exploit the finite training data to find a function able of making predictions for all the possible input elements [87].
A vast number of input features leads to system complexity; therefore, feature selection can be done [14].
A crucial step for MaLe is represented by the learning process. The most commonly used learning techniques include supervised, unsupervised and RL [14] (Table 2.2).

Supervised learning In this type of learning, the input data is an array of features vectors - also called *predictors* - associated to their respective response or label. The aim of training is to assess a function that best maps the relation between input feature vectors and the corresponding response [14]. Such purpose is typically referred to as *regression*, for continuous outputs, or *classification*, for discrete outputs [87].
In the framework of gait studies, several supervised algorithms have been explored:

- **SVM**: SVM has good generalisation capability even for relatively small datasets. SVM is parameterized by kernels, which allow it to deal with both linear and nonlinear problems; moreover, it can be used for multi-class classification as well as binary classification [44][83].

- **ANN**: ANNs in the shape of single or multi-layer perceptrons have proved to be a valid resource for prediction and pattern recognition in gait studies. Still, explainability of the network response remains as issue, as many times it acts as a black box [14]. Training of ANNs use different variants of the back-propagation (BP) algorithm.
- **DT**: DTs are used to model the highly nonlinear and complex relations between variables [14]. DTs are typically easily interpretable; however, they usually fail when the degree of task complexity becomes too critical.
- **Random forest (RF)**: A RF is basically an ensemble of randomised DTs. Therefore, the predicted output is assigned after a process of voting i.e. the output with the maximum number of DT votes is chosen [14].
- **HMM**: HMM models have the peculiarity of maintaining the imposed sequence of events for the phenomenon that they refer to, hence, they are particularly suitable to perform tasks of gait phases recognition [88].
- **K-nearest neighbours (KNN)**: The KNN classifier works by calculating the distance of an object from the representative objects of each class and assigning that object to the class that is less distant from, according to a certain distance metric. KNN is extensively used in real-time applications [69][80], as it is free from implied presumptions about the dataset distribution [14].
- **Fuzzy techniques**: Fuzzy techniques find application in gait asymmetry studies, where linguistic information is expressed in the shape of varying membership functions rather than numerically [72][89]. However, spreading of such techniques in gait analysis is being limited by the struggle in defining a number of linguistic variables and optimal choice of membership functions.

In the gait analysis panorama, supervised techniques of MaLe have been applied to several tasks such as gait activity detection, gait events detection (GED), disorder detection, asymmetry detection and neurological studies [14].

Unsupervised learning In unsupervised learning - sometimes referred to with the term *clustering* - no labels are provided to the learning algorithm, but rather the algorithm itself deduces the relationship among several inputs to assess an output [14]. The vast majority of clustering algorithms identify clusters according to the distance between all feature vectors. Such methods were less investigated in gait studies, since the accurate definition of the learning objectives and the handling of a large amount of feature vectors are challenging [14]. Nevertheless, such techniques can be employed when the relationship between different observations is not known.

Typically, unsupervised methods are complemented by dimensionality reduction methods, especially when the size of the dataset is significant. Unsupervised methods of MaLe are able of learning separate patterns for particular pathological conditions, starting from the appropriate selection of distance metrics depending on the given problem [14].

RL In gait studies, RL is mainly employed with exoskeletons or walking assistive devices to make those systems interact with a dynamic environment. Actually, in the context of rehabilitation devices, RL techniques coupled with DeLe proved to be able of capturing user's variability and subject-specific needs, resulting in automation [14].

Learning model		Complexity	Interpretability	Strengths	Drawbacks
Supervised	SVM	Moderate	Moderate	Generalisation Stability	Feature scaling Selection of kernel function
	ANN			No need for hand-crafted features complex relationships detection	Poor explainability
	DT			Probabilistic approach	Lacks predictability of continuous attributes Instability
	RF			Combination of multiple DTs Fast training	Large memory consumption for large datasets Tuning of hyper-parameters
	KNN			Instance-based learning	Feature scaling sensitive to outliers
	Fuzzy			Rule-based approach with low complexity	Selection of membership function Large number of linguistic features
Unsupervised		Moderate	Difficult	Ease of implementation	Time complexity
RL		Moderate	Moderate	Continuous learning No model training	Limited learning

Table 2.2: Strengths and drawbacks of some of the most explored MaLe models for gait studies [14].

DeLe

As MaLe can be seen as a subset of the wider ensemble of methods that goes under the name of AI, DeLe can be seen as a subset of MaLe itself [90].
DeLe basically extends the concept of neural networks by training "deeper" and more complex neural network models with a higher number of layers in respect with MaLe models [90]. The peculiarity of DeLe is the use of significantly larger datasets in the training phase. Actually, while performance of MaLe neural networks reaches a plateau even when trained with big datasets, DeLe networks manage to unleash the complete potential of data: theoretically, if the available training dataset was sufficiently large, the performance of the trained DeLe model would be impeccable [91].

Feature selection and extraction

Dimensionality of data should be taken into account when designing a MaLe algorithm, as it affects complexity and processing capabilities [14]. Often, input dataset feature with a high number of predictors; however, some of them may be poorly relevant, redundant or even detrimental for the target task [90]. To reduce computational complexity and prevent performance from being lowered by confounding features, dimensionality reduction techniques are typically adopted [81]. A distinction is typically made between feature selection and feature extraction techniques [14].

- **Feature selection**: Selection of a subset of features from the input data. Selection can progress forward or backward: in the first case, features are added iteratively until an optimal vector of features is determined, in the second case, features are iteratively removed from an initial vector of features, until an optimal set of features is identified.
- **Feature extraction**: Mathematical transformation of original features to derive new ones. When performed "manually", such approach is considered more "expensive", as all the original features need to be collected anyway [90]. DeLe methods typically employ feature extraction techniques to craft new low-level features from high level features or raw data, releasing the operator from the burden of features hand-crafting [91].

In gait analysis, several techniques of dimensionality reduction have been investigated, such as principal component analysis (PCA) [44], hill climbing [64] and

discriminate analysis[7]. However, some classifiers like ANN, deep neural network (DNN) and convolutional neural network (CNN) automatically select/extract optimum features from the original features set [91].

2.3.2 Machine learning and deep learning applied to gait analysis

GED, gait activity detection (GAD), gait disorder detection (GDD), gait asymmetry detection (GAsD) and neurological studies in gait have represented the most explored application areas of MaLe methods to gait analysis for the last years [14] (Figure 2.13).

The assessment of STP typically requires the previous identification of the gait

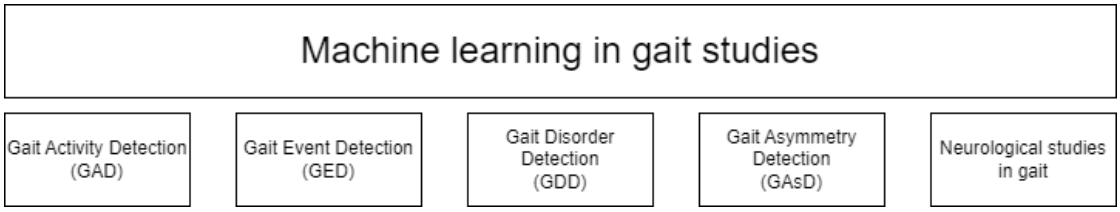


Figure 2.13: Taxonomy of machine learning applications [14].

phases and sub-phases: in literature, such type of study goes under the name of GED [14].

The detection of the gait phases - swing and stance - relies on the determination of ICs and FCs (see 2.1.3). Sometimes, sub-phases of gait i.e. loading, push off, swing and terminal swing are also targeted, as they can provide insight into individual variations and dynamic assessment of one's gait [66]. GED can also support control strategies of rehabilitation robots in order to prevent them from harming the patients, as well as pivoting FES, real-time orthosis control and gait rehabilitation [66].

In the last decade, GED has been addressed by a number of authors and research groups. Souza et al. [69] managed to train a ANN classifier for the recognition of human body gait parameters, reporting an accuracy superior to 99%. Farah et al. [66] succeeded in the identification of four gait phases across different walking conditions employing local sensor signals from thigh and knee with a logistic model DT. Paulo et al. [65] applied one-class SVM to automatically detect shifts in gait

[7]linear discriminate analysis (LDA) allow the projection of features from higher dimension space to lower dimensions [14]. If linear separation of the dataset is not feasible, then non-discriminate classifiers can be employed, such as quadratic discriminate analysis (QDA), flexible discriminate analysis (FDA) and FDA [14].

scheme. Finally, adaptive learning and RL were exploited for adaptive controllers with hemiplegia patients [85].

Chapter 3

The INDIP system

In the present Chapter, the basic notions regarding the functioning of inertial sensors i.e., accelerometers and gyroscopes are provided. Then, the INertial module with Distance sensors and Pressure insoles (INDIP) system used for data collection is described in detail.

3.1 Inertial sensors

Inertial sensors can track the motion of an object with respect to an inertial reference frame [92]. Inertial sensors directly measure inertial signals i.e. accelerations and/or angular velocities. From such quantities, linear velocities, linear displacements and angular displacements can be directly determined through subsequent integration. Inertial sensors are massively employed in many fields (from military to customer services); as they are generally inexpensive and sufficiently reliable for many applications. However, inertial sensing technology is also affected by some issues, such as drift errors and the need for frequent calibration (Table 3.1). As mentioned in Sub-subsection 2.2.4, inertial sensors - together with magnetometers - are often embedded within MIMUs.

3.1.1 Accelerometer

Accelerometers measure acceleration along one, two or three sensitivity axis (accelerometers that measure accelerations in all the three directions are commonly called tri-axial accelerometers).
Accelerometers measure the *proper linear acceleration*[1] a_p, which is the vectorial

[1]Also called *g-force* or *specific force*.

Strengths	Drawbacks
• Non-invasiveness • Low cost • Low power consumption • Low encumbrance • No environmental restrictions (real-world applicability) • Multi-parametric measures • Wearable • Scaling-up availability • Industry-ready	• Drift errors • Instability of measurements • Low accuracy • Electromagnetic (electro-magnetic (EM)) interference • Affected by temperature and noise • Require periodic calibration • Poor displacement estimation

Table 3.1: Strengths and drawbacks of inertial sensors for customers applications.

difference between the coordinate acceleration a_c - which is the rate of change of the velocity of the sensor - and the acceleration of gravity, $\vec{g}$:

$$a_p = a_c - \overrightarrow{\mathrm{g}} \tag{3.1}$$

Then, the output of the accelerometer does not represent the effective acceleration that the sensor is experiencing, but rather the difference between such acceleration and the acceleration of gravity. As a consequence, a 1D-accelerometer that is free-falling with the positive axis parallel to gravity will give $0\,\mathrm{m/s^2}$ as output, while it will give $|\vec{g}| = 9.81\,\mathrm{m/s^2}$ when it is stationary.

Principle of functioning

An accelerometer can be modelled as a second order spring-mass-damper system with proof mass m, elasticity constant k and damping factor b (Figure 3.1).

When a force $\overrightarrow{F_{applied}}$ is applied to the system, the proof mass, the spring and the damper tend to react with forces opposed to the applied force. According to the Newton's second law - which states that the algebraic sum of all the forces equals the inertial force of the proof mass - the vectorial sum of the applied force and the forces exerted by the spring and by the damper (respectively $\overrightarrow{F_{spring}}$ and $\overrightarrow{F_{damper}}$)

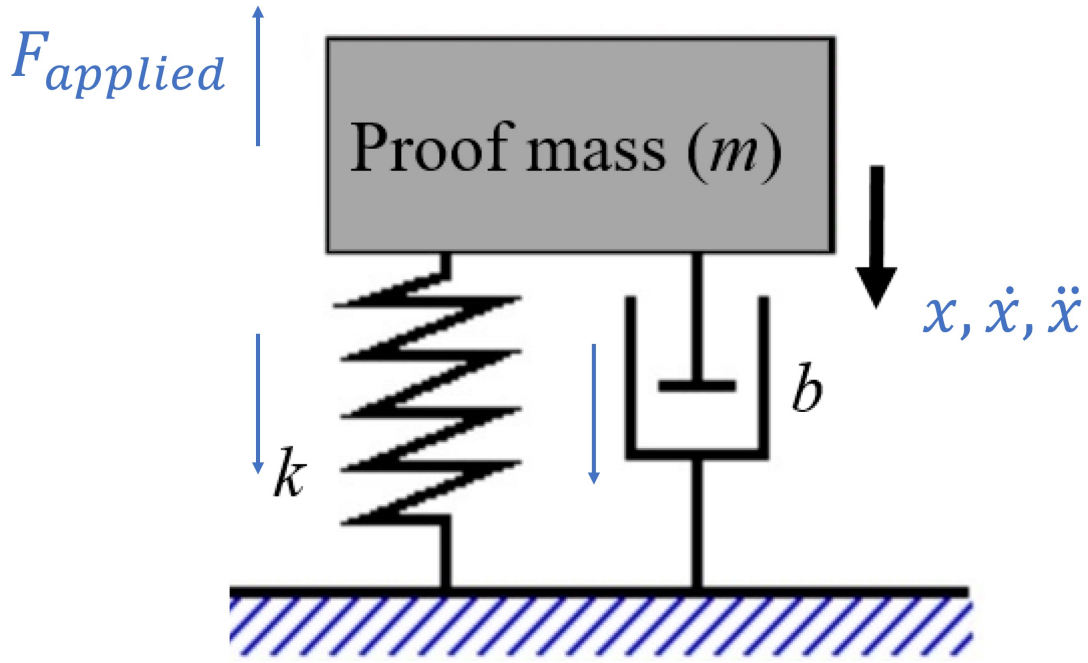


Figure 3.1: Dynamic model of a 1D-accelerometer [93].

equals the inertial force acting on the proof mass $\overrightarrow{F_{mass}}$:

$$\overrightarrow{F_{applied}} + \overrightarrow{F_{spring}} + \overrightarrow{F_{damper}} = \overrightarrow{F_{mass}} \tag{3.2}$$

Linear displacement x, velocity $\dot{x}$ and acceleration $\ddot{x}$ of the mass can be explicited by substituting the definitions of the damping, elastic and inertial forces in Equation 3.2:

$$F_{applied} - kx - b\dot{x} = m\ddot{x} \tag{3.3}$$

The applied force can be also written as the scalar product between the proof mass and the acceleration of the system a_c:

$$kx + b\dot{x} + m\ddot{x} = F_{applied} = ma_c \tag{3.4}$$

The acceleration a_c represents the input to the accelerometer and - in absence of gravity - it corresponds both to coordinate and proper accelerations. If the sensitivity axis is parallel to gravity, the gravity acceleration exerts on the mass a force proportional to the mass itself. Then, the accelerometer will measure the proper acceleration a_p, which incorporates the contribution of gravity:

$$kx + b\dot{x} + m\ddot{x} = m(a_c - g) = ma_p \tag{3.5}$$

Being Equation 3.4 a second order non-homogeneous differential equation, its solution can hardly be computed in the time domain, while it is trivial in the Laplace domain:

$$kx(s) + bsx(s) + ms^2x(s) = F(s) = ma(s) \tag{3.6}$$

By rearranging the terms of Equation 3.6, the ratio between the output displacement $x(s)$ and the input acceleration $a(s)$ can be easily derived:

$$H(s) = \frac{x(s)}{a(s)} = \frac{1}{s^2 + \frac{b}{m}s + \frac{k}{m}} = \frac{1}{s^2 + \frac{\omega_0}{Q}s + \omega_0^2} \tag{3.7}$$

$H(s)$ is the *transfer function* of the system, while Q and ω_0 correspond respectively to the *quality factor* and to the *resonance angular frequency*:

$$Q = \frac{m\omega_0}{b} \tag{3.8}$$

$$\omega_0 = \sqrt{\frac{k}{m}} \tag{3.9}$$

The transfer function of the accelerometer represents the dynamic sensitivity of the sensor, whose value changes according to the frequency of the input acceleration. As it can be easily demonstrated, the sensitivity of the accelerometer $H(s)$ decreases with the square of frequency from a value $H(0)$ when the input acceleration is constant to a value of $0\,\mathrm{m/s^2}$ when the input frequency approaches infinite. As a consequence, accelerometers are typically designed to work at frequencies much lower than their resonance frequency ($\omega \ll \omega_0$). $H(0)$ - also referred to as *static sensitivity* - is inversely proportional to the square of the resonance frequency (Equation 3.10).

$$H(0) = \frac{1}{\omega_0^2} \tag{3.10}$$

In order to achieve a large sensing bandwidth, a high resonance frequency is needed. However, this reduces the sensitivity of the device. Therefore, the design of an accelerometer is usually a compromise between the sensitivity and bandwidth [93].

Technical specifications

In the following, some of the main technical specifications for accelerometers are listed [56]:

- **Sensitivity (mV/g)**: Sometimes called *scale factor*, it is defined - at a given frequency - as the output voltage signal generated per unit input acceleration in a given direction:

$$S_x = \frac{\text{Output voltage generated(mV)}}{\text{Input acceleration along x-axis(} g\text{)}} \tag{3.11}$$

If the accelerometer has more than one sensitive axis, S_y and S_z can be defined as well.

- **Cross-axis sensitivity (mV/g)**: the output voltage generated due to an acceleration component orthogonal to a sensitive axis

$$S_{x,ay} = \frac{\text{Output voltage generated(mV)}}{\text{Input acceleration along y-axis(} g\text{)}} \tag{3.12}$$

$$S_{x,az} = \frac{\text{Output voltage generated(mV)}}{\text{Input acceleration along z-axis(} g\text{)}} \tag{3.13}$$

  For a tri-axial accelerometer, each axis has two cross-axis sensitivities.

- **Bandwidth (Hz)**: Range of vibration frequencies to which the accelerometer responds. The bandwidth also determines the frequency at which an accelerometer provides its readings. Since the range of frequencies created by the human body motion hardly goes beyond 12 Hz; a bandwidth of 40 Hz to 60 Hz is typically enough for sensing a tilt or human motion.

- **Voltage noise density (µg/Hz$^{0.5}$)**: Voltage noise changes with the inverse square root of the bandwidth: the faster the accelerometer provides readings, the worse the accuracy. Noise strongly affects the performance of the accelerometers when operating at lower accelerations i.e., with a smaller output signal.

- **Zero-g-voltage (V)**: expected voltage at 0 g.

- **Dynamic range (g)**: Range between the smallest and the largest detectable amplitudes before incurring in distortion or clipping of the output signal.

Manufacturers of accelerometers generally provide the customer with a datasheet that includes values and range of tolerance for each of the above mentioned specifications.

Transduction mechanisms

Several sensing mechanisms can be adopted to convert the proof mass displacement due to the input acceleration into a measurable quantity. According to it, accelerometers can belong to one of the following classes:

- **Piezoresistive**: Piezoresistive accelerometers feature with simple design structure, fabrication process and readout circuits; however, they generally have low sensitivity, bulky designs and are strongly affected by temperature [93].

- **Piezoelectric**: Piezoelectric accelerometers exploit the property of piezoelectric crystals to generate electric charge in response to a mechanical stress along a sensitive axis. Piezoelectric accelerometers have the advantages of being self-powered and providing directly a digital output. In addition, they generally require a simple interface circuitry. On the other hand, leakage of piezoelectric materials lowers the performance in DC conditions and the size of the devices is quite large [93].
- **Capacitive**: In capacitive accelerometers, the displacement in the proof mass is first converted to a proportional capacitance change and then into an amplified voltage signal. Capacitive accelerometers are characterized by high sensitivity, good DC performance, low noise, low drift and low temperature sensitivity. On the other hand, they are strongly influenced by EM interference [93].
- **Optical**: Optical accelerometers are highly sensitive and mostly immune to EM interference. However, they struggle to be integrated with the readout circuitry [93].
- **Tunneling**: Tunneling accelerometers are highly sensitive and have low footprint; nevertheless, they are significantly affected by noise at low frequencies and require expensive fabrication process [93].

Other less used types of accelerometers exploit resonant and thermal sensing technologies [93].

3.1.2 Gyroscope

Gyroscopes measure angular velocity along one, two or three sensitivity axis (gyroscopes that measure angular velocities in all the three directions are commonly called tri-axial gyroscopes). Angular velocities measurements are generally expressed in degrees per second (dps).
In combination with accelerometers, gyroscopes can be employed in several applications that need an integrated solution for inertial sensing and motion processing problems [94].

Principle of functioning

Although several types of technologies can be employed for sensing angular velocity, the most common type of gyroscopes is based on the *Coriolis force*; hence, they are called *Coriolis vibrating gyroscopes* or *Vibrating forks gyroscopes*. Every time that an object with mass m rotates and translates respectively at angular rate $\overrightarrow{\omega}$ and linear speed $\overrightarrow{v_t}$, a force $\overrightarrow{F_c}$ applied to the object is generated - which goes under

the name of Coriolis force (Equation 3.14).

$$\overrightarrow{F_c} = -2m\overrightarrow{v_t} \times \overrightarrow{\omega} \tag{3.14}$$

Vibrating forks gyroscopes embed a pair of proof masses that oscillate with the same amplitude, but in opposite directions. At rest, the tines resonate in anti-phase in the plane of the fork (drive mode). When the sensor is put into rotation, the tines begin to oscillate also along the orthogonal direction to the plane. Such oscillation generates a torque that triggers the torsional mode around the gyroscope stem. Forks can feature with one, two or more tines: the more the tines, the higher the sensitivity and rejection to common-mode errors (Figure 3.2).

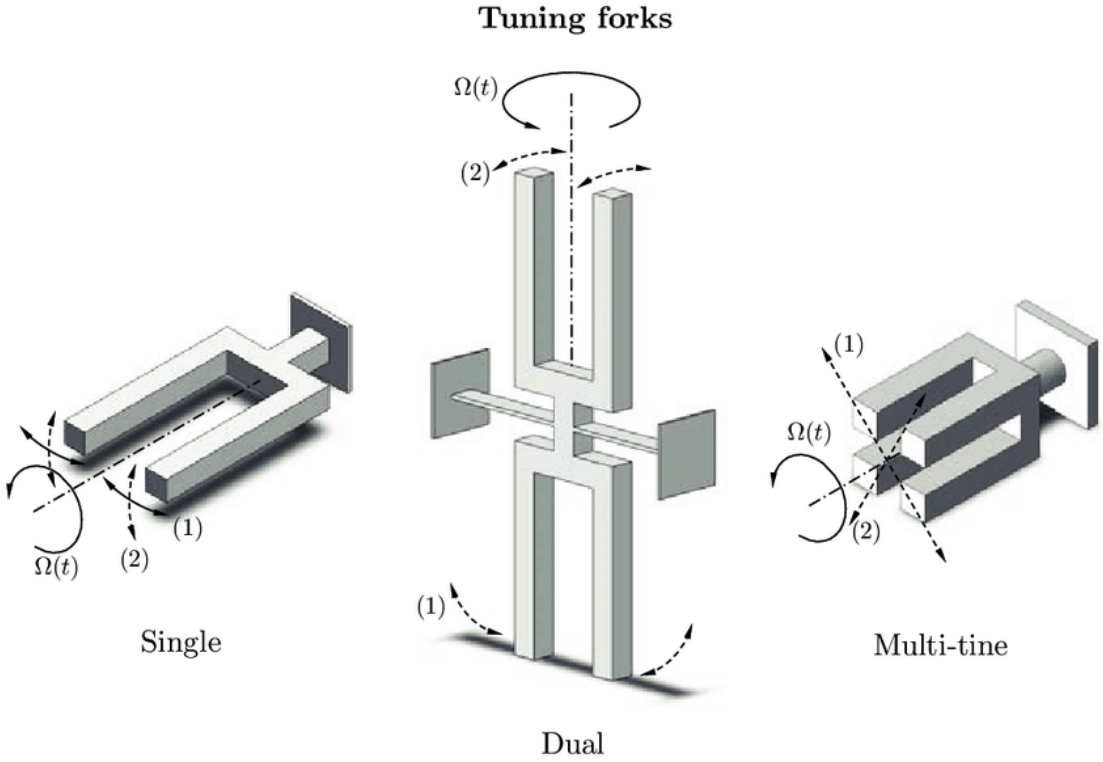


Figure 3.2: From the left, a single, dual and multi-tine configuration of vibrating forks gyroscopes [94]. Ω_t is the input angular velocity, (1) is the drive mode and (2) is the vibration response due to the Coriolis force.

Technical specifications

In the following, some of the main technical specifications for gyroscopes are listed:

- **Input range (dps)**: The input range represents the range of measurable values of angular velocity. The limits that define such range delimit the range

of input values for which the specified performance accuracy is valid. The amplitude of the input range is typically referred to as full-scale range (FSR).

- **Accuracy (%FSR)**: Also known as *linearity error*, it represents the average deviation of the output from a least-squares linear fit of the input-output data. The definition of accuracy implicitly assumes that the sensor exhibits a linear input-output behavior - ideally.
- **Scale factor (V/° /s)**: The scale factor represents the ratio between a change in the output and a change in the input. It is computed as the slope of the least squares straight line fit to input-output data.
- **Resolution (%dps)**: The smallest detectable input change for inputs greater than the noise level. Usually, it is evaluated as the minimum input change that produces a change in output equal to a specified percentage (e.g., 50 %) of the change in output calculated through the nominal scale factor.
- **Drift rate**: The share of the gyroscope's output that is functionally independent on the input rotation. The drift rate can be referred to two different components:
  - **Systematic**: The *systematic drift rate* is dependent on two contributes:
    - **Bias (dps)**: The bias - also known as zero rate output - is the average value of the gyroscope output over a specified time interval when operating at conditions uncorrelated to the input rotation.
    - **Environment**: The environmentally sensitive drift rate expresses the portion of drift rate that is dependent on temperature, acceleration, vibration and other quantities.
  - **Random**: The *random drift rate* is dependent on two contributes:
    - **Angle random walk ($^\circ$ /s/h$^{0.5}$)**: Angular error increases with time due to white noise.
    - **Bias instability ($^\circ$ /h)**: Random variation in bias computed over a specified time interval.

3.2 The INDIP kit

The INDIP is a multi-sensor system designed by Università degli Studi di Sassari. The INDIP system was originally devised as a system for validating other devices for gait analysis in the context of a European Project[2]; therefore, it features with

[2]Mobilise-D, a project sponsored by the European Union (EU) for the assessment of digital mobility outcomes.

high reliability both at hardware and software levels.
The complete INDIP setup includes a number of inertial, pressure and distance wearable sensor, which can be employed separately or at the same time (Figure 3.3).

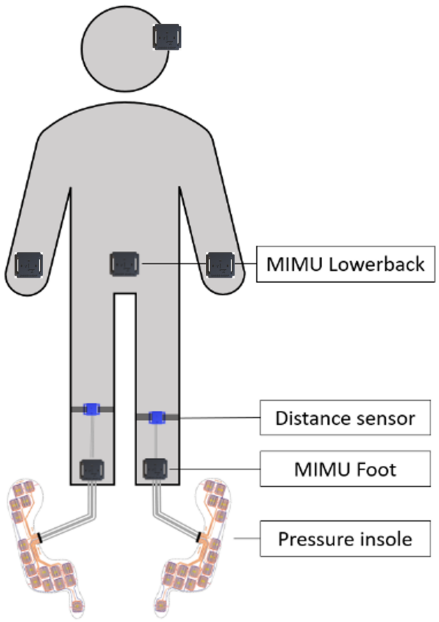
- **MIMUs**: A total amount of 7 MIMUs placed on the head, on the wrists, on the lower back and on the feet. MIMUs are firmly attached to the body through velcro bands to prevent any slippery or motion artifact.
- **Pressure insoles**: Two pressure insoles (one for each foot) to be placed within each shoe of the participant. Each pressure insole hosts a total amount of 16 pressure sensors, allowing a fine mapping of the foot-to-ground contact. Pressure insoles are usually protected by rubber insoles to prevent the sensing components from being damaged.
- **Distance sensors**: Two IR distance sensors attached to the shanks via velcro.

The sampling frequency of the INDIP system is 100 Hz. Before every recording, MIMUs should be previously connected one-by-one via USB to a computer in order to synchronize their timestamp with the current date: such procedure - performed through a custom graphic user interface (GUI) developed in Matlab[3] - is crucial to prevent timestamp of data recorded from different sensors to show any delay with respect to each other. After that, MIMUs are placed on the subject body and connected via bluetooth low-energy (BLE) to a custom app developed in Matlab, which allows the user to trigger the acquisition of all the connected sensors. Once that the acquisition has started, the participant is free to move: the MIMUs disconnect when the participant walks too far from the computer; however, once that the sensors have started recording, they continue to record data offline. At the end of the recording, sensors may be reconnected to the app via BLE to stop the acquisition and then connected via USB to the GUI to download recorded data.

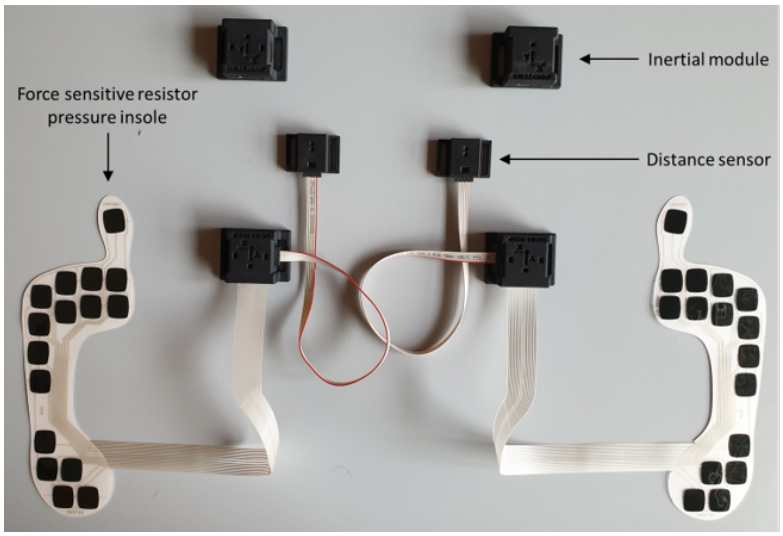
3.2.1 INDIP MIMU

Each MIMU consists in a printed circuit board (PCB) that hosts the inertial sensors, transmission modules and electronic circuitry for front-ending and data storage [95]. The PCB is embedded within a 3-D printed plastic case (Figure 3.4a). The INDIP system features with 6 MIMUs, positioned on 4 different body spots:

[3]MATLAB Release 2022b, The MathWorks, Inc., Natick, Massachusetts, United States.



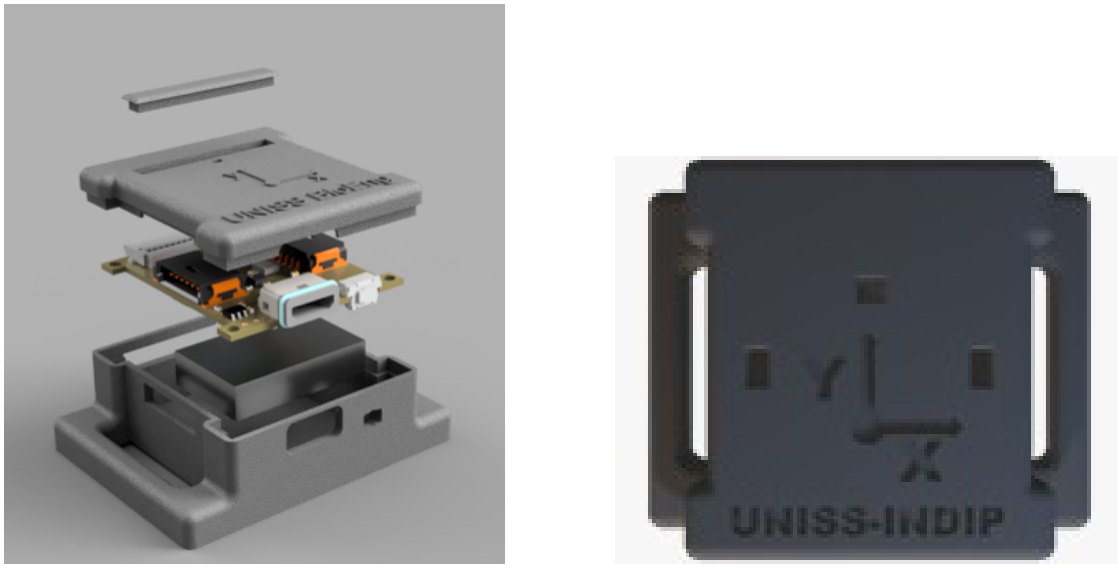
(a)



(b)

Figure 3.3: (a) Schematic configuration of the INDIP setup on the participant's body. (b) A top view of the INDIP setup.

- **Head**: The H-MIMU is positioned on the left side of the head of the participant with the y and x axis oriented approximately along the vertical (V) and anterior-posterior (AP) anatomical axis.



(a) **(b)**

Figure 3.4: (a) 3-D overview of an INDIP MIMU [95]. (b) Top-view of an INDIP MIMU. The x and y components lie on the sensor's plane, while the z-component points up perpendicularly.

- **Lower-back**: One MIMU is positioned on the lower back of the participant (L1-L2) with the y and x axis oriented approximately along the V and ML anatomical axis.
- **Feet**: Two MIMUs are positioned on the feet of the participant with the y and x axis oriented approximately along the V and ML anatomical axis.
- **Wrists**: Two MIMUs are positioned on the wrists of the participant with the y axis approximately oriented along the longitudinal axis of the forearm.

The MIMU hosts three main sensing elements, being a tri-axial accelerometer, a tri-axial gyroscope and a tri-axial magnetometer. All of them feature with selectable FSR and output data rate (ODR) and with low zero-measurement offset and noise (Table 3.2). Each MIMU contains a battery that allows it to operate for several hours, as well as a memory slot for storing acquisition data.

3.2.2 INDIP pressure insoles

The pressure insole includes 16 force sensing resistors leaning on a thin flexible plastic substrate shaped as a shoe insole [96]. Force sensing resistors are distributed along the insole length such as 9 for the forefoot, 2 for the midfoot and 5 for the

TRI-AXIAL ACCELEROMETER	
Measurement range	Up to ±16 g selectable FSR
Zero-g offset	±40 mg
Rate noise density	1.8 - 3.0 mg (Root Mean Squared (RMS))
Output data rate	1.6 to 6664 Hz
TRI-AXIAL GYROSCOPE	
Measurement range	Up to ±2000 dps selectable FSR
Zero-rate offset	±1 dps
RMS noise	0.075 dps
Output data rate	1.6 to 6664 Hz
TRI-AXIAL MAGNETOMETER	
Measurement range	±50 G
Zero-G offset	dynamically cancelled
Rate noise density	3 mG (RMS)
Output data rate	10 to 100 Hz

Table 3.2: Specifications of the sensors of the INDIP MIMU [95],[96].

rearfoot (Figure 3.6a). The pressure insoles do not have any batteries as they are meant to be connected to their corresponding foot MIMU through wires (Figure 3.6b). ODR of the pressure insoles can span over a range from 0 to 200 Hz.

3.2.3 INDIP distance sensors

The IR-time of flight (ToF) proximity sensor (VL6180X, [97]) provides proximity estimates in the range of 0 mm to 600 mm. The distance is estimated by measuring the phase shift between the radiated and the reflected IR waves (Figure 3.7a). The proximity sensor is encased within a plastic case together with a front-end for interfacing the inertial module, to which the distance sensor is wired (Figure 3.7b).

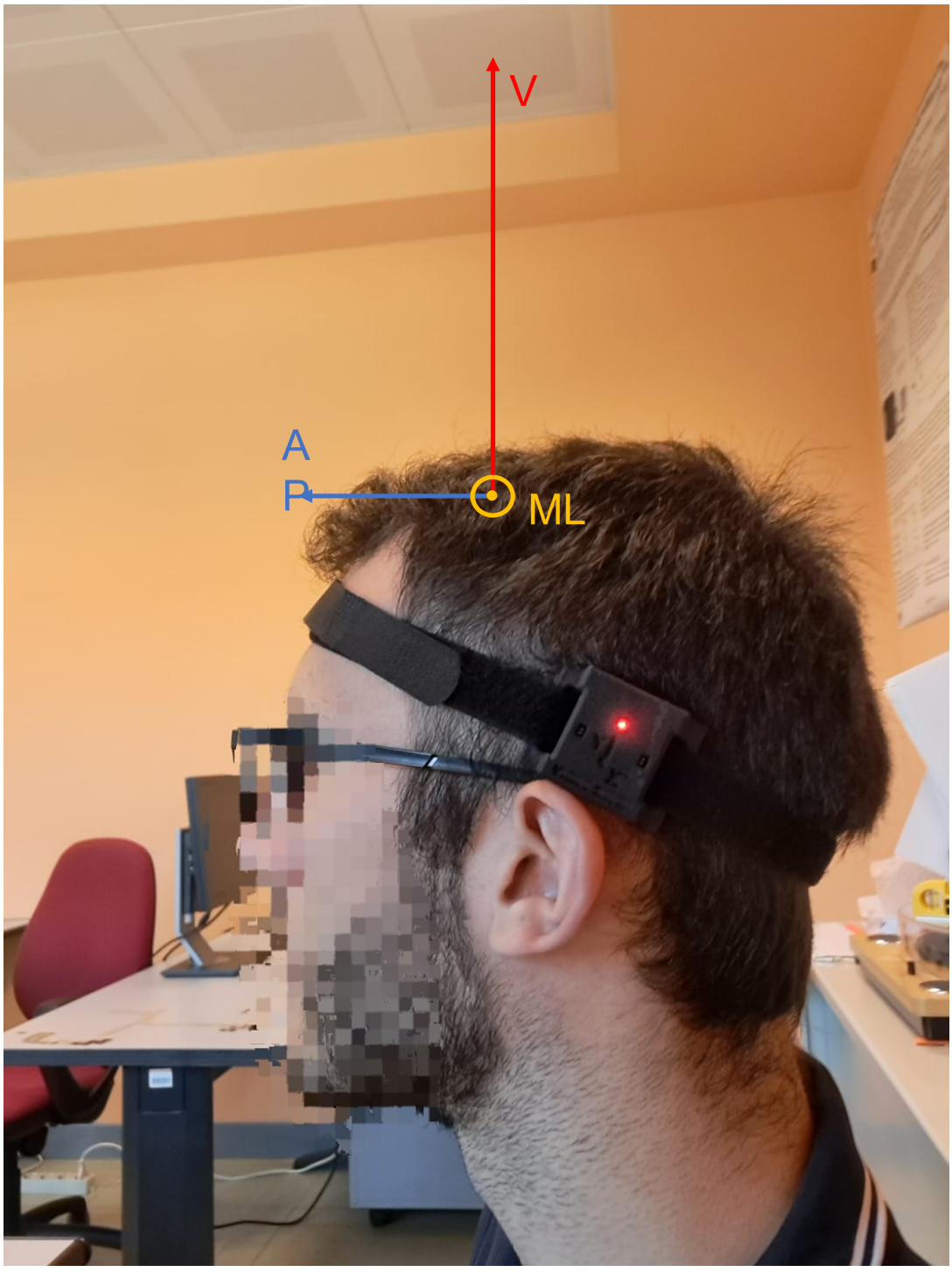
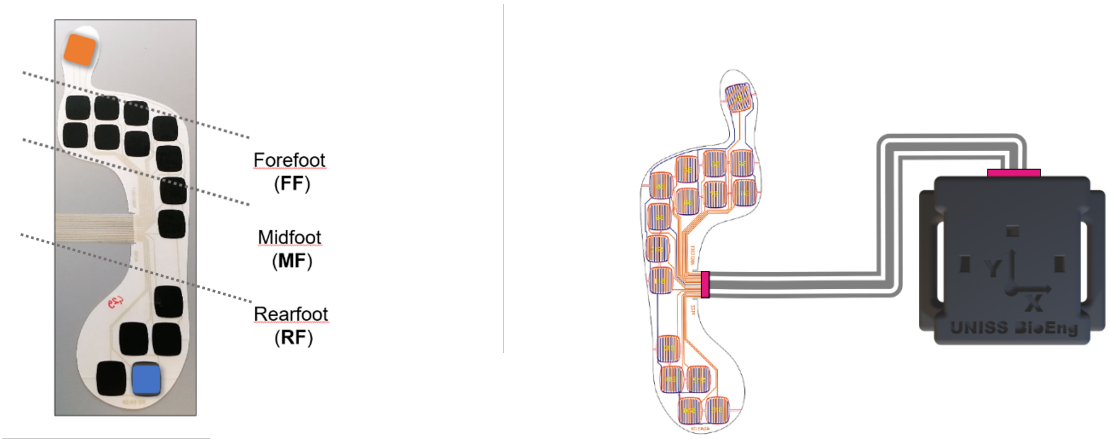
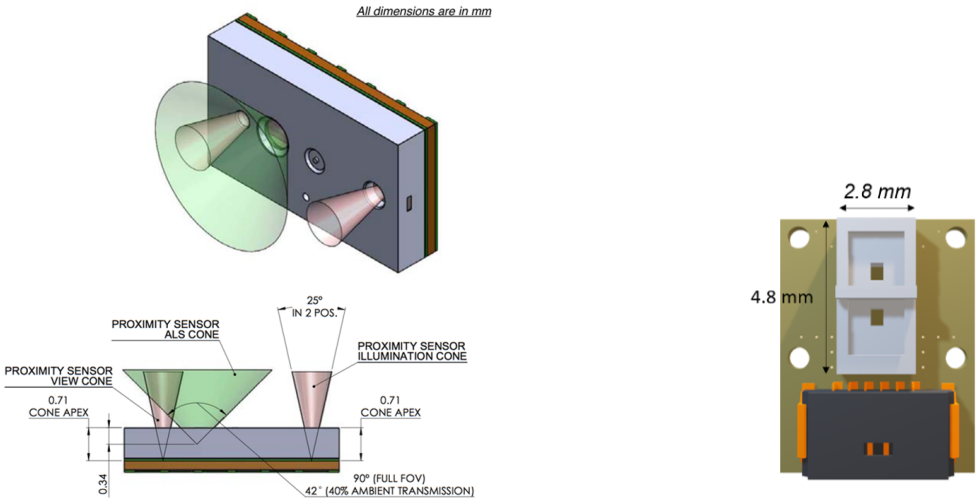


Figure 3.5: Head-MIMU mounted on the head of one participant. Note that the axis of the sensor are in general rotated with respect to the anatomical axis.



(a) **(b)**

Figure 3.6: (a) Top view of one pressure insole. (b) A pressure insole connected to its corresponding foot MIMU [96].



(a) **(b)**

Figure 3.7: (a) Features and dimensions (mm) of a a VL6180X IR-ToF proximity sensor [97]. (b) Top view of one distance sensor's internal components. Grey: VL6180X proximity sensor, Black: connector to the inertial module.

Chapter 4

Data collection

MaLe and DeLe usually require a relevant amount of data for training, validating and testing the algorithms. On the side of the learning algorithm, abundance of data is advised to increase generalizability of the trained model, as more data in the training stage translates into an extended capability to correctly classify or predict unseen data at the final usage. Validation - which is usually performed in parallel to training - typically requires a smaller dataset in respect to the one exploited for learning, as its purpose is not to train the model weather to prevent it from overfitting the training data. Eventually, some data should be left aside for testing the model's capabilities of generalizing on new data at the end of training. The size of the test set is usually comparable to the one of the validation set; however, having a model with good performance on a relatively large test set should be considered a plus.
For the thesis purposes, two sets of acquisitions have been carried out. The first set of acquisitions - referred to as *In-Lab* acquisitions or In-Lab dataset - includes gait of 11 subjects recorded in laboratory settings according to a well-defined experimental protocol. The second set of acquisitions - referred to as *Free-Living* acquisitions or Free-Living dataset - includes gait of 3 subjects recorded outside in unsupervised conditions. The purpose of the In-Lab dataset consists in training, validating and testing the developed algorithms (see Chapter 5), while the Free-Living dataset has been used only for testing.
The collected data is intended to train, validate and test a set of MaLe and DeLe models which are supervised (see Paragraph 2.3.1). Therefore, predictors data must be labeled i.e., they have to be associated to their respective response, such as a target class or value. As explained in Chapter 5, the goal of the developed algorithms will be the estimation of the temporal parameters and stride speed values from the data recorded from the head-MIMU only; hence, the other sensors of the INDIP kit are exploited to extract reference values for the temporal parameters and the stride speed values, through already validated algorithms [96] devised at

Università degli Studi di Sassari (see Appendix C.3).
In the following, the experimental protocol adopted for the acquisition of the two datasets will be described in detail. All the subjects involved in the study have signed an informed consent for the treatment of data recorded during the acquisition sessions.

4.1 In-lab acquisitions

The first tranche of acquisitions was conducted inside the Dipartimento di Elettronica e Telecomunicazioni (DET) of Politecnico di Torino.

4.1.1 Experimental protocol

Gait of 11 healthy participants (Table 4.1) was recorded through the INDIP system (Chapter 3) according to a well-defined experimental protocol.
The protocol for each in-lab acquisition is structured in seven *Tests* performed

Subject	Gender (M/F)	Age (y.o.)	Height (cm)	Weight (kg)	Shoe Size (EU)
1	M	26	177	63	42
2	F	28	163	55	38
3	F	26	164	56	37,5
4	M	29	185	77	45
5	F	25	165	55	38
6	M	23	179	56	42
7	M	23	179	70	46
8	F	27	170	65	37
9	M	24	178	75	43
10	F	27	165	52	37
11	M	32	183	67	43
Avg.	54% (M)	26,4	173,5	62,8	41,1
Std.	-	2,7	8,2	8,7	12,8

Table 4.1: Summary of participants for in-lab acquisitions.

under the supervision of one or two operators. Each test is composed of one to three repetitions hereinafter called *Trials*. In the following, the structure and purpose of each Test is briefly described.

- **Static test (Test 1)**: The Static Test is required to perform a Quality Check on the sensors performance. Signals recorded during this Test are submitted to

the INDIP Data Quality Check activities of daily life (ADL)- FISM GUI[1] that processes the signals and checks the presence of any artifacts or misbehaviors (Figure 4.1).

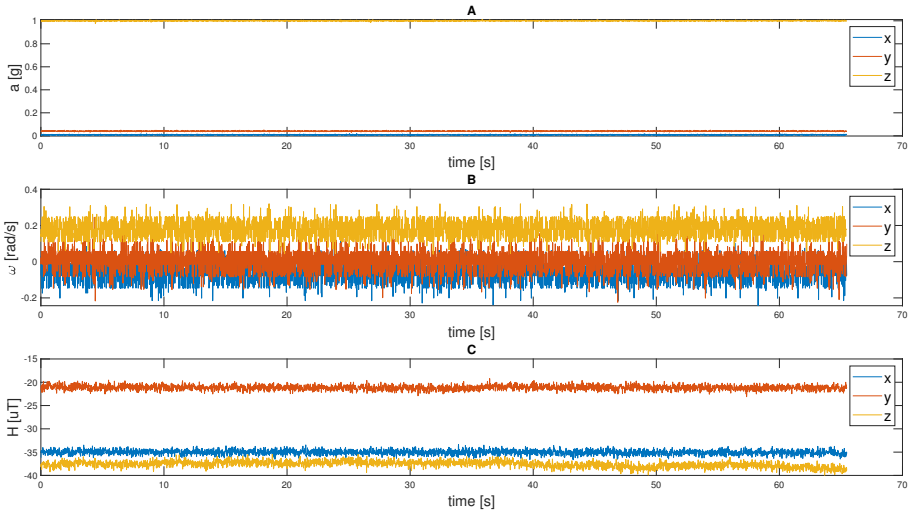


Figure 4.1: H-MIMU readings during the static acquisition (Test 1) for patient 2. A) Accelerometer readings; B) Gyroscope readings; C) Magnetometer readings. The only non-null acceleration component is the one parallel to gravity (z-component); moreover, since the MIMU is not undergoing any rotations, the readings of the 3 components of angular velocity are close to zero.

- **IMU static test**: This test consists in a static acquisition using the 6 INDIP MIMUs. The operator places all the devices on a flat surface and starts an acquisition of at least 60s, during the one knocking or moving the sensors should be avoided.
- **Pressure insoles static test**: This test is used to verify that the pressure insoles are properly working. An operator records the pressure insole signals while pressing the individual sensing units one at a time (the order in which the sensing elements are pressed is not relevant). This must be performed for both right and left pressure insoles (one acquisition at a time).

[1]Matlab GUI developed at Università degli Studi di Sassari for assessing the quality of data recorded by the INDIP system.

As already mentioned, data recorded in static conditions during Test 1 are used for the Quality Check. For the MIMUs, the GUI checks if the inertial signals have appropriate mean value (equal to zero), accelerometer norm within the expected range, not-too-high standard deviation, correct FSR or if they stopped recording during the ongoing acquisition. For the pressure insoles, the GUI checks if the insole has deteriorated its performance or if it stopped recording during the ongoing acquisition. After the performing of Test 1, static data are manually downloaded through the Matlab INDIP GUI, saved into the *Spot Check* folder of their respective *Participant Folder* and submitted to the INDIP *Quality Check* GUI (see Section C.2 for more information about the folder structure). If the quality check underlines any issues with the sensors, the malfunctioning sensors are replaced and the Quality Check is re-performed, otherwise, the operator(s) can go on with the other Tests.

- **Standing test (Test 2)**: This is a short static acquisition with the participant wearing all the INDIP MIMUs, pressure insoles and distance sensors (Figure 4.2). The participant is asked to stand still for at least 10s. Such Test is required to estimate the MIMUs orientation in the global framework, which allows to compute the rotation matrix to reorient the MIMUs recordings during the dynamic acquisitions (Figure 4.3, see A.3).

- **Data Personalization (Test 3)**: Such test is to check the correct placement of the pressure insoles inside the shoes and to override those sensors that do not activate properly (Figure 4.4).

  - Stand still for at least 10 s;
  - Lift up the left foot for at least 5 s (single right support);
  - Stand still for at least 5 s (double support);
  - Lift up the right foot for 5 s (single left support);
  - Stand still for at least 5 s (double support);
  - Raise and lower the left arm in the sagittal plane;
  - Raise and lower the right arm in the sagittal plane;
  - Walk at comfortable speed along a 12 m straight path (Figure 4.9a).

- **Slow straight walking (Test 4)**: The participant is asked to walk along a 12 m-straight path (Figure 4.9a) at slow speed (Figure 4.5). Such Test is repeated 3 times (Trials).

- **Normal straight walking (Test 5)**: The participant is asked to walk along a 12 m-straight path (Figure 4.9a) at comfortable speed (Figure 4.6). Such Test is repeated 3 times (Trials).

Figure 4.2: Front view of a participant during the Standing acquisition (Test 2). During the acquisition, participants are asked to stand still with their arms along their hips.

- **Fast straight walking (Test 6)**: The participant is asked to walk along a 12 m-straight path (Figure 4.9a) at fast speed (Figure 4.7). Such Test is repeated 3 times (Trials).
- **Round walking (Test 7)**: The participant is asked to walk for two rounds along an approximately 24 m-ring path (Figure 4.9b) at comfortable speed (Figure 4.8). Such Test is repeated 3 times (Trials). Such test was added to the experimental protocol to introduce some portions of non-straight walking in the dataset.

Repeating each Test three times provides a backup solution in case that one Trial

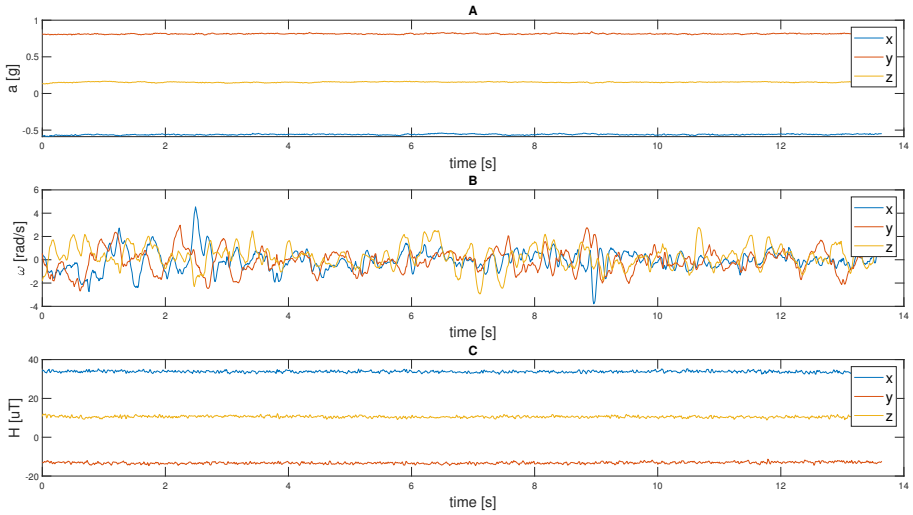


Figure 4.3: H-MIMU readings during the Standing acquisition (Test 2) for patient 2. A) Accelerometer readings; B) Gyroscope readings; C) Magnetometer readings. Since the sensor's axis are not aligned to the anatomical axis (Figure 3.5), the y-component of acceleration - which is associated to the vertical axis - does not equal gravity.

results in corrupted or damaged data. At the same time, disposing of three Trials for the same Test increases the redundancy of the training set. Actually, while on one side the inclusion of data referring to three different speed ranges and to non-straight walking increases data variability - promoting the generalization power of the trained models - on the other side introduces the risk of data underfitting, as the model is forced to learn to recognize a larger number of patterns.

4.1.2 Data preparation

After the performing of all the Tests, data recorded by each sensor are manually downloaded through the Matlab INDIP GUI and saved into the *Experimental Protocol* folder of their respective *Participant Folder* (see Section C.2) as text files (TXT). TXT files are submitted to the INDIP *Renaming* GUI and automatically saved as renamed TXT files into the *Laboratory* folder of their respective *Participant Folder*. Then, renamed TXT files are automatically standardized into a Matlab structure (`data.mat`) through a set of predefined Matlab scripts and functions and saved into the *Results* folder of their respective *Participant Folder*. Eventually, the `data.mat` structure is given as input to a set of Matlab scripts and functions developed by Università degli Studi di Sassari for the Mobilise-D

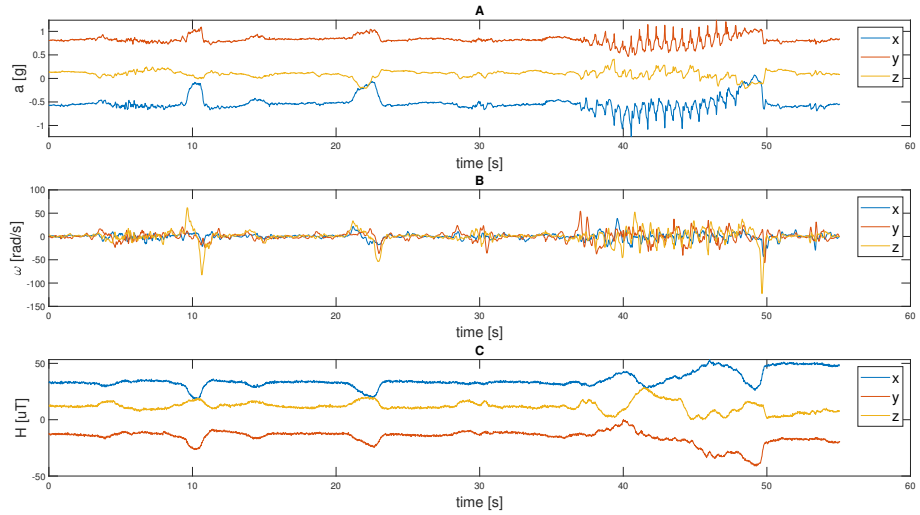


Figure 4.4: H-MIMU readings during the Data Personalization acquisition (Test 3) for patient 2. A) Accelerometer readings; B) Gyroscope readings; C) Magnetometer readings.

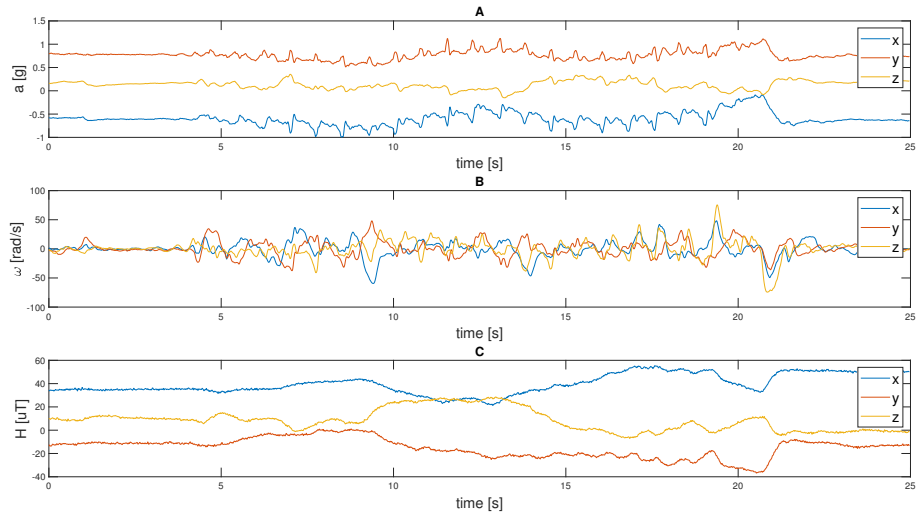


Figure 4.5: H-MIMU readings during the straight walking test at slow speed (Test 4) for patient 2. A) Accelerometer readings; B) Gyroscope readings; C) Magnetometer readings.

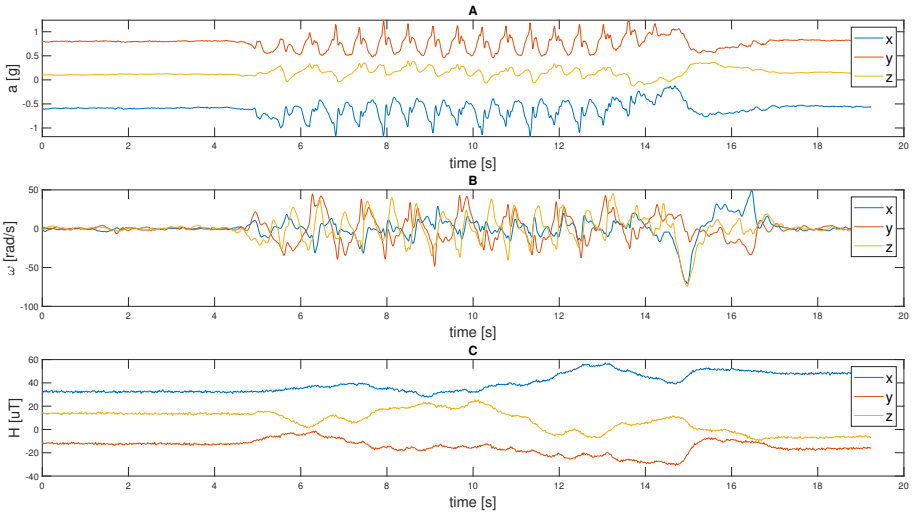


Figure 4.6: H-MIMU readings during the straight walking test at comfortable speed acquisition (Test 5) for patient 2. A) Accelerometer readings; B) Gyroscope readings; C) Magnetometer readings.

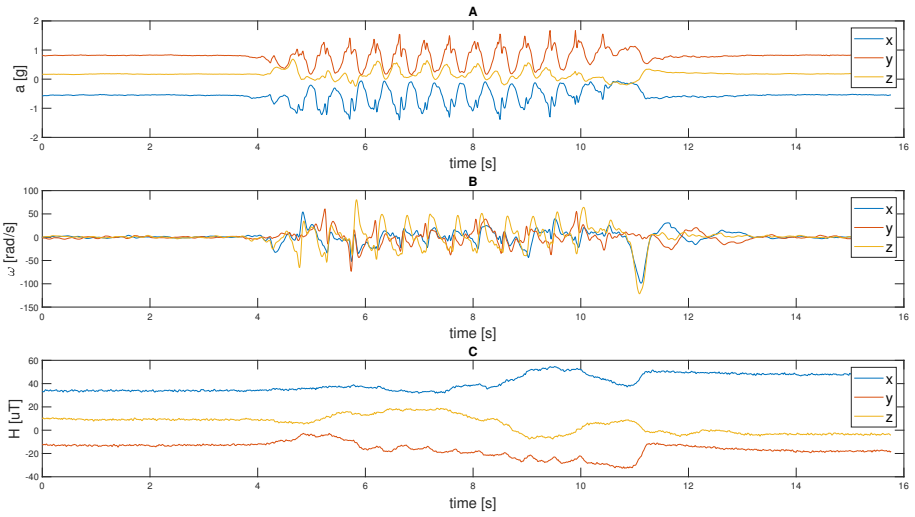


Figure 4.7: H-MIMU readings during the straight walking test at fast speed (Test 6) for patient 2. A) Accelerometer readings; B) Gyroscope readings; C) Magnetometer readings.

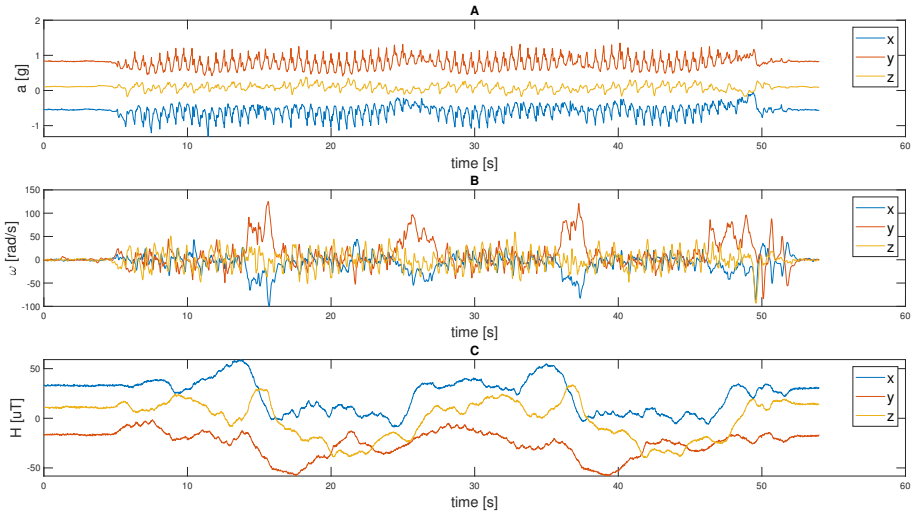


Figure 4.8: H-MIMU readings during the round walking test at comfortable speed (Test 7) for patient 2. A) Accelerometer readings; B) Gyroscope readings; C) Magnetometer readings.

project. Such algorithms return a Matlab structure (`data.mat`) that contains both the standardized data and the INDIP standard i.e., the GS for the estimation of STPs such as the stride lengths, the times and speeds (see Section C.3).
At the end of the In-Lab acquisitions, a total amount of about 1 and a half hours of gait data had been gathered.

4.2 Free-living acquisitions

Besides the in-lab acquisitions, acquisitions in unconstrained conditions of free-living were performed. While the In-Lab dataset was used to train and validate models described in Chapter 5, data gathered from this second tranche of acquisitions were used only for testing purposes.

4.2.1 Experimental protocol

Gaits of three healthy young (HY) subjects (Table 4.2) were collected. Each acquisition consisted of 4 separate recordings.

- **Static test (Recording 1)**: See Section 4.1.
- **Standing test (Recording 2)**: See Section 4.1.

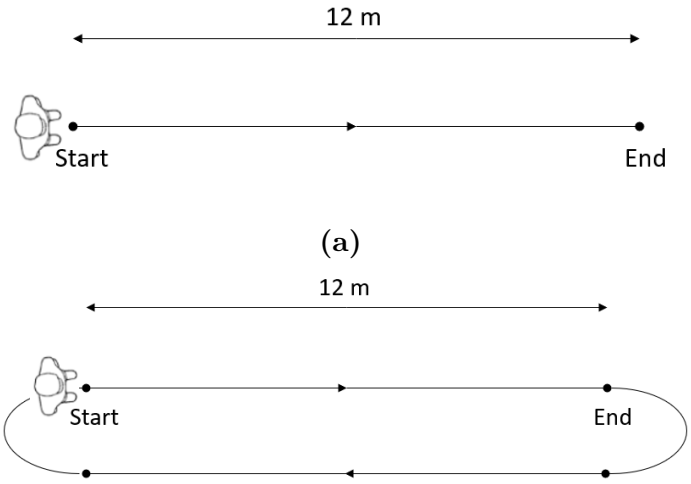


Figure 4.9: (a) Straight path for Tasks 3, 4, 5 and 6. (b) Ring path for Task 7. The 12 m straight path is marked with two adhesive bands on the floor on both ends, while the ring path is marked with two cones on both ends. Participants are asked to start walking approximately two seconds after the acquisition trigger, and to stand still for approximately two seconds after reaching the end of the path.

ID	Gender (M/F)	Age (y.o.)	Weight (kg)	Height (cm)	Shoe Size (EU)
13	M	22	70	179	46
14	M	21	84	175	46
15	F	23	52	160	37
Avg.	67% (M)	22,0	68,7	171,3	43,0
Std.	-	1,0	16,0	10,0	5,2

Table 4.2: Summary of participants for free-living acquisitions.

- **Data personalization (Recording 3)**: See Section 4.1.
- **Free-living recording (Recording 4)**: Participant are recorded for two and a half hours while they perform ADL. During the time of the recording, participants are free to walk, sit, drink and perform any other kind of daily life activity except from taking elevators and exposing the sensors to water or unsafe conditions (e.g. high temperatures). Participants are asked to walk as much as possible and to perform, at least once, each one of the following activities:

- Rise from a chair and walk to another room.
- Walk to the kitchen and have a drink.
- Walk up and down a set of stairs.
- Walk outside.
- Walk up and down an inclined path.

4.2.2 Data preparation

Recorded raw data are prepared according to the same procedure applied to the in-lab dataset (see 4.1.2). At the end of the data preparation phase, a Matlab structure (`data.mat`) including recorded raw data and the INDIP standard was available for each Participant. At the end of the Free-Living acquisitions, a total amount of about 7 and a half hours of gait data had been gathered.

Chapter 5

Methods

In the present Chapter, the implementation of the MaLe and DeLe methods is presented. First, the organizational aspects of the work are outlined; then, the construction of the datasets is presented. Eventually, the training, validation and testing stages of the implemented algorithms are described[1].

5.1 Overview

As stated in the previous Chapters, the ultimate goal of the thesis work is represented by the estimation of STPs from the inertial signals recorded by the H-MIMU only. In particular, the development of the MaLe and DeLe methods has been oriented towards the accomplishment of two separate - however linked - functional tasks:

- **Task 1 - Temporal parameters estimation**: First, time steps at which gait events occur are detected; then, they are used to define 4 gait temporal parameters (step time, stride time, single support (SS) time and double support (DS) time).
- **Task 2 - Stride speed estimation**: Detected ICs are used to segment the strides. Then, a set of features of the acceleration norm signal during each stride are used to predict the value of the stride speed.

As a matter of fact, the two tasks are sequential i.e., the output of Task 1 (gait events) is supposed to be the input to Task 2. Depending on the task, a specific set of predictors i.e., time-series or features is given as input to a model. In turn,

[1]In the following Chapters, the term "validation" is used to denote the process of evaluating the performance of the model at training time to prevent overfitting, while the term "testing" refers to the assessment of the trained model performance when it is fed with unseen new data.

the model provides a response i.e., a prediction of the output class or value. As mentioned in Section 4.2, while for training and validation of the models only In-Lab data were employed, testing involved both the in-lab dataset and the Free-Living dataset. Since the models for the estimation of the temporal parameters and the ones for the estimation of the stride speed are intended to address different tasks, the datasets used for training them have different features - however both referring to raw data from the In-Lab dataset. Instead, at prediction, the models act sequentially to predict first gait events from raw data and secondarily stride speed from gait events.

5.2 Estimation of the temporal parameters

The first functional task is represented by the estimation of the gait events. Such task can be divided itself into three main functional sub-tasks.

- **SS/DS phase classification**
- **GED**
- **Temporal parameters definition**

In the following, each of these sub-tasks will be described in detail.

5.2.1 Single/double support phase classification

The first step in the process of estimating temporal parameters is represented by the sequence-to-sequence (Seq2Seq) classification[2] of each time step of the input time-series as belonging to the SS or DS phase. Actually, once that SS and SS time intervals are determined, gait events can be identified by exploiting the definition of single support and double support.
Similarly, gait events could have been determined by classifying input signals as belonging to the stance or swing phases of one of the two lower limbs; however, since the recording MIMU is positioned on the head, the dominant component of acceleration on the head (i.e., the vertical component) reflects both left and right lower limb activities; therefore, the tediousness of recognizing the swing phase of only one limb.
For the achievement of such task, two categories of DeLe models have been employed.

[2]Seq2Seq learning consists in training MaLe or DeLe models to convert sequences from one domain (e.g., sentences in Italian) to sequences in another domain (e.g. the same sentences translated to English) [98]. Seq2Seq classification means translating the input sequence into an output sequence of the same length whose time steps correspond to labels for the input sequence.

- **Long-short term memory (LSTM) neural network**: A LSTM network is a typology of recurrent neural network (RNN) that is able to learn long-term dependencies between time steps of sequence data. A LSTM network allows to input sequence data into a network, and make predictions based on the single time steps of the sequence data [99].
- **Temporal convolutional network (TCN)**: Convolutional neural networks can result in similar or even better performance with respect to RNN on typical sequence modeling tasks [100]. Among the benefits of using convolutional networks, there are better parallelism, better control over the receptive field size, better control of the memory footprint of the network during training, and more stable gradients [99]. Just like RNN, convolutional networks can operate on variable length input sequences and can be used to model Seq2Seq or sequence-to-one tasks. Among the several types of CNNs, TCN are particularly suited to be applied to Seq2Seq classification tasks; in addition, they have already been successfully applied to GED and STP estimation [57].

After defining the architecture of the TCN and LSTM networks (see Sub-subsection 5.2.1), 5 randomly-selected combinations of TRS and VS were employed to train and validate the models, resulting in 5 couples of TCN and LSTM trained networks.

Sequence modeling

The task for which the models have been trained consists in a Seq2Seq classification task; therefore, before deepening into the architectures and training algorithms for the developed models, a preliminary stage to introduce the nature of the sequence modeling task is recommended [101].
Consider an input sequence $x_0, ..., x_T$. The goal of a Seq2Seq task is the prediction of a corresponding output sequence $y_0, ..., y_T$. The underlying hypothesis is that output at time t only depends on inputs from time 0 to t rather than the future inputs (Equation 5.1):

$$\hat{y}_0, ..., \hat{y}_0 = f(x_0, ..., x_T) \tag{5.1}$$

Where $\hat{y}_t$ represents the predicted output at time step t and f represents the function that maps each input to its corresponding output (Equation 5.2):

$$f : X^{T+1} \longrightarrow Y^{T+1} \tag{5.2}$$

The goal of learning in a sequence modeling task consists in finding the function f that minimizes the expected loss L between the targets y and the predictions $\hat{y} = f(x)$ (Equation 5.3).

$$L(y_0, ..., y_T; f(x_0, ..., x_T)). \tag{5.3}$$

Dataset construction

In the following Subsection, the steps carried out for the construction of the dataset are described.
The sequential data available for training and validating the models consists in the tri-axial accelerations, angular velocities and magnetic field amplitudes recorded by the H-MIMU (Figure 4.5) for the 11 participants to the In-Lab acquisitions (see Table 4.1) walking under four different conditions (see 4.1).
Participants were initially divided into a construction set and a TS. The method used to perform such division is represented by block sampling, which leaves intact the integrity of one's gait data [3]. The construction set served to train and validate the models and it included Participants from 1 to 10 of the In-Lab dataset. The test set served for testing the trained model and assessing its generalization capabilities, and it was composed of Participant 11 only. The construction set was further divided into a TRS - for training the models - and a VS - for preventing data overfitting during training (Figure 5.2).

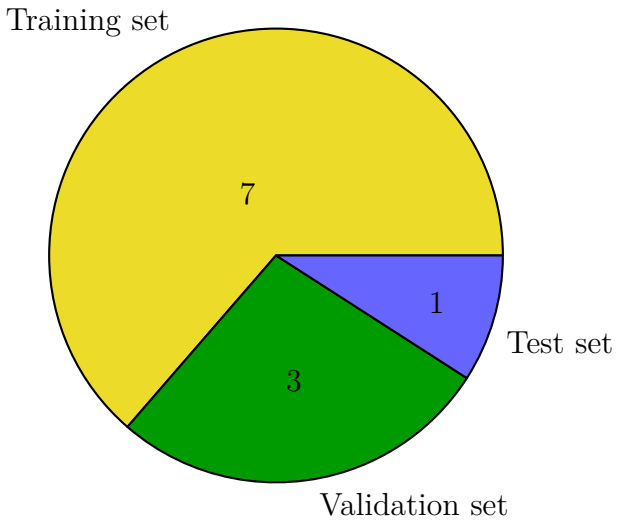


Figure 5.2: The pie chart shows the number of participants for each of the In-Lab TRS, VS and TS.

[3]Block sampling - otherwise known as *convenience* or *clustered* sampling - consists in sampling entire subgroups of data (e.g., data from the same center or patient) instead of selecting single individuals or elements. Other possible sampling methods include random sampling and stratified sampling.

Five couples of training set and validation set have been employed for constructing the models. In this way, the robustness of the trained models has been evaluated[4]. In addition, data of the 3 Participants to the Free-Living acquisitions were used to construct a second - much larger - TS. The operations for constructing the dataset that are described in the next paragraphs were carried out for all the Trials of the In-Lab dataset and for all the micro walking bout of the Free-Living dataset (see C).

Data loading As described in Section C.3, the Matlab structure `data.mat` stored in the *Results* folder of the current Participant folder contains the tri-axial accelerations, angular velocities and magnetic field strength for each MIMU, together with the INDIP standard. In particular, the accelerations and the angular velocities were employed for training the models. From these ones, a set of 8 predictors time-series has been derived, as similarly done by Gadaleta et al. [102]:

- **V acceleration**
- **ML acceleration**
- **AP acceleration**
- **Acceleration norm** (Equation 5.4)
- **V angular velocity**
- **ML angular velocity**
- **AP angular velocity**
- **Angular velocity norm** (Equation 5.5)

$$a_n = \sqrt{a_v^2 + a_{ml}^2 + a_{ap}^2} \qquad (5.4)$$

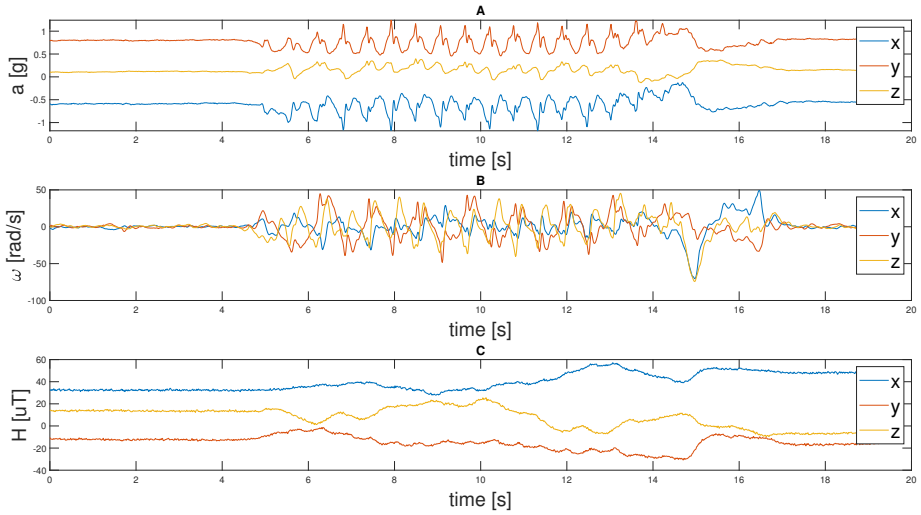
$$\omega_n = \sqrt{\omega_v^2 + \omega_{ml}^2 + \omega_{ap}^2} \qquad (5.5)$$

First of all, `data.mat` is uploaded to the Matlab workspace. In Matlab, predictor signals are organized as a 2D-array, where each row corresponds to one single time step and each column corresponds to one predictor.

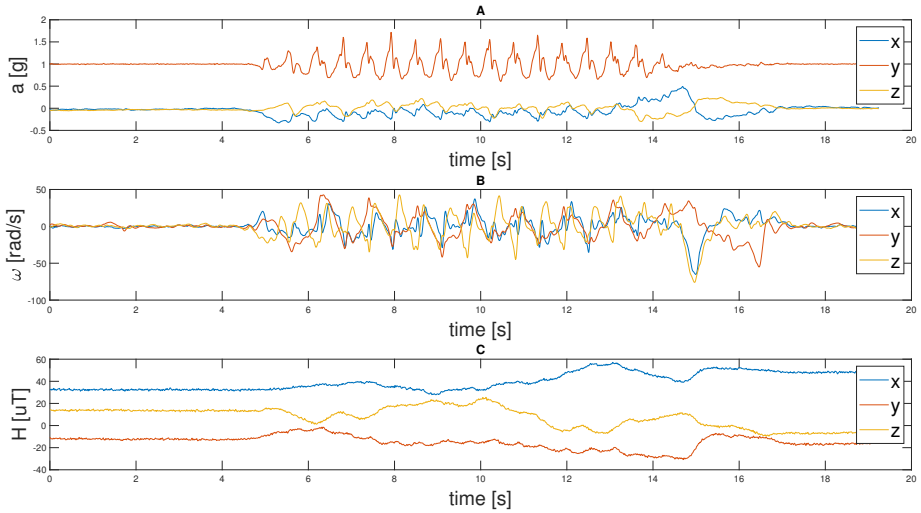
[4]A supervised method is generally considered robust when its performance are not significantly affected by a change in its training data.

Data checking During the acquisitions, the operators and/or the participant may commit technical issues linked to wrong sensor positioning or experiments execution, such as positioning the H-MIMU on the right side instead of the left side, walking a longer distance or forgetting to stop the acquisition as the participant reaches the end of the path. These issued have to be recognized and corrected. Having uploaded the signals, correction of the detected issues is performed through a Matlab custom function `manage_subjects.m` (see A.1).

H-MIMU re-orientation The H-MIMU is positioned on the left side of the head so that the sensor axes (local frame) are approximately parallel to the anatomical axes (global frame). However, due to the positioning variability, the sensor's axes may not perfectly correspond to the anatomical axes (Figure 3.5). Therefore, the recorded raw signals - which should refer to the anatomical axes - actually refer to a frame that is slightly rotated with respect to the anatomical one (Figure 4.3). To prevent data from being affected by such error, the H-MIMU is virtually rotated with respect to the ideal gravity by applying a rotation matrix to the recorded raw signals. The applied rotation matrix represents the relationship between the global and the local frame, and it is computed by processing the accelerations acquired during the Standing acquisition (Test 2 for the In-Lab dataset, Recording 2 for the Free-Living dataset). The rotation matrix is computed through the Matlab custom function `calc_R.m` and is applied to the accelerations and angular velocities through the Matlab custom function `reorient_head.m` (see Section A.3). Once that the H-MIMU is virtually rotated, signals are supposed to refer to the anatomical global frame (Figure 5.4).



(a)

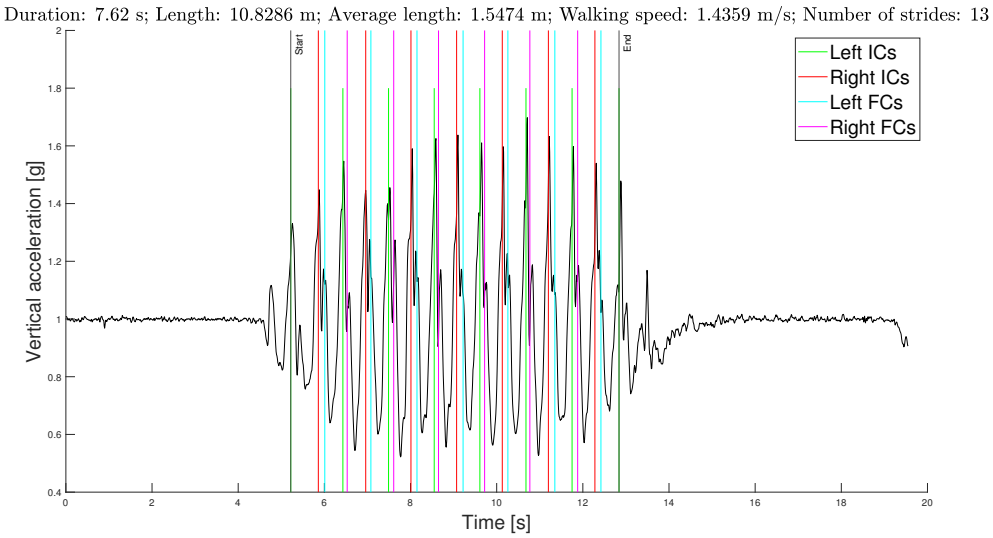


(b)

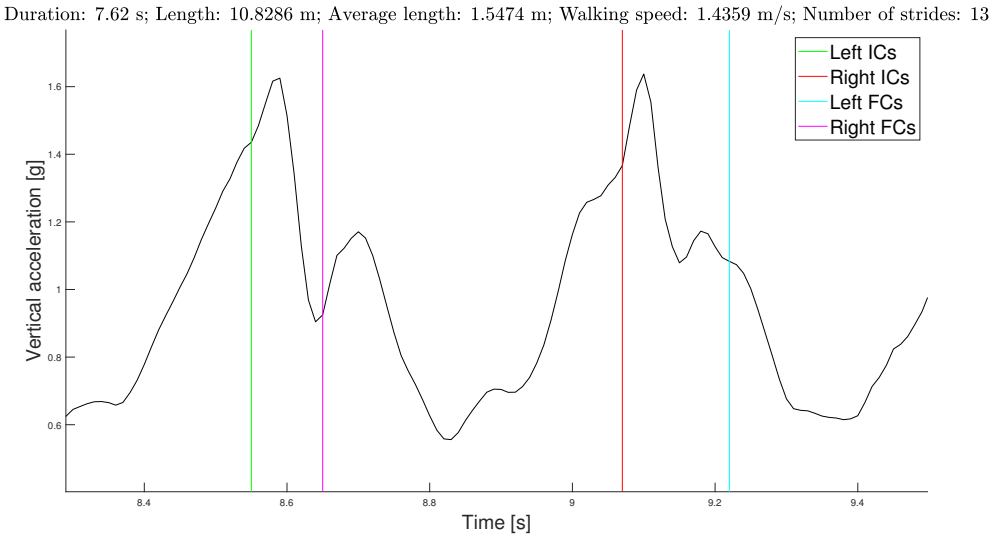
Figure 5.4: H-MIMU readings during the straight walking acquisition at comfortable speed (Test 5) for patient 2. A) Accelerometer readings; B) Gyroscope readings; C) Magnetometer readings. (a) Raw signals (b) Re-oriented signals. Notice the difference in the values of V-acc in the first seconds of the Trial between the raw and the reoriented signals: in the first case, V-acc values deviate from the expected ones (V-acc = 1 g), while they match the expected value after re-orientation. Moreover, the magnetic field strength is not affected by the H-MIMU virtual re-orientation, as the rotation matrix is applied to accelerations and angular velocities only.

Notice that, as the rotation matrix is computed from data recorded during a static acquisition, re-orientation fails to correct the effect of any misalignments that arise during the dynamic acquisitions (Tests 3 to 7).

Gait events from the INDIP standard Once that the H-MIMU has been virtually re-oriented, the INDIP standard is used as a reference for extracting the gait events referring to the current signals (see Section C.3). For each Trial, the time samples of all the left and right ICs and FCs are retrieved from the Matlab `data.mat` structure (Figure 5.5).



(a)



(b)

Figure 5.5: INDIP gait events for a Trial of straight walking at comfortable speed performed by Participant 2. The V-acc signal is plotted in black. Some relevant parameters extracted from the INDIP standard are reported, such as the duration and length of the continuous walking bout, the average stride length, the average walking speed and the total number of strides. (a) Full Trial (b) A portion of the Trial referring to a single stride.

Labeling Supervised classification methods require a response associated to predictors, which must be provided in the same format of the expected output. As mentioned in Subsection 5.2.1, the intended task to be performed by the models is a Seq2Seq classification task i.e., a classification task that consists in classifying each sample of the predictors as belonging to the SS phase class or to the DS phase class. Therefore, the response associated to predictors must be provided as a binary signal (*label*) that has the same number of samples of each predictor.
To decide whether a sample belongs to the SS or DS phase class, the information regarding the occurrence instants of gait events is exploited. In particular, a sample belongs to the SS phase if it occurs between a FC of one foot and the next IC of the same foot, while it belongs to the SS phase if it occurs between an IC of one foot and the next FC of the other foot.

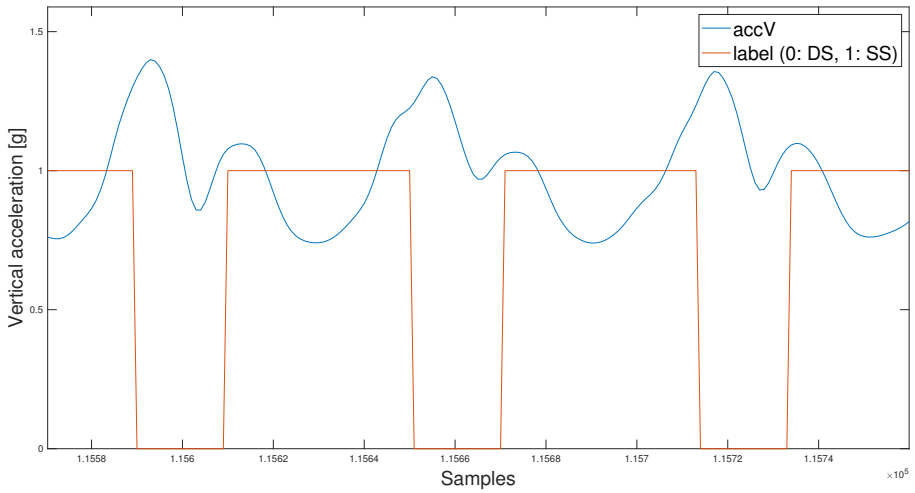


Figure 5.6: Filtered V-acceleration and label signal for a Trial of straight walking at comfortable speed performed by Participant 2.

The purpose of the Seq2Seq classification task consists in recognizing the SS and DS phases within gait. Therefore, only portions of the signal corresponding to walking must be included in the dataset. To determine such portions, predictors of each Trial are trimmed between the first and the last IC retrieved by the INDIP standard, so that eventually the label is a binary time-series, where 0 correspond to the DS phase class and 1 to the SS phase class (Figure 5.6). Similarly, data of the Free-Living dataset had to be properly "trimmed". In particular, non-gait and inclined gait portions of the signals were cut off.

Moving average filter Predictors are filtered with a 5-points moving average filter through the Matlab predefined function `smooth` (Figure 5.7).

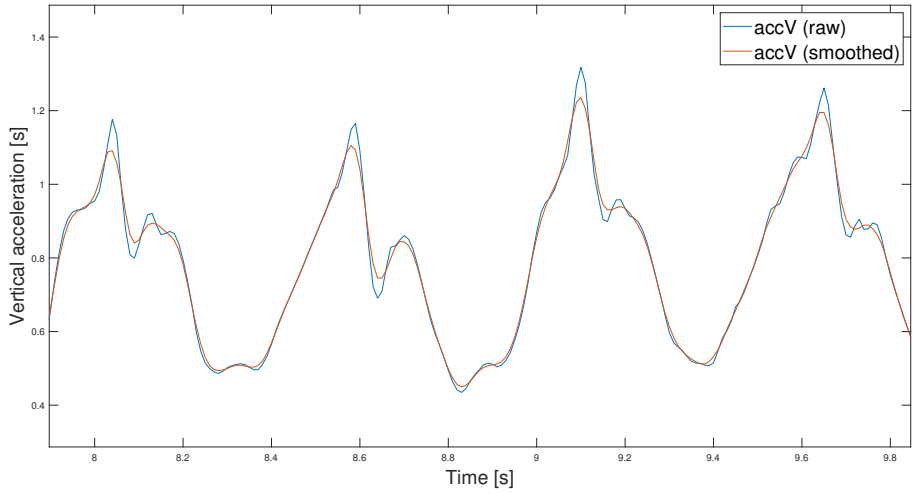


Figure 5.7: Raw and filtered V-acceleration. The effect of the moving average filter is a mild smoothing of the signals, which deletes rapid variations without compromising the morphology of the signal.

Such filtering is intended to reduce the effect of high-frequency noise and the rapid movements of the head which may occur during walking.

Concatenation The steps described in the previous paragraphs are carried out separately for each Trial of each participant. After the removal of non-gait and inclined gait portions from the recorded signals, about 40 minutes of effective gait had remained out of the initial 90 for the In-Lab dataset, while only 4 and a half hours of Free-Living data remained out of the initial 7 and a half.
Regarding the In-Lab signals, resulting predictors and responses are concatenated into a single array of 9 columns (1-8: predictors, 9: class) and 231745 rows, each of them corresponding to a time sample (Table 5.1).

Timestamp [s]	AP-acc	V-acc	ML-acc	accNorm	AP-gyr	V-gyr	ML-gyr	gyrNorm	Class
0	-0,081	1,115	-0,024	1,119	11,850	-5,390	-22,972	26,998	1
0,01	-0,078	1,139	-0,031	1,143	11,911	-5,951	-21,905	26,286	1
0,02	-0,073	1,164	-0,038	1,168	11,969	-6,503	-20,883	25,637	1
0,03	-0,067	1,189	-0,046	1,194	11,986	-7,039	-19,927	25,049	0
0,04	-0,060	1,214	-0,055	1,218	11,923	-7,553	-19,049	24,519	0
...	...	...	...	...	...	...	...	...	...
2317,43	-0,045	1,252	-0,072	1,257	11,428	-8,530	-17,534	23,592	0
2317,44	-0,037	1,263	-0,080	1,268	10,972	-9,010	-16,874	23,183	0

Table 5.1: Input array of concatenated predictors and responses. The dataset is composed of a total amount of 231745 data points. Since the value of the sampling frequency is 100 Hz, the dataset comprises approximately 39 minutes (2318 s) of walking.

Once that data of all the participants have been processed and concatenated, the resulting dataset is block-sampled into a TRS, a VS and a TS, as described in Sub-subsection 5.2.1. At the end of the dataset construction process, a total amount of 3754 strides were segmented from the In-Lab dataset. Of those, 3426 belong to the construction set, while the remaining 328 to the TS (Table 5.2). Regarding the Free-Living TS, a total amount of 26558 strides were segmented at the end of the dataset construction process.

	Total	**Construction set (TRS/VS≈ 70/30%)**	**Test set**
Number of strides	3754	3426	328

Table 5.2: Summary table reporting the number of segmented strides for the In-Lab dataset. Regarding the construction set, 7 patients out of the total 10 were randomly assigned to the TRS for each TRS/VS combination; therefore, approximately 70% of the strides were assigned to the TRS, while data referring to the remaining strides ended in the VS.

Network architectures

In the following Sub-subsection, a brief introduction on the architectures of DeLe and MaLe models for Seq2Seq classification is provided. Then, the architectures of the two models developed for the SS/DS phases classification will be described in detail. For a refresher on DeLe architectures, see Appendix D. The Matlab code used for defining the architecture of the networks is available in Appendix B.

Long-short term memory (LSTM) network The Seq2Seq classification task for the recognition of the SS and DS phases was initially addressed by developing

a LSTM RNN.
In the present work, the input is represented by a sequence of size 8 (the number of predictors). The LSTM layer has 200 hidden units (hidden size = 200), and gives the full sequence as output. The LSTM layer is followed by a fully connected layer (FCL), a softmax layer and a classification layer (Figure 5.8).

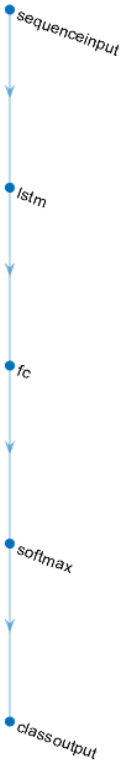


Figure 5.8: Layer graph illustrating the architecture of the developed LSTM network. The information flows from the top of the graph to its bottom. For a deeper explanation of each of the nodes of the graph, see Subsection D.3.2. The hidden size of the LSTM layer is 200, while the size of the FCL is 2.

In the following, the layers of the network will be described in detail.

- **Sequence input layer**: The sequence input layer inputs sequence data to the network.
  - *Input size*: **8**. The sequence input layer inputs 8 sequences to the network (see Paragraph 5.2.1.
  - *Minimum sequence length of input data*: **1**. Since the LSTM network can process input sequences of every length, such value does not affect the

network performance. Instead, a CNN would require input sequences to have at least as many samples as the size of the filters for convolution.

- **LSTM layer**: The core component of an LSTM network is represented by the LSTM layer. In the following, a description of the hyperparameters and learnable parameters for the trained LSTM layer - together with their final values - is provided[5].
  - **Hyperparameters**:
    - *Input size*: **8**.
    - *Hidden size*: **200**. The hidden size i.e., the number of hidden units corresponds to the quantity of information remembered between time steps i.e., the hidden state. Such amount of information can refer to all previous time steps, no matter the sequence length. A large hidden size increases the risk of overfitting; then, its value typically ranges from a few dozen to a few thousand [99].
    - *State activation function*: **Hyperbolic tangent (tanh)**. Activation function to update the cell and hidden state.
    - *Gate activation function*: **Sigmoid**. Activation function to apply to the gates.
  - **Learnable parameters**:
    - *Input weights size*: **800 x 8**.
    - *Recurrent weights size*: **800 x 200**.
    - *Bias size*: **800**.
  - **Weights and bias initialization methods**:
    - *Input weights*: **Glorot initializer**[6]. The Glorot initialization method independently samples from a uniform distribution with mean μ and variance ν given as follow [103]:

      $$\mu = 0 \qquad \nu = \frac{2}{InS + 4HS}$$

      Where InS and HS denote the input size and the hidden size, respectively.

[5]In the field of MaLe and DeLe, learnable parameters are those parameters that are automatically adjusted during training i.e., weights and bias. Hyperparameters are those parameters of the model/algorithm that are set at the beginning of training. Hyperparameters are not adjusted during training; however, they can be automatically or manually optimized during the tuning phase to improve the model's performance.

[6]Also known as Xavier initializer [99].

  - ∗ *Recurrent weights*: **Orthogonal initializer**. The orthogonal initialization method initializes the recurrent weights R with Q, the orthogonal matrix resulting from the QR decomposition of $Z = QR$ for a random matrix Z sampled from a unit normal distribution [104].
  - ∗ *Bias*: **"Unit-forget-gate" initializer**. The "unit-forget-gate" initialization method initializes the bias of the forget gate with ones and the remaining biases with zeros.

- **FCL**: A FCL multiplies the layer input by a weight matrix W and then adds a bias vector b. All the neurons in a FCL are connected to all the neurons in the previous layer. If the FCL is fed with a sequence - as it happens in an LSTM network - then the FCL works separately on each time step. For instance, if the size of the array X given as input to the FCL is $D \times N \times S$, then the FCL outputs an array Z of size $O \times N \times S$, where O is the output size for the FCL. At time step t, the output Z corresponds to the entry $WX_t + b$, where X_t represents time step t of X.
  - *Output size*: **2**. The number of classes for the Seq2Seq classification task determines the output size of the FCL.
  - **Weights and bias initialization methods**:
    - ∗ *Input weights*: **Glorot initializer**. See the weights and bias initialization methods item for the LSTM layer.
    - ∗ *Bias*: **"Zeros" initializer**. The "zero" initialization method initializes the bias of the FCL with zeros.

- **Softmax layer**: The final FCL is typically followed by a softmax layer and then a classification layer. A softmax layer applies a *softmax function* - otherwise known as *normalized exponential* - to the layer input [105]:

$$y_r(x) = \frac{e^{a_r(x)}}{\sum_{j=1}^{k} e^{a_r(x)}}$$

  Where:

$$0 \leq y_r \leq 1$$
$$\sum_{j=1}^{k} y_j = 1$$

- **Classification layer**: A classification layer calculates the cross-entropy loss (CEL) for classification tasks with mutually exclusive classes. The layer determines the number of classes from the output size of the previous layer,

which is typically represented by a softmax layer. For instance, to enter the number of classes K of the network for the target task, a FCL with output size K and a softmax layer should be inserted before the classification layer. During training, output values from the softmax function are processed using the CEL function for a 1-of-K coding scheme to assign every input to one of the K mutually exclusive classes [105]:

$$CEL = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{K} w_i t_{ni} \ln y_{ni}$$

Where N is the number of samples, K is the number of classes, w_i is the weight for class i, t_{ni} is the indicator that the n^th sample belongs to the i^th class, and y_{ni} is the output for sample n for class i, which in this case, is the value from the softmax function. As a matter of fact, y_{ni} represents the probability that the network associates the n^th input with class i.

Temporal Convolutional Network (TCN) In addition to the LSTM network, a TCN network has been developed to compare performance.
In the present work, the input is represented by a sequence of size 8 (the number of predictors), as for the LSTM network. The network is composed by a TCN core of 4 residual blocks followed by a FCL, a softmax layer and a classification layer.

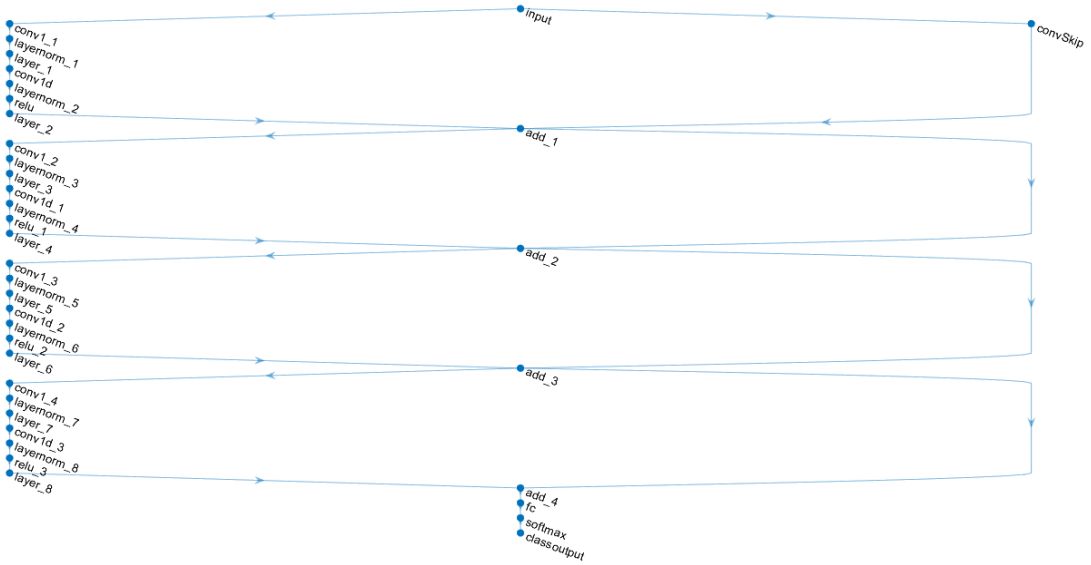


Figure 5.9: Layer graph illustrating the architecture of the developed TCN. The information flows from the top of the graph to its bottom. For a deeper explanation of each of the nodes of the graph, see Subsection D.4.4.

In the following, the layers and the hyperparameters values specified for defining the architecture of the TCN are presented (Figure 5.9).

- **TCN layer**: The core of the developed network is represented by 4 residual blocks, each of them containing two sets of dilated causal convolution layers with the same dilation factor, followed by weight normalization, rectified linear unit (ReLu) activation function and spatial dropout layers (Figure 5.9). See Subsection D.4.4 for further information about the architecture of residual blocks.
  - *Number of residual blocks*: **4**.
  - *Dilation base*: **2**. Specifying a dilation base of 2 means that the dilation factor of each block is twice the dilation factor of the previous block (the first residual block has dilation factor equal to 1).
  - *Number of filters*: **64**. The number of filters corresponds to the number of kernel vectors used for the 1D convolution.
  - *Filter size*: **5**. The kernels are mono-dimensional vectors of weights of size 5
  - *Dropout factor*: **0.005**. During training, the dropout layer randomly sets input elements to zero with probability equal to the dropout factor. Such operation has proved to change the underlying network architecture from one iteration to the other, eventually helping to prevent overfitting [106], [107]. A higher dropout factor yields to more elements being dropped during training. During prediction, the output of the dropout layer is equal to its input.
- **FCL**: Same settings of the LSTM network.
- **Softmax layer**: Same settings of the LSTM network.
- **Classification layer**: Same settings of the LSTM network.

Training and validation

In the following Sub-subsection, training and validation of the developed models will be described in detail. As described in Sub-subsection 5.2.1, the training stage of each model architecture was repeated 5 times with 5 different combinations of TRS and VS to evaluate the robustness of the model. For each TRS/VS combination, a random hold-out validation scheme was observed; meaning that 7 of the 10 subjects in the construction set were randomly assigned to the TRS, while data from the remaining 3 subjects were used to construct the VS. Therefore, a total number of 10 DeLe models were trained and validated - 5 TCNs and 5 LSTM networks (Figure 5.10).

Training
In the present work, the TCN and the LSTM networks have been trained according to the following training options.

- *Solver (learning rule)*: **Adaptive moment estimation (ADAM)**. The ADAM solver, similarly to other BP algorithms such as the root mean square propagation (RMSProp), attempts to enhance network training by adopting learning rates that differ by parameter and is able to automatically adapt to the loss optimization function [99]. The ADAM solver keeps an element-wise moving average of both the parameter gradients (m) and their squared values (v):

$$m_l = \beta_1 m_{l-1} + (1 - \beta_1)\nabla E(\theta_l)$$
$$v_l = \beta_2 v_{l-1} + (1 - \beta_2)\nabla [E(\theta_l)]^2$$

  Where l is the iteration number, θ is the parameter vector, ∇E(θ) is the loss function and β_1 and β_2 are respectively the gradient and squared gradient decay factors. Then, the network parameters are updated accordingly to the computed moving averages [108]:

$$\theta_{l+1} = \theta_l - \frac{\alpha m_l}{\sqrt{v_l} + \epsilon}$$

  Where α is the learning rate and ϵ is the denominator offset. In the following, the parameters values for the ADAM algorithm employed for training the LSTM network are listed:

  - *Gradient decay factor (β_1)*: **0.9**.
  - *Squared gradient decay factor (β_2)*: **0.999**. Such value corresponds to an averaging length of 1000 parameters updates.
  - *Denominator offset (ϵ)*: $\mathbf{1.5e-10}$. Such parameter avoids division by zero in the network parameter updates.
  - *Learning rate (α)*: **0.001**. Such value is a trade off between the need for a short training time (achieved with high learning rates) and stability and optimization of the results (high learning rates can result in sub-optimal outcomes or provoke divergence). The learning rate is not dropped during training.

- *Number of epochs*: **60**. Maximum number of epochs to use for training. An epoch is the complete passage of the training algorithm over the whole training set. An epoch comprises several iterations: an iteration represents one step taken into the gradient descent algorithm towards the minimization of the loss function using a mini-batch of training set examples [99].

- *Mini-batch size*: A mini-batch is a subset of the training set that is used to compute the gradient of the loss function and update the parameters. The sizes of the mini-bacthes for the LSTM and TCN networks were set to **128** and **1**, respectively.

- *Shuffle*: **Never**. Since the input predictors that are fed to the model are time-series, shuffling should be avoided to prevent time-dependencies between samples at different time steps to be lost.

- L_2 *regularization coefficient*: **0.0001**. L_2 regularization consists in adding a weight regularization term - otherwise known as *weight decay* - to the the loss function $E(\theta)$ to reduce overfitting [105][109]. The regularized loss function $E_R(\theta)$ takes the following form:

  $$E_R(\theta) = E(\theta) + \lambda\Omega(w)$$

  Where w is the weight vector, λ is the regularization coefficient and $\Omega(w)$ is the energy density of the weight vector:

  $$\Omega(w) = \frac{1}{2}w^T w$$

- *Gradient clipping*: **Yes**. Exponentially increasing gradients denote an unstable training that can diverge within relatively few iterations. Such "gradient explosion" is typically accompanied by a training loss that either diverges or converges. Gradient clipping helps avoid gradient explosion by making training stable at higher learning rates and even when outliers are present [110]. Gradient clipping allows faster training and does not usually affect accuracy.

  - *Gradient threshold*: **2**.
  - *Gradient threshold method*: $\mathbf{L_2}$ **norm**. If the L_2 norm of the gradient of a trainable parameter is higher than the gradient threshold, the gradient is scaled to match the L_2 norm to the gradient threshold.

- *Sequence options*: LSTM networks accept input sequences with varying lengths. During training, passed sequences are padded, truncated, or split so that all the sequences in every mini-batch have the same length.

  - *Sequence length*: Sequences in each mini-batch are padded to the length of the **longest** sequence. This strategy does not throw away any data; however, it can bring noise to the network.

– *Padding direction*: **Right**. The sequences in the mini-batch start at the same time step and are padded at their end. Such strategy is convenient when training sequence-to-sequence networks, as padding in the first time steps may affect the prediction for the earlier time steps.

– *Padding value*: **0**. Sequences are zero-padded.

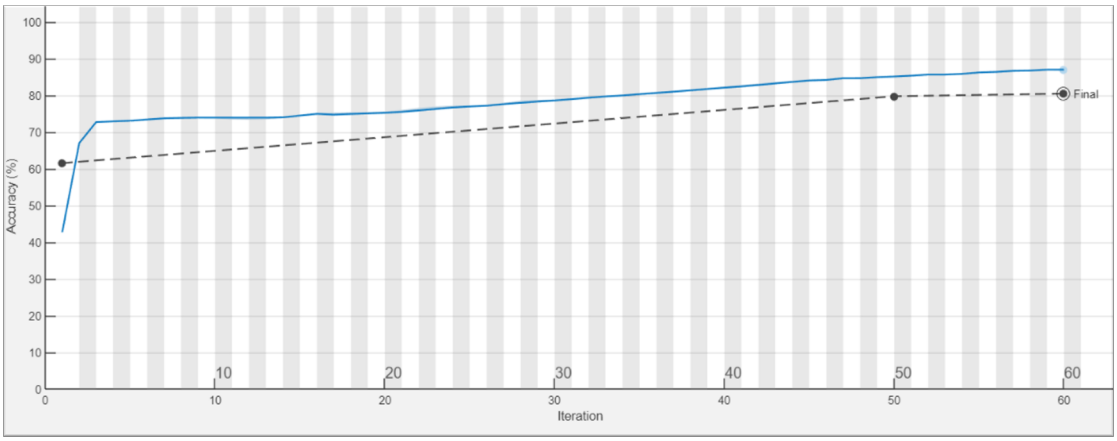


Figure 5.10: Training plot for the first trained LSTM network. Values of TRS accuracy on the TRS (blue) and VS (dotted black) for each training epoch are shown.

Validation

As mentioned in Sub-subsection 5.2.1, for each TRS/VS combination, data of 3 randomly selected patients have been employed to validate the networks during training, in order to prevent overfitting.
In the following, the adopted validation settings are described.

- *Validation frequency*: **50**. The validation frequency is the number of iterations between evaluations of validation metrics.
- *Validation patience*: **Infinite**. Validation patience represents the number of times that the loss value on the validation set can be larger than or equal to the previously smallest loss before that the network training is stopped.

5.2.2 Gait events detection

The second step in the process of estimating temporal parameters is represented by GED. Actually, once that gait data are segmented into the SS and DS phases, information related to the phases' onset and end can be leveraged to extract the gait events.

Edge detection

As described in Subsection 2.2.3, the SS and DS phases are delimited by specific gait events (Table 2.1):

- SS phase: portion of the gait cycle comprised between a FC of one foot and the next IC of the opposite foot.
- DS phase: portion of the gait cycle comprised between an IC of one foot and the next FC of the opposite foot.

Such definitions can be exploited to predict the time steps at which gait events occur from the SS/DS label response predicted by the models. Since the predicted label signal is a sequence of zeros and ones, edges of the signal can be detected by simple differentiation. Since the label is a binary sequence of zeros and ones, differential results in a sequence of ones and minus ones, which correspond respectively to falling and rising edges in the predicted response label. i.e., ICs and FCs (Figure 5.11).

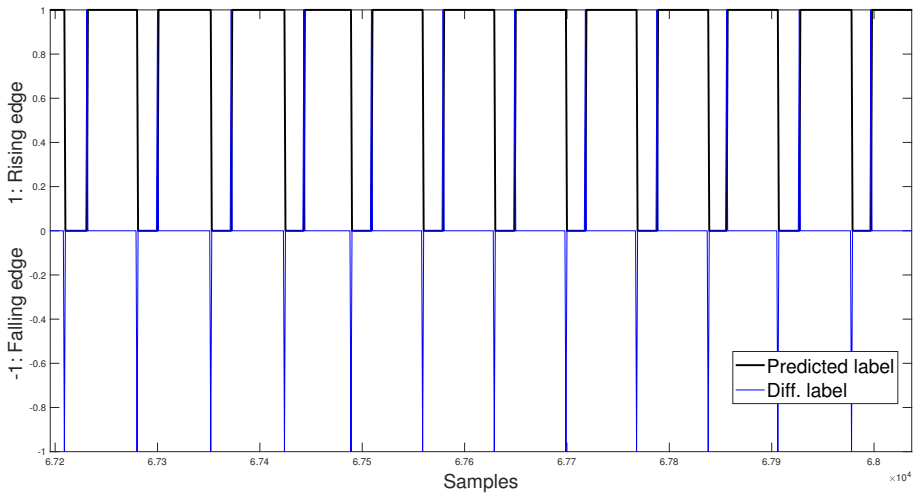


Figure 5.11: A portion of the predicted response label and the corresponding differentiated label. The differentiation operation has been carried out through the use of the Matlab built-in function `diff`. Notice that, as `diff` subtracts each element of a signal from the next one, the edges of the predicted response label are predicted one sample before the effective transition i.e., the `diff` function shifts the edges to the left by one time step.

Such operation is performed for both the predicted and the target response signals, so that a comparison between the target and predicted gait events becomes

possible.

Pairing of gait events

In general, time steps at which a predicted gait event occurs may differ from the time step of the corresponding target gait event. For computing most of the performance metrics (see Section 5.4), predicted gait events must be unequivocally paired to target gait events. To achieve such coupling, an empirical algorithm has been adopted. Consider the i^th gait event predicted by the INDIP standard IC_{INDIP_i} and the j^th gait event predicted by the H-MIMU IC_{H-MIMU_j}:

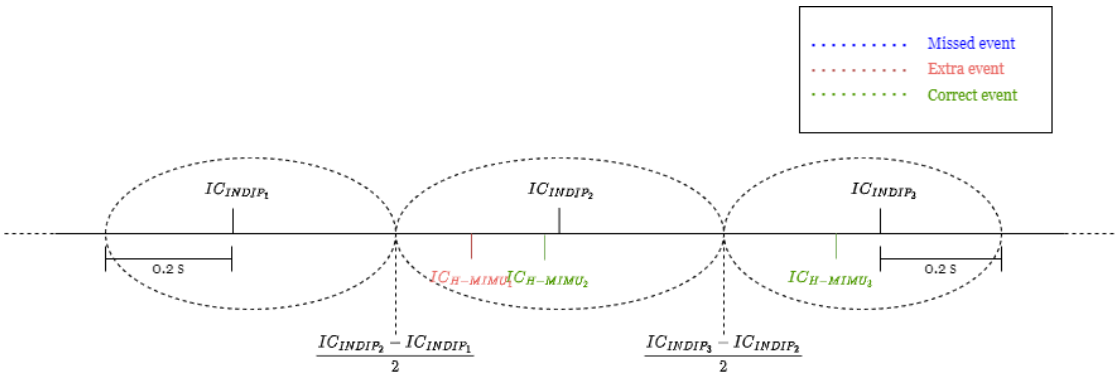


Figure 5.12: Schematic illustration of the method for determining the correspondence between predicted and target gait events. Notice that, for each gait event predicted by the GS, only one missed event is possible, while multiple extra events can occur.

1. A time neighborhood centered around IC_{INDIP_i} is considered. For most of the gait events, the neighborhood spans from half the distance between IC_{INDIP_i} and $IC_{INDIP_{i-1}}$ to half the distance between IC_{INDIP_i} and $IC_{INDIP_{i+1}}$ (Figure 5.12). A reasonable interval of 0.2 s before and after the first and last gait events of the sequence is considered for the first and the last INDIP standard gait events of the sequence, respectively.

2. gait event predicted by the INDIP standard IC_{H-MIMU_j} is associated to gait event predicted by the INDIP standard IC_{INDIP_i} if it falls in the defined neighborhood.

   - If no H-MIMU events fall within the defined neighborhood, a *missed event* is counted.
   - If multiple H-MIMU events fall within the defined neighborhood, the closest to the INDIP standard gait event is associated to it, while the others are counted as *extra events*.

A Matlab custom function `COMPARISON_MIMU_SP.m` is employed to perform such association. `COMPARISON_MIMU_SP.m` returns two vectors of the same lengths containing time steps at which coupled gait events occur. Missed events are reported as not a number (NaN) to preserve the left/right foot alternation. `COMPARISON_MIMU_SP.m` also returns the time steps at which extra and missed events predicted by the H-MIMU occur.

5.2.3 Temporal parameters definition

After detection and pairing of correspondent gait events (Figure 5.13), temporal parameters can be estimated.

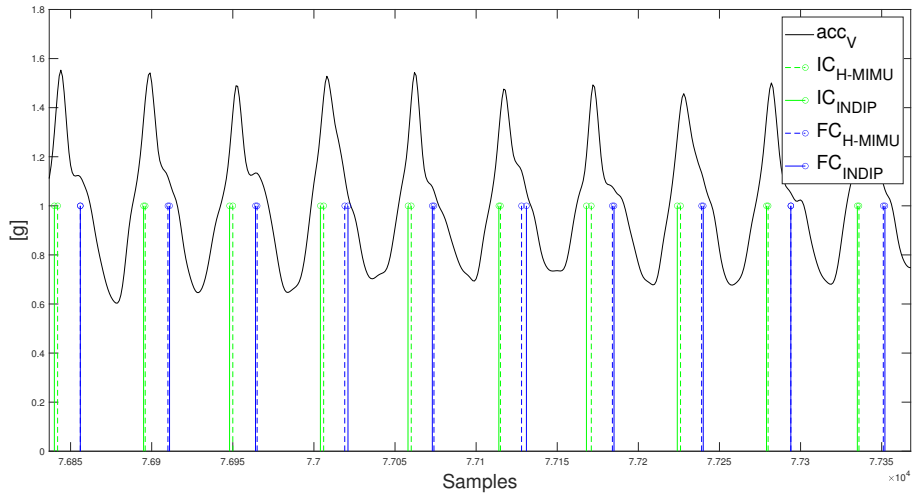


Figure 5.13: Comparison between predicted and target gait events plotted on the V-acc signal. acc_V: V-acc; IC_{H-MIMU}, FC_{H-MIMU}: ICs and FCs predicted by the H-MIMU, respectively; IC_{INDIP}, FC_{INDIP}: ICs and FCs predicted by the INDIP standard, respectively. Notice the presence of missed events.

Time steps at which gait events occur are leveraged to define the following temporal parameters:

- **Stride time**: Given an IC occurring at a generic time step, the time duration of the stride starting at that time step is given as follows:

$$Stride\ time = IC_i - IC_{i-2} \qquad [s]$$

  Where IC_i denotes the i^{th} IC.

- **Step time**: Given an IC occurring at a generic time step, the time duration of the step starting at that time step is given as follows:

$$Step\ time = IC_i - IC_{i-1} \qquad [s]$$

  Where IC_i denotes the i^{th} IC.

- **SS phase time**: Given a FC followed by an IC occurring at two generic time steps, the time duration of the SS phase starting at the time step of the FC is given as follows:

$$SS\ time = IC_i - FC_i \qquad [s]$$

  Where IC_i and FC_i denote the i^{th} IC and FC, respectively.

- **DS phase time**: Given a IC followed by a FC occurring at two generic time steps, the time duration of the DS phase starting at the time step of the IC is given as follows:

$$DS\ time = FC_i - IC_i \qquad [s]$$

  Where IC_i and FC_i denote the i^{th} IC and FC, respectively.

Notice that no distinction is made between left and right strides/steps, as the purpose of GED consists in the identification of gait events regardless of their side[7]. See Table 2.1 for a more detailed description of the aforementioned parameters.

5.3 Estimation of the stride speed

The second main block of the work consists in the algorithms for the estimation of the stride speed. At prediction, such algorithms are intended to be fed with strides segmented through detected gait events and predict the average WS for each stride - otherwise known as *stride speed* (see Table 2.1). At training, gait events retrieved from the INDIP standard are exploited for segmenting the strides that constitute the algorithm's construction set instead.
Five different MaLe regression methods have been trained to predict the value of the stride speed from a set of features referred to segmented strides:

- **gaussian process regression (GPR)**: Four GPR algorithms based on different kernels (exponential, matern 5/2, rational quadratic, quadratic exponential) trained for predicting the stride speed value from a set of predictors

[7]While some sensors configurations intrinsically provide information about the corresponding side to detected gait events (e.g., shanks, feet), H-MIMUs do not provide such information, as the morphology of the H-MIMU signal is scarcely affected by bio-mechanical changes in the ML direction. However, the task of side determination can be addressed in a second moment.

derived from the tri-axial acceleration norm. GPR models are characterized by hard interpretability [99]. An approach based on a GPR model for the estimation of the stride speed from H-MIMU data has already been proposed by Zihajehzadeh et al. [12].

- **Linear SVM**: a SVM algorithm based on a linear kernel trained for predicting the stride speed value from a set of predictors derived from the tri-axial acceleration norm. Linear SVMs are characterized by easy interpretability [99].

All the five MaLe models have been trained and validated with the Matlab `Regression Learner` application. In the next Subsections, every step to the development of the models is described - from the construction of the dataset to the training and validation of models.

5.3.1 Dataset construction

The same data from the In-Lab dataset used to train and validate the DeLe models are employed to train and validate the MaLe models for the prediction of the stride speed.
Raw inertial data are combined to the gait events retrieved from the INDIP standard to segment the gait cycles within the signals i.e., to extract portions of the signal belonging to the same gait cycle. Then, segmented signals are processed to compute a set of 136 features in the time and frequency domains (Figure 5.14).

Data loading

As described in Section C.3, the Matlab structure `data.mat` stored in the *Results* folder of the current Participant folder (see Subsection 4.1.2) contains the tri-axial accelerations, angular velocities and magnetic field strength for each MIMU, together with the INDIP standard.
In particular, V-, AP- and ML-accelerations of each Trial - together with anthropometric data of the respective subject - are employed to derive the predictors used to train the MaLe models.
For each subject, `data.mat` is uploaded to the Matlab workspace. Then, a custom Matlab function `retrieve_path_info.m` is called to read the Operator Table associated to each subject (see Section C.2) and extract the following information about the subject's anthropometry:

- **Height** (**cm**)
- **Weight** (**kg**)

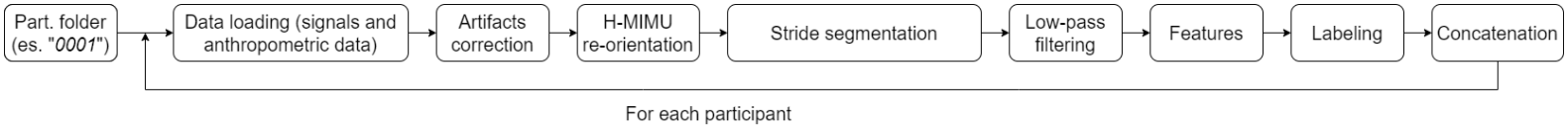


Figure 5.14: Operative workflow for the construction of the dataset for the stride speed estimation.

Height and weight are used as raw predictors by the MaLe models. Since height and weight can affect one's way of walking, the morphology of the underlying inertial signals and eventually the value of the WS can reflect their influence.

Correction of the acquisition artifacts

See page 68.

H-MIMU re-orientation

See page 68.

Stride segmentation

For each Trial, left and right ICs are retrieved from the INDIP standard and used to determine the end and the beginning of each stride, defined as the time interval between an IC of one foot and the next IC of the same foot (Figure 5.15).

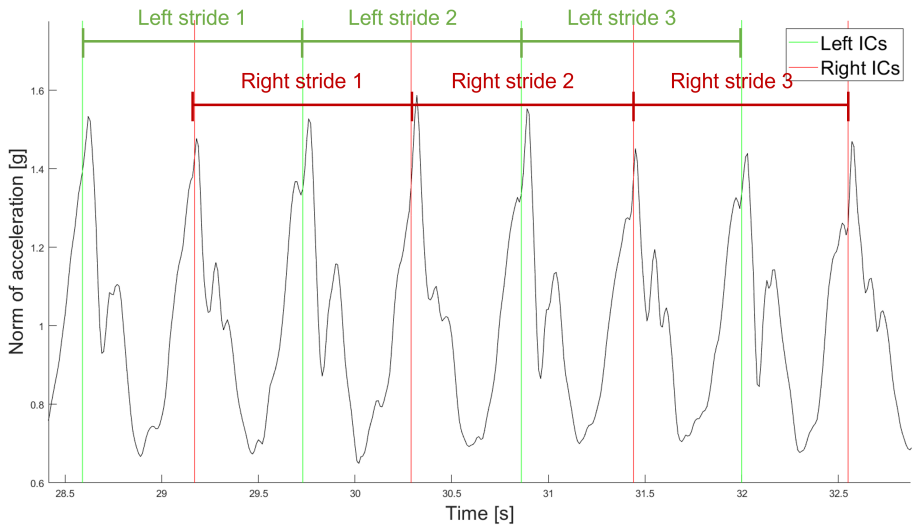


Figure 5.15: A portion of the acceleration norm signal (Participant 2, Test 7, Trial 2) segmented into three left strides and three right strides.

At the end of the stride segmentation process, a total amount of 4154 strides is segmented from the In-Lab acquisitions. Of those, 366 are given to the TS (Participant 11), while the remaining 3788 constitute the construction set.

Low-pass filtering

For each stride, the acceleration norm signal is calculated from the tri-axial acceleration over the stride duration (Equation 5.4) through the Matlab built-in function `vecnorm`.
Since changes in the H-MIMU acceleration norm during regular gait typically involve frequencies lower than 5 Hz, the acceleration norm is filtered with a low-pass filter to prevent high frequency noise from corrupting the signal. To avoid altering the informative content of the signal in the pass band, it is filtered with a digital infinite impulse response (IIR) Butterworth filter[8] designed in order to comply to the following constraints:

- **Pass band**: 0 to 5 Hz.
- **Stop band**: 5 to 10 Hz.
- **Pass band ripple**: 3 dB.
- **Stop band attenuation**: 60 dB

The filter hyperparameters were optimized through the Matlab built-in function `buttord` in order to find the lowest order Butterworth filter that complied to the required constraints [111]. As a result, a Butterworth low-pass filter with the following characteristics was implemented (Figure 5.16):

[8]Butterworth filters show a magnitude response which is maximally flat in the pass band and monotonic overall. On the other hand, Butterworth filters have a decreased roll-off steepness. Other filters (e.g., elliptic and Chebyshev filters) typically achieve steeper roll-off for a given filter order.

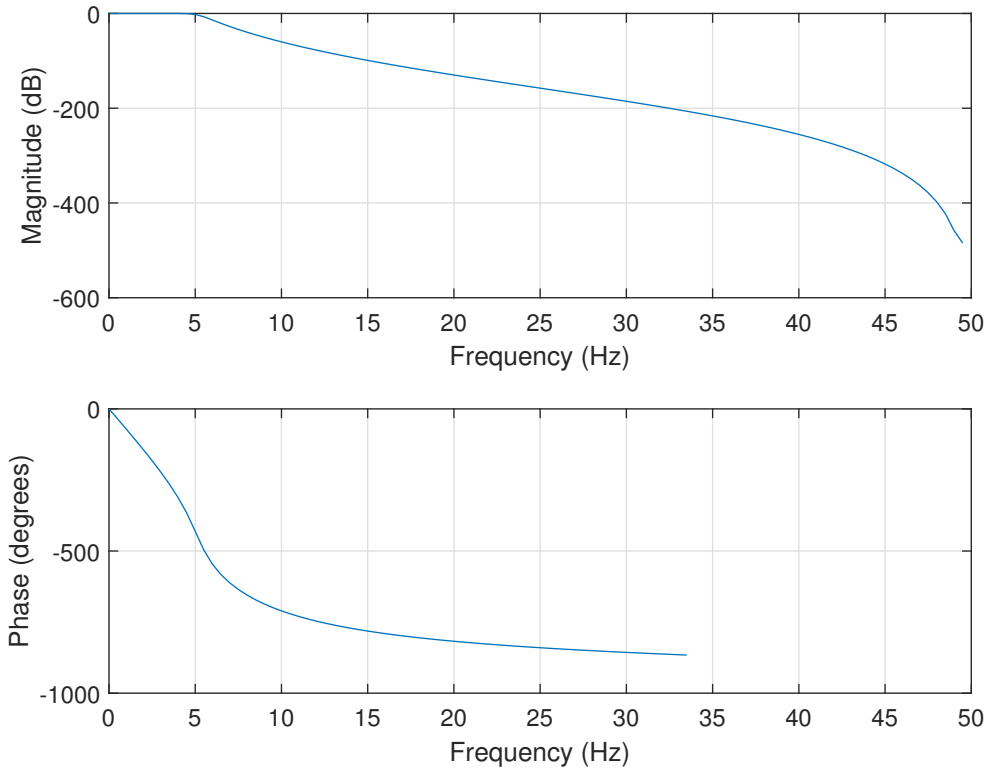


Figure 5.16: Magnitude and phase Bode diagrams referring to the Butterworth digital filter for the acceleration norm. As required, the stop band goes from 5 Hz to 10 Hz with less than 3 dB attenuation in the pass-band and an attained attenuation of 60 dB at the end of the stop band.

- **Order**: 4.
- **Cut-off frequency**: 5.14 Hz.

The transfer function coefficients of the filter are computed through the Matlab built-in function `butter`, while the signals are filtered through the Matlab built-in function `filtfilt`[9] (Figure 5.17).

[9] `filtfilt` carries out a zero-phase digital filtering by processing the input signal in both the forward and reverse directions [99].

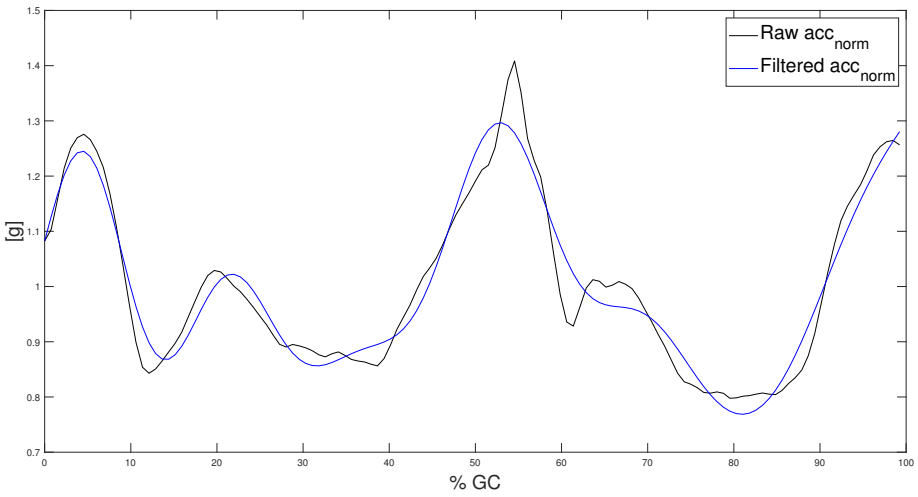


Figure 5.17: Raw and filtered acceleration norm signals acc_{norm} over a stride. x-coordinate is reported as % GC.

Low-pass filtering is a quasi-mandatory step when dealing with human accelerations, as basically every conscious or unconscious gesture affects the values and morphology of the acceleration signal. Filtering is particularly relevant for accelerations recorded by H-MIMUs, as during gait the head is subject to nods and rotations that are uncorrelated to the biomechanics of walking.

Stride features

For each stride, filtered acceleration norm acc_{norm} is used to derive a set of 136 features in the time and frequency domains, as proposed by Zihajehzadeh et al. [12]. Median, mode, signal magnitude area (SMA), energy, zero-crossing rate and mean absolute value (MAV) are computed in the time domain, while magnitudes of 128 fast Fourier transform (FFT) coefficients are computed in the frequency domain (Table 5.3).

Domain	Parameter	Unit	Definition
Time (6)	Median	g	m (see Equation 5.6).
	Mode	g	Most repeated value of acc_{norm}
	SMA	g	$\sum_{i=1}^{N}\|(acc_{norm})_i\|$
	Energy	g^2	$\sum_{i=1}^{N}(acc_{norm})_i^2$
	Zero-crossing rate	-	Number of times that acc_{norm} crosses the average value-level.
	MAV	g	$\frac{1}{N}\sum_{i=1}^{N}\|(acc_{norm})_i\|$
Frequency (128)	FFT coefficients	g	$Y(k),\ k = [1, 128]$ (see Equation 5.7)
Anthropometric (2)	Weight	kg	Participant's weight.
	Height	m	Participant's height.

Table 5.3: Predictors used by the MaLe models to predict the stride speed value for each stride. acc_{norm}: samples of the acceleration norm for the current stride; N: number of samples of acc_{norm}; m: median; $Y(k)$: FFT coefficients. The Matlab built-in functions `median`; `mode`; `mean`; `sum`, `abs`, `zerocrossrate`, `fft` - or a combination of them - were employed to compute the time and frequency domains features.

$$m = \begin{cases} (acc_{norm})_{\frac{N+1}{2}}, & \text{if N is odd} \\ \frac{(acc_{norm})_{\frac{N}{2}} + (acc_{norm})_{\frac{N}{2}+1}}{2}, & \text{otherwise} \end{cases} \tag{5.6}$$

$$Y(k) = \left| \sum_{i=1}^{N} (acc_{norm})_i e^{\frac{-2\pi \mathrm{i}}{N} - (i-1)(k-1)} \right| \tag{5.7}$$

Such features are assumed to be predictors for the stride speed. In Matlab, predictors are organized within a 2D-array, where each row corresponds to one single stride and each column corresponds to one feature of the respective stride. In addition to the time and frequency domains features, weight and height are added to the predictors array as additional anthropometric features.

Labeling

Supervised regression methods require a response associated to predictors, which must be provided in the same format of the expected output.
As mentioned in Subsection 5.3, the intended task to be performed by the models is a regression task i.e., the developed MaLe models are supposed to infer the value of the stride speed from the predictors associated to each stride. Therefore, the response associated to predictors must be provided as a vector of scalar values that has the same number of rows of the predictors array. The i^th element of the aforementioned vector represents the true stride speed value of the i^th stride expressed in m/s.

True stride speed values for each stride are retrieved form the INDIP standard and used to construct the label vector, which contains the response values of the stride speed that represent the target for the MaLe methods.

5.3.2 Stride speed prediction

Once that the dataset has been created, a regression model is used to predict the value of stride speed for each of the strides included in the construction set. Based on the same procedure followed for the DeLe networks for the estimation of the temporal parameters, a set of MaLe models with different settings is trained, in order to determine the optimal set of hyperparameters. As mentioned in Section 5.3, four GPR models and one SVM are trained and tested. Each one of the models receives as input a set of 136 predictors related to the stride features in the time and frequency domains (see 5.3.1). As output, the model returns the value of stride speed associated with the input stride.

Gaussian Process Regression (GPR)

Four GPR models are trained to infer the value of the stride speed from the computed stride features. GPR models are a class of non-parametric probabilistic models that model the response with a probability distribution over a space of functions [99]. See Section D.1 for further details about the theoretical notions behind the implementation of GPR models.

GPR general settings In the following, the design choices for the training and validation of the developed GPR models are presented.

- *Basis function*: **Constant**. The basis function defines the form of the prior mean function of the GPR model [99]. For the present models, a constant basis function represented by a vector of ones is chosen.
- σ_0: $\dfrac{\mathbf{std(y)}}{\sqrt{\mathbf{2}}}$, where `std(y)` denotes the standard deviation of the target response. σ_0 represents the initial value for the noise standard deviation of the Gaussian process model [99]. After that σ_0 is set, it is optimized during training.
- *Normalization*: **Standard**. Predictor data are standardized i.e., normalized with respect to their mean and standard deviation, so that the standardized predictor has mean 0 and unit standard deviation. Standardizing cancels the dependence on arbitrary scales in the predictors and improves overall performance [99].

- *Fit method*: **Subset of data elements approximation**. Method to estimate parameters of the GPR model [99].
- *Optimizer*: **Quasi-Newton optimizer**. Optimization method used to estimate the model's parameters. The quasi-Newton optimizer consists in a dense, symmetric rank-1-based, quasi-Newton approximation to the Hessian [99] [112].
- *Validation*: **5-folds cross-validation**. Data is randomly partitioned into 5 sets. At each fold, one set is used as TS and the remaining $k-1$ sets are used as TRS. Eventually, the performance of the k trained models are aggregated to provide a summary of the model's skill.

Kernel functions To find the optimal architecture of the GPR model for the stride speed prediction task, four GPR models with different kernel functions have been trained and evaluated:

- **Exponential**
- **Squared exponential**
- **Matern 5/2**
- **Rational quadratic**

As described in Subsection D.1.3, all the kernel functions are defined by two parameters, respectively the length scale σ_l and the signal standard deviation σ_f. In addition, the rational quadratic kernel is also defined by the scale-mixture parameter α. The initial values of the kernel parameters θ are set at the beginning of training (Equation 5.8) and then optimized during the training process.

$$\begin{cases} \sigma_l = mean(std(x)) = 0{,}1618 \\ \sigma_f = std(y)/2 = 0{,}2139 \\ \alpha = 1, \qquad \text{(Only for rational quadratic).} \end{cases} \tag{5.8}$$

Where X represents the predictor matrix, y represents the target response vector and `mean` and `std` are respectively the mean and standard deviation operators. Notice that the correlation length scale is the same for all the predictors.

Support Vector Machine (SVM)

Besides the GPR models, one linear SVM regression model is developed - hereinafter referred to as linear support vector regression (SVR) model. SVR models

are a class of non-parametric MaLe models that aim at finding the best function that at the same time fits the training data and observe a set of pre-defined constraints regarding the prediction errors [99][113]. Linear SVR has an easy level of interpretability; however, it can have low flexibility and can yield to underfitting in certain applications. See Section D.2 for further details about the theoretical notions behind the implementation of SVR models.
In the following, the general settings for the training and validation of the developed SVR model are presented.

- *Box constraint*: **0,2412**. The box constraint is calculated as $\frac{\texttt{iqr(y)}}{1.349}$, which is an estimate of the standard deviation of the target response variable Y [99]. `iqr` denotes the interquartile range operator.
- *Kernel function*: **Linear**. The kernel function defines the transformation applied to the data before the SVM is trained. The linear kernel function has the following form:
$$G(x_j, x_k) = x_j' x_k \tag{5.9}$$
The kernel function is employed to derive the Gram matrix [10].
- *Kernel scale*: **2,8951**. Value of kernel scale has been determined though a heuristic procedure that used subsampling [99]. All observations of the predictor matrix X are divided by the value of the kernel scale. Then, the appropriate kernel norm is applied to compute the Gram matrix. The kernel scale determines the scale of the predictors on which the kernel varies significantly [99]. A smaller kernel scale allows for a more flexible model.
- ϵ: **0,02412**. Half the width of the ϵ-insensitive band (see Section D.2). The value is calculated as $\frac{\texttt{iqr(y)}}{13.49}$, which is an estimate of a tenth of the standard deviation of the response variable Y. Prediction errors that fall in the ϵ-insensitive band are treated as equal to zero. A smaller value of ϵ allows more flexibility. [99].
- *Normalization*: **Standard**. See Sub-subsection 5.3.2.
- *Solver*: **sequential minimal optmization (SMO)**. Optimization routine.
- *Validation*: **5-folds cross-validation**. See Sub-subsection 5.3.2.

[10]The *Gram matrix* associated to a set of n vectors $\{x_1, ..., x_n; x_j \in \mathbb{R}^p\}$ consists in an $n \times n$ matrix with element (j, k) corresponding to $G(x_j, x_k) =< \Phi(x_j), \Phi(x_k) >$, an inner product of the transformed predictors through the kernel function Φ [99].

- *Maximum number of iterations*: $\mathbf{1 \times 10^6}$.
- *Gap tolerance*: $\mathbf{1 \times 10^{-3}}$. Feasibility gap tolerance to stop the SMO. The SMO stopped after 16015 iterations at a gap value of $8.335{,}3 \times 10^{-4}$.

The gap tolerance was achieved before reaching the maximum number of iterations. As a result, the feasibility gap convergence criterion forced the SMO algorithm to stop.

5.4 Testing methodology and metrics

In the following Section, the methodology used for testing the developed MaLe and DeLe models and the metrics used to measure their performance are presented. All the models were tested both on the full In-lab and Free-living datasets.

5.4.1 Temporal parameters

Hereinafter, the expressions TRS, VS and TS will be used to address respectively the training, validation and test sets to perform the GED task. The expression Free-living TS will refer instead to the data collected in free-living conditions not used for training the models.
The 10 DeLe models allow to achieve three different tasks:

1. **Gait phase classification**: Prediction of the gait phase at a given time step.
2. **Gait events detection**: Prediction of the time steps at which the ICs and the FCs occur.
3. **Temporal parameters prediction**: The model predicts the value of four different temporal parameters (see Table 2.1):
   - **Stride time**
   - **Step time**
   - **SS time**
   - **DS time**

The three tasks are sequential, meaning that the output of classification is post-processed to obtain the time steps of gait events and the values of the temporal parameters, as described in Subsections 5.2.2 and 5.2.3.
In the following, the evaluation metrics employed for quantifying the performance of the models in the achievement of each one of the above-mentioned tasks are described.

Gait phases

Once trained, the DeLe classification models are used for prediction over the training and validation data and over the unseen test data.
As mentioned in Sub-subsection 5.2.1, data from Participant 11 were not included in neither the TRS nor the VS; and instead it constituted the In-lab TS. Similarly, data recorded in free-living conditions from Participants 13, 14 and 15 were left aside from training and used for testing purposes only.
At prediction, the raw output of the models has the same format of the target response used for training i.e., it is a 1D sequence of ones and zeroes corresponding to the SS phase and DS phase, respectively (Figure 5.18).

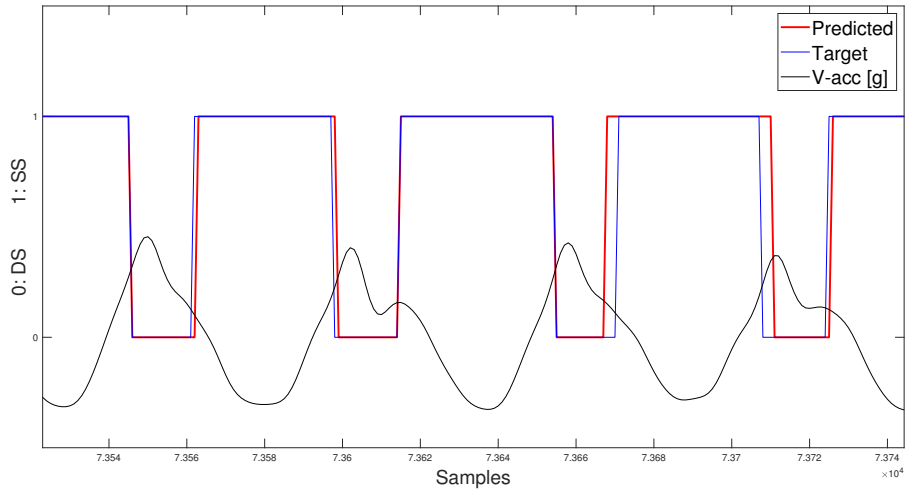


Figure 5.18: Visual comparison between the target response signal (blue) and the predicted response signal (red). The smoothed V-acc (black) shows how the labels correspond to the underlying inertial signals. The current example refers to a Trial belonging to the TRS and predicted by a LSTM model.

Confusion matrix Since the SS/DS phase classification task is binary, a 2×2 confusion matrix (CM) is computed every time that a model is tested on a TRS, VS or TS. CMs - otherwise called *contingency matrices* or *error matrices* [114] - represent a powerful tool for visualizing the outcome of classification altogether, specially in the case of binary classification tasks. Each column of the matrix contains the instances in an actual class while each row contains the instances in a predicted class, or vice versa – use of both variants has been reported by the literature [115]. In binary classification tasks - and specially in the field of health data analysis - class labeled by 0 is commonly referred to as "negative", while the

other one is generally referred to as "positive". In binary tasks, a CM is an array of 4 elements, each of them referring to a different type of instance (Figure 5.19).

- **True negative (TN)**: Instance of class 0 correctly classified in class 0.
- **False positive (FP)**: Instance of class 0 mis-classified in class 1.
- **False negative (FN)**: Instance of class 1 mis-classified in class 0.
- **True positive (TP)**: Instance of class 1 correctly classified in class 1.

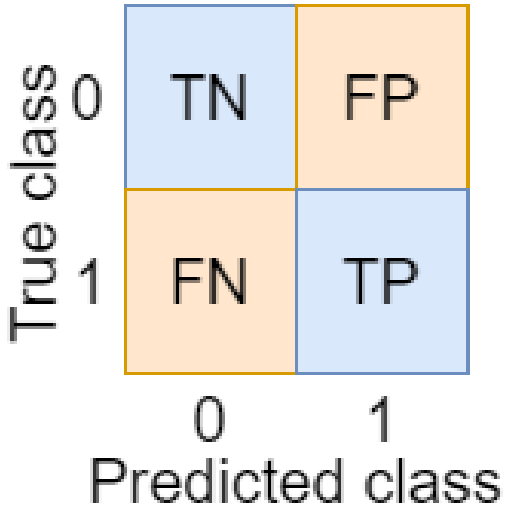


Figure 5.19: Illustration to show how TPs, TNs, FPs and FNs are disposed within a CM. *Blue*: correctly classified instances; *red*: mis-classified instances.

Since the SS and DS phase classes are respectively labeled by 1 and 0, the SS phase class is referred to as "positive", while the DS phase class is referred to as "negative" (Figure 5.20).

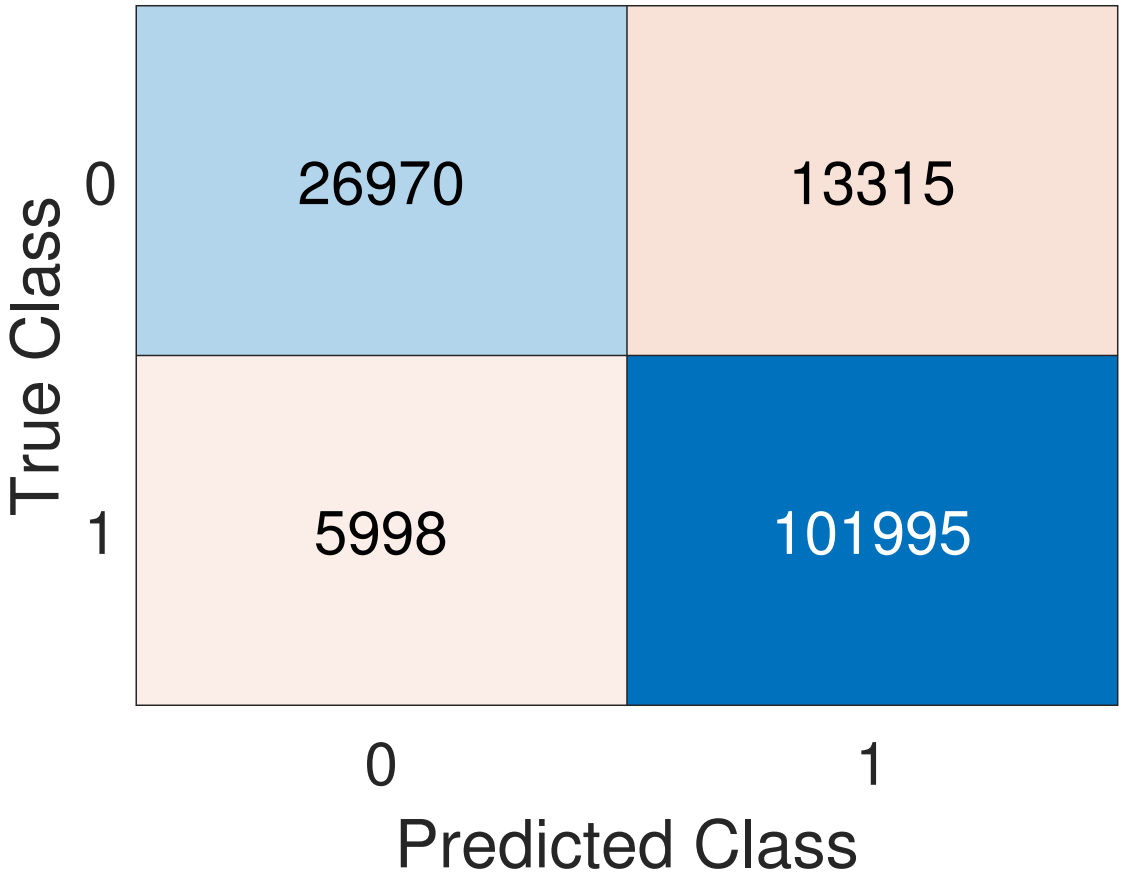


Figure 5.20: Example of a CM for the SS/DS binary classification task showing the absolute values of the number of TPs, TNs, FPs and FNs. The sum of the elements in the CM adds up to the total number of observations.

Therefore, for the present work, the above-mentioned terms take on the following meaning:

- **TN**: A sample of the DS phase class correctly classified in the DS phase class.
- **FP**: A sample of the DS phase class mis-classified in the SS phase class.
- **FN**: A sample of the SS phase class mis-classified in the DS phase class.
- **TP**: A sample of the SS phase class correctly classified in the SS phase class.

Since the number of tested models is 10 and the number of tested datasets is 4 (TRS, VS, in-lab TS, Free-living TS), a total amount of 40 CMs has been printed out.

Evaluation metrics A 2×2 CM allows for the calculation of several useful evaluation metrics that allow to quantify the performance of a classifier. In the following, the adopted evaluation metrics are listed and described:

- **TP**: The absolute number of TPs.
- **TN**: The absolute number of TNs.
- **Number of trues (T)**: The absolute number of correctly classified samples:

$$T = TP + TN$$

- **Accuracy**: It measures the percentage of correctly classified instances with respect to the total number of predictions:

$$Accuracy(\%) = 100 \times \frac{TP + TN}{TP + TN + FP + FN}$$

  The accuracy is arguably the most comprehensive and robust indicator for quantifying the performance of a classifier.

- **Sensitivity**: Otherwise known as true positive rate (TPR), it measures the percentage of correctly classified class 1 instances with respect to the total amount of instances in class 1:

$$Sensitivity(\%) = 100 \times \frac{TP}{TP + FN}$$

- **Specificity**: Otherwise known as true negative rate (TNR), it measures the percentage of correctly classified class 0 instances with respect to the total amount of instances in class 0:

$$Specificity(\%) = 100 \times \frac{TN}{TN + FP}$$

- **Positive predictive value (PPV)**: Otherwise known as *precision*, it measures what percentage is truly positive out of all the positive predicted:

$$PPV(\%) = 100 \times \frac{TP}{TP + FP}$$

- **Negative predictive value (NPV)**: Otherwise known as *recall*, it measures what percentage is truly negative out of all the negative predicted:

$$NPV(\%) = 100 \times \frac{TN}{TN + FN}$$

- **F_1-score (%)**: While accuracy is the arithmetic mean of sensitivity and specificity, the F_1-score represents the harmonic mean of sensitivity and PPV:

$$F_1 = 2\,\frac{Sensitivity\ PPV}{Sensitivity + PPV}$$

Such metrics have been computed for each CM; therefore, 40 sets of evaluation metrics have been collected.

Gait events detection

As described in Subsection 5.2.2, the output of classification is exploited to detect the time steps at which the gait events (ICs and FCs) occur.
Once that the gait events are detected for each dataset, the following error metrics are computed:

- **Extra ICs**: The absolute number of times that the model has predicted an IC not associated with a target IC. The number of extra ICs can also be referred to the number of ICs predicted by the GS (the INDIP standard) and expressed as percentage:

$$\%\ \text{extra ICs} = 100 \times \frac{\text{extra ICs}}{\text{ICs}_{\text{INDIP}}} \tag{5.10}$$

- **Missed ICs**: The absolute number of times that the model has missed to predict an IC. The number of missed ICs can also be referred to the number of ICs predicted by the GS (the INDIP standard) and expressed as percentage:

$$\%\ \text{missed ICs} = 100 \times \frac{\text{missed ICs}}{\text{ICs}_{\text{INDIP}}} \tag{5.11}$$

- **Extra FCs**: The absolute number of times that the model has predicted an FC not associated to a target FC. The number of extra FCs can also be referred to the number of FCs predicted by the GS (the INDIP standard) and expressed as percentage:

$$\%\ \text{extra FCs} = 100 \times \frac{\text{extra FCs}}{\text{FCs}_{\text{INDIP}}} \tag{5.12}$$

- **Missed FCs**: The absolute number of times that the model has missed to predict a FC. The number of missed FCs can also be referred to the number of FCs predicted by the GS (the INDIP standard) and expressed as percentage:

$$\%\ \text{missed FCs} = 100 \times \frac{\text{missed FCs}}{\text{FCs}_{\text{INDIP}}} \tag{5.13}$$

- **Mean absolute error (MAE)**: The average unsigned time delay of the predicted gait event with respect to the target gait event:

$$MAE\,(\mathrm{s}) = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{5.14}$$

  Where y and $\hat{y}$ denote respectively the target and predicted gait events and N is the total number of predicted events.

- **Mean error (ME)**: The average signed time delay of the predicted gait event with respect to the target gait event:

$$ME\,(\mathrm{s}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i) \tag{5.15}$$

MAE and ME are computed for both type of gait events (ICs and FCs). Since there are 10 tested models and 4 tested datasets, 40 sets of error metrics have been evaluated.

Temporal parameters

As described in Subsection 5.2.3, detected gait events are exploited to predict the step time, the stride time and the SS and DS times, whose values are compared to their GS correspondents by means of the following error metrics:

- **MAE**: The average unsigned difference between the values of the predicted and target parameter:

$$MAE\,(\mathrm{s}) = \frac{1}{N}\sum_{i=1}^{N}\left|T_i - \hat{T}_i\right| \tag{5.16}$$

  Where T_i and $\hat{T}_i$ denote respectively the target and predicted values of the parameter and N is the total number of the parameter's values. For the SS and DS times, %MAE is calculated as well:

$$MAE\,(\%) = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{(T_{phase})_i - (\hat{T}_{phase})_i}{(T_{stride})_i}\right| \tag{5.17}$$

  Where $(T_{phase})_i$ and $(\hat{T}_{phase})_i$ denote respectively the target and predicted values of the gait phase time, $(T_{stride})_i$ denotes the GS stride time value and N is the total number of strides.

- **ME**: The average signed difference between the values of the predicted and target parameter:

$$ME\,(\mathrm{s}) = \frac{1}{N}\sum_{i=1}^{N}(T_i - \hat{T}_i) \tag{5.18}$$

  For the SS and DS times, %ME is calculated as well:

$$ME\,(\%) = \frac{1}{N}\sum_{i=1}^{N}\frac{(T_{phase})_i - (\hat{T}_{phase})_i}{(T_{stride})_i} \tag{5.19}$$

MAE and ME are computed for all the four types of gait events. Since SS time and DS time are typically expressed as % GC, %MAE and %ME for the SS and DS times are computed as well. Since there are 10 tested models and 4 tested datasets, 40 sets of error metrics have been evaluated.

5.4.2 Stride speed

Testing machine learning models on strides segmented by the INDIP standard

At the end of training, the performance of the MaLe models are tested on the strides of the In-Lab construction set and test set segmented by the INDIP standard (see Subsection 5.3.1). Such validation does not evaluates the performance of the whole GED/stride speed estimation pipeline; however, it provides an immediate feedback about the performance of the developed models without the influence of DeLe-based GED, which is addressed in a second moment (see Sub-subsection 5.4.2).
As described in Sub-subsection 5.3.2, performance of the trained MaLe models at training time is evaluated via 5-folds cross-validation. First, the results of regression on each In-Lab dataset for each of the trained regression models are used to produce three types of plots, which give an immediate visual feedback about the general outcome of regression:

- **Response plot**: predicted and true response are plotted versus the record number. This type of plot was drawn only for the results of the models on the stride speed construction set.

- **Predicted VS actual response**: predicted stride speed is plotted against the actual, true stride speed. An ideal regression model should be characterized by a predicted response that is perfectly equal to the true response, so that all the points lie on a diagonal line. The vertical distance from the line to any point represents the error of the prediction for that point. A model with good performance has limited errors, which means that the predictions are

scattered symmetrically close to the line [99]. This type of plot was drawn for the results of the models on both the stride speed construction set and TS.

- **Residual plot**: Prediction residuals are plotted versus the true response. In general, a model with good performance shows residuals distributed nearly symmetrically around the zero-level [99]. This type of plot was drawn for the results of the models on both the stride speed construction set and TS.

Then, the agreement between the two set of measurements is quantified by the following set of error metrics. In particular, the predicted stride speed values y_{H-MIMU} are compared to the target stride speed values y_{INDIP} provided by the INDIP standard:

- **MAE**: Mean unsigned difference between the N predicted stride speed values y_{INDIP} and the N corresponding target stride speed values y_{INDIP}:

$$MAE\,(\mathrm{m/s}) = \frac{1}{N}\sum_{i=1}^{N}|y_{INDIP} - y_{H-MIMU}| \tag{5.20}$$

- **Mean squared error (MSE)**: Mean squared difference between the N predicted stride speed values y_{INDIP} and the N corresponding target stride speed values y_{INDIP}:

$$MSE\,(\mathrm{m^2/s^2}) = \frac{1}{N}\sum_{i=1}^{N}(y_{INDIP} - y_{H-MIMU})^2 \tag{5.21}$$

- **Root mean squared error (RMSE)**: Root mean square difference between the N predicted stride speed values y_{INDIP} and the N corresponding target stride speed values y_{INDIP}:

$$RMSE\,(\mathrm{m/s}) = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(y_{INDIP} - y_{H-MIMU})^2} \tag{5.22}$$

- **Coefficient of determination ($\mathbf{R^2}$)**: R^2 is a statistical metric used to represent the share of the variance for a dependent variable explained by an independent variable or variables in a regression model:

$$R^2 = \frac{1}{N}\sum_{i=1}^{N}\frac{(y_{INDIP} - y_{H-MIMU})^2}{(y_{INDIP} - \bar{y}_{H-MIMU})^2} \tag{5.23}$$

  Where $\bar{y}_{H-MIMU}$ represents the mean of the predicted stride speed values for the given dataset. R^2 is based on the hypothesis that every independent predictor in the model helps to explain variance for the dependent variable. R^2 ranges between 0 and 1.

In addition, the training time in seconds and the prediction speed (obj/s) are reported as well.

Testing machine learning models on strides segmented by deep learning models

As explained in 5.3.2, the output of the DeLe classification model becomes the input to the MaLe regression model. In particular, the output of GED (Subsection 5.2.2) is used to segment the strides within the acceleration norm signal and compute a set of 136 stride predictors that are used by the MaLe models to infer the value of the stride speed for each input stride vector (see Sub-subsection 5.3.1 and Section 5.3.2).
Since there are ten DeLe models for GED, five MaLe models for stride speed prediction and four datasets, a total amount of two hundreds model - dataset combinations have been tested. For each one of them, a the same error metrics described in Sub-subsection 5.4.2 are computed. In addition, the following metrics are computed as well:

- **Mean absolute percentage error (MAPE)**: Mean unsigned difference between the N predicted stride speed values y_{INDIP} and the N corresponding target stride speed values y_{INDIP} expressed as percentage of y_{INDIP}:

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_{INDIP} - y_{H-MIMU}|}{y_{INDIP}} \tag{5.24}$$

  The MAPE scales the variable units as percentage units; therefore facilitating its interpretability [116]. The MAPE ranges between 0 and 1.

- **Adjusted ρ^2**: Adjusted R^2 (adR^2) is used to represent the share of variance for a dependent variable explained by only those independent variables that really explain the dependent variable. If any variables that do not help in predicting the dependent variable are introduced, the value of adR^2 decreases.

$$adR^2 = 1 - \frac{(1-R^2)(N-1)}{(N-p-1)} \tag{5.25}$$

  Where p is the number of independent predictors. adR^2 ranges between 0 and 1.

- **Pearson correlation index (ρ)**: The Pearson correlation index between two random variables measures their linear dependence [99]:

$$\rho = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{y_{INDIP} - \bar{y}_{INDIP}}{\sigma_{INDIP}}\right)\left(\frac{y_{H-MIMU} - \bar{y}_{H-MIMU}}{\sigma_{H-MIMU}}\right) \tag{5.26}$$

  Where $\bar{y}_{H-MIMU}$ and $\bar{y}_{INDIP}$ respectively represents the mean of the predicted and target stride speed values for the given dataset. σ_{H-MIMU} and σ_{INDIP} respectively represents the standard deviations of the predicted and target stride speed values for the given dataset. ρ ranges between 0 and 1.

Chapter 6

Results

In the present Chapter, the results of the testing stage are presented (see Section 5.4).

6.1 Temporal parameters

As described in Section 5.4, the first task to be evaluated is the GED and the successive estimation of the temporal parameters. As mentioned in Section 5.2, the task aimed at estimating the temporal parameters can be further divided into three functional sub-tasks (SS/DS classification, GED, definition of the temporal parameters).
In this Section, the errors and general performance for each one of the three functional sub-tasks are presented. First, the results obtained on the In-Lab dataset are presented; then, results on the Free-Living dataset are shown as well.

6.1.1 In-Lab dataset

As described in Section 5.4, each one of the trained DeLe models is applied to its respective In-Lab TRS, VS and TS. Since there are 10 trained models - 5 TCNs and 5 LSTM networks - and 3 In-Lab datasets, 30 summaries of results have been drafted. The summary of results of a DeLe model includes the classification metrics, the number of missed and extra events, the time error between predicted and target gait events and the error in the estimation of the temporal parameters.

LSTM networks

Five LSTM networks were trained to classify each time step of the predictor signals in the SS or DS phase. Then, the output of the five classifiers was processed to define the gait events and the values of four temporal parameters.

Classification metrics Table 6.1 shows the performance of classification for each of the 5 trained LSTM networks on each In-Lab dataset.

Classification metrics - LSTM							
Rep.	Set	Acc. (%)	Sens. (%)	Spec. (%)	PPV (%)	NPV (%)	F_1 (%)
1	TRS	86,98	94,45	66,95	88,45	81,81	91,35
	VS	80,44	91,37	53,22	82,94	71,24	86,95
	TS	69,74	87,47	39,28	71,24	64,57	78,52
2	TRS	89,73	94,50	76,84	91,68	83,80	93,07
	VS	79,05	85,77	62,59	84,89	64,21	85,33
	TS	69,61	81,37	49,40	73,43	60,66	77,20
3	TRS	90,62	94,96	78,81	92,41	85,19	93,67
	VS	79,59	89,15	56,54	83,19	68,36	86,06
	TS	70,23	87,88	39,88	71,54	65,69	78,87
4	TRS	89,51	94,06	76,91	91,87	82,35	92,95
	VS	78,42	91,09	49,23	80,52	70,58	85,48
	TS	71,72	85,25	48,45	73,98	65,64	79,22
5	TRS	87,57	95,38	67,12	88,37	84,73	91,74
	VS	81,40	91,55	54,72	84,16	71,13	87,70
	TS	69,50	87,60	38,37	70,96	64,29	78,41

Table 6.1: Values of the adopted classification metrics for the binary SS/DS classification task obtained by each LSTM model on the In-Lab dataset. Each row of the Table refers to the performance achieved by the model (column 1) on the corresponding In-Lab dataset (column 2). *Rep.*: Repetition; *Acc.*: Accuracy; *Sens.*: Sensitivity; *Spec.*: Specificity.

For each model, the values of each metric over the three datasets are aggregated to derive the average and standard deviation values.
The average values are used to draw a bar diagram that allows a faster comparison between the performance of the different models, while the values of the standard deviation are used to superimpose error bars to each data series (Figure 6.1).

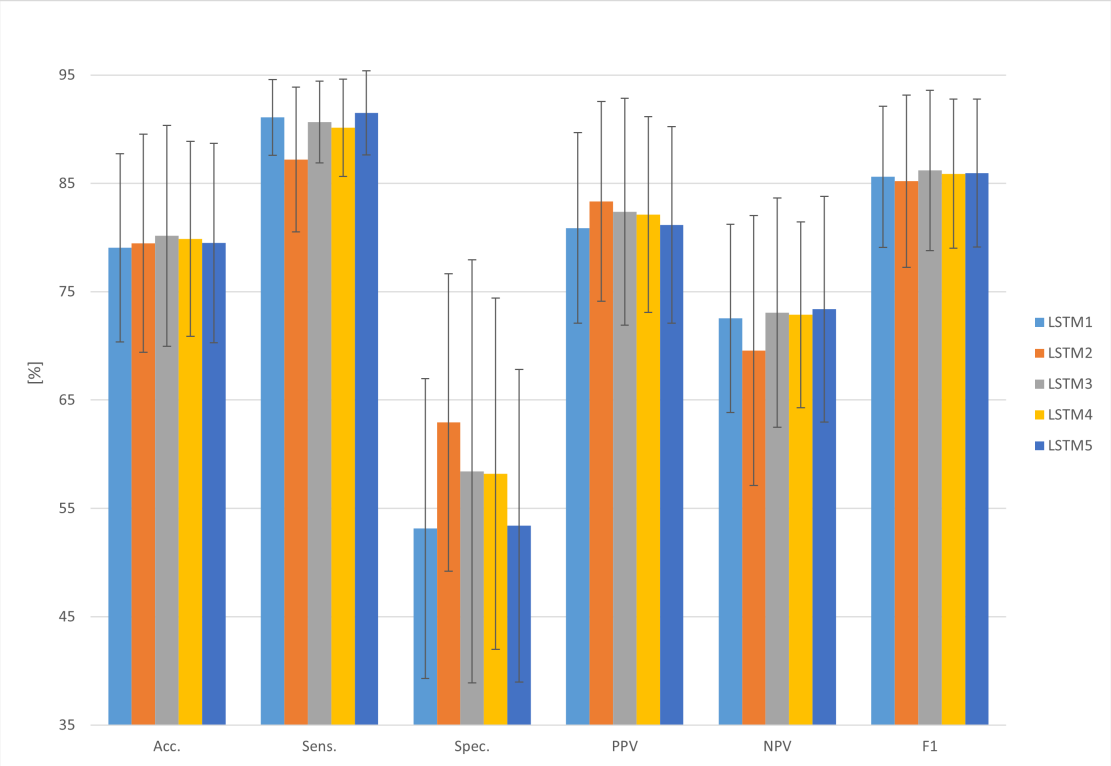


Figure 6.1: Bar diagram that summarizes the classification performance of the LSTM models on the In-Lab dataset. Each group of bars denotes the values of the corresponding metric for each of the 5 models. For each bar, the average values are plotted as mean ± standard deviation of the In-Lab TRS, VS and TS. The present graph was designed on Microsoft®Excel®.

Similarly, for each model, average values of each metric are used to draw line plots (Figure 6.2) and radar plots (Figure 6.3). Plots referred to different models are overlapped to highlight differences in the metrics values.

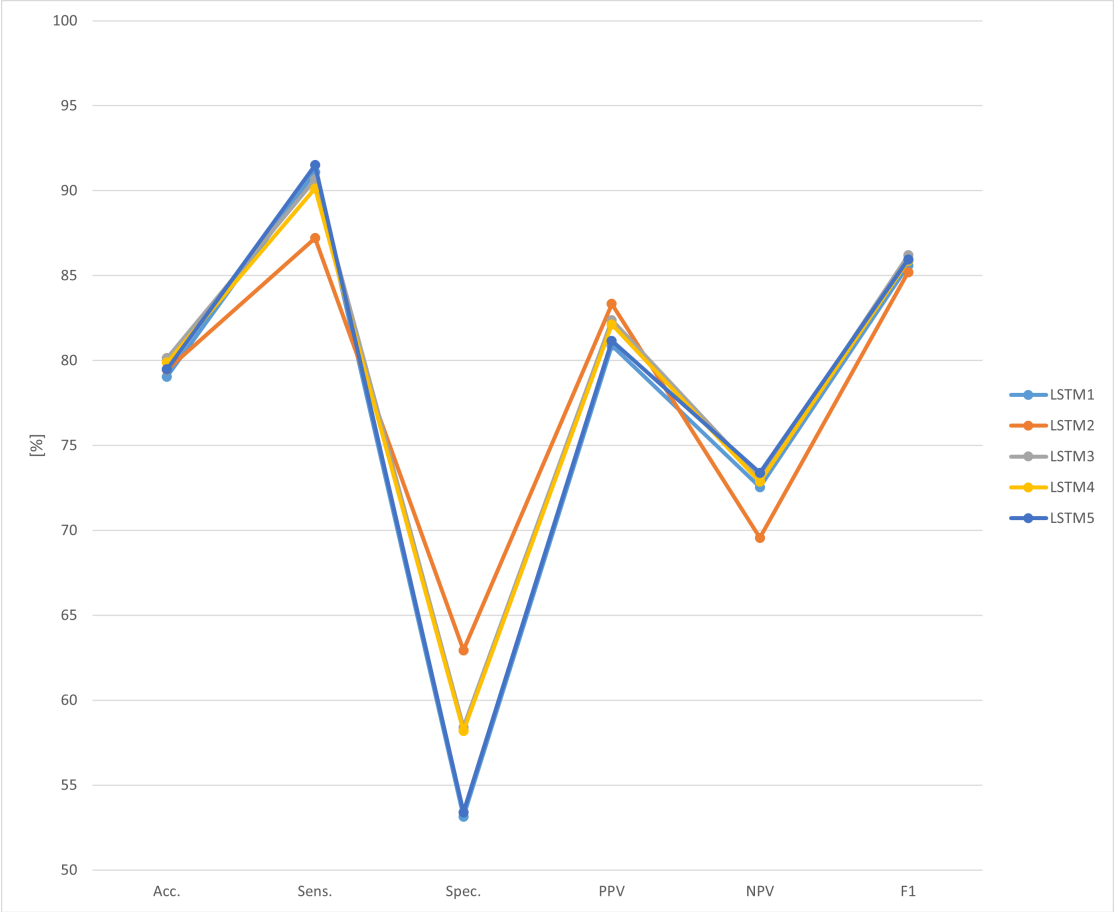


Figure 6.2: Lines plot that summarizes the classification performance of the LSTM models on the In-Lab dataset. Each line represents a model, while each point of the line represents the value of the corresponding metric on the x-axis for that model. The present graph was designed on Microsoft®Excel®.

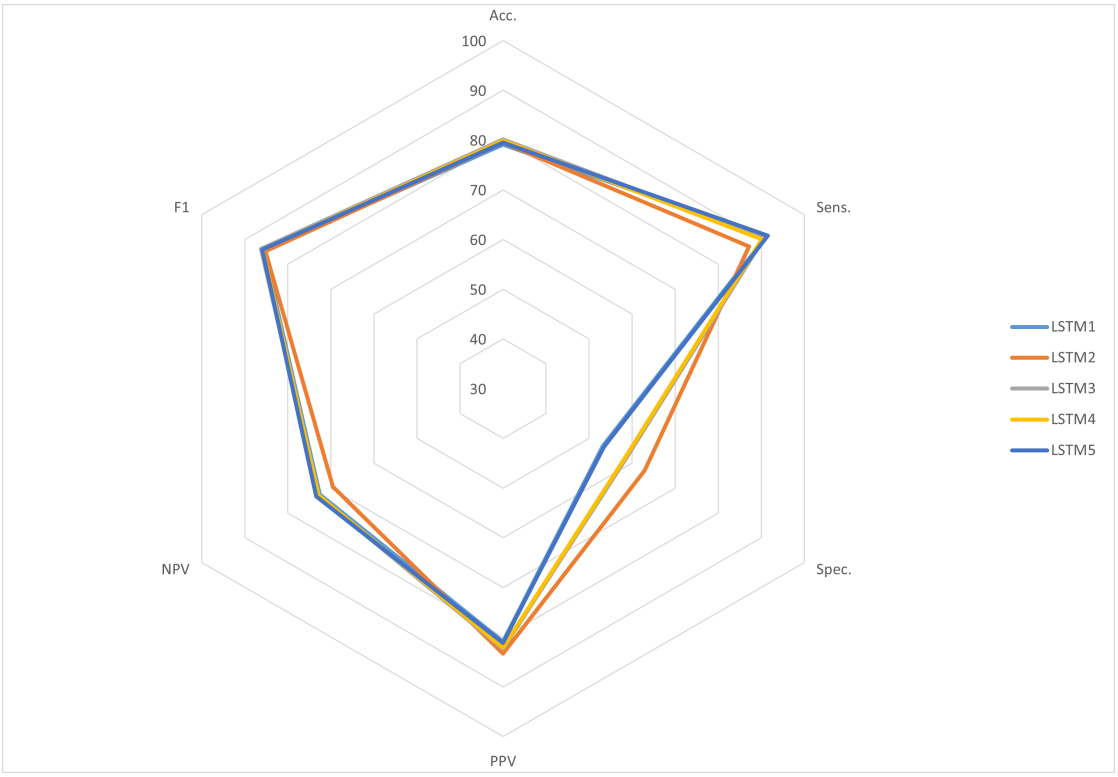


Figure 6.3: Radar plot that summarizes the classification performance of the LSTM models on the In-Lab dataset. Each polygon represents a model, while each point of the polygon represents the value of the corresponding metric on the outer hexagon. The present graph was designed on Microsoft®Excel®.

GED errors Table 6.2 shows the absolute and relative numbers of extra and missed events for each of the five trained LSTM networks on each In-Lab dataset.

Missed/extra events - LSTM					
Repetition	SET	extra IC	missed IC	extra FC	missed FC
1	TRS	92 (3,79 %)	278 (11,44 %)	94 (3,87 %)	280 (11,52 %)
	VS	71 (7,10 %)	162 (16,20 %)	71 (7,10 %)	162 (16,20 %)
	TS	21 (6,36 %)	98 (29,70 %)	20 (6,06 %)	97 (29,39 %)
2	TRS	81 (3,36 %)	170 (7,05 %)	80 (3,32 %)	170 (7,05 %)
	VS	54 (5,30 %)	99 (9,72 %)	56 (5,50 %)	102 (10,02 %)
	TS	12 (3,64 %)	68 (20,61 %)	14 (4,24 %)	72 (21,82 %)
3	TRS	63 (2,60 %)	157 (6,48 %)	60 (2,48 %)	156 (6,44 %)
	VS	57 (5,65 %)	123 (12,20 %)	61 (6,05 %)	127 (12,60 %)
	TS	13 (3,94 %)	88 (26,67 %)	12 (3,64 %)	87 (26,36 %)
4	TRS	79 (3,23 %)	193 (7,88 %)	79 (3,23 %)	194 (7,92 %)
	VS	56 (5,70 %)	180 (18,33 %)	61 (6,21 %)	184 (18,74 %)
	TS	18 (5,45 %)	79 (23,94 %)	14 (4,24 %)	75 (22,73 %)
5	TRS	84 (3,54 %)	277 (11,66 %)	82 (3,45 %)	274 (11,54 %)
	VS	43 (4,08 %)	166 (15,73 %)	43 (4,08 %)	168 (15,92 %)
	TS	12 (3,64 %)	116 (35,15 %)	14 (4,24 %)	119 (36,06 %)

Table 6.2: Number of missed and extra gait events (FCs and ICs) on the In-Lab TRS, VS and TS for each LSTM model. In parentheses, the percentage of misclassified events is reported i.e., the ratio between the number of extra/missed events and the total number of events detected by the INDIP standard (see Subsubsection 5.4.1).

Next, the time errors between the target events and the detected events are computed in terms of MAE and ME (Table 6.3).

LSTM		**ICs**		**FCs**	
Repetition	SET	MAE (s)	ME (s)	MAE (s)	ME (s)
1	TRS	0,03±0,04	0,01±0,05	0,04±0,04	0,00±0,06
	VS	0,05±0,06	0,03±0,07	0,06±0,05	0,01±0,07
	TS	0,08±0,07	0,04±0,09	0,08±0,07	0,01±0,11
2	TRS	0,02±0,03	0,00±0,04	0,03±0,03	0,00±0,05
	VS	0,05±0,06	0,03±0,07	0,07±0,06	0,04±0,08
	TS	0,09±0,07	0,05±0,10	0,08±0,07	0,04±0,10
3	TRS	0,02±0,03	0,00±0,04	0,03±0,03	0,00±0,04
	VS	0,06±0,06	0,02±0,08	0,06±0,07	0,01±0,09
	TS	0,08±0,07	0,05±0,10	0,08±0,06	0,00±0,10
4	TRS	0,03±0,03	0,00±0,04	0,03±0,03	0,00±0,05
	VS	0,06±0,06	0,02±0,09	0,07±0,07	0,00±0,09
	TS	0,07±0,06	0,03±0,09	0,08±0,06	0,01±0,10
5	TRS	0,03±0,04	0,01±0,04	0,04±0,04	-0,01±0,06
	VS	0,05±0,06	0,01±0,07	0,06±0,06	0,01±0,08
	TS	0,08±0,07	0,05±0,09	0,08±0,06	0,02±0,09

Table 6.3: Time error between the time steps at which predicted and target gait events occur obtained by each LSTM model on the In-Lab dataset. Results obtained for each LSTM net on the In-Lab dataset are expressed in terms of mean ± standard deviation. Unsigned residuals are considered for MAE, while signed residuals are considered for ME (see Sub-subsection 5.4.1).

Errors on the temporal parameters Table 6.4 shows the values of MAE and ME in the estimation of the step and stride times.

LSTM		**Step time errors**		**Stride time errors**	
Repetition	SET	MAE (s)	ME (s)	MAE (s)	ME (s)
1	TRS	0,04±0,05	0,00±0,07	0,04±0,05	0,00±0,07
	VS	0,06±0,06	0,00±0,09	0,05±0,06	0,00±0,08
	TS	0,11±0,09	0,00±0,14	0,09±0,09	0,00±0,12
2	TRS	0,03±0,04	0,00±0,05	0,04±0,04	0,00±0,06
	VS	0,06±0,07	0,00±0,09	0,06±0,08	0,00±0,10
	TS	0,11±0,09	0,00±0,14	0,10±0,08	0,00±0,13
3	TRS	0,03±0,04	0,00±0,05	0,03±0,04	0,00±0,05
	VS	0,07±0,09	0,00±0,11	0,06±0,08	0,00±0,10
	TS	0,10±0,09	0,00±0,14	0,10±0,09	0,00±0,13
4	TRS	0,04±0,04	0,00±0,05	0,04±0,04	0,00±0,06
	VS	0,07±0,08	0,00±0,11	0,06±0,08	0,00±0,10
	TS	0,10±0,08	-0,01±0,13	0,09±0,08	0,00±0,12
5	TRS	0,04±0,05	0,00±0,06	0,04±0,04	0,00±0,06
	VS	0,06±0,07	0,00±0,09	0,06±0,07	0,00±0,09
	TS	0,10±0,10	0,01±0,14	0,08±0,09	-0,01±0,12

Table 6.4: Time errors between target and predicted step and stride times obtained by each LSTM model on the In-Lab dataset. Results obtained for each LSTM net on the In-Lab dataset are expressed in terms of mean ± standard deviation. Unsigned residuals are considered for MAE, while signed residuals are considered for ME (see Sub-subsection 5.4.1).

Similarly, Tables 6.5 and 6.6 show the absolute and relative errors in the estimation of the SS and DS times.

DS time errors - LSTM			
Repetition	SET	MAE (s)	ME (s)
1	TRS	0,04±0,04 (6,94±6,21 %)	0,01±0,06 (1,90±9,12 %)
	VS	0,06±0,06 (9,41±8,24 %)	0,02±0,08 (2,62±12,23 %)
	TS	0,07±0,06 (12,28±10,21 %)	0,04±0,09 (5,39±15,05 %)
2	TRS	0,04±0,03 (5,93±5,48 %)	0,00±0,05 (0,26±8,07 %)
	VS	0,05±0,05 (8,72±7,84 %)	-0,01±0,08 (-1,96±11,56 %)
	TS	0,07±0,06 (11,85±10,62 %)	0,01±0,09 (2,27±15,76 %)
3	TRS	0,03±0,03 (5,29±4,98 %)	0,00±0,05 (0,26±7,26 %)
	VS	0,06±0,05 (9,60±7,75 %)	0,01±0,08 (1,64±12,24 %)
	TS	0,08±0,07 (13,39±11,14 %)	0,04±0,09 (6,97±15,98 %)
4	TRS	0,04±0,04 (5,94±5,40 %)	0,00±0,05 (-0,29±8,02 %)
	VS	0,06±0,06 (9,71±8,19 %)	0,03±0,08 (4,72±11,80 %)
	TS	0,07±0,06 (11,44±9,92 %)	0,03±0,08 (4,45±14,49 %)
5	TRS	0,04±0,04 (6,68±5,89 %)	0,02±0,05 (2,78±8,46 %)
	VS	0,05±0,05 (8,70±7,78 %)	0,01±0,07 (1,84±11,53 %)
	TS	0,08±0,07 (13,89±11,63 %)	0,04±0,10 (6,28±17,01 %)

Table 6.5: Time errors between target and predicted DS times obtained by each LSTM model on the In-Lab dataset. Results obtained for each LSTM net on the In-Lab dataset are expressed in terms of mean ± standard deviation. Unsigned residuals are considered for MAE, while signed residuals are considered for ME (see Sub-subsection 5.4.1). Values between parentheses denote the error with respect to the target stride time.

SS time errors - LSTM			
Repetition	SET	MAE (s)	ME (s)
1	TRS	0,05±0,06 (7,59±8,45 %)	-0,01±0,07 (-1,42±11,28 %)
	VS	0,06±0,07 (9,86±9,40 %)	-0,01±0,09 (-1,89±13,49 %)
	TS	0,12±0,11 (19,80±17,39 %)	-0,04±0,15 (-7,77±25,22 %)
2	TRS	0,04±0,05 (6,37±7,00 %)	0,00±0,06 (0,04±9,47 %)
	VS	0,07±0,07 (10,35±9,72 %)	0,01±0,10 (1,99±14,07 %)
	TS	0,11±0,08 (18,44±15,50 %)	-0,01±0,13 (-3,57±23,85 %)
3	TRS	0,04±0,04 (5,71±6,40 %)	0,00±0,05 (-0,25±8,57 %)
	VS	0,08±0,09 (11,78±11,94 %)	-0,01±0,12 (-1,41±16,72 %)
	TS	0,11±0,09 (19,91±18,54 %)	-0,05±0,14 (-8,93±25,73 %)
4	TRS	0,04±0,05 (6,25±6,76 %)	0,00±0,06 (0,38±9,20 %)
	VS	0,08±0,09 (11,47±13,02 %)	-0,02±0,12 (-3,35±17,03 %)
	TS	0,11±0,09 (19,91±15,76 %)	-0,03±0,14 (-6,37±24,61 %)
5	TRS	0,05±0,05 (7,34±7,76 %)	-0,02±0,07 (-2,39±10,42 %)
	VS	0,06±0,08 (9,86±11,35 %)	-0,01±0,10 (-1,41±14,97 %)
	TS	0,10±0,08 (16,83±14,16 %)	-0,03±0,12 (-6,42±21,08 %)

Table 6.6: Time errors between target and predicted SS times obtained by each LSTM model on the In-Lab dataset. Results obtained for each LSTM net on the In-Lab dataset are expressed in terms of mean ± standard deviation. Unsigned residuals are considered for MAE, while signed residuals are considered for ME (see Sub-subsection 5.4.1). Values between parentheses denote the error with respect to the stride time.

TCN

Based on the same procedure followed for the LSTM networks, five TCNs were trained to classify each time step of the predictor signals in the SS or DS phase. Then, the output of the five classifiers was processed to define the gait events and the values of four temporal parameters.

Classification metrics Table 6.7 shows the performance of classification for each of the five trained TCNs on each In-Lab dataset.

Classification metrics - TCN networks							
Rep.	SET	Acc. (%)	Sens. (%)	Spec. (%)	PPV (%)	NPV (%)	F_1 (%)
1	TRS	93,45	96,28	85,88	94,81	89,59	95,54
	VS	92,05	96,26	81,57	92,86	89,76	94,53
	TS	86,49	94,90	72,05	85,37	89,15	89,88
2	TRS	92,75	97,19	80,77	93,17	91,42	95,14
	VS	91,14	97,38	75,85	90,81	92,19	93,98
	TS	83,65	96,19	62,11	81,36	90,45	88,15
3	TRS	92,92	96,75	82,53	93,77	90,33	95,24
	VS	91,56	96,85	78,81	91,68	91,20	94,19
	TS	85,74	95,63	68,75	84,03	90,14	89,45
4	TRS	92,89	96,07	84,07	94,36	88,53	95,21
	VS	91,29	98,35	75,03	90,07	95,18	94,03
	TS	87,55	96,95	71,39	85,35	93,17	90,78
5	TRS	92,46	95,18	85,34	94,44	87,13	94,81
	VS	90,95	94,03	82,86	93,52	84,08	93,77
	TS	84,48	90,81	73,58	85,53	82,33	88,09

Table 6.7: Values of the adopted classification metrics for the binary SS/DS classification task obtained by the TCN models. Each row of the Table refers to the performance achieved by the model (column 1) on the corresponding In-Lab dataset (column 2). *Rep.*: Repetition; *Acc.*: Accuracy; *Sens.*: Sensitivity; *Spec.*: Specificity.

For each model, the values of each metric over the three datasets are aggregated to derive the average and standard deviation values.
The average values are used to draw a bar diagram that allows a faster comparison between the performance of the different models, while the values of the standard deviation are used to superimpose error bars to each data series (Figure 6.4).

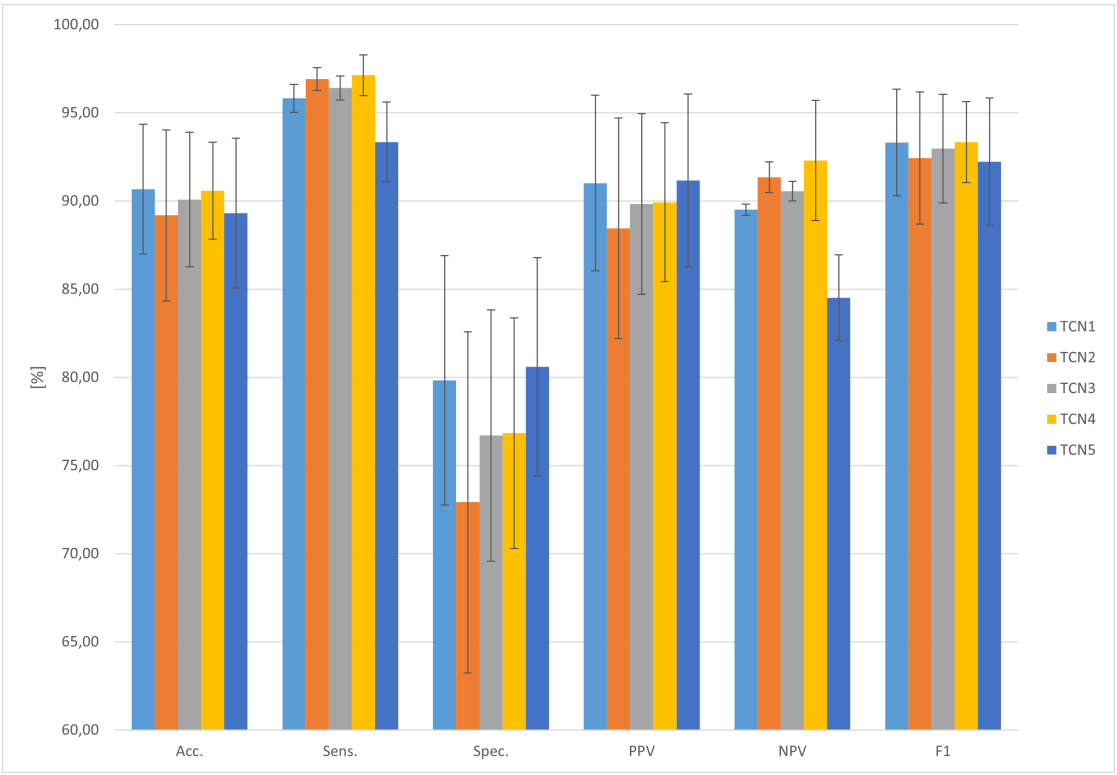


Figure 6.4: Bar diagram to summarize the classification performance of the TCN models on the In-Lab dataset. Each group of bars denotes the values of the corresponding metric for each of the 5 models. For each bar, the average values are plotted as mean ± standard deviation of the In-Lab TRS, VS and TS. The present graph was designed on Microsoft®Excel®.

Similarly, for each model, average values of each metric are used to draw line plots (Figure 6.5) and radar plots (Figure 6.6). Plots referred to different models are overlapped to highlight differences in the metrics values.

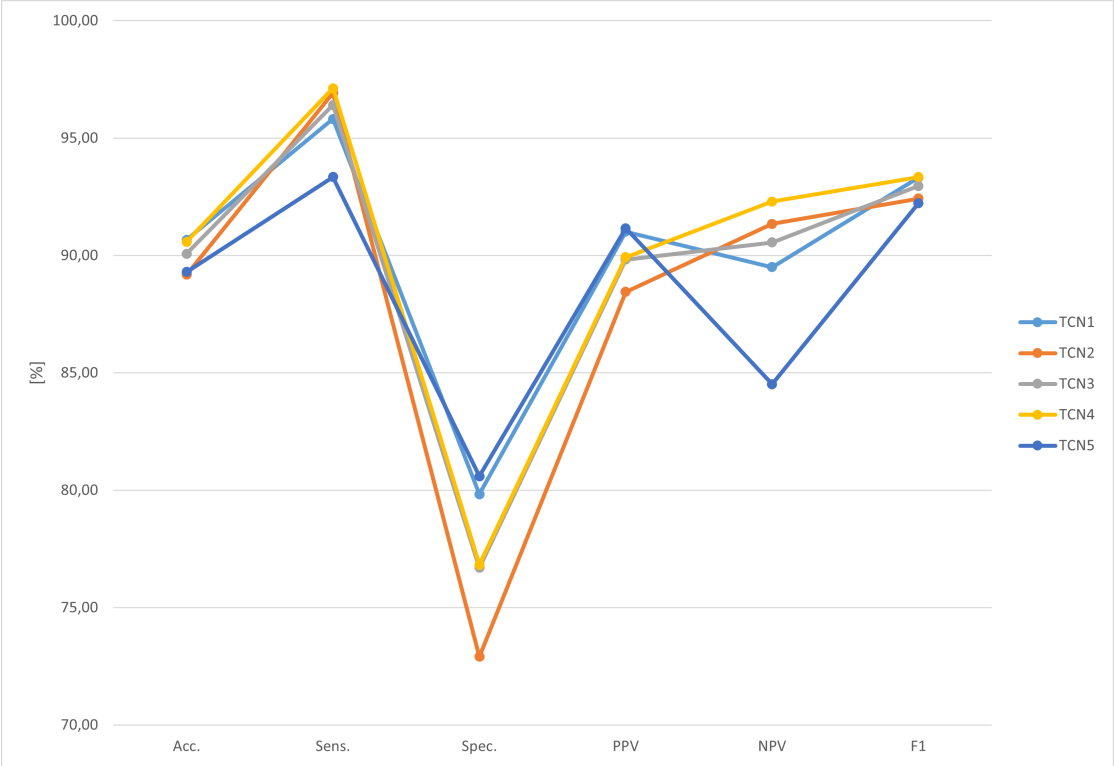


Figure 6.5: Lines plot that summarizes the classification performance of each TCNs model on the In-Lab dataset. Each line represents a model, while each point of the line represents the value of the corresponding metric on the x-axis for that model. The present graph was designed on Microsoft®Excel®.

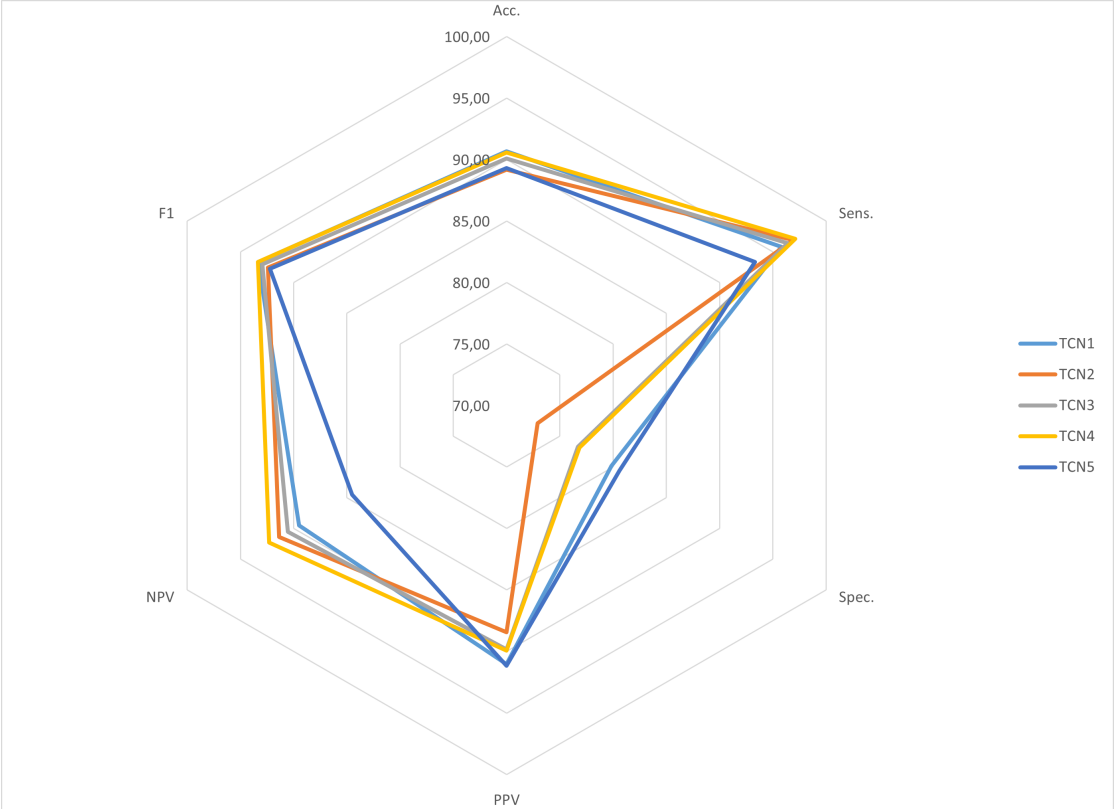


Figure 6.6: Radar plot that summarizes the classification performance of each TCNs model on the In-Lab dataset. Each polygon represents a model, while each point of the polygon represents the value of the corresponding metric on the outer hexagon. The present graph was designed on Microsoft®Excel®.

GED errors Table 6.8 shows the absolute and relative numbers of extra and missed events for each of the five trained TCNs on each In-Lab dataset.

Missed/extra events - TCN					
Repetition	SET	extra IC	missed IC	extra FC	missed FC
1	TRS	61 (2,51 %)	4 (0,16 %)	60 (2,47 %)	4 (0,16 %)
	VS	33 (3,30 %)	6 (0,60 %)	34 (3,40 %)	6 (0,60 %)
	TS	6 (1,82 %)	0 (0,00 %)	6 (1,82 %)	0 (0,00 %)
2	TRS	61 (2,53 %)	6 (0,25 %)	60 (2,49 %)	6 (0,25 %)
	VS	15 (1,47 %)	1 (0,10 %)	14 (1,38 %)	1 (0,10 %)
	TS	5 (1,52 %)	2 (0,61 %)	5 (1,52 %)	2 (0,61 %)
3	TRS	68 (2,81 %)	9 (0,37 %)	68 (2,81 %)	9 (0,37 %)
	VS	16 (1,59 %)	0 (0,00 %)	15 (1,49 %)	0 (0,00 %)
	TS	4 (1,21 %)	1 (0,30 %)	4 (1,21 %)	1 (0,30 %)
4	TRS	74 (3,02 %)	7 (0,29 %)	74 (3,02 %)	7 (0,29 %)
	VS	32 (3,26 %)	1 (0,10 %)	34 (3,46 %)	1 (0,10 %)
	TS	6 (1,82 %)	0 (0,00 %)	6 (1,82 %)	0 (0,00 %)
5	TRS	62 (2,61 %)	4 (0,17 %)	61 (2,57 %)	4 (0,17 %)
	VS	30 (2,84 %)	9 (0,85 %)	30 (2,84 %)	9 (0,85 %)
	TS	3 (0,91 %)	0 (0,00 %)	3 (0,91 %)	0 (0,00 %)

Table 6.8: Number of missed and extra gait events (FCs and ICs) on the In-Lab TRS, VS and TS for each TCN model. In parentheses, the percentage of misclassified events is reported i.e., the ratio between the number of extra/missed events and the total number of events detected by the INDIP standard (see Sub-subsection 5.4.1).

Next, the time errors between the target events and the detected events are computed in terms of MAE and ME (Table 6.9).

TCN		ICs		FCs	
Repetition	SET	MAE (s)	ME (s)	MAE (s)	ME (s)
1	TRS	0,02±0,02	0,00±0,02	0,02±0,02	0,00±0,03
	VS	0,01±0,01	0,00±0,02	0,03±0,02	-0,02±0,04
	TS	0,05±0,04	0,05±0,04	0,03±0,02	0,01±0,04
2	TRS	0,02±0,02	0,01±0,02	0,02±0,02	-0,01±0,03
	VS	0,02±0,02	0,02±0,02	0,03±0,03	-0,02±0,04
	TS	0,07±0,04	0,07±0,04	0,03±0,02	0,00±0,04
3	TRS	0,02±0,02	0,01±0,02	0,03±0,02	-0,01±0,03
	VS	0,02±0,02	0,02±0,02	0,03±0,03	-0,01±0,04
	TS	0,06±0,04	0,05±0,04	0,03±0,02	0,00±0,04
4	TRS	0,02±0,02	0,00±0,02	0,02±0,02	-0,01±0,03
	VS	0,02±0,02	0,01±0,02	0,04±0,03	-0,03±0,03
	TS	0,05±0,04	0,05±0,04	0,02±0,02	-0,01±0,03
5	TRS	0,02±0,01	0,01±0,02	0,03±0,03	0,00±0,04
	VS	0,02±0,02	0,01±0,02	0,03±0,02	0,01±0,04
	TS	0,05±0,04	0,05±0,04	0,04±0,02	0,03±0,04

Table 6.9: Time error between the time steps at which predicted and target gait events occur obtained by each TCN model on the In-Lab dataset. Results obtained for each TCN net on the In-Lab dataset are expressed in terms of mean ± standard deviation. Unsigned residuals are considered for MAE, while signed residuals are considered for ME (see Sub-subsection 5.4.1).

Errors on the temporal parameters Table 6.10 shows the values of MAE and ME in the estimation of the step and stride times.

TCN		Step time errors		Stride time errors	
Repetition	SET	MAE (s)	ME (s)	MAE (s)	ME (s)
1	TRS	0,02±0,02	0,00±0,03	0,01±0,02	0,00±0,02
	VS	0,02±0,02	0,00±0,02	0,01±0,02	0,00±0,02
	TS	0,06±0,03	0,00±0,07	0,02±0,03	0,00±0,03
2	TRS	0,02±0,02	0,00±0,03	0,02±0,02	0,00±0,02
	VS	0,02±0,02	0,00±0,02	0,01±0,01	0,00±0,02
	TS	0,06±0,03	0,00±0,07	0,03±0,03	0,00±0,04
3	TRS	0,02±0,02	0,00±0,02	0,01±0,02	0,00±0,02
	VS	0,02±0,02	0,00±0,03	0,01±0,02	0,00±0,02
	TS	0,06±0,03	0,00±0,07	0,02±0,03	0,00±0,03
4	TRS	0,02±0,02	0,00±0,03	0,02±0,02	0,00±0,02
	VS	0,02±0,02	0,00±0,02	0,01±0,02	0,00±0,02
	TS	0,06±0,03	0,00±0,07	0,02±0,03	0,00±0,04
5	TRS	0,02±0,02	0,00±0,02	0,01±0,02	0,00±0,02
	VS	0,02±0,02	0,00±0,03	0,02±0,02	0,00±0,02
	TS	0,06±0,03	0,00±0,07	0,02±0,03	0,00±0,04

Table 6.10: Time errors between target and predicted step and stride times obtained by each TCN model on the In-Lab dataset. Results obtained for each TCN net on the In-Lab dataset are expressed in terms of mean ± standard deviation. Unsigned residuals are considered for MAE, while signed residuals are considered for ME (see Sub-subsection 5.4.1).

Similarly, Tables 6.11 and 6.12 show the absolute and relative errors in the estimation of the SS and DS times.

SS time errors - TCN			
Repetition	SET	MAE (s)	ME (s)
1	TRS	0,03±0,03 (4,78±4,59 %)	-0,01±0,04 (-1,10±6,54 %)
	VS	0,03±0,03 (4,87±3,63 %)	-0,02±0,04 (-2,43±5,57 %)
	TS	0,05±0,04 (9,34±8,72 %)	-0,04±0,05 (-8,14±9,86 %)
2	TRS	0,04±0,04 (5,51±5,08 %)	-0,02±0,05 (-2,91±6,91 %)
	VS	0,04±0,04 (6,36±5,05 %)	-0,03±0,05 (-4,73±6,61 %)
	TS	0,07±0,05 (13,34±10,96 %)	-0,07±0,06 (-12,60±11,80 %)
3	TRS	0,03±0,03 (5,16±4,50 %)	-0,02±0,04 (-2,25±6,47 %)
	VS	0,04±0,04 (6,02±5,50 %)	-0,03±0,05 (-3,58±7,33 %)
	TS	0,06±0,05 (10,40±9,99 %)	-0,05±0,05 (-9,75±10,63 %)
4	TRS	0,03±0,03 (4,73±4,79 %)	-0,01±0,04 (-1,36±6,59 %)
	VS	0,05±0,03 (7,09±3,89 %)	-0,04±0,03 (-6,94±4,16 %)
	TS	0,06±0,05 (11,18±11,04 %)	-0,05±0,06 (-10,29±11,87 %)
5	TRS	0,03±0,03 (5,46±4,80 %)	-0,01±0,05 (-0,40±7,26 %)
	VS	0,04±0,03 (5,67±4,88 %)	0,00±0,05 (0,42±7,47 %)
	TS	0,05±0,04 (8,37±7,97 %)	-0,02±0,06 (-4,72±10,56 %)

Table 6.11: Time errors between target and predicted SS times obtained by each TCN model on the In-Lab dataset. Results obtained for each TCN net on the In-Lab dataset are expressed in terms of mean ± standard deviation. Unsigned residuals are considered for MAE, while signed residuals are considered for ME (see Sub-subsection 5.4.1). Values between parentheses denote the error with respect to the stride time.

DS time errors - TCN			
Repetition	SET	MAE (s)	ME (s)
1	TRS	0,03±0,03 (4,62±4,05 %)	0,01±0,04 (0,99±6,06 %)
	VS	0,03±0,03 (4,99±3,89 %)	0,02±0,04 (2,38±5,86 %)
	TS	0,06±0,04 (9,07±6,08 %)	0,04±0,05 (6,61±8,70 %)
2	TRS	0,03±0,04 (5,33±4,65 %)	0,02±0,04 (2,83±6,48 %)
	VS	0,04±0,04 (6,35±4,73 %)	0,03±0,05 (4,63±6,42 %)
	TS	0,08±0,05 (12,23±6,94 %)	0,07±0,06 (11,02±8,73 %)
3	TRS	0,03±0,03 (5,06±4,25 %)	0,02±0,04 (2,17±6,24 %)
	VS	0,04±0,04 (5,96±4,86 %)	0,03±0,05 (3,42±6,88 %)
	TS	0,06±0,04 (10,20±6,41 %)	0,05±0,06 (8,17±8,86 %)
4	TRS	0,03±0,03 (4,43±4,32 %)	0,01±0,04 (1,27±6,06 %)
	VS	0,04±0,03 (6,93±3,45 %)	0,04±0,03 (6,86±3,60 %)
	TS	0,06±0,04 (9,52±5,56 %)	0,05±0,04 (8,73±6,73 %)
5	TRS	0,03±0,03 (5,32±4,41 %)	0,01±0,04 (0,31±6,91 %)
	VS	0,04±0,04 (5,60±4,81 %)	0,00±0,05 (-0,41±7,37 %)
	TS	0,05±0,04 (7,35±5,16 %)	0,02±0,05 (3,13±8,43 %)

Table 6.12: Time errors between target and predicted DS times obtained by each TCN model on the In-Lab dataset. Results obtained for each TCN net on the In-Lab dataset are expressed in terms of mean ± standard deviation. Unsigned residuals are considered for MAE, while signed residuals are considered for ME (see Sub-subsection 5.4.1). Values between parentheses denote the error with respect to the stride time.

6.1.2 Free-Living dataset

As described in Section 5.4, each one of the trained DeLe models is applied to the Free-Living TS. Since there are ten trained models - five TCNs and five LSTM networks - and one Free-Living TS, ten summaries of results have been drafted.

Classification metrics Table 6.13 shows the performance of classification for each of the ten trained DNNs on the Free-Living TS.

Classification metrics						
NET	Acc. (%)	Sens. (%)	Spec. (%)	PPV (%)	NPV (%)	F_1 (%)
LSTM1	71,23	89,39	27,26	74,85	51,47	81,47
LSTM2	72,44	89,26	31,72	75,99	54,94	82,09
LSTM3	71,62	88,56	30,59	75,55	52,48	81,54
LSTM4	72,26	86,77	37,12	76,97	53,67	81,57
LSTM5	71,55	90,55	25,53	74,65	52,74	81,83
TCN1	84,06	93,28	61,73	85,51	79,14	89,23
TCN2	82,79	92,99	58,10	84,31	77,38	88,44
TCN3	82,43	91,92	59,43	84,58	75,24	88,10
TCN4	84,05	93,52	61,11	85,35	79,56	89,25
TCN5	80,76	88,41	62,24	85,01	68,93	86,68

Table 6.13: Values of the adopted classification metrics for the binary SS/DS classification task obtained by each DNN on the Free-Living dataset. Each row of the Table refers to the performance achieved by the model (column 1) on the Free-Living TS. *Acc.*: Accuracy; *Sens.*: Sensitivity; *Spec.*: Specificity.

Values of the classification metrics are used to draw bar diagrams that allows a faster comparison between the performance of the different TCN (Figure 6.7) and LSTM models (Figure 6.8).

GED errors Table 6.14 shows the absolute and relative numbers of extra and missed events for each of the ten trained DNNs on the Free-Living dataset.

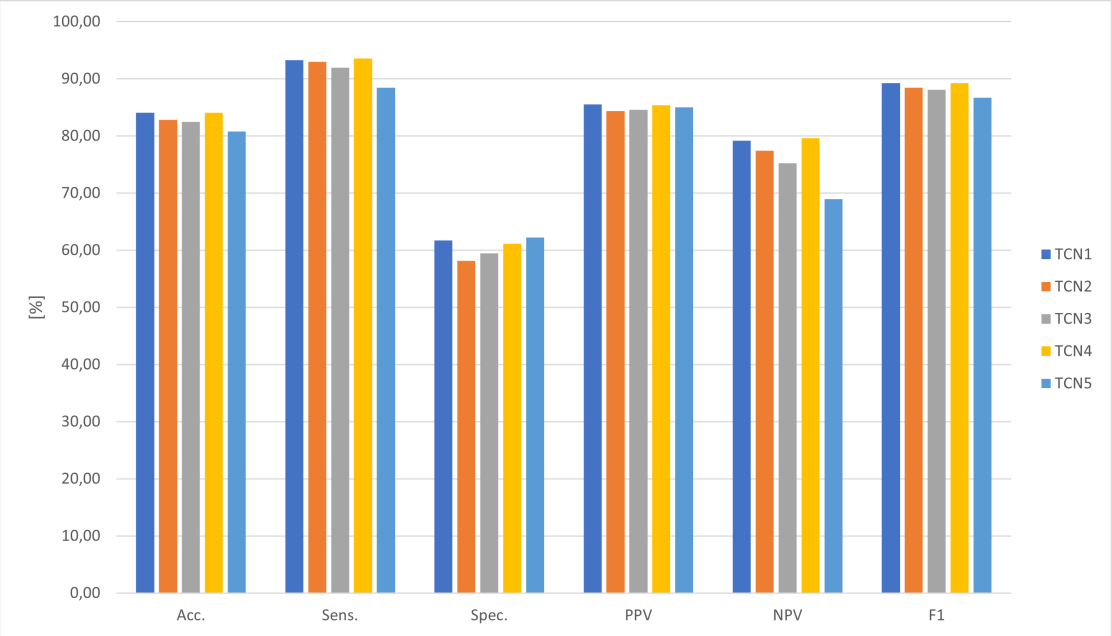


Figure 6.7: Bar diagram that summarizes the classification performance of the TCN models on the Free-Living test set. Each group of bars denotes the values of the corresponding metric for each of the five LSTM models. The present graph was designed on Microsoft®Excel®.

Missed/extra events				
NET	**extra IC**	**missed IC**	**extra FC**	**missed FC**
LSTM1	4225 (15,47 %)	8001 (29,30 %)	4368 (16,00 %)	8111 (29,70 %)
LSTM2	3962 (14,51 %)	7618 (27,90 %)	4033 (14,77 %)	7668 (28,08 %)
LSTM3	3815 (13,97 %)	7504 (27,48 %)	3923 (14,37 %)	7596 (27,82 %)
LSTM4	4455 (16,31 %)	6358 (23,28 %)	4544 (16,64 %)	6392 (23,41 %)
LSTM5	3736 (13,68 %)	9300 (34,06 %)	3889 (14,24 %)	9400 (34,42 %)
TCN1	550 (2,01 %)	300 (1,10 %)	540 (1,98 %)	287 (1,05 %)
TCN2	806 (2,95 %)	218 (0,80 %)	819 (3,00 %)	208 (0,76 %)
TCN3	635 (2,33 %)	252 (0,92 %)	630 (2,31 %)	238 (0,87 %)
TCN4	572 (2,09 %)	244 (0,89 %)	556 (2,04 %)	223 (0,82 %)
TCN5	694 (2,54 %)	213 (0,78 %)	688 (2,52 %)	202 (0,74 %)

Table 6.14: Number of missed and extra gait events (FCs and ICs) on the Free-Living dataset for each DNN. In parentheses, the percentage of misclassified events is reported i.e., the ratio between the number of extra/missed events and the total number of events (see Sub-subsection 5.4.1).

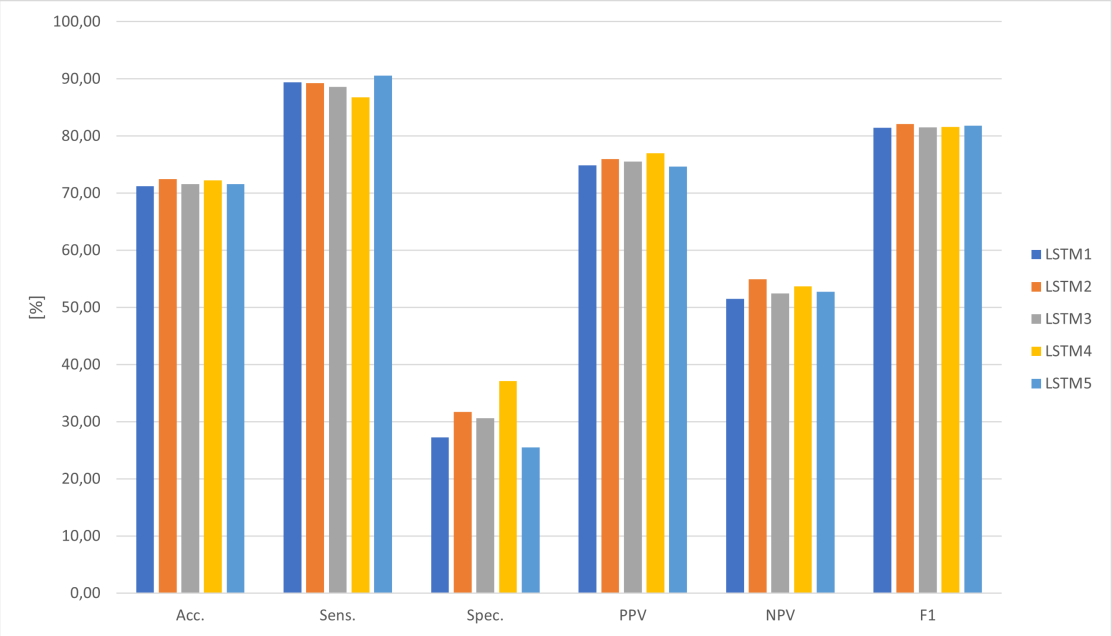


Figure 6.8: Bar diagram that summarizes the classification performance of the LSTM models on the Free-Living test set. Each group of bars denotes the values of the corresponding metric for each of the five LSTM models. The present graph was designed on Microsoft®Excel®.

Next, the time errors between the target events and the detected events are computed in terms of MAE and ME (Table 6.15).

	ICs		FCs	
NET	**MAE (s)**	**ME (s)**	**MAE (s)**	**ME (s)**
LSTM1	0,09±0,08	0,04±0,11	0,08±0,08	0,00±0,11
LSTM2	0,08±0,08	0,04±0,10	0,07±0,07	0,00±0,10
LSTM3	0,08±0,08	0,04±0,11	0,08±0,07	0,01±0,11
LSTM4	0,08±0,08	0,03±0,10	0,07±0,07	0,00±0,10
LSTM5	0,09±0,08	0,04±0,11	0,08±0,08	0,00±0,11
TCN1	0,05±0,04	0,05±0,04	0,04±0,04	0,02±0,05
TCN2	0,06±0,04	0,05±0,05	0,04±0,04	0,02±0,05
TCN3	0,05±0,05	0,05±0,05	0,04±0,04	0,02±0,05
TCN4	0,05±0,04	0,05±0,04	0,04±0,04	0,01±0,05
TCN5	0,05±0,04	0,05±0,05	0,05±0,05	0,04±0,06

Table 6.15: Time error between the time steps at which predicted and target gait events occur obtained by each DNN on the Free-Living dataset. Each pair of values (mean ± standard deviation) denotes the result of a model (column 1) on the Free-Living dataset. Unsigned residuals are considered for MAE, while signed residuals are considered for ME (see Sub-subsection 5.4.1).

Errors on the temporal parameters Table 6.16 shows the values of MAE and ME in the estimation of the step and stride times.

	Step time errors		Stride time errors	
NET	**MAE (s)**	**ME (s)**	**MAE (s)**	**ME (s)**
LSTM1	0,11±0,11	0,00±0,16	0,10±0,11	0,00±0,15
LSTM2	0,10±0,11	0,00±0,15	0,09±0,10	0,00±0,14
LSTM3	0,10±0,11	0,00±0,15	0,10±0,11	0,00±0,14
LSTM4	0,10±0,11	0,00±0,15	0,09±0,10	0,00±0,14
LSTM5	0,11±0,12	0,00±0,16	0,10±0,11	0,00±0,15
TCN1	0,04±0,05	0,00±0,06	0,02±0,04	0,00±0,05
TCN2	0,04±0,05	0,00±0,06	0,02±0,04	0,00±0,05
TCN3	0,04±0,05	0,00±0,07	0,02±0,05	0,00±0,06
TCN4	0,04±0,05	0,00±0,06	0,02±0,04	0,00±0,05
TCN5	0,04±0,05	0,00±0,06	0,02±0,04	0,00±0,05

Table 6.16: Time errors between target and predicted step and stride times obtained by each DNN on the Free-Living dataset. Results obtained for each net on the Free-Living dataset are expressed in terms of mean ± standard deviation. Unsigned residuals are considered for MAE, while signed residuals are considered for ME (see Sub-subsection 5.4.1).

Similarly, Tables 6.17 and 6.18 show the absolute and relative errors in the estimation of the SS and DS times.

DS time errors		
NET	**MAE (s)**	**ME (s)**
LSTM1	0,09±0,07 (14,58±10,43 %)	0,05±0,10 (8,20±15,94 %)
LSTM2	0,08±0,08 (13,14±10,20 %)	0,04±0,10 (6,29±15,40 %)
LSTM3	0,08±0,08 (13,67±10,53 %)	0,04±0,10 (6,48±16,00 %)
LSTM4	0,08±0,08 (13,54±10,92 %)	0,03±0,11 (4,74±16,74 %)
LSTM5	0,08±0,08 (14,37±10,78 %)	0,05±0,11 (8,41±15,87 %)
TCN1	0,05±0,05 (7,63±6,46 %)	0,03±0,06 (4,82±8,76 %)
TCN2	0,05±0,05 (8,00±6,82 %)	0,04±0,06 (5,78±8,78 %)
TCN3	0,05±0,05 (8,02±6,55 %)	0,03±0,07 (4,53±9,31 %)
TCN4	0,05±0,05 (7,68±6,40 %)	0,03±0,06 (5,29±8,49 %)
TCN5	0,05±0,05 (8,15±6,35 %)	0,01±0,07 (1,18±10,26 %)

Table 6.17: Time errors between target and predicted DS times obtained by each DNN on the Free-Living dataset. Results obtained for each net on the Free-Living dataset are expressed in terms of mean ± standard deviation. Unsigned residuals are considered for MAE, while signed residuals are considered for ME (see Subsubsection 5.4.1). Values between parentheses denote the error with respect to the stride time.

SS time errors		
NET	**MAE (s)**	**ME (s)**
LSTM1	0,12±0,12 (20,31±17,89 %)	-0,03±0,16 (-6,43±26,29 %)
LSTM2	0,11±0,11 (18,53±16,76 %)	-0,03±0,15 (-5,07±24,46 %)
LSTM3	0,11±0,11 (18,88±17,54 %)	-0,03±0,15 (-5,00±25,28 %)
LSTM4	0,11±0,11 (17,99±16,57 %)	-0,02±0,15 (-3,97±24,13 %)
LSTM5	0,12±0,12 (20,00±17,77 %)	-0,03±0,16 (-5,73±26,13 %)
TCN1	0,05±0,05 (8,38±8,34 %)	-0,03±0,06 (-5,26±10,58 %)
TCN2	0,05±0,06 (9,13±8,86 %)	-0,04±0,07 (-6,21±11,11 %)
TCN3	0,05±0,06 (8,95±9,05 %)	-0,03±0,07 (-4,98±11,71 %)
TCN4	0,05±0,06 (9,34±9,06 %)	-0,03±0,07 (-5,81±11,65 %)
TCN5	0,05±0,05 (9,05±8,36 %)	-0,01±0,07 (-1,65±12,21 %)

Table 6.18: Time errors between target and predicted SS times obtained by each DNN on the Free-Living dataset. Results obtained for each net on the Free-Living dataset are expressed in terms of mean ± standard deviation. Unsigned residuals are considered for MAE, while signed residuals are considered for ME (see Sub-subsection 5.4.1). Values between parentheses denote the error with respect to the stride time.

6.2 Stride speed

The second task to be evaluated is the estimation of the stride speed. In particular, a set of MaLe models were employed: four GPR models and one linear SVM.
In this Section, the errors and general performance for the developed models are presented. First, the models are evaluated on the strides of the In-Lab dataset segmented by the INDIP standard; then, the models are evaluated on the strides from the In-Lab and Free-Living datasets segmented by the DeLe models.

6.2.1 In-Lab dataset

First, results obtained on the In-Lab dataset are presented including both the results on the strides segmented by the INDIP standard and by TCN 1.

Strides segmented by the INDIP standard

Table 6.19 summarizes the performance in the estimation of the stride speed achieved by the MaLe models on the strides segmented by the INDIP standard. As anticipated in Sub-subsection 5.4.2, five plots were drawn to visually represent

Model type	Kernel	SET	RMSE (m/s)	R^2	MSE (m^2/s^2)	MAE (m/s)	Prediction sp. (obj/s)	Training time (s)
GPR	Exp	CS	0,07	0,95	0,00	0,05	4500	590,96
		TS	0,07	0,93	0,01	0,01		
	Rat	CS	0,07	0,95	0,00	0,05	3100	700,2
		TS	0,08	0,92	0,01	0,06		
	Exp^2	CS	0,07	0,95	0,00	0,05	4800	197,31
		TS	0,08	0,90	0,01	0,06		
	5/2	CS	0,07	0,95	0,00	0,05	4500	228,38
		TS	0,08	0,91	0,01	0,06		
SVM	Linear	CS	0,09	0,91	0,01	0,07	6200	25,87
		TS	0,09	0,90	0,01	0,06		

Table 6.19: Values of the error metrics for each MaLe model. Predictors used for regression were derived from the strides of the In-Lab dataset segmented by the INDIP standard. *CS*: Construction set (Participants 1-10); *TS*: Test set (Participant 11); *Prediction sp.*: Prediction speed; *Exp*: exponential kernel; *Rat*: rational quadratic kernel; *Exp^2*: squared exponential; *5/2*: Matern 5/2.

the achieved results of regression for each model. Since the five MaLe models have similar performance, only the five plots referring to the exponential GPR model are presented for the sake of simplicity.
Figure 6.9 shows the response plot referring to the results of regression for the exponential GPR model on the strides of the construction set segmented by the INDIP standard.

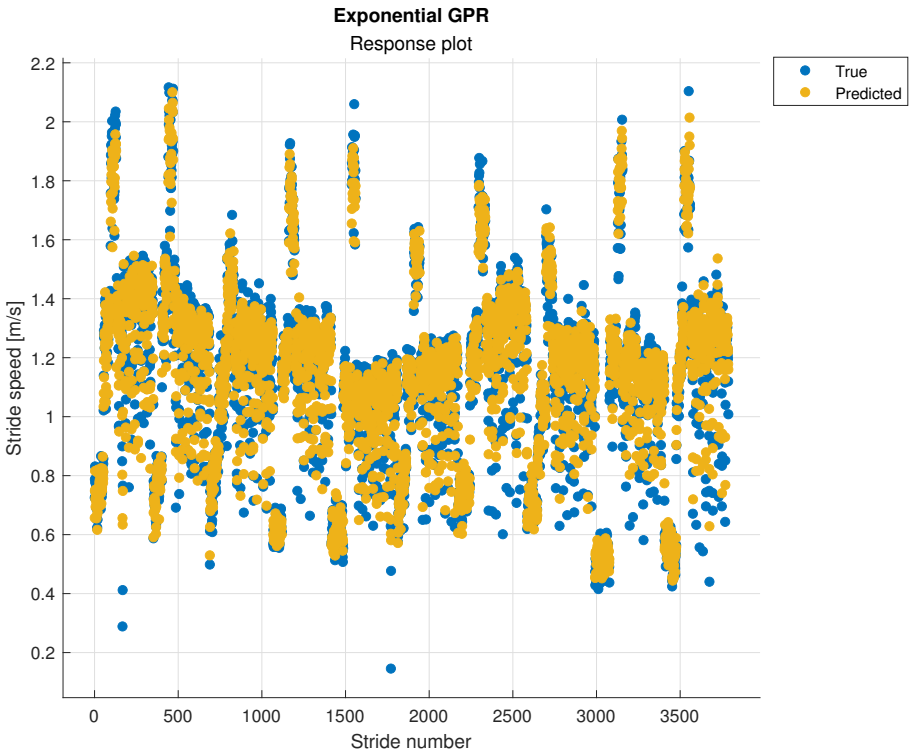


Figure 6.9: Plot of the target response and of the response estimated by the exponential GPR model for the estimation of the stride speed on the In-Lab construction set. Predicted and actual stride speeds are plotted versus the stride number. Similar plots have been drawn for the other GPR and SVM models as well.

Figure 6.10 shows the actual versus predicted responses of regression for the exponential GPR model on the strides of the construction set segmented by the INDIP standard.

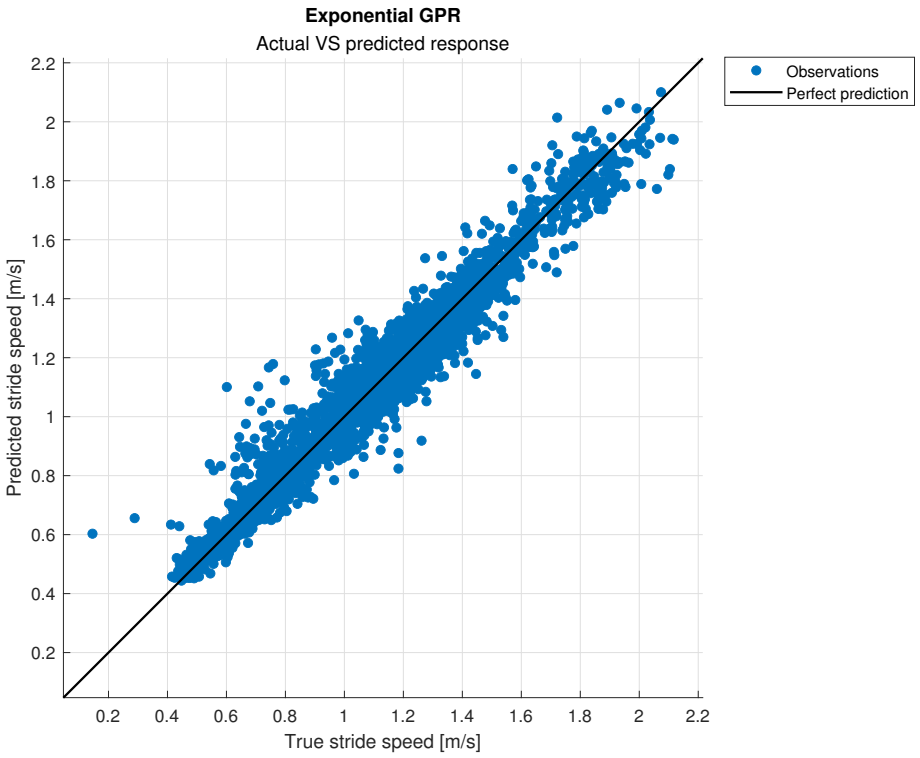


Figure 6.10: Plot of the actual responses versus responses predicted by the exponential GPR model for the estimation of the stride speed. Predicted stride speed is plotted versus the actual speed of the strides of the construction set segmented by the INDIP standard. The black line represents the ideal mapping between predicted and target response. Similar plots have been drawn for the other GPR and SVM models as well.

Figure 6.11 shows the residuals of regression achieved by the exponential GPR model on the strides of the construction set segmented by the INDIP standard. Figure 6.12 shows the actual versus predicted responses of regression for the exponential GPR model on the strides of the TS segmented by the INDIP standard.

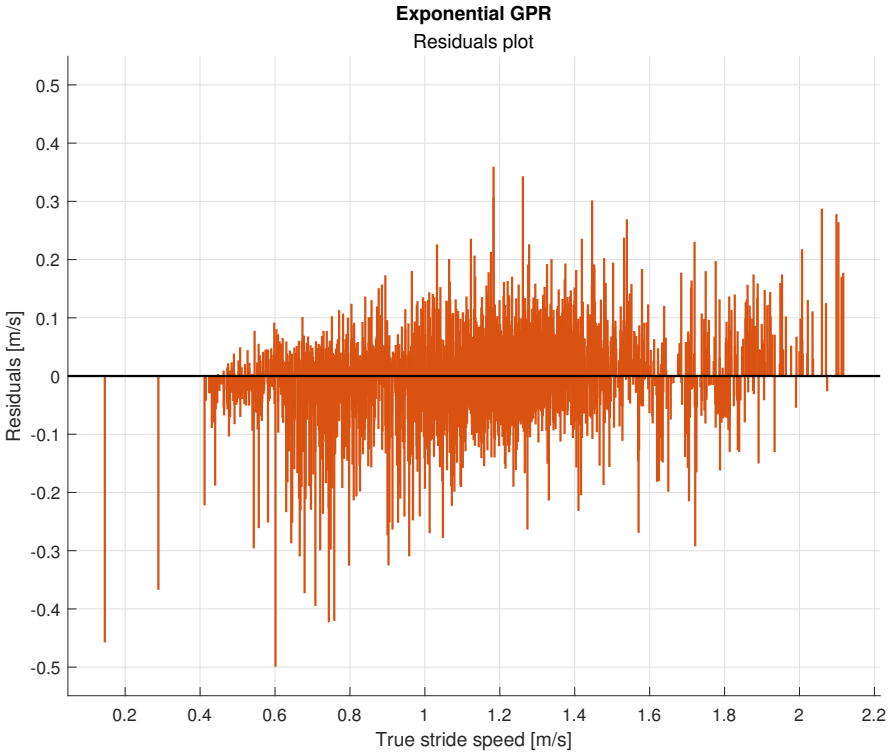


Figure 6.11: Plot of the residuals of regression achieved by the exponential GPR model for the estimation of the stride speed. The stride speed residuals are plotted versus the true value of the stride speed for the strides of the In-Lab construction set segmented by the INDIP standard. Similar plots have been drawn for the other GPR and SVM models as well.

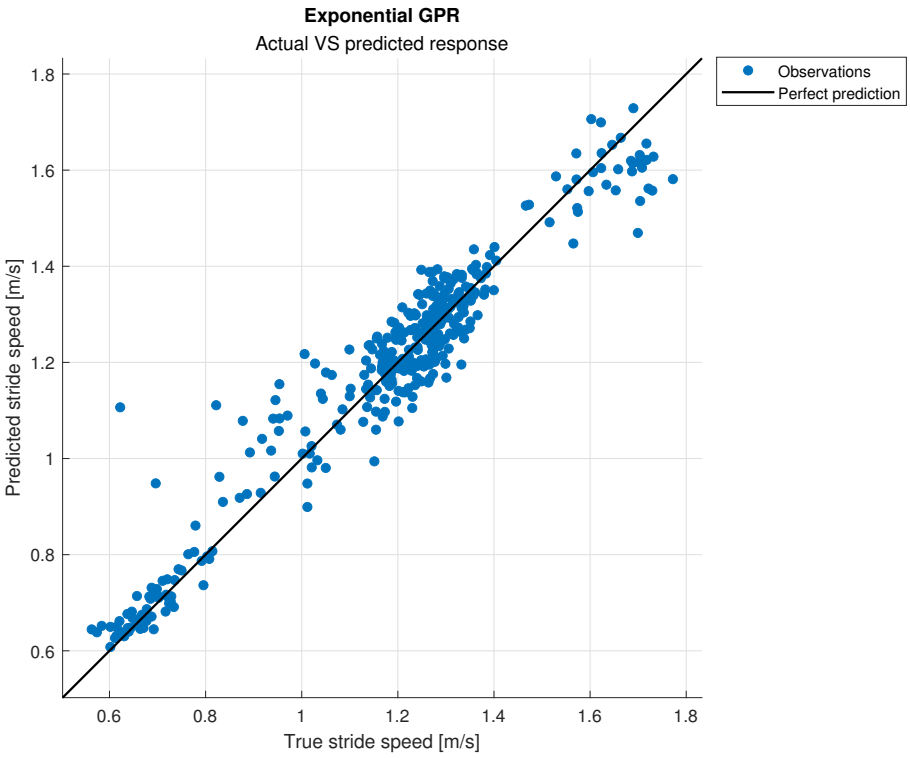


Figure 6.12: Plot of the actual responses versus responses predicted by the exponential GPR model for the estimation of the stride speed. Predicted stride speed is plotted versus the actual speed of the strides of the TS segmented by the INDIP standard. The black line represents the ideal mapping between predicted and target response. Similar plots have been drawn for the other GPR and SVM models as well.

Figure 6.13 shows the residuals of regression achieved by the exponential GPR model on the strides of the TS segmented by the INDIP standard.

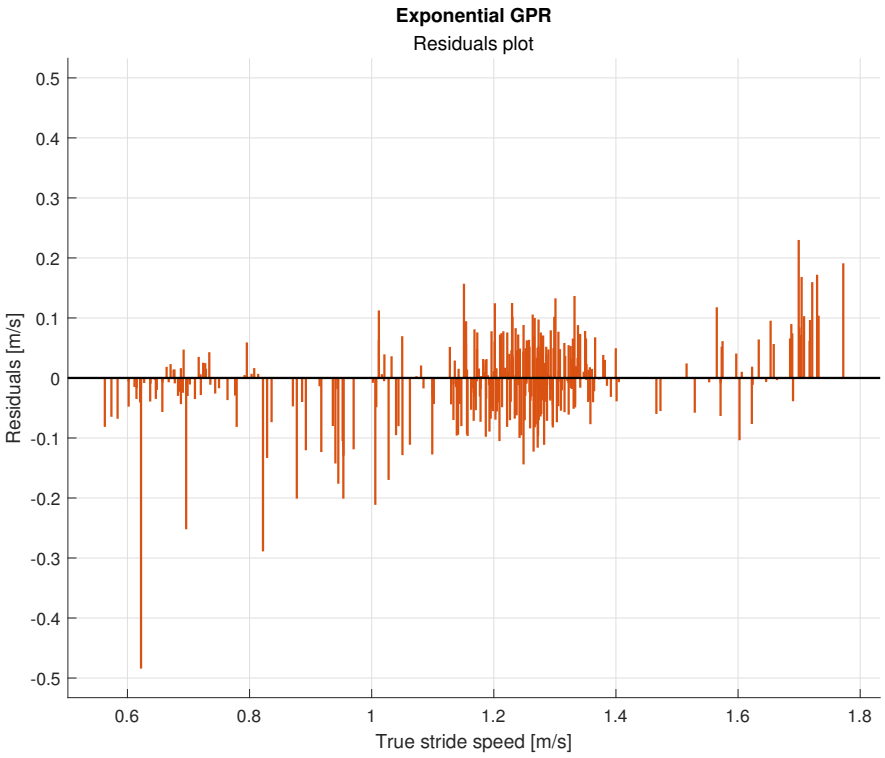


Figure 6.13: Plot of the residuals of regression achieved by the exponential GPR model for the estimation of the stride speed. The stride speed residuals are plotted versus the true value of the stride speed for the strides of the In-Lab TS segmented by the INDIP standard. Similar plots have been drawn for the other GPR and SVM models as well.

Strides segmented by TCN 1

Given its superior performance (see Chapter 7), the first TCN model (TCN 1) was chosen as the best candidate for the segmentation of strides functional to the stride speed estimation. Table 6.20 summarizes the performance for the estimation of the stride speed achieved by the MaLe models. The models are tested on the strides of the In-Lab dataset segmented by TCN 1.

6.2.2 Strides from the Free-Living temporal parameters dataset

Table 6.21 summarizes the performance for the estimation of the stride speed achieved by the MaLe models. The models are tested on the strides of the Free-Living dataset segmented by TCN 1.

Model type	Kernel	SET	MAE (m/s)	MSE (m^2/s^2)	RMSE (m/s)	MAPE	R^2	adR^2	ρ
GPR	Exp	CS	0,05	0,00	0,07	0,05	0,95	0,89	0,97
		TS	0,07	0,01	0,08	0,06	0,90	0,70	0,97
	Rat	CS	0,05	0,00	0,07	0,05	0,94	0,89	0,97
		TS	0,07	0,01	0,09	0,07	0,89	0,66	0,97
	Exp^2	CS	0,05	0,01	0,07	0,05	0,94	0,88	0,97
		TS	0,08	0,01	0,10	0,07	0,88	0,63	0,97
	5/2	CS	0,05	0,01	0,07	0,05	0,94	0,89	0,97
		TS	0,08	0,01	0,10	0,07	0,88	0,63	0,97
SVM	Linear	CS	0,07	0,01	0,09	0,07	0,91	0,82	0,95
		TS	0,06	0,01	0,08	0,05	0,92	0,74	0,97

Table 6.20: Values of the error metrics for each MaLe model. Predictors used for regression were derived from the strides of the In-Lab dataset segmented using TCN 1. *CS*: Construction set (Participants 1-10); *TS*: Test set (Participant 11); *Exp*: exponential kernel; *Rat*: rational quadratic kernel; *Exp²*: squared exponential; *5/2*: Matern 5/2.

Type of model	Kernel	MAE (m/s)	MSE (m^2/s^2)	RMSE (m/s)	MAPE	R^2	adR^2	ρ
GPR	Exp	0,10	0,02	0,13	0,09	0,61	0,36	0,80
	Rat	0,10	0,02	0,13	0,09	0,62	0,38	0,80
	Exp2	0,12	0,02	0,16	0,11	0,48	0,22	0,74
	Matern	0,12	0,02	0,15	0,10	0,53	0,27	0,77
SVM	Linear	0,11	0,02	0,14	0,10	0,56	0,31	0,79

Table 6.21: Values of the error metrics for each MaLe model. Predictors used for regression were derived from the strides of the Free-Living dataset segmented using TCN 1. *Exp*: exponential kernel; *Rat*: rational quadratic kernel; *Exp²*: squared exponential; *5/2*: Matern 5/2.

Chapter 7

Discussion

The current study aimed at validating a MaLe approach to the estimation of STP from a single MIMU attached to the left side of the head. Data from the H-MIMU were used for training several DNNs to detect gait events from walking Trials performed by HY subjects. Eleven Participants walked along a 12 m indoor path at three different self-selected walking speeds. Furthermore, other three Participants walked both indoor and outdoor for 2.5-hour sessions unsupervised. The gait events predicted by the networks were compared with those detected by a common reference method, i.e., the INDIP system. Then, a set of relevant stride-specific temporal parameters were derived. In addition, a set of models of MaLe was trained to infer the value of the stride speed from the acceleration signal of the segmented strides.

7.1 Temporal parameter estimation

First, the DeLe models for the assessment of temporal parameters were evaluated. To assess the robustness of the proposed architectures, five models with different TRS/VS combinations were trained for each type of architecture.

7.1.1 Performance on the In-Lab dataset

Initially, the models were tested on the In-Lab TRS and VS used for training, and on the In-Lab TS. Such test aimed at assessing the performance of the models on data equal (or similar) to the ones used for training.

Performance of LSTM networks

At the beginning, a set of five LSTM networks were trained to assess general suitability of the LSTM architecture to the task and evaluate its robustness.

A first measure for the models performance was provided by the classification metrics (Table 6.1). For all the LSTM networks, accuracy was significantly high on the TRS (min: 87.0 %, max: 90.6 %), while - as expected - lower values of accuracy were achieved when the models were tested on unseen data of the TS. In addition, sensitivity attained much higher percentages than specificity (max. sensitivity on the TRS: 95.4 %, max. specificity on the TRS: 78.9 %), meaning that the LSTM models tend to oversegment the SS phase with respect to the DS phase.
The average accuracy was best for model 3 (LSTM 3); however, all the 5 LSTM models showed similar average values of accuracy (see Figure 6.1), confirming that DeLe approach is relatively immune to variations of the dataset, as long as the dataset size is not reduced. The average sensitivity value for model 3 (LSTM 3) was lower than the other models; however, LSTM 3 reached a higher average value of specificity, resulting in a more balanced performance.
In general, the performance achieved by all the five LSTM models were quite similar, suggesting that the LSTM architecture is robust enough i.e., its performance does not depend much on the data used for training.
The number of extra and missed gait events is a direct consequence of the classification output, as instants at which gait events occur are retrieved by simply differentiating the predicted response signal. Results presented in Table 6.2 show that overall the LSTM networks lead to a higher number of missed events with respect to extra events, meaning that the LSTM networks struggle to catch all the events detected by the INDIP standard. In general, the rate of mis-classification for ICs is comparable to that obtained for FCs; therefore, the model tends to commit similar mis-classification errors on both types of events. LSTM 5 showed a lower number of extra events both for FCs and ICs with respect to the other LSTM models while the best performance regarding missed events were achieved by LSTM 2.
Another relevant aspect regarding the models' performance for GED was represented by the time errors between the predicted and target occurring time steps of gait events (Table 6.3). Such deviations were computed after that target and predicted gait events were matched (see Sub-subsection 5.2.2); therefore, they refer only to matched gait events. As it can be observed from Table 6.3, average time errors were extremely small (0.02 s to 0.10 s for MAE, -0.01 s to 0.05 s for ME across all LSTM models), suggesting that LSTM networks tend to detect a gait event within a range of ± 0.1 s (10 samples) around the occurring of the true gait event. For most of the LSTM models, ME values showed a positive bias, meaning that the models tend to detect gait events in advance with respect to true gait events. In general, errors for FCs were comparable with errors for ICs. Notice how ME values are slightly lower than MAE, as signed errors tend to annihilate with each other.

At this point, errors in the estimation of step, stride, SS and DS times were computed. Those errors directly derive from the time deviations between predicted and true gait events. Since the definition of temporal parameters relies upon two separate gait events, errors in the estimation of the time duration for that parameter will reflect errors referring to both the events, leading to greater errors. In general, errors on the step and on the stride times were comparable and lower than 150 m s (Table 6.4). Similar values were obtained for the estimation of the SS and DS phases time duration (Tables 6.6 and 6.5). For the LSTM models, standard deviations for the errors referring to temporal parameters identified quite large ranges, suggesting that the error in the estimation of temporal parameters is not consistent. Notice that, while the sign of ME for the SS time was negative for most of the models, it was positive for DS times, meaning that the outcome of classification yields to overestimate the SS with respect to the DS.

Performance of TCNs

Beyond the five LSTM networks, five TCNs were trained as well to assess if an approach based on dilated causal 1D convolutions could bring any advantages with respect to RNNs.
A first measure for the models' performance was provided by the classification metrics (Table 6.7). For all the TCN networks, the values of accuracy were slightly higher than the ones achieved by corresponding LSTM networks trained on the same In-Lab dataset. Over the TRS, all the TCNs attained levels of accuracy higher than 92.4 %, with encouraging performance on the TS as well (≥ 83.7 % for all the TCNs). In addition, TCNs showed much more balanced performance with respect to LSTM networks. Especially, as visually shown in Figure 6.4, TCN 1 achieved the best trade-off between accuracy ((90.7 ± 3.7) %), sensitivity ((95.8 ± 0.8) %) and specificity ((79.83 ± 7.0) %) over the three In-Lab datasets. Also TCN 5 has balanced sensitivity and accuracy; however, it reached lower levels of accuracy and F_1-score than TCN 1 (Figures 6.5 and 6.6). Even if TCNs seem to show more balanced performance on the SS/DS classification task, the models were still more sensitive to the SS class rather than to the DS class.
In general, the performance achieved by all the five TCN models were quite similar, suggesting that the TCN architecture is robust enough i.e., its performance does not depend much on the data used for training.
Regarding the task of GED, the number of extra and missed events were remarkably lower than the ones achieved by the LSTM networks (Table 6.8). The average percentage of missed FCs and ICs was 0.25 %, meaning that the TCN models correctly detect gait events in the majority of cases. Still, the TCN models strive to detect more events than the ones that are actually present, as the number of extra events was in general slightly higher than the one of missed events. For most

of the models, the numbers of missed and extra events on the In-Lab TS were comparable to or even lower than the corresponding numbers on the In-Lab TRS, suggesting that TCNs have good generalization skill.
The values of time deviation between the associated predicted and true gait events achieved by the TCNs were extremely low and balanced between FCs and ICs, meaning that the prediction error on the time step at which gait events occur hardly exceeds a couple of hundredths of a second (Table 6.9). The signs of ME for the FCs and the ICs had opposite sign. In particular, models tend to detect ICs in advance and FCs late, meaning that most of the ICs are shifted in a direction and most of the FCs are shifted in the opposite direction.
As a direct consequence of the limited errors on the estimation of gait events, errors on the time duration of step times and stride times were also low with respect to LSTM models (Table 6.10).
Similarly to LSTM networks, the SS and DS times represent the temporal parameters with the greater errors, and for which the errors on the In-Lab TS are much higher than the ones on the TRS and VS (Tables 6.11 and 6.12). Such difference may be due to the fact that the estimation of the SS and DS time requires the knowledge of the time instants at which both FCs and ICs occur, while the step and stride time only depend on the ICs. Since ICs and FCs are shifted in opposite directions, the duration of a SS or DS phase has a higher probability to be shrunk or stretched, therefore the greater errors in the estimation of the SS and DS times. However, values of standard deviation are much more limited than the ones observed with LSTM networks.

7.1.2 Performance on the Free-Living dataset

Typically, methods validated only on supervised walking sessions struggle to achieve the same performance when applied to unsupervised outdoor walking [13]. Therefore, models were tested also on data from the Free-Living dataset, in order to assess the extent to what the developed methods are able to perform when applied to unsupervised walking.

Performance of LSTM networks

As shown in Table 6.13, classification accuracy for all the LSTM models on the Free-Living dataset did not exceed 72 %. Similarly to the In-Lab dataset, sensitivity values are significantly higher than specificity values, meaning that the performance of the LSTM models is strongly unbalanced in favor of the SS class. This trend is confirmed by the bar diagram in Figure 6.8, which clearly shows how specificity values did not match the corresponding sensitivity values. All the models showed

similar values of classification metrics; however, LSTM 4 showed a more balanced performance, as it reached lower sensitivity and higher specificity than the other LSTM models.
The performance of the LSTM models also get worse on the side of the detection of gait events. Once again, the number of missed events was considerably higher than the number of extra events; moreover, for data recorded during walking in free-living conditions, both numbers of extra and missed events were remarkably greater than the previous case of standardized walking.
Regarding the time errors between predicted and true gait events, a slight increase in the MAE (from 0.07 s to 0.09 s across all LSTM models) and ME (from 0.00 s to 0.04 s across all LSTM models) was observed for data recorded in free-living conditions. The main issue is probably represented by the increase in the average standard deviation of errors (up to 0.11 s across all LSTM models), which is an indicator of the greater variability of data contained in the Free-Living dataset.
Concerning step and stride times, MAE values seemed to be significantly higher than ME values, which instead were near to zero (Tables 6.16, 6.18, 6.17). Thus, the number of times that the LSTM models tend to overestimate step and stride times is averagely equal to the number of times that step and stride times are underestimated. Nevertheless, the increased variability in the morphology and characteristics of the predictors belonging to the Free-Living dataset yield to lower performance and greater errors, as denoted by the increased standard deviation values.
Similarly, SS and DS times attained greater errors than on the In-Lab dataset (from 0.08 s to 0.12 s for MAE, from −0.03 s to 0.05 s for ME across all LSTM models). Again, while the sign of ME for the SS time was negative for most of the models, it was positive for DS times, meaning that the predictions made by the models yield to overestimate the SS with respect to the DS.

Performance of TCNs

Once that TCNs were tested on the In-Lab dataset, their performance were evaluated on the Free-Living dataset.
TCN 1 confirmed to have the best trade-off between accuracy (84.1 %), sensitivity (93.3 %) and specificity (61.73 %) also for the Free-Living dataset (Figure 6.7) compared to LSTM models. Moreover, similarly to what observed for the In-Lab dataset, levels of accuracy achieved by the TCNs on the Free-Living dataset were consistently higher than the ones achieved by LSTM networks (Table 6.13).
As shown by Tables 6.2) and 6.3, TCNs seem to have performance on the Free-Living dataset similar to one on the In-Lab dataset, both in terms of missed and extra events (from 1.98 % to 3.00 % for extra events, from 0.74 % to 1.10 % for missed events across all TCN models) and time deviations (from 0.04 s to 0.05 s for

MAE, from 0.01 s to 0.05 s for ME across all TCN models).
As shown in Tables 6.16, 6.17 and 6.18, the MAE values in the estimation of the temporal parameters were much reduced with respect to the ones committed by the LSTM models either in terms of step times (0.04 s), stride times (0.02 s), SS times (0.05 s) and DS times (0.05 s). Furthermore, errors obtained for the Free-Living dataset did not differ substantially from the ones obtained on the In-Lab dataset, suggesting that the trained TCN models do not overfit data and instead are able to generalize on unseen data with greater variability.

7.2 Stride speed estimation

The ultimate goal of the project was represented by the estimation of the stride speed. Actually, this parameter is a strong indicator of mobility and general wellness of the individual. While values of temporal parameters can be inferred from the inertial signals relatively easily, the estimation of the stride speed requires information regarding the space domain. Therefore, conventional methods for stride speed estimation generally exploit time integration of inertial time series - despite it being potentially affected from significant errors - or biomechanical models of gait that account to the anthropometric features of the individual. The challenge for the present study was to find a method for retrieving the value of the stride speed without any signal integration or assumption on walking by a set of MaLe regression models, in order to avoid the tedious task of estimating spatial parameters.

7.2.1 Results on the strides segmented by the INDIP standard

Table 6.19 shows the performance of the trained GPR and SVR models on the strides of the In-Lab dataset segmented by the INDIP standard. All the GPR models showed high performance in terms of errors (MAE $< 0.06\,\mathrm{m/s}$) and correlation coefficient ($R^2 > 0.90$). Performance achieved by the linear SVM regression model were also good; however SVR resulted in larger errors than the GPR models, as the process covariance was more hardly parameterized through a linear kernel. On the other hand, the SVR model required much less time for training (25.87 s). The prediction speeds of all the regression models were comparable and of the order of thousands of objects per second.
As shown in Figures 6.10 and 6.12, observations of the predicted vs actual response plot for the exponential GPR model were scattered symmetrically around the ideal line, both for the In-Lab construction set and the TS, suggesting that the model fits well the data distribution. In addition, residuals of regression were mainly

contained within a narrow range of $\pm 0.2\,\mathrm{m/s}$ (Figures 6.11 and 6.13) - except for some outliers.

7.2.2 Results on the strides segmented by TCN 1

Results achieved by the MaLe regression models on the strides of the In-Lab dataset segmented by TCN 1 strongly resembled to the ones achieved on strides segmented by the INDIP standard (Table 6.20).
Instead, a dramatic decay was observed when the models were tested on the strides of the Free-Living dataset segmented by TCN 1 (Table 6.21). While the increase in the value of errors is not drastic (MAE: $0.010\,\mathrm{m/s}$ - $0.012\,\mathrm{m/s}$), a severe decrease in correlation between the predictors and response was observed (R^2 <0.62, $\rho < 0.80$), suggesting that the models manage to explain only a small portion of the process variance. Notice that the values of adR^2 in Tables 6.20 and 6.21 are definitely smaller than the corresponding R^2, meaning that many of the employed features poorly contribute to the result of regression.

7.3 General considerations

The observations made in the previous Sections yield to some general considerations about the performance and limits of the developed models.
Regarding the detection of gait events, TCN models have shown better performance overall than the LSTM models. For the classification task on the In-Lab dataset, TCNs outperformed LSTM networks in terms of average accuracy (TCN: 90.66 %, LSTM: 79.88 %), sensitivity (TCN: 95.81 %, LSTM: 90.13 %) and specificity (TCN: 79.82 %, LSTM: 59.18 %). In addition, the TCN classifiers showed values of sensitivity and specificity that are more balanced with respect to LSTM networks, resulting in more balanced errors.
Concerning the detection of gait events, time deviations between the predicted and target gait events of the In-Lab dataset ranges between 0.02 s and 0.08 s. Such results match the performance achieved by Hwang et al. [117] in the H-MIMU-based GED task through peak detection methods and Galadeta et al. [102] in the GED task based on B-MIMU and F-MIMUs through TCNs.
As previously discussed, both LSTM and TCN seem to be more sensitive to SS rather than DS. As mentioned in Subsection 2.1.2, a phase of SS takes around 80 % GC, while a phase of DS takes around 20 % GC. As a consequence, the SS class is overly represented in the TRS with respect to the DS class, resulting in models that are better at recognizing SS rather than DS. To virtually reduce such unbalance, one could think to increase the number of DS time steps in the TRS by performing data augmentation or by performing a stratified sampling instead of block sampling. However, such approaches inevitably alter the natural sequence

of events happening at walking time. To prevent the ratio between SS and DS to be altered, one could instead think of applying weights during training in order to increase the relevance of the less represented class, in this case DS.
As expected, the performance of both the LSTM and TCN models show a deterioration at prediction time on unseen data belonging to the Free-Living dataset. However, while classification accuracy of the LSTM networks drops under 75 %, TCNs maintain levels of accuracy around 85 %, suggesting a good generalization skill of the TCN models.
On the side of the estimation of the stride speed, all the regression models achieved similar performance on the strides of the In-Lab dataset, attaining small errors (MAE: 0.01 m/s - 0.07 m/s) and high correlation (R^2: 0.90 - 0.95). Among all the trained MaLe models, the exponential GPR showed slightly more consistent performance with respect to other regression models. Nevertheless, correlation coefficient severely decreases ($R^2 < 0.6$) when regression models are asked to predict the stride speed from unseen data of the Free-Living dataset, suggesting that MaLe models only explain a small portion of the variance of the process. Such lack of generalization probably depends on the increased variability of gait in free-living conditions, which implies that the signals recorded in standardized conditions significantly differ from the ones recorded in free-living conditions. Actually, during every-day life, one performs a number of gestures and activities during walking that involve the head, such as looking at the traffic lights, or answering the phone. Moreover, gait alternates with other activities typical of daily life in which the contribute of head segment can significantly vary. Such gestures reflect on the head acceleration signal and behave as *confounding factors* for models that try to extract knowledge from the head signals, eventually decreasing their performance. Then, to improve the performance of regression models in the estimation of the stride speed, the TRS should be expanded by including data referred to gait in free-living conditions.
Some of the issues of the developed models for both GED and stride speed estimation could be addressed through hyperparameters optimization. Being one of the "first tries" in the direction of the assessment of STP with a single H-MIMU, the present work did not focus on a sensitivity study of hyperparameters; however, hyperparameters tuning is generally a required step to assess the full potential of an algorithm [14].
Training outcomes may benefit from a more thoughtful approach to data partition. In the present work, data have been randomly divided into TRS, VS and TS; however, stratification with unsupervised methods such as k-fold or dendrograms may help to craft datasets that evenly represent all the clusters of the data distribution. Finally, techniques of dimensionality reduction should be sought to exclude predictors that do not concur to - or even hinder - the outcome of prediction [14].

Chapter 8

Conclusions

8.1 General results

The approach proposed in the present thesis is based on the use of MaLe and DeLe techniques to predict the values of several STP from data recorded by a single H-MIMU in supervised and unsupervised conditions.
The models have been developed with data acquired through the INDIP system, a reliable multi-sensor system - to which the H-MIMU belongs. Data recorded by the H-MIMU were used to extract the predictors, while data recorded by the remaining sensors of the INDIP system were used to define the GS values of the target STP.
The output of the thesis work is represented by a set of DeLe and MaLe models that are supposed to work sequentially i.e., the output of the DeLe model for GED represents the input to the MaLe model for the stride speed estimation. When used together, they define a complete pipeline for the estimation of spatio-temporal parameters from inertial raw data (see Section 5.1).
The methods have been validated on a number of HY subjects acquired in different walking conditions, included walking at fast and slow speeds and walking outside in a free-living environment. Results showed that the STP estimation errors are consistent with those highlighted by other similar studies that exploit other single or multi-sensor configurations of MIMUs [12][102]. The models achieved good performance for all the analyzed walking speeds. In addition, the methods showed robust performance when trained on different data, suggesting that the proposed architectures are appropriate to the requested tasks.
Since most of the related studies exploit one or more MIMUs positioned at lower body sites (shanks, feet, pelvis, etc...) [96], [8], an approach based on signals recorded at the head may be considered an innovation in the field, and it is believed to pave the way for further research regarding the exploitation of H-MIMUs for gait analysis.

8.2 Future directions and related research

Magnetic-inertial sensors are able to track the human motion with a degree of repeatability and accuracy that matches the one of stereophotogrammetry. In addition, magneto-inertial sensing technology is suitable to be employed during daily life and prolonged observation time. Inertial sensors are typically encased in units of few cubic millimeters or less, therefore, they may be integrated in rigid or flexible electronic substrates to enable continuous and non-invasive data collection. With regard to this, the proposed methods - which are based on the data recorded by a single H-MIMU and validated on real-world data - seem to be an option for trying and answer the question on "how do we walk during daily life?", which still represents a limit for lab-based technologies such as stereophotogrammetry and FSRs.
The large number of publications concerning the deployment of MaLe in gait analysis [14] suggests that it represents a valid resource for automatic gait phases detection, which is required in the rehabilitation sector - from FES to diagnosis to therapy planning. As suggested by the achieved results, MaLe can be successfully exploited to determine gait spatio-temporal parameters from raw inertial data recorded at the head. The task can be facilitated by the use of DeLe, which allows automatic crafting of features and pruning of the pre-processing pipeline.
The proposed methods represent a remarkable starting point for the use of H-MIMUs and MaLe for the estimation of spatio-temporal parameters. However, some issues will need to be addressed in the future.

Validation and optimization of the proposed algorithms on a larger sample of subjects The validity of H-MIMU-based methods for the estimation of spatio-temporal parameters was proven in the present work on a number of HY subjects. The extension of the developed algorithms to other age groups and walking conditions represents the next step for assessing the range of applicability of the proposed methods. In addition, validation of H-MIMU-based methods on pathological gait is still an open issue.

Explainability of machine learning and deep learning MaLe methods manage to achieve excellent performance for the requested task; however, their results are often hardly explainable, as the computation of the output is the product of a series of convolutions and transformations that profoundly change the layout of data, eventually preventing human operators to grasp the "reasoning" behind the obtained results. Such issue is even more relevant in the case of DeLe, for which a

larger number of transformations is applied to data and the model appears as a "black-box". In the clinical field, the difficulty in understanding the results of AI can prevent a method or device from being adopted, eventually limiting its deployment. Efforts have been made recently to implement explainable MaLe techniques [118], with the purpose of providing a pronounced descriptive approach to algorithms as well as additional information to users, therefore improving data insights.

Development of a shared codebase for machine learning Often, data-driven methods lack in reproducibility due to diversity and complexity of current approaches. Different hyperparameter choices, discrepancies between programming languages and the lack of a publicly available codebase all represent significant barriers to the validation of methods developed by other fellow researchers. In the recent years, several studies have tried to develop shared resources and libraries that different research groups can employ and that allow for faster and more consistent comparison between obtained results (e.g., the DANCE Python library to support DeLe models for the analysis of single-cell gene expression at scale [119]). However, such resources have not been yet implemented for the purposes of the analysis of the human motion; therefore, a systematic benchmarking procedure is today required to fully evaluate methods.

In the future, the performing of more rigorous trials to validate accuracy of MaLe methods and the implementation of real-time MaLe-based techniques should be addressed.
Efforts should be dedicated towards feature engineering that exploits domain knowledge. Actually, gait analysis differs from other disciplines due to its non-stationary, temporal and complex nature. Therefore, the task of quantifying the underlying biomechanics in learning models is not trivial [14]. Despite such inconvenience, the challenging job of feature engineering is overcome by DeLe, which allows automatic crafting of features.
Eventually, other factors should be taken into account for the design of MaLe methods applied to gait analysis, such as the training and testing of algorithms on pathological populations, the exploitation of transfer learning and the overcoming of the black-box nature of MaLe models, since it often hinders the comprehension of the relationship between MaLe outputs and the underlying gait phenomenon and ultimately prevents AI from being accepted and adopted as a tool in clinical applications.
The success of MaLe techniques of the recent years is due to its capacity of providing accurate, robust and fast classification by extracting simple features from highly temporal and nonlinear gait data [14]. The variability, instinctive, flexibility and adaptability of such dynamic approaches put them ahead of conventional methods that rely on thresholding or observational algorithms [14]. Moreover, MaLe eases

the walking assessment by introducing a more objective approach to gait analysis with respect to observational methods.
Nevertheless, until now, predictive models have been scarcely employed as diagnostics tool. More exhaustive research is needed before exploiting MIMUs and MaLe methods as a standardized clinical diagnostic tool [14]; however - once deployed - they will bring benefits for people's health and quality of life.

Appendix A

Useful Matlab functions

In the present Chapter, some of the Matlab custom functions employed to process data are presented.

A.1 `manage_subjects.m`

`manage_subjects.m` performs a correction of the experimental issues. As described at page 68, some errors may arise while performing the acquisitions. Once that all the acquisitions were completed, signals of each Participant were carefully examined in order to detect possible problems and try to correct them.
In the following, detected issues are presented:

- **Participant 7**: Sensors did not stop recording immediately at the end of the round walking (RW) Test. Then, the last 40 s of all the 3 Trials of the RW Test were cut off.
- **Participant 9**: The H-MIMU was erroneously positioned on the left side instead of the right side of the head. Then, the sign of the AP and ML acceleration and angular velocity is changed to virtually reorient the H-MIMU.

`manage_subjects.m` was personally written by the candidate.

A.2 `retrieve_path_info.m`

`retrieve_path_info.m` reads the Operator Table of the Participant and writes their personal and anthropometric features (name, date of birth, height, weight, dominant hand, shoe size) to a Matlab `struct` variable.
`retrieve_path_info.m` was personally written by the candidate.

A.3 `calc_R.m`, `reorient_head.m`

The function `calc_R.m` compares the tri-axial acceleration recorded by the H-MIMU during the Standing Test to the expected acceleration ($[0g0]$). Then, it computes the angle difference between the expected and measured gravity vectors and use it to create a rotation axis. The identified axis is then exploited to get the orientation of the H-MIMU through quaternions algebra and eventually its final orientation expressed in quaternions is converted to a rotation matrix R.
R is passed as input to `reorient_head.m` and is applied through linear algebra to all the measured accelerations of the various Trials to virtually rotate the H-MIMU. Both `calc_R.m` and `reorient_head.m` were developed at Università degli Studi di Sassari.

A.4 `main_synch_head_validation.m`

`main_synch_head_validation.m` is used to standardize data from the In-Lab and Free-Living acquisitions. Standardization in this case means reading the TXT files in the *Laboratory* or *Free-living* folder of each Participant into the Matlab workspace (see Section C.2), applying the stored calibration matrices to each sensor to correct calibration error and writing the output of calibration into an organized Matlab `struct` (see Section C.3). Both `main_synch_head_validation.m` were developed at Università degli Studi di Sassari.

A.5 INDIP algorithms for the spatio-temporal parameters estimation

As mentioned in Section C.3, the standardized Matlab `struct data.mat` is given as input to a set of algorithms for the estimation of gait STP. Such algorithms, developed at Università degli Studi di Sassari, exploit data from the B-MIMU, F-MIMUs and pressure insoles to obtain STP of gait in conditions of regular, inclined and non-straight walking [96]. The INDIP algorithms have been developed by Università degli Studi di Sassari to represent a standard for algorithms based on a lower number of sensors; then, in the present thesis, they are assumed as the GS for the estimation of STP.

Appendix B

Matlab code for the network architectures

B.1 Long-short term memory network

```
% Define the LSTM architecture
numFeatures = 8;
numHiddenUnits = 200;
numClasses = 2;
% network architecture
layers = [
    sequenceInputLayer(numFeatures)
    lstmLayer(numHiddenUnits,'OutputMode','sequence')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer];
% training options
options = trainingOptions('adam', ...
    'MaxEpochs',60, ...
    'GradientThreshold',2, ...
    'Verbose',0, ...
    'Shuffle','never',...
    'ValidationData',validation_data,...
    'Plots','training-progress');
%% training
[net,info] = trainNetwork(XTrain,YTrain,layers,options);
```

B.2 Temporal convolutional network

```
%% Definizione rete TCN
% https://it.mathworks.com/help/deeplearning/ug/sequence-to-...
    sequence-classification-using-1-d-convolutions.html
numFeatures =  size(XTrain{1},1);  % 8
numClasses = 2;
% network architecture
numFilters = 64;
filterSize = 5;
dropoutFactor = 0.005;
numBlocks = 4;

layer = sequenceInputLayer(numFeatures,Normalization="rescale-...
    symmetric",Name="input");
lgraph = layerGraph(layer);

outputName = layer.Name;

for i = 1:numBlocks
    dilationFactor = 2^(i-1);

    layers = [
        convolution1dLayer(filterSize,numFilters,DilationFactor...
    =dilationFactor,Padding="causal",Name="conv1_"+i)
        layerNormalizationLayer
        spatialDropoutLayer(dropoutFactor)
        convolution1dLayer(filterSize,numFilters,DilationFactor...
    =dilationFactor,Padding="causal")
        layerNormalizationLayer
        reluLayer
        spatialDropoutLayer(dropoutFactor)
        additionLayer(2,Name="add_"+i)];

    % Add and connect layers.
    lgraph = addLayers(lgraph,layers);
    lgraph = connectLayers(lgraph,outputName,"conv1_"+i);

    % Skip connection.
    if i == 1
        % Include convolution in first skip connection.
        layer = convolution1dLayer(1,numFilters,Name="convSkip...
    ");

        lgraph = addLayers(lgraph,layer);
```

```
39          lgraph = connectLayers(lgraph,outputName,"convSkip");
40          lgraph = connectLayers(lgraph,"convSkip","add_" + i + ...
    "/in2");
41      else
42          lgraph = connectLayers(lgraph,outputName,"add_" + i + ...
    "/in2");
43      end
44
45      % Update layer output name.
46      outputName = "add_" + i;
47  end
48
49  layers = [
50      fullyConnectedLayer(numClasses,Name="fc")
51      softmaxLayer
52      classificationLayer];
53  lgraph = addLayers(lgraph,layers);
54  lgraph = connectLayers(lgraph,outputName,"fc");
55  % training options
56  options = trainingOptions("adam", ...
57      MaxEpochs=60, ...
58      miniBatchSize=1, ...
59      Plots="training-progress", ...
60      Verbose=0,...
61      ValidationData = validation_data);
62  %NB remember to specify validation data
63  %% training
64  net = trainNetwork(XTrain,YTrain,lgraph,options);
```

B.3 Custom spatial dropout layer

```
1
2  classdef spatialDropoutLayer < nnet.layer.Layer & ...
       nnet.layer.Formattable
3      % Example custom spatial dropout layer.
4
5      properties
6          DropoutFactor
7      end
8
9      methods
10         function layer = spatialDropoutLayer(dropoutFactor,...
       NameValueArgs)
```

```
11          % layer = spatialDropoutLayer creates a spatial ...
    dropout layer
12          % with dropout factor 0.02;
13          %
14          % layer = spatialDropoutLayer(dropoutProb) creates ...
    a spatial
15          % dropout layer with the specified probability.
16          %
17          % layer = spatialDropoutLayer(___,Name=name) also ...
    specifies the
18          % layer name using any of the previous syntaxes.
19
20          % Parse input arguments.
21          arguments
22              dropoutFactor = 0.02;
23              NameValueArgs.Name = ""
24          end
25          name = NameValueArgs.Name;
26
27          % Set layer properties.
28          layer.Name = name;
29          layer.Description = "Spatial dropout with factor " ...
    + dropoutFactor;
30          layer.Type = "Spatial Dropout";
31          layer.DropoutFactor = dropoutFactor;
32      end
33
34      function Z = predict(layer, X)
35          % Forward input data through the layer at ...
    prediction time and
36          % output the result.
37          %
38          % Inputs:
39          %         layer - Layer to forward propagate ...
    through
40          %         X     - Input data
41          % Output:
42          %         Z     - Output of layer forward function
43
44          % At prediction time, the output is unchanged.
45          Z = X;
46      end
47
48      function Z = forward(layer, X)
49          % Forward input data through the layer at training
50          % time and output the result and a memory value.
51          %
52          % Inputs:
```

```
53              %           layer - Layer to forward propagate ...
    through
54              %           X     - Input data
55              % Output:
56              %           Z - Output of layer forward function
57
58              dropoutFactor = layer.DropoutFactor;
59
60              % Mask dimensions.
61              fmt = dims(X);
62              maskSize = size(X);
63              maskSize(ismember(fmt,'ST')) = 1;
64
65              % Create mask.
66              dropoutScaleFactor = single(1 - dropoutFactor);
67              dropoutMask = (rand(maskSize,'like',X) > ...
    dropoutFactor) / dropoutScaleFactor;
68
69              % Dropout.
70              Z = X .* dropoutMask;
71          end
72      end
73  end
```

Appendix C

INDIP data structure

In the present Chapter, information regarding how recorded data are systematically stored and organized is given.

C.1 Operator Table

Every time that an acquisition session takes place, the operator(s) are asked to fill the *Operator Table* for the session. This table is an Excel sheet that contains the Participant's personal and anthropometric information, information on the employed sensors, any technical problems and annotations:

- **Date of the session (DD/MM/YYYY)**
- **ID of the acquisition center**
- **Name**
- **Date of birth (DD/MM/YYYY)**
- **Gender (M/F)**
- **Height (cm)**
- **Weight (kg)**
- **Shoe size (EU)**
- **Waist width (cm)**
- **Dominant hand (L/R)**
- **List of sensors**: The Operator Table includes the IDs of the all employed sensors, plus the IDs of sensors eventually substituted during the acquisitions.

- **Annotations**: For each Test/Recording, the operator is asked to report any issues regarding misbehavior of the sensors or non-adherence to the protocol.

During the acquisition (or right at the end of it), the Operator Table is filled in with all the required information and - eventually - any other relevant annotation. The Operator Table must be named after the Participant's ID and saved in the *Mobility Test* or *ADL Test* subfolders as described in Section C.2.

C.2 Participant folder

As mentioned in Chapter 3, at the end of each acquisition, recorded data stored in each INDIP MIMU has to be properly downloaded to the Participant's folder.
To ease navigation and automated function of the INDIP software and GUIs, the Participant folder must observe a predefined structure (Figure C.1). First, the

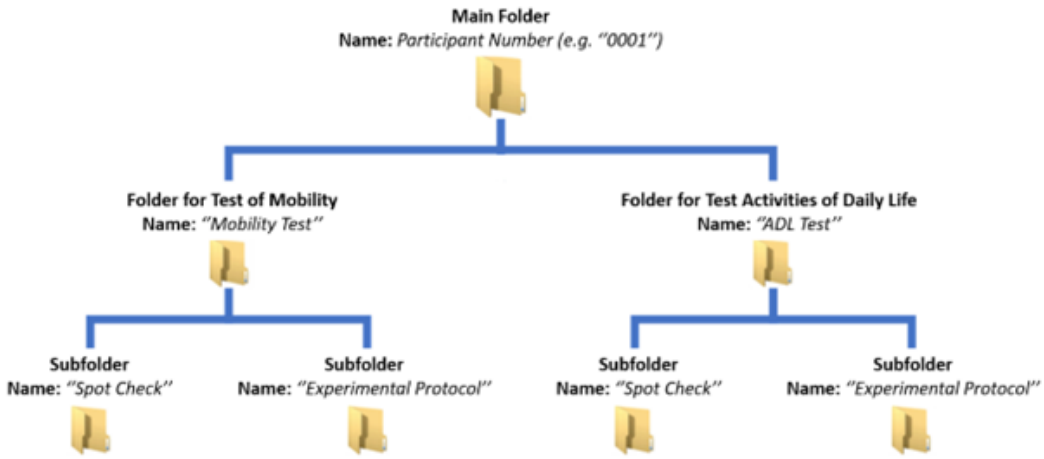


Figure C.1: Skeleton of the Participant folder.

Participant's *Main Folder* named after the Participant's number is created. The ID number for each Participant is chosen randomly. Then, two sub-folders are created:

- *Mobility Test*: Such folder is created if the Participant has performed the In-Lab acquisitions, otherwise not.
- *ADL Test*: Such folder is created if the Participant has performed the Free-Living acquisitions, otherwise not.

For the present study, no Participants have both the *Mobility Test* and the *ADL Test* folders i.e., the Participants included in the In-Lab dataset differ from the

ones included in the Free-Living dataset.
Once that the *Mobility Test* and the *ADL Test* folders are created, they are filled with the Operator Table (see Section C.1) and with other two sub-folders:

- *Spot Check*: The *Spot Check* folder contains data for the Quality Check (Test/Recording 1). Data contained in the *Spot Check* folder are saved as TXT data and manually renamed with the convention "pppp-INDIP#xxx-Static01-ddmmyyyy", where "pppp" denotes the 4 digits ID of the Participant (e.g., "0001") and "xxx" denotes the three digits ID of the INDIP MIMU (e.g., "096"). Since the Static Test involves 6 MIMUs and 2 pressure insoles, the *Spot Check* folder contains a total amount of 8 TXT files.

- *Experimental Protocol*: The *Experimental Protocol* folder contains data of the other Tests/Recordings (Tests 2-7, Recordings 2-4). Data contained in the *Experimental Protocol* folder are saved as TXT data with the convention "INDIP#xxx_DD-MM-YYYY_hhmmss". Since each session requires 14 Trials and involves a total amount of 6 MIMUs, the *Experimental Protocol* folder contains a total amount of 84 TXT files.

For the INDIP algorithms to function, TXT files in the *Experimental Protocol* folder have to be renamed according to the same convention adopted by the *Spot Check* folder. Given the high amount of files in the Experimental Protocol folder, such renaming is performed through a Matlab Renaming GUI designed at Università degli Studi di Sassari. Renamed files of the *Experimental Protocol* folder are saved to the *Laboratory* folder, which contains also TXT files of the *Spot Check* folder and the Operator Table (Figure C.2). Eventually, TXT files are standardized to return the Matlab `struct` variable `data.mat` through the Matlab custom functions `main_synch_head_validation_INDOOR.m` and `main_synch_head_validation_OUTOOR.m`, respectively if the files refer to the *Mobility Test* or *ADL Test* folder. Such functions have been developed at Università degli Studi di Sassari. `data.mat` is saved to the *Standardized* or *Free-living* sub-folder if data refers to the In-Lab or Free-Living acquisitions, respectively (see Section C.3). Once that the standardized `data.mat` is saved to the *Standardized* folder, a set of algorithms developed at Università degli Studi di Sassari[1] are employed to extract a set of STP and return another Matlab `struct` - still called `data.mat` - that contains raw data as well as the reference values for the estimated STP i.e., the INDIP standard (see C.3). The resulting Matlab `struct` is saved to the subfolder *Results* of the *Mobility Test/ADL Test* folder (Figure C.3).

[1]Such algorithms have been devised by Università degli Studi di Sassari to validate methods for the estimation of STP in the framework of the EU project MobiliseD; therefore, they are highly reliable and can be used as a GS for the purposes of the present thesis.

Figure C.2: Skeleton of the Participant folder including the *Laboratory* folder.

(a) Content of the *Mobility Test* folder.

(b) Content of the *ADL Test* folder.

Figure C.3: Content of the *Mobility test*/*ADL Test* folder at the end of data standardization.

C.3 INDIP standard

In the present Section, the structure of `data.mat` is described in detail. `data.mat` is a Matlab 1×1 `struct` with a number of fields and sub-fields. Consider to request the access to the accelerations recorded by the H-MIMU during Trial 1 of Test1 of the In-Lab acquisitions. To perform such request, one should type the following Matlab command:

```
1    data.TimeMeasure1.Test1.Trial1.SU_INDIP.Head.Acc
```

The resulting output is an $N \times 3$ Matlab double array of tri-axial accelerations values, where N is the number of recorded time steps. Notice that, to access data of a MIMU, the number of the Test, the number of the Trial and the position of the MIMU have to be specified. Similarly, to access head accelerations recorded by the H-MIMU during Recording 4 of the Free-Living acquisitions, one should use the following Matlab command:

```
1    data.TimeMeasure1.Recording4.SU_INDIP.Head.Acc
```

Once that `data.mat` is filled with the reference values of the STP estimated through the INDIP algorithms, it will contain a number of STP (see Table 2.1):

- *Start (s)*: first IC of the Trial.
- *End*: last IC of the Trial.
- *Stride frequency (Hz)*
- *Cadence (Hz)*
- *Duration (s)*: defined as End − Start.
- *Length (m)*: walked length during the Trial.
- *Walking speed (m/s)*: defined as $\frac{\text{Length}}{\text{Duration}}$
- *Average stride length (m)*: sum of the stride lengths divided by the number of strides.
- *Number of strides*: occurring during the Trial.

- *Initial contacts (s)*: Vector containing all the ICs for the Trial.
- *Final contacts (s)*: Vector containing all the FCs for the Trial.
- *Stride times (s)*
- *Stride lengths (m)*
- *Stride speeds (m/s)*
- *Stance times (s)*
- *Swing times (s)*
- *Stance lengths (m)*
- *Swing lengths (m)*
- *Stance speeds (m/s)*
- *Swing speeds (m/s)*
- *SS times (s)*
- *DS times (s)*
- *Step times (s)*

A substantial difference exists between the INDIP standard of the Matlab `struct data.mat` of the In-Lab and Free-Living acquisitions. In the In-Lab version of `data.mat`, each Trial is associated to a single set of STP. Instead, in the Free-Living version of `data.mat`, each Recording has a certain number of *micro walking bouts*, each of them associated to a different set of STP. A micro walking bout is defined as a walking period composed of at least two strides - inclined walking excluded. Then, the INDIP standard for the Free-Living acquisitions only refers to those portions of gait during the ones the Participant is effectively walking regularly.

Appendix D

Notions on machine learning and deep learning

In the following Chapter, the MaLe and DeLe techniques and models employed for the thesis purposes are described in detail. Most of the information reported in this Chapter refers to the Matlab documentation [99] and topic-related scientific papers and websites [87][100][107][106][101][120][121][122][123].

D.1 Gaussian Process Regression (GPR)

Gaussian processes provide a practical probabilistic approach to learning based on kernels [87]. In the next Section, the mathematical notions behind the implementation of GPR models are introduced.

D.1.1 Linear regression models

Consider a training set:

$$(x_i, y_i); \qquad i = 1, 2, ..., n$$

Where x_i is the i^{th} set of d real predictors and y_i represents the i_{th} real target. The purpose of a generic regression model is to predict the value of the response variable y_{new} of an input vector of predictors x_{new}.
A linear regression model is characterized by the following form:

$$y = x^T \beta + \epsilon \tag{D.1}$$

Where ϵ is the prediction error characterized by a Gaussian distribution that has mean of 0 and variance σ^2. The error variance σ^2 and the coefficients β are derived from the training data, typically through the least square errors (LSE) algorithm

or one of its variants.

D.1.2 GPR models

GPR models are a class of non-parametric kernel-based probabilistic models [87]. A GPR model introduces latent variables $f(x_i)$ from a Gaussian process (Equation D.2) and explicit basis functions h to explain the response:

$$f(x_i); \qquad i = 1,2,...,n \tag{D.2}$$

The covariance function of the latent variables captures the smoothness of the response, while basis functions project the input predictors x into a p-dimensional feature space. A Gaussian process is defined as a set of random variables that have a joint Gaussian distribution when taken in any finite number. Consider a Gaussian process:

$$\{f(x), \qquad x \in \mathbb{R}^d\} \tag{D.3}$$

Given a set of n observations $\{x_1, x_2, ..., x_n\}$, the joint distribution of the random variables $\{f(x_1)(x_2), ..., f(x_n)\}$ is Gaussian. If $\{f(x), x \in \mathbb{R}^d\}$ is a Gaussian process, then it is fully described by its mean $m(x)$ and its covariance function $k(x, x')$ (Equation D.4).

$$\begin{cases} E(f(x)) = m(x) \\ Cov(f(x), f(x')) = E(\{f(x) - m(x)\}\{f(xi) - m(x')\}) = k(x, x') \end{cases} \tag{D.4}$$

Given a Gaussian process that has mean equal to 0 and covariance $k(x, x')$, a GPR is defined as follows:

$$h(x)^T\beta + f(x) \tag{D.5}$$

Where $f(x)$ is a function of the process, $h(x)$ represents a set of basis functions that transform the original feature vector $x \in \mathbb{R}^d$ into a new feature vector $h(x) \in \mathbb{R}^p$ and β is a $p \times 1$ vector of basis function coefficients.
GPR models are probabilistic, meaning that an instance of response y can be modeled as follows:

$$P(y_i|f(x_i), x_i) \sim N(y_i|h(x)^T\beta + f(x), \sigma^2) \tag{D.6}$$

Notice that the latent variable $f(x_i)$ introduced for each observation x_i makes the GPR model non-parametric. In vector form, this model is equivalent to. Equation D.6 can also be expressed in terms of vectors and matrices:

$$P(y|f, X) \sim N(y|H\beta + f, \sigma^2 I) \tag{D.7}$$

Where X is a matrix of observations, y is the vector of responses, H is a matrix of basis functions coefficients, f are the functions of the Gaussian process and I is the identity matrix (Equation D.8).

$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \qquad H = \begin{bmatrix} h(x_1^T) \\ h(x_2^T) \\ \vdots \\ h(x_n^T) \end{bmatrix} f = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix} \tag{D.8}$$

The latent variables $f(x_1), f(x_2), \ldots, f(x_n)$ in the GPR model have a Gaussian joint distribution with mean of 0 and covariance expressed by the matrix $K(X, X)$. Notice that $K(X, X)$ has the same form of the linear regression case (Equation D.9).

$$K(X, X) = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \ldots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \ldots & k(x_2, x_n) \\ \vdots & \vdots & \vdots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \ldots & k(x_n, x_n) \end{bmatrix} \tag{D.9}$$

During training, the basis function coefficients β, the noise variance σ^2 and the kernel function hyperparameters θ of the GPR model are estimated. Since GPR models are probabilistic, confidence intervals can be computed by the trained model at prediction time.

D.1.3 GPR kernels

In general, elements with similar predictor values x are expected to have close target response values y. In a Gaussian process, such similarity is expressed by the *covariance function* [87]. The covariance function denotes the covariance between the two latent variables $f(x_i)$ and $f(x_j)$ i.e., how the response at one input x_i is affected by responses at other inputs $x_{j \neq i}, j = 1, 2, \ldots, n$ (both x_i and x_j are $d \times 1$ observations vectors).
The covariance function $k(x_i, x_j)$ can be defined by various kernel functions [99]. The kernel parameters contained in vector θ allow to parameterize the covariance function, hence, it can be expressed as $k(x_i, x_j|\theta)$ to explicitly denote the dependence on θ.
For most of the standard kernel functions, θ is based on the signal standard deviation σ_f and the characteristic length scale σ_l, which briefly define the minimal distance between the input values x_i and x_j for which their responses can be considered uncorrelated. σ_f and σ_l are supposed to be greater than 0, so that the unconstrained parametrization vector θ can be expressed as follows:

$$\theta_1 = \log \sigma_l \qquad \theta_2 = \log \sigma_f \tag{D.10}$$

Covariance functions that implement automatic relevance determination (ARD) can also have a separate length scale for each predictor $\{\sigma_m, m = 1, 2, \ldots, d\}$ [124]. In such case, the unconstrained parametrization θ becomes as follows:

$$\theta_m = \log \sigma_m \qquad \theta_d + 1 = \log \sigma_f \qquad m = 1, 2, \ldots, d. \tag{D.11}$$

The most common covariance functions (kernels) for each predictor x_i are defined as follows:

Squared exponential kernel One of the most commonly employed covariance functions:

$$k\left(x_i, x_j \mid \theta\right) = \sigma_f^2 \exp\left[-\frac{1}{2}\frac{\left(x_i - x_j\right)^T \left(x_i - x_j\right)}{\sigma_l^2}\right] \tag{D.12}$$

Where σ_f denotes the standard deviation of the signal and σ_l denotes the characteristic scale length.
The ARD squared exponential kernel is defined as follows:

$$k\left(x_i, x_j \mid \theta\right) = \sigma_f^2 \exp\left[-\frac{1}{2}\sum_{m=1}^{d}\frac{\left(x_{im} - x_{jm}\right)^2}{\sigma_m^2}\right] \tag{D.13}$$

Exponential kernel

$$k\left(x_i, x_j \mid \theta\right) = \sigma_f^2 \exp\left(-\frac{r}{\sigma_l}\right) \tag{D.14}$$

Where σ_l denotes the characteristic scale length and r denotes the Euclidean distance between x_i and x_j (Equation D.15).

$$r = \sqrt{\left(x_i - x_j\right)^T \left(x_i - x_j\right)} \tag{D.15}$$

The ARD exponential kernel is defined as follows:

$$k\left(x_i, x_j \mid \theta\right) = \sigma_f^2 \exp(-r) \tag{D.16}$$

Where r is defined as follows:

$$r = \sqrt{\sum_{m=1}^{d}\frac{\left(x_{im} - x_{jm}\right)^2}{\sigma_m^2}} \tag{D.17}$$

Matern 3/2 kernel

$$k\left(x_i, x_j \mid \theta\right) = \sigma_f^2(1 + \frac{\sqrt{3}r}{\sigma_l})\exp(-\frac{\sqrt{3}r}{\sigma_l}) \tag{D.18}$$

Where σ_l denotes the characteristic scale length and r denotes the Euclidean distance between x_i and x_j (Equation D.15).
The ARD Matern 3/2 kernel is defined as follows:

$$k(x_i, x_j \mid \theta) = \sigma_f^2(1 + \sqrt{3}r)\exp(-\sqrt{3}r) \tag{D.19}$$

See Equation D.17 for the definition of r.

Matern 5/2 kernel

$$k\left(x_i, x_j\right) = \sigma_f^2\left(1 + \frac{\sqrt{5}r}{\sigma_l} + \frac{5r^2}{3\sigma_l^2}\right)\exp\left(-\frac{\sqrt{5}r}{\sigma_l}\right) \tag{D.20}$$

Where σ_l denotes the characteristic scale length and r denotes the Euclidean distance between x_i and x_j (Equation D.15).
The ARD Matern 5/2 kernel is defined as follows:

$$k(x_i, x_j \mid \theta) = \sigma_f^2(1 + \sqrt{5}r + \frac{5}{3}r^2)\exp(-\sqrt{5}r) \tag{D.21}$$

See Equation D.17 for the definition of r.

Rational quadratic kernel

$$k\left(x_i, x_j \mid \theta\right) = \sigma_f^2\left(1 + \frac{r^2}{2\alpha\sigma_l^2}\right)^{-\alpha} \tag{D.22}$$

Where σ_l denotes the characteristic scale length, α is a scale-mixture parameter ($\alpha geq 0$), r denotes the Euclidean distance between x_i and x_j (Equation D.15).
The ARD rational quadratic kernel is defined as follows:

$$k\left(x_i, x_j \mid \theta\right) = \sigma_f^2\left(1 + \frac{1}{2\alpha}\sum_{m=1}^{d}\frac{(x_{im} - x_{jm})^2}{\sigma_m^2}\right)^{-\alpha} \tag{D.23}$$

D.2 Support Vector Machine (SVM) regression

SVMs are a class of MaLe tools used for classification and regression. SVMs have been proposed for the first time by Vladimir Vapnik and his research group in 1992 [125]. Thereafter, the method has gained popularity and today it is systematically employed for solving tasks in a number of field areas.
SVM regression - otherwise known as SVR - relies on kernel functions; therefore, it is considered a non-parametric technique. SVR offers the flexibility to define the

extent to which a prediction error is tolerable and allows to find an appropriate line (or hyperplane if there are more than one dimension) to fit the training data [113]. One of the most widely used SVRs is the linear epsilon-insensitive SVM (ϵ-SVM) regression, also referred to as L_1 loss [99]. In ϵ-SVM regression, the training set includes predictor variables and target response values. The goal of ϵ-SVM regression consists in finding a function $f(x)$ that deviates from yn by no greater value than ϵ for each element of the training set x, and contemporaneously is as *flat*[1] as possible [99].

D.2.1 Linear SVM regression - primal form

Consider a training set of N multivariate observations x_n with observed target response values y_n. The linear function that is supposed to perform the mapping from x_n to y_n has the following form:

$$f(x) = x'\beta + b \tag{D.24}$$

For ϵ-SVM, $f(x)$ has to be maximally flat; at the same time, all residuals must be lower than the quantity ϵ (Equation D.25).

$$\forall n : |y_n - (x'_n\beta + b)| \leq \epsilon \tag{D.25}$$

For $f(x)$ to be flat, the norm J of the coefficients vector β must be minimized (Equation D.26).

$$J(\beta) = \frac{1}{2}\beta\beta' \tag{D.26}$$

There is the possibility that no $f(x)$ exists to fulfill such conditions for all points. Then, slack variables ξ_n and ξ_n^* can be introduced for each point to deal with unfeasible constraints (Equation D.27).

$$J(\beta) = \frac{1}{2}\beta\beta' + C\sum_{n=1}^{N}(\xi_n + \xi_n*) \tag{D.27}$$

The slack variables allow regression residuals to exist up to the values of ξ_n and ξ_n^*, yet still satisfy the required conditions (Equation D.28).

$$\begin{cases} \forall n : |y_n - (x'_n\beta + b)| \leq \epsilon + \xi_n \\ \forall n : |(x'_n\beta + b) - y_n| \leq \epsilon + \xi_n^* \\ \forall n : \xi_n \geq 0 \\ \forall n : \xi_n^* \geq 0 \end{cases} \tag{D.28}$$

[1]A function is flat is all of its derivatives vanish at a given point [126].

The constant C in Equation D.27 is the *box constraint*. It is a scalar value greater than zero that controls the penalty imposed on observations that fall outside the ϵ margin and aids to avoid overfitting. C defines the compromise between the flatness of $f(x)$ and the extent up to which residuals greater than ϵ are accepted [99]. As C increases, the range of tolerance for errors greater than ϵ becomes larger. As C tends to 0, the range of tolerance approaches 0 and Equation D.27 collapses into Equation D.26 [113].
The linear ϵ-SVM loss function overrides errors in the range $[-\epsilon, \epsilon]$ by squashing them to zero. The loss function value is derived from the distance between observed value y and the ϵ boundary (Equation D.29).

$$L_\epsilon = \begin{cases} 0 & \text{if: } |y - f(x)| \leq 0 \\ |y - f(x)| - \epsilon & \text{otherwise} \end{cases} \tag{D.29}$$

D.2.2 Linear SVM regression - dual form

The optimization problem described in Subsection D.2.1 can be solved more easily in its Lagrange dual formulation - from a computational point of view - as it allows for a lower bound to the solution of the primal problem. The optimal values of the primal and dual problems can not be equal - their difference is commonly referred to as the "duality gap" - however, if the problem is convex and satisfies a constraint qualification condition, the value of the optimal solution to the primal problem corresponds to the solution of the dual problem [99].
The dual formula is obtained by using the primal form for constructing a Lagrangian function with non-negative multipliers α_n and α_n^* for each observation x_n (Equation D.30).

$$L(\alpha) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha_i^*) \left(\alpha_j - \alpha_j^*\right) x_i' x_j + \varepsilon \sum_{i=1}^{N} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{N} y_i (\alpha_i^* - \alpha_i) \tag{D.30}$$

The dual formula is subject to the following constraints:

$$\begin{cases} \sum_{n=1}^{N} (\alpha_n - \alpha_n^*) = 0 \\ \forall n : 0 \leq \alpha_n \leq C \\ \forall n : 0 \leq \alpha_n^* \leq C \end{cases} \tag{D.31}$$

The coefficients vector β can be portrayed as a linear combination of the training observations (Equation D.32).

$$\beta = \sum_{n=1}^{N} (\alpha_n - \alpha_n^*) x_n \tag{D.32}$$

Therefore, the function used to infer new values is dependent only on the support vectors (Equation D.33):

$$f(x) = \sum_{n=1}^{N} (\alpha_n - \alpha_n^*) x_n + b \tag{D.33}$$

To obtain optimal solutions, the Karush-Kuhn-Tucker complementarity conditions are required as optimization constraints. For linear SVM regression, such conditions take the following form:

$$\begin{aligned} &\forall n : \alpha_n \left(\varepsilon + \xi_n - y_n + x_n{}'\beta + b\right) = 0 \\ &\forall n : \alpha_n^* \left(\varepsilon + \xi_n^* + y_n - x_n'\beta - b\right) = 0 \\ &\forall n : \xi_n \left(C - \alpha_n\right) = 0 \\ &\forall n : \xi_n^* \left(C - \alpha_n^*\right) = 0. \end{aligned} \tag{D.34}$$

Conditions in Equation D.34 define that all observations completely inside the ϵ range have Lagrange multipliers $\alpha_n = 0$ and $\alpha_n^* = 0$. If one of them is instead different from zero, then the corresponding observation is referred to as a *support vector* [99].

D.2.3 Solver algorithms for SVR

The SVR minimization problem can be expressed and solved using traditional quadratic programming techniques. Though, since the Gram matrix could be too large to be stored in memory, quadratic programming algorithms have a high computational cost. On the other hand, exploiting a *decomposition method* may accelerate the computation and prevent memory from running out.
Decomposition methods - sometimes referred to as "chunking and working set methods" - work by dividing all observations into two non-overlapping sets: the *working set* and the *remaining set*. Only the observations in the working set are changed in each iteration. Therefore, only some columns of the Gram matrix are stored in each iteration, reducing the overall amount of memory required.
SMO is the most common algorithm for solving SVM minimization problems [127]. SMO is based on a series of two-point optimizations. In each iteration, according to a selection rule based on second-order information, two points are chosen to constitute the working set. After that, the Lagrange multipliers for the selected working set are solved analytically [128][129].
In SVR, the update of the gradient vector ∇L for the active set occurs after each iteration. The decomposed equation for the gradient vector is of the following form:

$$(\nabla L)_n = \begin{cases} \sum_{i=1}^{N} \left(\alpha_i - \alpha_i^*\right) G\left(x_i, x_n\right) + \varepsilon - y_n, n \leq N \\ -\sum_{i=1}^{N} \left(\alpha_i - \alpha_i^*\right) G\left(x_i, x_n\right) + \varepsilon + y_n, n > N \end{cases} \tag{D.35}$$

The SMO algorithm computes the gradient vector iteratively until the specified convergence criterion is achieved. The convergence criterion can be represented by a number of conditions [99]:

- **Feasibility gap**: The feasibility gap is expressed as follows:

$$\delta = \frac{J(\beta) + L(\alpha)}{J(\beta) + 1} \tag{D.36}$$

  Where $J(\beta)$ denotes the primal target and $L(\alpha)$ denotes the dual target. At the end of each iteration, the feasibility gap is evaluated. If the feasibility gap is less than a specified value; then, the convergence criterion is met and a solution is returned.

- **Gradient difference**: At the end of each iteration, the gradient vector ∇L is evaluated. If the difference in gradient vector values for the current iteration and the previous iteration is less than a specified value; then, the convergence criterion is met and a solution is returned.

- *Largest Karush-Kuhn-Tucker (KKT) violation*: At the end of each iteration, the KKT violation for all the values of the Lagrangian multipliers is evaluated. If the largest violation is less than a specified value; then, the convergence criterion is met and a solution is returned.

D.3 Long-short term memory (LSTM) network architecture

In the next Section, the basic notions for understanding the description of the LSTM architecture presented in Sub-subsection 5.2.1 are introduced.

D.3.1 Recurrent Neural Network (RNN)

A RNN is a DeLe network structure that exploits information of the past to enhance the performance of the network on current and future inputs. RNNs use a looping structure to store past information in a hidden state, so that the network can keep track of the past samples of the input time-series when it is predicting a sample at the current time step (Figure D.1).

The looping structure represents a sort of memory that allows RNNs to capture the information wrapped in the input sequence itself, while feed-forward nets typically fail to do it [120]. The ability of RNNs to find "long-term dependencies" i.e., correlations between events separated by many moments is basically a way to share weights over time, as an event downstream in time is affected by one or more

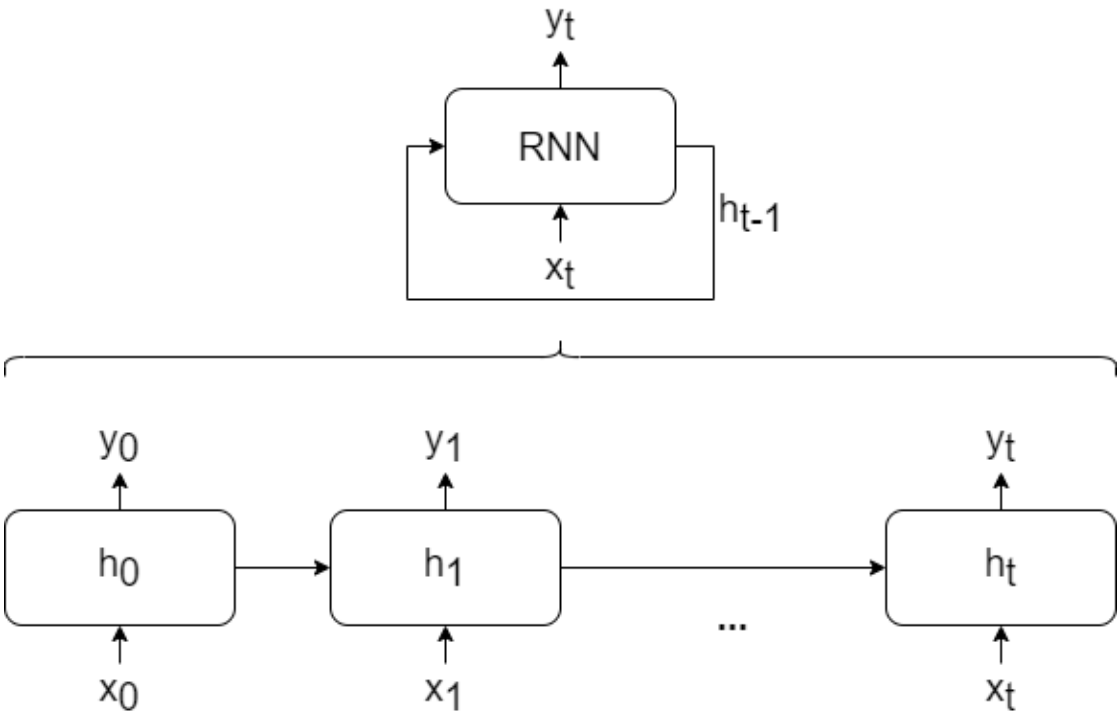


Figure D.1: Unrolling a single cell of a RNN to show how information moves through the network for a data sequence. Inputs are fed to the hidden state of the cell to produce the output, and the hidden state is passed to the next time step.

events that came before [130].
The process of conveying memory forward can be described mathematically (Equation D.37)

$$h_t = \Phi(Wx_t + Uh_t) \tag{D.37}$$

The hidden state at time step t h_t is a function of the input at the same time step x_t combined to a weight matrix W and added to the hidden state of the previous time step h_{t-1} combined to its own hidden-state-to-hidden-state matrix U, otherwise known as a *transition matrix* [130]. The weight arrays are basically filters that control the importance of the present input and the past hidden state in the determination of the hidden unit output. During training, the error generated by weights is backpropagated and used to adjust their values until error is sufficiently low. The combination of the weighted input and hidden state is fed to the function Φ – typically a tanh or a logistic sigmoid function (Equation D.38) – which allows to condense values in a very large or very small range into a logistic space and making gradients digestible for backpropagation (Figure D.2).

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^{2x} - 1}{e^{2x} + 1} \qquad \sigma(x) = (1 + e^{-x})^{-1} \tag{D.38}$$

Due to such characteristics, RNNs are well suited for addressing a number of problems that involve sequential data of varying length:

- **Signal classification**: Automatic classification of signals can decrease the manual time needed for large datasets or enable classification in real time. Raw signals can be submitted to deep networks or pre-processed to focus on some relevant signal features.
- **Natural language processing**
- **Video analysis**

However, basic RNNs struggle to learn longer-term dependencies. Multiplying a number for a factor slightly larger than 1 has almost no effect if the multiplication happens few times, while it causes the number to diverge if the multiplication is repeated many times. Similarly, if the multiplication factor is slightly lower than 1 and it is applied over and over, the number is pushed to 0.
The information flowing through RNN networks goes through many steps of multiplication; then, gradients - which are basically derivatives - are susceptible to "vanishing" or "exploding" i.e., the network weights either shrink or increase excessively.
The gradient denotes the change in all weights with regard to a variation of the error. If the gradient explodes, it can be truncated or "squashed" through the sigmoid or tanh function; however, the vanishing gradient problem can not be

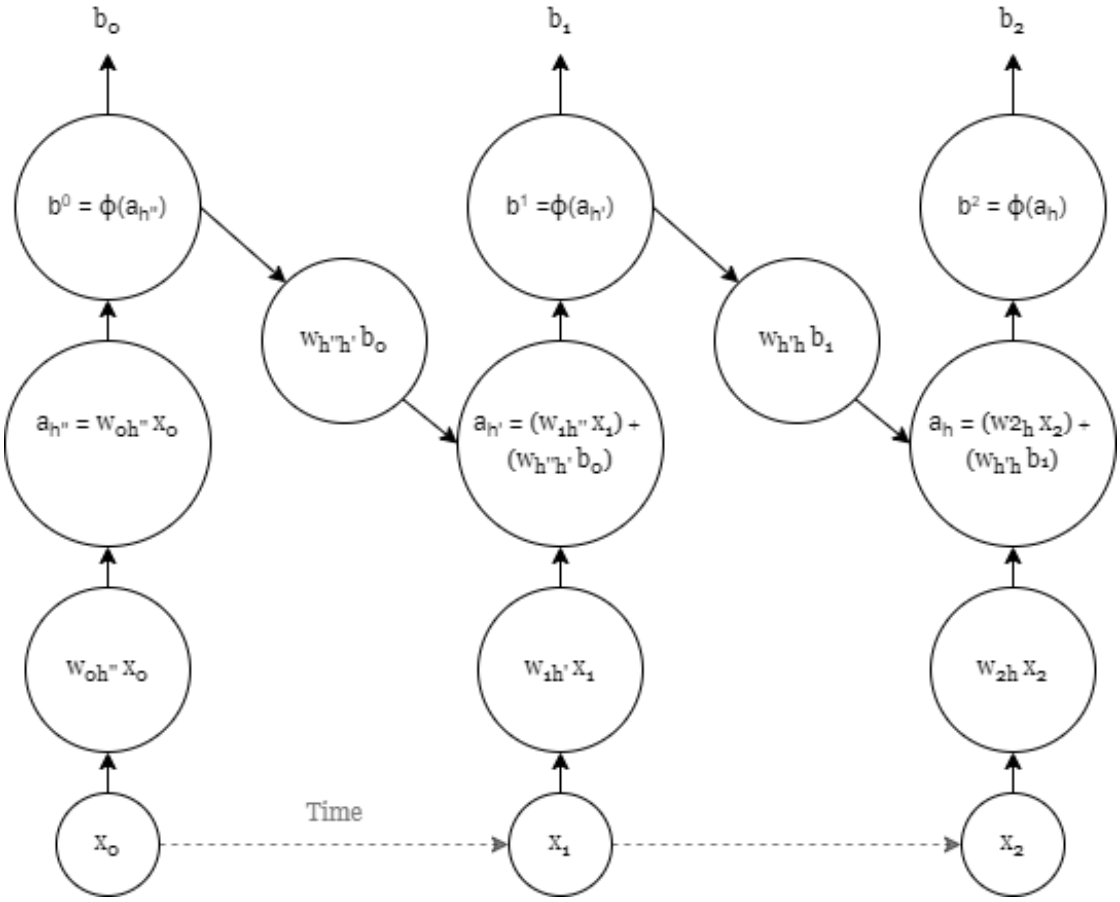


Figure D.2: Diagram for illustrating the functioning of a RNN. The vertical lines of nodes (hidden units) can be seen as feed-forward networks, whose output is forwarded to the next unit. Inputs x are filtered through the weights w and combined to the previous hidden state to compute the activation of the hidden layer a. The result of such operation is transformed by the function Φ and eventually passed to the next unit through the transition matrix $w_{h^i h^j}$.

solved as easily, since the gradients become too small for computers to work with. If the gradient is not knowable, the direction of decreasing error is not known and weights can not be adjusted, eventually stopping the network learning.

D.3.2 LSTM networks

LSTMs are able to overcome the obstacle of gradient vanishing/explosion by leveraging additional gates to control the flow of information from hidden cell to the output and the next hidden state, limiting the storing of information for relevant data only. Due to the "decision making" capacity of the network, LSTMs are able to learn relationships over more than 1000 time steps.
The state of a LSTM layer is determined by the *hidden state* - otherwise known as the *output state* - and the *cell state*. The hidden state at time step t carries the output of the LSTM layer for this time step, while the cell state carries the information learned from the previous time steps. At every time step, the layer adds or removes information from the cell state according to the control exerted by gates. Such gates improves the capabilities of the LSTM to learn long-term relationships in

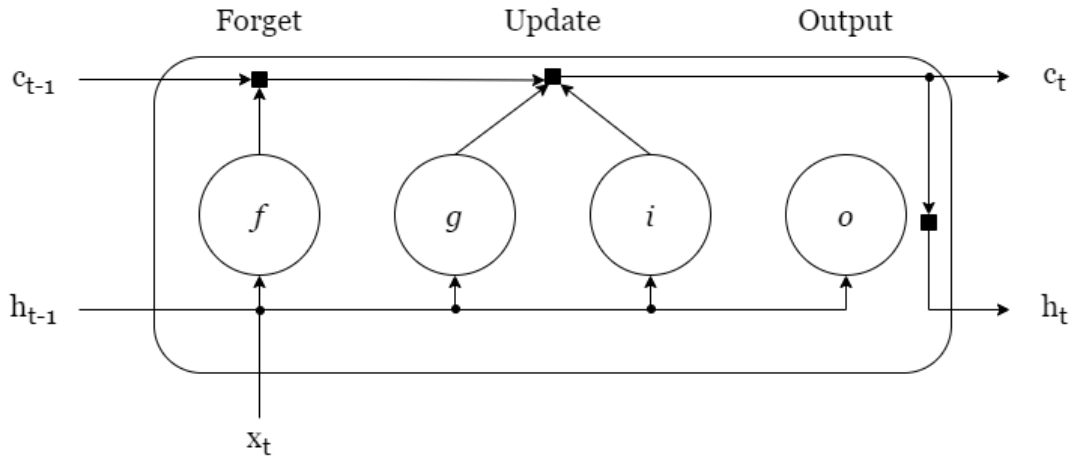


Figure D.3: The LSTM architecture has more gates to control information flow with respect to RNN. x_t: input, h_t: hidden state, c_t: Cell state, f_t: forget gate, g: memory cell, i: input gate, o: output gate.

the data; therefore, they are a widely implemented type of RNN. The architecture of a LSTM layer typically features with some fundamental elements [131] (Figure D.3):

- **Input gate i**: i controls the level of cell state update. The weights and biases to the input gate determine the extent to which a new value flows into the cell.

- **Forget gate f**: f controls the level of cell state reset (forget). The weights and biases to the forget gate determine the extent to which a value remains in the cell.
- **Memory cell g**: Also known as *cell candidate*, g adds information to the cell state.
- **Output gate o**: o controls the level of cell state added to hidden state. The weights and biases to the output gate determine the extent to which the value in the cell is used to compute the output activation of the LSTM block.

A LSTM network features with a set of learnable weights: the input weights W, the recurrent weights R and the bias b. The matrices W, R, and b bind the input weights, the recurrent weights, and the bias of each component of the LSTM layer (Equation D.39).

$$W = \begin{bmatrix} W_i \\ W_f \\ W_g \\ W_o \end{bmatrix} \qquad R = \begin{bmatrix} R_i \\ R_f \\ R_g \\ R_o \end{bmatrix} \qquad b = \begin{bmatrix} b_i \\ b_f \\ b_g \\ b_o \end{bmatrix} \tag{D.39}$$

Where i, f, g and o denote the input gate, the forget gate, the cell candidate and the output gate, respectively.
At time t, the cell state c_t and the hidden state h_t depend on the values of the layer components and on the *state activation function* σ_c (Equation D.40)[2].

$$\mathbf{c_t} = f_t \odot \mathbf{c_t} + i_t \odot g_t \qquad \mathbf{h_t} = o_t \odot \sigma_c(\mathbf{c_t}) \tag{D.40}$$

The values of the layer components at time step t are given as follows:

$$\begin{aligned} i_t &= \sigma_g(W_i\mathbf{x_t} + R_i\mathbf{h_{t-1}} + b_i) \\ f_t &= \sigma_g(W_f\mathbf{x_t} + R_f\mathbf{h_{t-1}} + b_f) \\ g_t &= \sigma_g(W_g\mathbf{x_t} + R_g\mathbf{h_{t-1}} + b_g) \\ o_t &= \sigma_g(W_o\mathbf{x_t} + R_o\mathbf{h_{t-1}} + b_o) \end{aligned}$$

Where σ_g represents the *gate activation function.*

[2]The symbol $\odot$ refers to the Hadamard product (element-wise product of vectors).

D.4 Temporal Convolutional Network (TCN) architecture

In the last decades, the task of sequence modeling in the framework of DeLe has been widely approached through RNN architectures, such as LSTM networks. However, recent studies [101] suggest that LSTM networks are obsolete, and that CNN architectures should be accounted as one of the main candidates for sequence modeling and classification. As demonstrated by Bai et al. [101], CNNs can attain better performance than RNNs in many tasks without the deficiencies of recurrent models, such as the exploding/vanishing gradient problem or the lack of memory retention. In addition, using a CNN instead of a RNN can lead to performance enhancements, as it enables multiple computation of outputs in parallel.
In particular, among the several types of convolutional networks, the TCN architecture emerged as the primary candidate for Seq2Seq classification tasks. A TCN is based on dilated, causal 1D convolutional layers having the same input and output lengths (see D.4.1 for a refresher on the 1D convolution operation).
TCNs rely on two principles [101]:

- The TCN returns an output that has the same length as the input.
- The prediction at time t is only influenced by the previous inputs i.e., there is no information leakage from the future into the past.

To achieve the first aspect, the TCN employs a 1D fully convolutional network (FCN) architecture[3] [123]. To accomplish the second point, the TCN employs causal convolutions i.e., convolutions where an output at time t is convoluted only with elements from time t and earlier in the previous layer. To reach a long effective history size, the network must be very deep or filters must be very large. Neither of these two accomplishments was feasible when the methods were first introduced; however, today such features can be met thanks to the technological advancements in the computational field. Nevertheless, TCNs usually have a larger memory footprint during inference with respect to RNNs, as the entire input sequence is required to calculate the output sample at the next time step [101].
The architecture of a TCN includes many blocks, layers and elements (Figure D.4) that will be discussed and presented in the next Section.

[3] In a FCN, each hidden layer is the same length as the input layer. Zero vectors of length (kernel size-1) is padded to keep the length of subsequent layers the same.

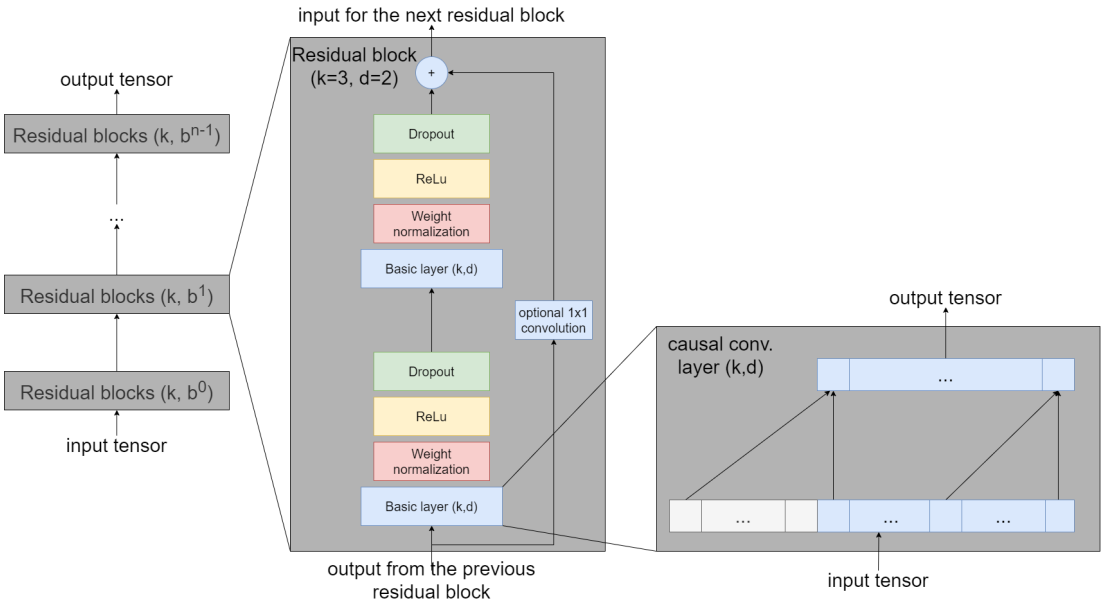


Figure D.4: Complete architecture of a TCN [101] [121].

D.4.1 1D Convolution

1D convolution represents the operation at the basis of CNNs. To visualize convolution, let's consider one single 1D convolutional layer that takes an input tensor of shape (*batch_size*, *input_length*, *nr_input_channels*) and returns an output tensor of shape (*batch_size*, *input_length*, *nr_output_channels*)[4], representing a batch of multivariate input and output sequences [121]. Then, let's consider a kernel of size *kernel_size*[5].
Let's focus on one single element of the batch (the same operations are applied to each element in the batch) and let's consider initially the univariate case, for simplicity:

$$nr\_input\_channels = nr\_output\_channels = 1$$

Such case corresponds to mapping a 1D input tensor to a 1D output tensor.
To calculate one element of the output of 1D convolution, a sub-sequence of consecutive elements of length kernel_size of the input is required. To obtain the output, the dot product of the sub-sequence of the input and a kernel vector of learned weights of the same length is calculated (Figure D.5). Such process is repeated for every element of the input sequence by shifting the sub-sequence window by one element to the right (Figure D.6). Notice that zero-padding at the beginning and the end of the input sequence is needed to ensure that the output sequence has the same length of the input sequence.
In the multivariate case ($nr\_input\_channels > 1$), the process described above is repeated for each input channel with a different kernel and the resulting *nr_input_channels* intermediate output vectors are summed. Such operation is equivalent to performing a 2D convolution with an input tensor of shape (*input_size*, *nr_input_channels*) and a kernel of shape (*kernel_size*, *nr_input_channels*). Since the sub-sequence window moves along a single axis only, it can be still considered 1D; however, a 2D kernel matrix is applied at every step (Figure D.7). If *nr_output_channels* is also larger than 1, the above procedure is just replicated for each output channel with a different kernel matrix. The output vectors are then piled up to create an output tensor of shape (*input_length*, *nr_output_channels*). The number of kernel weights equals to $kernel\_size \times nr\_input\_channels \times nr\_output\_channels$.

---

[4]Notice that in the case of TCNs, since every layer is of the same input and output length, just the third dimension of the input and output tensors differs [101]. If the input tensor consists in just one sequence ($nr\_input\_channels = 1$, univariate case), the sizes of the input and the output tensor will be both equal to one. In the more general multivariate case, *nr_input_channels* and *nr_output_channels* might be different.

[5]A kernel is basically a vector whose values are the multiplication factors i.e., the weights that are applied to the input elements values.

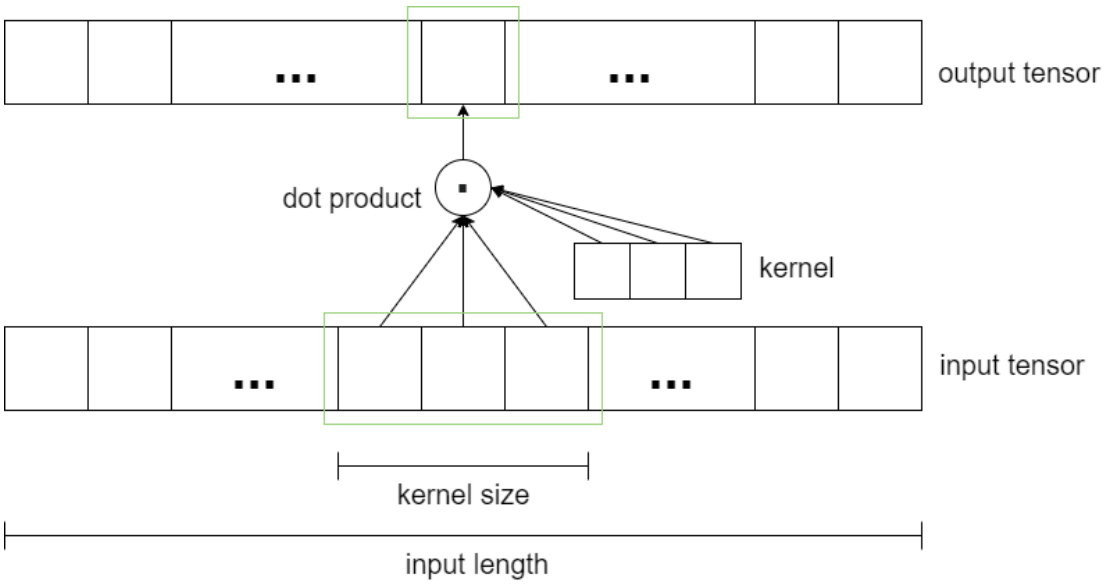


Figure D.5: The illustration unrolls the functioning of 1D convolution for a 1D tensor [121]. In this example, the kernel size is 3.

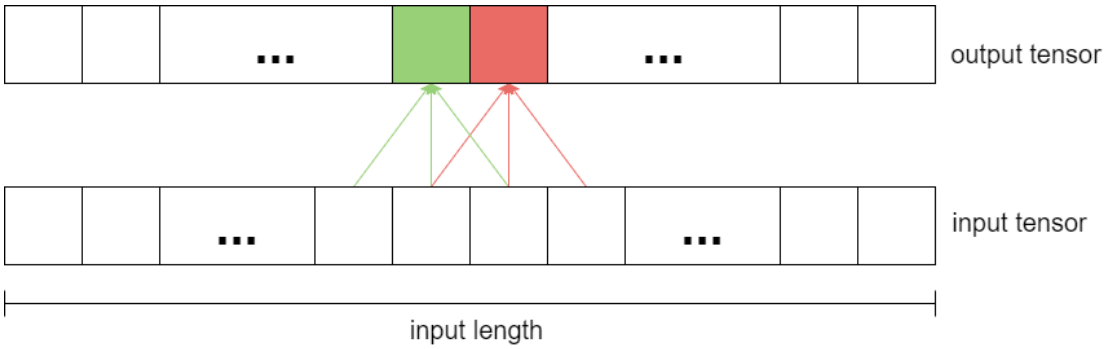


Figure D.6: Two consecutive output elements and their respective input subsequences convoluted by a kernel of size 3 [121]. To simplify the visualization, the dot product with the kernel vector is not shown anymore; however, it takes place for every output element with the same kernel weights.

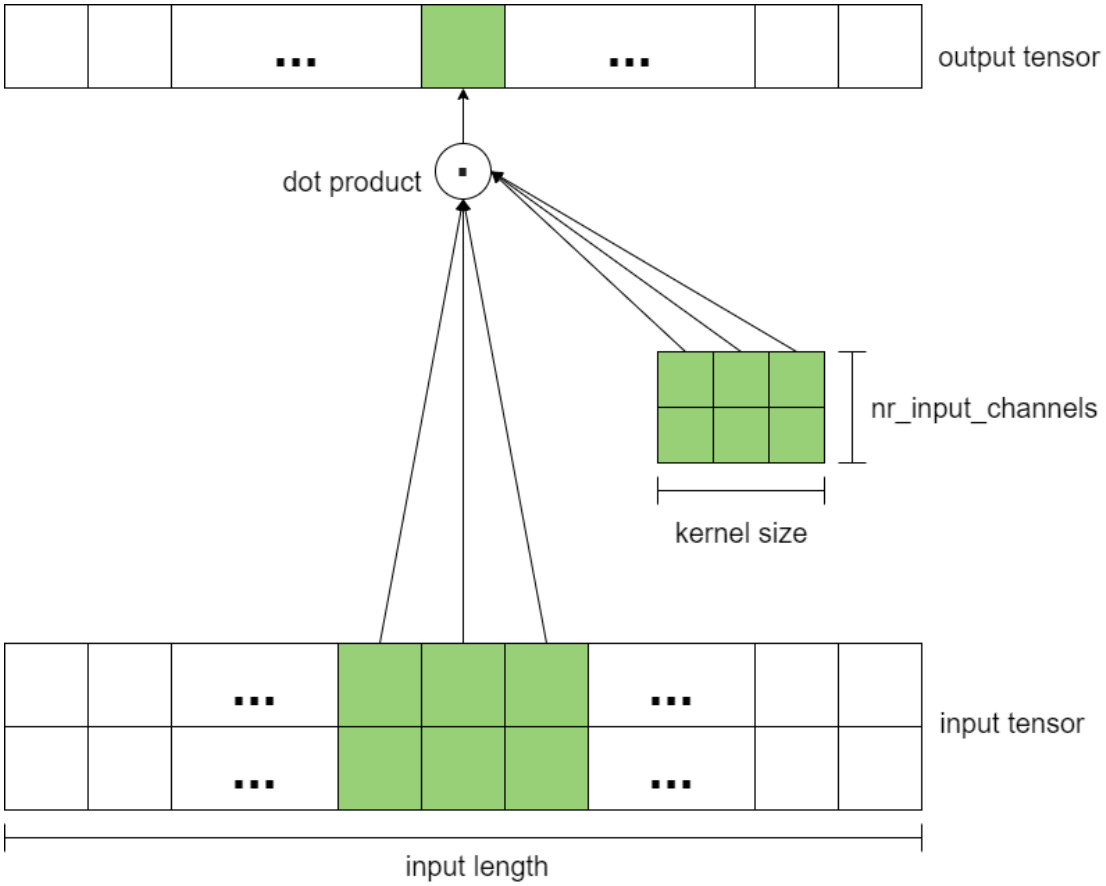


Figure D.7: 1D convolution with a multivariate tensor and a kernel vector is equivalent to a 2D convolution with a kernel matrix [121]. In this case, *kernel_size* equals to 3 and *nr_input_channels* equals to 2.

D.4.2 Causal convolution

As mentioned in 5.2.1, causal convolutions represent the operations at the core of a TCN. Given a causal convolutional layer, for every i in $[0, \ldots, input\_length - 1]$, the i^{th} element of the output channel only depends on the elements of the input sequence with indices from 0 to i i.e., it depends only on the previous input elements. To ensure convolution to be causal, zero padding can be applied only on the left side of the input tensor (Figure D.8). However, causal convolutions can only trace

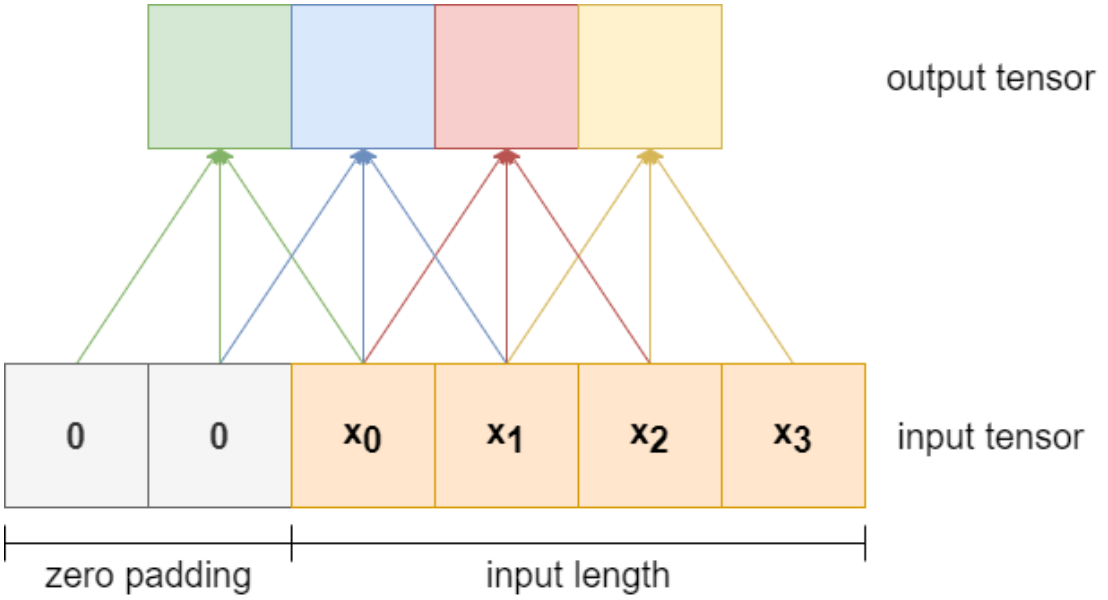


Figure D.8: Left-zero padding allows 1D causal convolution [121]. In this example, a kernel of size 3 is convoluted with an input tensor of length 4; therefore, 2 zeros must be appended to the left side of the input tensor to observe the causality condition.

back the input history with size linear, limiting the applicability of the causal convolution to sequence tasks with less deep history. Actually, we might want the output of a sequence forecasting model to depend on all previous entries in the input i.e., all entries occurring at a time step equal or previous to the current one [121]. Such quality is accomplished when the *receptive field* i.e., the set of entries of the original input that determine an individual entry of the output, has size equal to $input\_length$[6]. A simple 1D convolutional network - either causal or non-causal

[6]Such condition is called *full history coverage*.

- with n layers and a kernel of size k has a receptive field of size r (Equation D.41):

$$r = 1 + n(k - 1) \tag{D.41}$$

For instance, if we have a kernel of size 3, the 5^{th} element in the output will depend on elements 3, 4 and 5 of the input. As multiple layers are stacked on top of each other, the receptive filed is expanded according to Equation D.41. Referring to the previous example, by stacking two layers with kernel of size 3, a receptive field of size 5 is achieved i.e., the 5^{th} element in the output will be depending on elements from 1 to 5 in the input (Figure D.9). Notice that, given a fixed size of the kernel

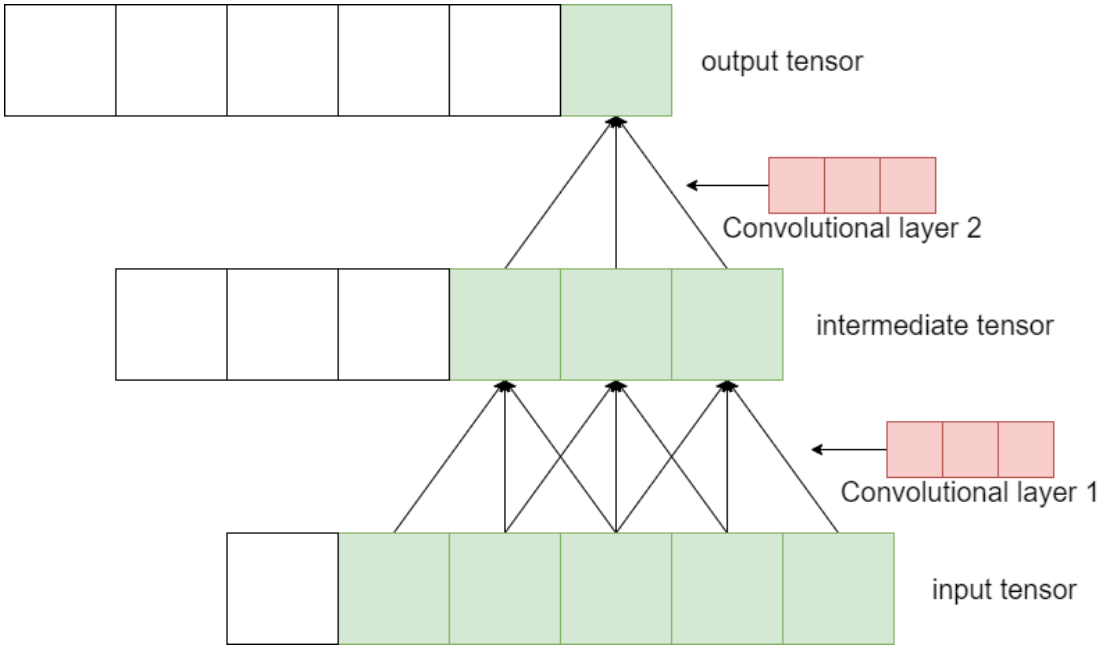


Figure D.9: Schematic illustration of the receptive field for a 2-layers convolutional network with kernels of size 3. The last sample of the output layer depends on the last 5 samples of the input layer.

k, the number of layers n required for full history coverage increases linearly to the length l of the input tensor (Equation D.42):

$$n = \frac{l - 1}{k - 1} \tag{D.42}$$

As a consequence, networks based on simple causal convolutional layers become rapidly very deep, resulting in models with a vast amount of learnable parameters i.e., a large number of kernel weights. In addition, a great number of layers has been correlated to degradation issues related to the loss function gradient [121].

D.4.3 Dilated convolution

Dilation allows to increase the size of the receptive field of the convolutional network without the need of too many layers. Dilating a convolutional layer means increasing the distance between the elements of the input sequence that are employed to compute the dot product that results in one entry of the output sequence. In general, a d-dilated layer with a kernel of size k has a receptive field that spreads across a length of $1+d(k-1)$ [121]. A simple convolutional layer could be considered as a 1-dilated layer, since the sub-sequence convolution window is composed by adjacent elements of the input sequence [121]. If d is fixed, the number of layers

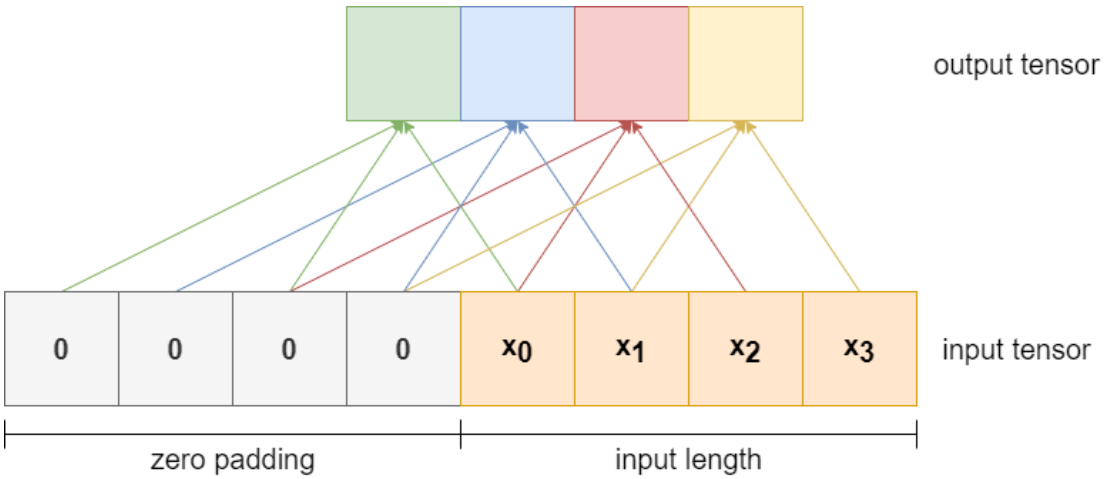


Figure D.10: The illustration displays an example of a 2-dilated layer with an input of length 4 [121]. Given a kernel of size 3, a 2-dilated convolutional layer has a receptive field of size 5, while a simple convolutional layer (1-dilated layer) would have a receptive field that spreads over a length of the same size of the kernel (3). Pink: zero padding; Orange: original input sequence.

needed to fully cover the length of the input tensor will still be linear to the input length. To avoid such inconvenience once for all, the value of d can be increased exponentially layer by layer. Such operation is performed by specifying a constant *dilation base* denoted by the integer b, which determines the value of the dilation factor d at the i^{th} layer (Equation D.43).

$$d = b^i \tag{D.43}$$

In such way, full input coverage can be achieved more easily (Figure D.11). In general, every layer adds a value of $d(k-1)$ to the current receptive field width, where d is computed according to Equation D.43. As a consequence, the size of the receptive field w of a TCN with exponential dilation of base b, kernel size k

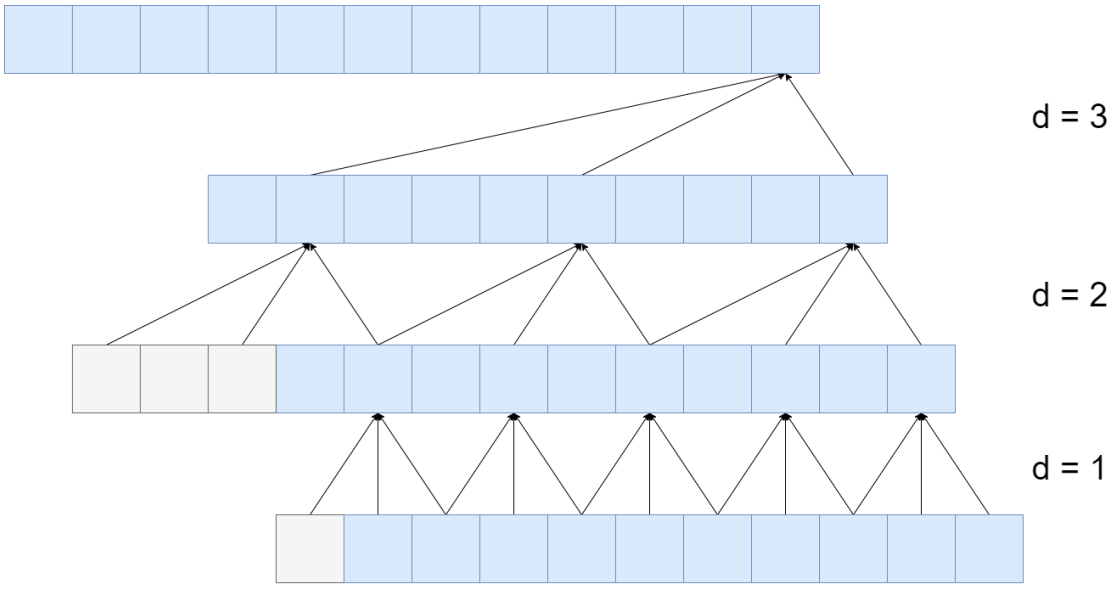


Figure D.11: The illustration shows a network with an input of length 10, a kernel of size 3 and a dilation base of 2 [121]. As a result of such hyperparameters, the network consists in 3 dilated convolutional layers that achieve full input coverage. For keeping the visualization simple, only the inputs that influence the last value of the output are shown. Similarly, only zero-padding entries that are needed for the last output value are displayed. Notice that, given the hyperparameters described above, full receptive field coverage could be maintained for lengths of the input up to 15. Pink: zero padding; Blue: original input sequence.

and number of layers n can be computed according to Equation D.44 [121].

$$w = 1 + \sum_{i=0}^{n-1}(k-1)b^i = 1 + (k-1)\frac{b^n - 1}{b-1} \tag{D.44}$$

Still, according to the values b and k, the resulting receptive field may have "holes", meaning that there are entries in the input sequence that the output value does not depend on i.e., some elements of the input sequence are ignored when the output is computed (Figure D.12). To fix such issue, two strategies may be adopted:

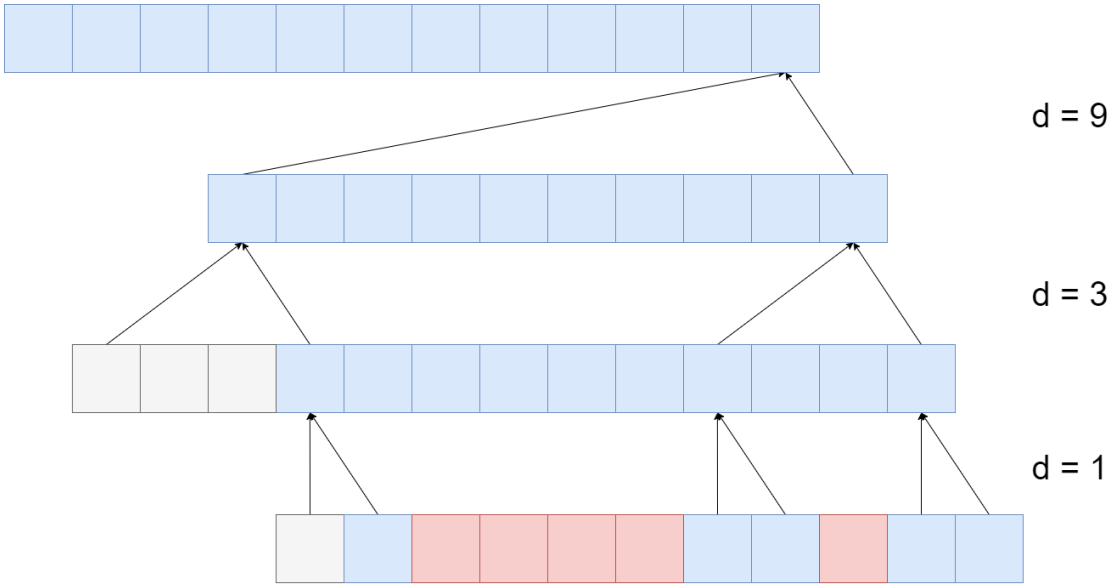


Figure D.12: The illustration shows the influence of the elements of the input sequence in the determination of the last output element for a network with dilation base of 3 and kernel size of 2 [121]. Elements in red represent holes in the receptive field.

1. Increase the size of the kernel.
2. Decrease the dilation base.

In general, one could easily demonstrate that, for a receptive field to have no holes, the kernel size k has to be greater than or equal to the dilation base b. To achieve full history coverage, the width of the receptive field w must overpass the input length l. Combining the condition $w > l$ to Equation D.41, the following inequality involving b, k, d and l is holds:

$$1 + \sum_{i=0}^{n-1}(k-1)b^i = 1 + (k-1)\frac{b^n - 1}{b-1} \geq l \tag{D.45}$$

Solving Equation D.45 for n, the minimum number of required layers can be determined (Equation D.46).

$$n = \log_b \frac{(l-1)(b-1)}{(k-1)} + 1 \tag{D.46}$$

Notice that the number of layers is not anymore linear in the length of the input, but rather logarithmic. Such improvement allows to reduce the number of required layers for achieving full receptive field coverage.
Eventually, if the number of layers below the current layer is i; then, the number of zero-padding entries p for the current layer to have full field coverage can be computed as follows [121]:

$$p = b^i(k-1) \tag{D.47}$$

D.4.4 Residual blocks

Recently, Bai et al. [101] introduced the use of *residual blocks* [122] for TCNs. A residual block is composed by 2 layers with the same dilation factor and a residual connection (Figure D.13). The output of the two convolutional layers within the residual block is added to the input of the residual block itself. The result is fed as input to the next residual block. For all inner blocks of the network except from the first and the last one, the widths of the input and output are the same - namely $num\_filters$. 1x1 convolution is typically used to adjust the widths of the residual tensors at the first and last residual blocks, as they may have different input and output channel widths [121].
Such variation affects the minimum amount of required layers for full history coverage. Appending a new residual block to a TCN extends the receptive field twice more than when appending a basic causal dilated layer, since it incorporates 2 such layers (Equation D.48).

$$w = 1 + \sum_{i=0}^{n-1} 2(k-1)b^i = 1 + 2(k-1)\frac{b^n - 1}{b-1} \tag{D.48}$$

Where r is the size of the receptive field, b is the dilation base, k is the kernel size ($k \geq b$) and n is the number of residual blocks for achieving a full history coverage. Then, the minimum number of residual blocks n for full history coverage of an input sequence of length l can be computed (Equation D.49).

$$n = \log_b \frac{(l-1)(b-1)}{2(k-1)} + 1 \tag{D.49}$$

D.4.5 Activation, Normalization, Regularization

Residual blocks include some additional elements (Figure D.14):

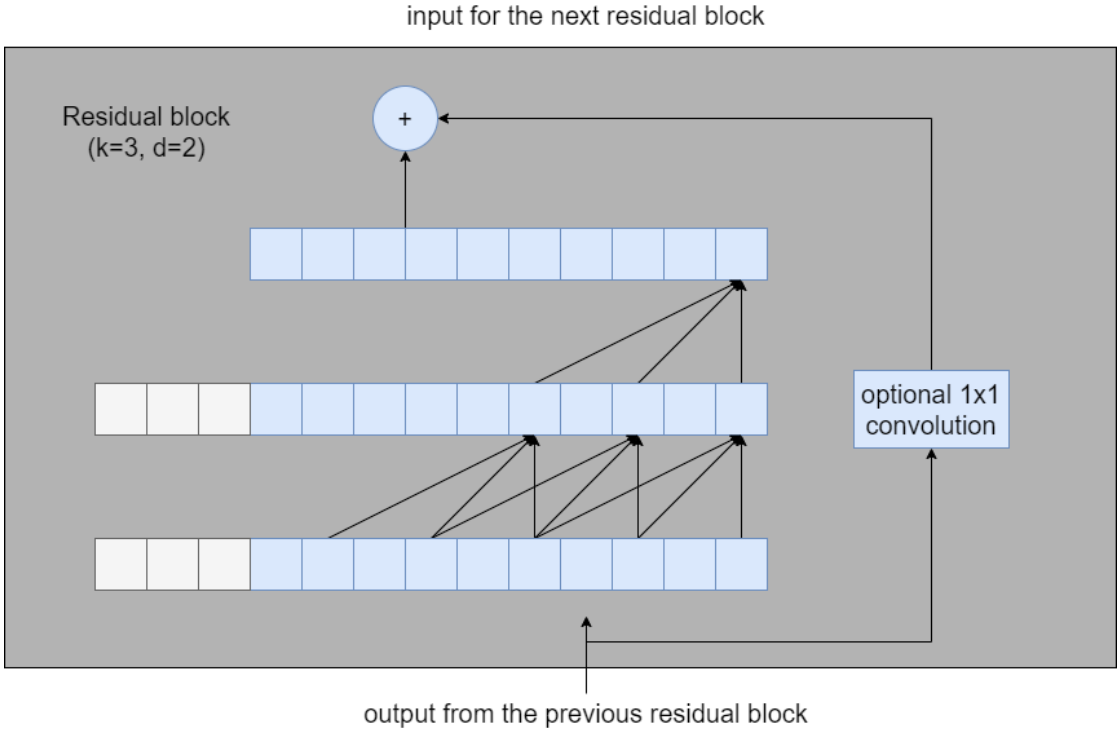


Figure D.13: Illustration of a residual block of a TCN with kernel of size 3 and dilation factor of 2 [121].

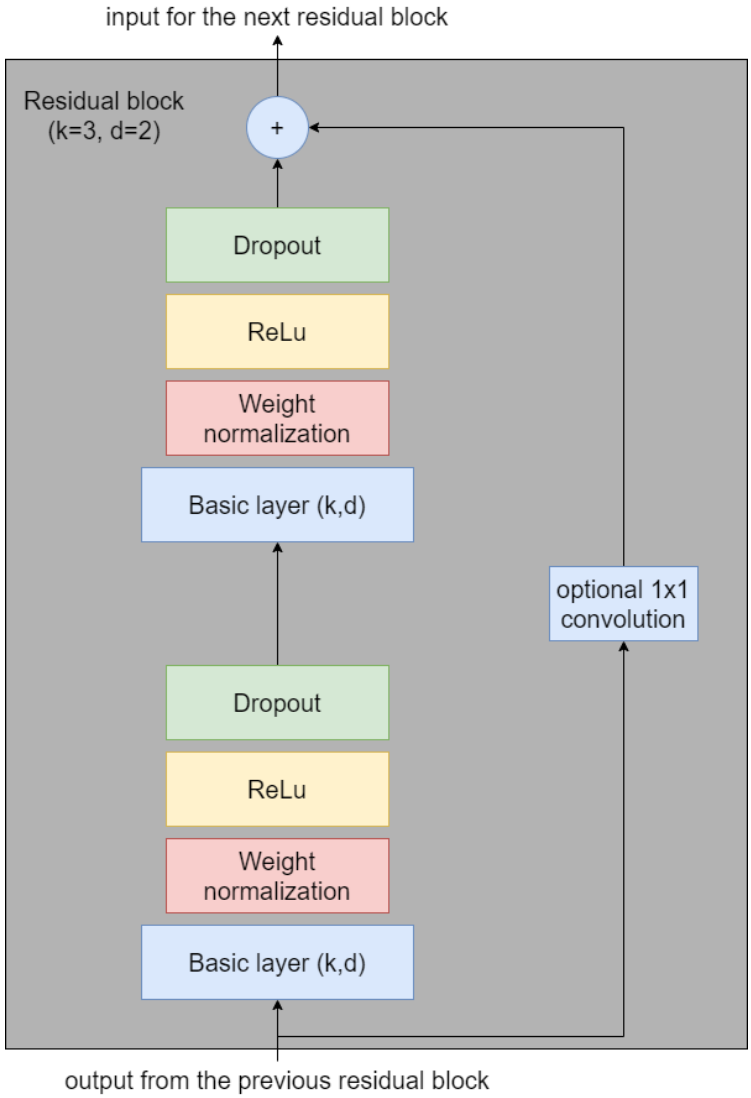


Figure D.14: Complete architecture of a residual block [101]. Notice that, at the last layer, the second ReLu is not present to allow the final output to display negative values as well [121].

- **Activation functions**: Activation functions are typically stacked on top of the convolutional layers to add non-linearities. Usually, ReLu activation functions are added to the residual blocks after both convolutional layers.
- **Normalization**: Normalization of the input of the hidden layers prevents the exploding gradient problems. Therefore, weight normalization is implemented at every convolutional layer.
- **Regularization**: Rregularization is introduced via *dropout* after every convolutional layer in every residual block to prevent overfitting.

D.5 Training algorithms

In the following Section, a brief introduction on training algorithms used for DeLe and MaLe neural networks is provided.
Generally, shallow and deep feed-forward neural networks are trained through BP. The BP learning algorithm moves backward from the final error through the outputs, weights and inputs of each hidden layer of the network. The algorithm assigns weights liability for a share of the error by computing their partial derivatives or the relationship between their rates of change. Those derivatives are then used by the gradient descent learning rule - otherwise known as solver- to adjust the weights in the direction of decreasing error.
On the other hand, RNNs typically rely on back-propagation through time (BPTT). Such algorithm extends the concept of BP to the time dimension, by expressing an ordered set of calculations that link one time step to the next one. When input sequences are relatively long, an approximation of BPTT called truncated BPTT is preferred to full BPTT, as full BPTT's computational cost per parameter update increases significantly over the time steps. On the other hand, with BPTT the gradient can not flow back to the full sequence, so the network struggles to learn much long-term dependencies [130].

Bibliography

[1] Claude Sammut and Geoffrey I. Webb, eds. *Encyclopedia of Machine Learning and Data Mining*. 2nd ed. Springer Reference. New York: Springer, 2017. ISBN: 978-1-4899-7685-7. DOI: 10.1007/978-1-4899-7687-1 (cit. on p. xxix).

[2] M. Bertoli, A. Cereatti, D. Trojaniello, et al. «Estimation of spatio-temporal parameters of gait from magneto-inertial measurement units: multicenter validation among Parkinson, mildly cognitively impaired and healthy older adults». In: *BioMed Eng OnLine* 17 (2018) (cit. on pp. 1, 16, 18, 27).

[3] Philipp Müller, Antonio J. del Ama, Juan C. Moreno, and Thomas Schauer. «Adaptive multichannel FES neuroprosthesis with learning control and automatic gait assessment». In: *Journal of NeuroEngineering and Rehabilitation* 17 (2020) (cit. on p. 1).

[4] K Aminian, P Robert, E Jéquier, and Y Schutz. «Incline, speed, and distance assessment during unconstrained walking». In: *Medicine and science in sports and exercise* 27.2 (Feb. 1995), pp. 226–234. ISSN: 0195-9131. URL: http://europepmc.org/abstract/MED/7723646 (cit. on p. 1).

[5] Magdalena Zuk, Magdalena Wojtków, Michal Popek, Jakub Mazur, and Katarzyna Bulinska. «Three-dimensional gait analysis using a virtual reality tracking system». In: *Measurement* 188 (2022), p. 110627. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2021.110627. URL: https://www.sciencedirect.com/science/article/pii/S0263224121014974 (cit. on p. 1).

[6] Aurelio Cappozzo, Ugo Della Croce, Alberto Leardini, and Lorenzo Chiari. «Human movement analysis using stereophotogrammetry: Part 1: theoretical background». In: *Gait and Posture* 21.2 (2005), pp. 186–196. ISSN: 0966-6362. DOI: https://doi.org/10.1016/j.gaitpost.2004.01.010. URL: https://www.sciencedirect.com/science/article/pii/S0966636204000256 (cit. on pp. 1, 24).

[7] Diana Trojaniello. «ASSESSMENT OF GAIT SPATIO-TEMPORAL PARAMETERS IN NEUROLOGICAL DISORDERS USING WEARABLE INERTIAL SENSORS». PhD thesis. Alma Mater Studiorum, Università di Bologna, 2015 (cit. on pp. 1, 2, 20, 23, 24).

[8] Anisoara Paraschiv Ionescu, Christopher J. Newman, Lena Carcreff, Corinna N. Gerber, Stephane Armand, and Kamiar Aminian. «Locomotion and cadence detection using a single trunk-fixed accelerometer: validity for children with cerebral palsy in daily life-like conditions». In: *Journal of NeuroEngineering and Rehabilitation* 16 (2019) (cit. on pp. 2, 15, 25, 27, 149).

[9] Giulia Pacini Panebianco, Maria Cristina Bisi, Rita Stagni, and Silvia Fantozzi. «Analysis of the performance of 17 algorythms from a systematic review: Influence of sensor position, analysed variable and computational approach in gait timing estimation from IMU measurements». In: *Gait & Posture* 66 (2019) (cit. on pp. 2, 17, 18, 27).

[10] Richard M. Kwasnicki, Raza Ali, Stevan J. Jordan, Louis Atallah, Julian J.H. Leong, Gareth G. Jones, Justin Cobb, Guang Zhong Yang, and Ara Darzi. «A wearable mobility assessment device for total knee replacement: A longitudinal feasibility study». In: *International Journal of Surgery* 18 (2015), pp. 14–20. ISSN: 1743-9191. DOI: https://doi.org/10.1016/j.ijsu.2015.04.032. URL: https://www.sciencedirect.com/science/article/pii/S1743919115001429 (cit. on p. 2).

[11] Christopher Buckley, Brook Galna, Lynn Rochester, and Claudia Mazzà. «Attenuation of Upper Body Accelerations during Gait: Piloting an Innovative Assessment Tool for Parkinson's Disease». English. In: *BioMed Research International* 2015 (Oct. 2015). ISSN: 2314-6133. DOI: 10.1155/2015/865873 (cit. on pp. 2–4, 8, 9).

[12] Shaghayegh Zihajehzadeh and Edward J. Park. «A Gaussian process regression model for walking speed estimation using a head-worn IMU». In: *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2017, pp. 2345–2348. DOI: 10.1109/EMBC.2017.8037326 (cit. on pp. 2, 87, 92, 149).

[13] Webren Zijlstra. «Assessment of spatio-temporal parameters during unconstrained walking». In: *European Journal of Applied Physiology* 92 (2004) (cit. on pp. 2, 7, 15, 16, 23, 144).

[14] Preeti Khera and Neelesh Kumar. «Role of machine learning in gait analysis: a review». In: *Journal of Medical Engineering & Technology* 44.8 (2020). PMID: 33078988, pp. 441–467. DOI: 10.1080/03091902.2020.1822940. eprint: https://doi.org/10.1080/03091902.2020.1822940. URL: https:

//doi.org/10.1080/03091902.2020.1822940 (cit. on pp. 2, 16, 25, 26, 28–35, 148, 150–152).

[15] John E.A. Bertram. «Locomotion: Why We Walk the Way We Walk». In: *Current Biology* (2015) (cit. on p. 7).

[16] J. Perry and M.G. Benedetti. *Analisi del movimento*. Ed. by Elsevier. 2005 (cit. on pp. 7–13).

[17] Herbert Elftman. «FORCES AND ENERGY CHANGES IN THE LEG DURING WALKING». In: *American Journal of Physiology* (1939) (cit. on p. 8).

[18] Los Amigos Research, Inc Education Institute, Rancho Los Amigos National Rehabilitation Center, Rancho Los Amigos National Rehabilitation Center. Pathokinesiology Service, and Rancho Los Amigos National Rehabilitation Center. Physical Therapy Department. *Observational Gait Analysis*. Los Amigos Research and Education Institute, Rancho Los Amigos National Rehabilitation Center, 2001. ISBN: 9780967633510. URL: https://books.google.it/books?id=sZdMPgAACAAJ (cit. on p. 9).

[19] Nikola Mijailoviü, Marijana Gavriloviü, and Stefan Rafajloviü. «Gait Phases Recognition from Accelerations and Ground Reaction Forces: Application of Neural Networks». In: *Telfor Journal* (2009) (cit. on p. 9).

[20] T.P. Andriacchi, J.A. Ogle, and J.O. Galante. «Walking speed as a basis for normal and abnormal gait measurements». In: *Journal of Biomechanics* 10.4 (1977), pp. 261–268. ISSN: 0021-9290. DOI: https://doi.org/10.1016/0021-9290(77)90049-5. URL: https://www.sciencedirect.com/science/article/pii/0021929077900495 (cit. on p. 10).

[21] J C Otis and A H Burstein. «Evaluation of the VA-Rancho Gait Analyzer, Mark I.» In: *Bulletin of prosthetics research* 10-35 (1981), pp. 21–5 (cit. on p. 10).

[22] W Zijlstra and AL. Hof. «Assessment of spatio-temporal gait parameters from trunk accelerations during human walking.» In: *Gait Posture* 18 (2003), pp. 1–10 (cit. on pp. 10, 23, 25).

[23] D Mayich, Alison Novak, Daniel Vena, Timothy Daniels, and James Brodsky. «Gait Analysis in Orthopedic Foot and Ankle Surgery-Topical Review, Part 1: Principles and Uses of Gait Analysis». In: *Foot & ankle international. / American Orthopaedic Foot and Ankle Society [and] Swiss Foot and Ankle Society* 35 (Nov. 2013). DOI: 10.1177/1071100713508394 (cit. on p. 13).

[24] Richard Baker. «The history of gait analysis before the advent of modern computers». In: *Gait & Posture* 26 (2007) (cit. on pp. 14, 15).

[25] Aristotle. *Parts of animals, movement of animals, progression of animals*. Harvard University, 1968 (cit. on p. 14).

[26] Descartes Rene. *Treatise of man*. Prometheus Books, 1972 (cit. on p. 14).

[27] Borelli Giovanni. *On the movement of animals*. Springer-Verlag, 1989 (cit. on p. 14).

[28] W Weber and E Weber. *Mechanics of the human walking apparatus*. Springer-Verlag, 1991 (cit. on pp. 14, 15).

[29] M. Braun. *Picturing time: the work of Etienne-Jules Marey (1830 - 1904)*. University of Chicago Press, 1992 (cit. on p. 14).

[30] R Taft. *Introduction. In: Muybridge E, editor. The human figure in motion*. Dover, 1955 (cit. on p. 14).

[31] W Braune and O. Fischer. *On the centre of gravity of the human body*. Springer-Verlag, 1984 (cit. on p. 14).

[32] J. Amar. *Trottoir dynamographique*. Comptes rendus hebdomadaires des seances de l'Academie des Sciences, 1916 (cit. on p. 14).

[33] J.D.M. Saunders, V.T. Inman, and H.D. Eberhart. «The major determinants in normal and pathological gait». In: *J Bone Joint Surg* (1953) (cit. on p. 14).

[34] V. Inman, H. Ralston, and F. Todd. *Human walking*. Williams and Wilkins, 1981 (cit. on p. 14).

[35] J Perry. *gait analysis*. SLACK, 1992 (cit. on p. 15).

[36] J Perry. «Clinical gait analyzer». In: *Bull Prosthet Res* (1974) (cit. on p. 15).

[37] D. Sutherland and J. Hagy. «Measurement of gait movements from motion picture film». In: *BJ Bone Joint Surg* (1972) (cit. on p. 15).

[38] E.Y. Chao. «Justification of triaxial goniometer for the measurement of joint rotation». In: *J Biomec* (1980) (cit. on p. 15).

[39] *International Classification of Functioning, Disability and Health*. Standard. World Health Organization, May 2001 (cit. on p. 15).

[40] E. Odding, H.A. Valkenburg, H.J. Stam, and Albert Hofman. «Determinants of locomotor disability in people aged 55 years and over: The Rotterdam study». In: *European Journal of Epidemiology* 17 (2001) (cit. on p. 16).

[41] A König, L. Klaming, M. Pijl, A. Demeurraux, R. David, and P. Robert. «Objective measurement of gait parameters in healthy and cognitively impaired elderly using the dual-task paradigm». In: *Aging Clin Exp Res.* 29 (2017), pp. 1181–1189 (cit. on p. 16).

[42] G.K. Rose. «Clinical gait assessment: a personal view». In: *Journal of Medical Engineering and Technology* (1983) (cit. on p. 16).

[43] E. Dolatabadi, B. Taati, and A. Mihailidis. «An automated classification of pathological gait using unobtrusive sensing technology». In: *IEEE Trans Neural Syst Rehabil Eng* 25 (2017), p. 2336 (cit. on pp. 16, 28).

[44] Chen W. «Classification of normal and pathological gait in young children based on foot pressure data». In: *Neuroinformatics* 15 (2017), p. 13 (cit. on pp. 16, 30, 34).

[45] Michael W. Whittle. «Clinical gait analysis: a review». In: *Human Movement Science* (1996) (cit. on p. 16).

[46] Peter McGinnis. *Biomechanics of Sport and Exercise*. Ed. by College at Cortland : Human Kinetics. 2013 (cit. on pp. 16, 17).

[47] F. Bugané, M.G. Benedetti, G. Casadio, S. Attalaa, F. Biagi, M. Mancac, and A. Leardini. «Estimation of spatial-temporal gait parameters in level walking based on a single accelerometer: Validation on normal subjects by standard gait analysis». In: *Computer Methods and Programs in Biomedicine* 108 (2012) (cit. on p. 18).

[48] Elisa Digo. *Comparison of different inertial sensors setups and algorithms to estimate gait spatio-temporal parameters*. 2018 (cit. on pp. 18–22).

[49] Stacy Fritz and Michelle Lusardi. «Walking Speed: the Sixth Vital Sign». In: *Journal of Geriatric Physical Therapy* 32 (2009), pp. 2–5 (cit. on p. 18).

[50] R. Simon Sheldon. «Quantification of human motion: gait analysis—benefits and limitations to its application to clinical problems». In: *Journal of Biomechanics* 37 (2004), pp. 1869–1880 (cit. on p. 18).

[51] Benoît Sijobert, Mourad Benoussaad, Jennifer Denys, Roger Pissard-Gibollet, Christian Geny, and Christine Azevedo Coste. «Implementation and Validation of a Stride Length Estimation Algorithm, Using a Single Basic Inertial Sensor on Healthy Subjects and Patients Suffering from Parkinson's Disease». In: *Health* 7 (2015), pp. 704–714 (cit. on pp. 18, 27).

[52] Kathleen A. Lamkin-Kennard and Marko B. Popovic. «4 - Sensors: Natural and Synthetic Sensors». In: *Biomechatronics*. Ed. by Marko B. Popovic. Academic Press, 2019, pp. 81–107. ISBN: 978-0-12-812939-5. DOI: https://doi.org/10.1016/B978-0-12-812939-5.00004-5. URL: https://www.sciencedirect.com/science/article/pii/B9780128129395000045 (cit. on p. 20).

[53] GAITRITE. 2022. URL: https://www.gaitrite.com/ (visited on 11/15/2022) (cit. on p. 21).

[54] Valentina Agostini, Gabriella Balestra, and Marco Knaflitz. «Segmentation and Classification of Gait Cycles». In: *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society* 22 (Nov. 2013). DOI: 10.1109/TNSRE.2013.2291907 (cit. on p. 22).

[55] Zhuolin Yang, Chen Song, Feng Lin, Jeanne Langan, and Wenyao Xu. «A Smart Environment-Adapting Timed-Up-and-Go System Powered by Sensor-Embedded Insoles». In: *IEEE Internet of Things Journal* 6.2 (2019), pp. 1298–1305. DOI: 10.1109/JIOT.2018.2844837 (cit. on p. 23).

[56] Andrea Cereatti. *Slide del corso di Teleriabilitazione*. University Lecture. 2022 (cit. on pp. 26, 40).

[57] Robbin Romijnders, Elke Warmerdam, Clint Hansen, Gerhard Schmidt, and Walter Maetzler. «A Deep Learning Approach for Gait Event Detection from a Single Shank-Worn IMU: Validation in Healthy and Neurological Cohorts». In: 22.10 (2022). ISSN: 1424-8220. DOI: 10.3390/s22103859. URL: https://www.mdpi.com/1424-8220/22/10/3859 (cit. on pp. 26, 65).

[58] Arash Salarian, Heike Russmann, François Vingerhoets, C. Dehollain, Yves Blanc, Pierre Burkhard, and Kamiar Aminian. «Gait Assessment in Parkinson's Disease: Toward an Ambulatory System for Long-Term Monitoring». In: *IEEE transactions on bio-medical engineering* 51 (Sept. 2004), pp. 1434–43. DOI: 10.1109/TBME.2004.827933 (cit. on p. 27).

[59] Tecla Bonci, Alison Keogh, Silvia Del Din, Kirsty Scott, Claudia Mazzà, and on behalf of the Mobilise-D consortium. «An Objective Methodology for the Selection of a Device for Continuous Mobility Assessment». In: *Sensors* 20.22 (2020). ISSN: 1424-8220. DOI: 10.3390/s20226509. URL: https://www.mdpi.com/1424-8220/20/22/6509 (cit. on p. 27).

[60] Irina Galperin et al. «Associations between daily-living physical activity and laboratory-based assessments of motor severity in patients with falls and Parkinson's disease». In: *Parkinsonism and Related Disorders* 62 (May 2019). DOI: 10.1016/j.parkreldis.2019.01.022 (cit. on p. 27).

[61] Delaram Jarchi, Benny Lo, Charence Wong, Edmund Ieong, Dinesh Nathwani, and Guang-Zhong Yang. «Gait Analysis From a Single Ear-Worn Sensor: Reliability and Clinical Evaluation for Orthopaedic Patients». In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 24 (Jan. 2015), pp. 1–1. DOI: 10.1109/TNSRE.2015.2477720 (cit. on p. 27).

[62] A. Soltani, K Aminian, C. Mazza, Cereatti A., L. Palmerini, T. Bonci, and A. Paraschiv-Ionescu. «Algorithms for Walking Speed Estimation Using a Lower-Back-Worn Inertial Sensor: A Cross-Validation on Speed Ranges.» In: *IEEE Trans Neural Syst Rehabil Eng* (2021) (cit. on p. 27).

[63] Diana Trojaniello, Andrea Cereatti, and Ugo Della Croce. «Accuracy, sensitivity and robustness of five different methods for the estimation of gait temporal parameters using a single inertial sensor mounted on the lower trunk». In: *Gait and Posture* 40 (Sept. 2014). DOI: 10.1016/j.gaitpost.2014.07.007 (cit. on p. 27).

[64] Palaniswami M. «Computational intelligence in gait research: a perspective on current applications and future challenges». In: *IEEE Trans Inf Technol Biomed* 13 (2009), p. 687 (cit. on pp. 28, 34).

[65] Amorim P. «Trajectory-based gait pattern shift detection for assistive robotics applications». In: *Intel Serv Robot* 12 (2019), p. 255 (cit. on pp. 28, 35).

[66] Lemaire ED. «Design, development, and evaluation of a local sensor-based gait phase recognition system using a logistic model decision tree for orthosis-control». In: *J Neuroeng Rehabil* 16 (2019), p. 22 (cit. on pp. 28, 35).

[67] Zhu H. «Support vector machine for classification of walking conditions using miniature kinematic sensors». In: *Med Biol Eng Comput* 46 (2008), p. 563 (cit. on p. 28).

[68] Kamruzzaman J. «Neural networks for detection and classification of walking pattern changes due to ageing». In: *Aust Phys Eng Sci Med* 29 (2006), p. 188 (cit. on p. 28).

[69] Stemmer MR. «Extraction and classification of human body parameters for gait analysis». In: *J Control Autom Electr Syst* 29 (2018), p. 586 (cit. on pp. 28, 31, 35).

[70] Sangmin L. «Multi-channel electromyography pattern classification using deep belief networks for enhanced user experience». In: *J Central South Univ* 22 (2015), p. 1801 (cit. on p. 28).

[71] Yap MH. «Automated analysis and quantification of human mobility using a depth sensor». In: *IEEE J Biomed Health Inform* 21 (2017), p. 939 (cit. on p. 28).

[72] Iskandar PM. «A knowledge-based intelligent framework for anterior cruciate ligament rehabilitation monitoring». In: *Appl Soft Comput* 20 (2014), p. 127 (cit. on pp. 28, 31).

[73] Lello ED. «Probabilistic gait classification in children with cerebral palsy: a Bayesian approach». In: *Res Dev Disabil* 32 (2011), p. 2542 (cit. on p. 28).

[74] Findlow AH. «Automated nonlinear feature generation and classification of foot pressure lesions». In: *IEEE Trans Inf Technol Biomed* 14 (2010), p. 418 (cit. on pp. 28, 29).

[75] Lopez M. «Bag-of-steps: predicting lower-limb fracture rehabilitation length». In: *Neurocomputing* 268 (2017), p. 109 (cit. on p. 28).

[76] Li H. «Intelligent prediction of human lower extremity joint moment: an artificial neural network approach». In: *IEEE Access Spec Sect Data Enab Intell Dig Health* 7 (2019), p. 29973 (cit. on p. 28).

[77] Pappas E. «Classification of gait patterns in patients with unilateral anterior cruciate ligament deficiency based on phase space reconstruction». In: *Soft Comput* 24 (2020), p. 1851 (cit. on p. 28).

[78] Brower R. «Application of wearable sensors for human gait analysis using fuzzy computational algorithm». In: *Eng Appl Artif Intell* 24 (2011), p. 1018 (cit. on p. 28).

[79] Owen B. «Support vector machines for automated gait classification». In: *IEEE Trans Biomed Eng* 52 (2005), p. 828 (cit. on pp. 28, 29).

[80] Ubeda A. «Decoding the attentional demands of gait through EEG gamma band features». In: *PLoS One* 11 (2016), e0154136 (cit. on pp. 28, 31).

[81] Tan KC. «Spatio-spectral representation learning for electroencephalographic gait-pattern classification». In: *IEEE Trans Neural Syst Rehabil Eng* 26 (2018), p. 1858 (cit. on pp. 28, 29, 34).

[82] Nunes UJ. «ISR-AIWALKER: robotic walker for intuitive and safe mobility assistance and gait analysis». In: *IEEE Trans Hum Mach Syst* 47 (2017), p. 1110 (cit. on pp. 28, 29).

[83] Schache A. «Classification of gait disorders following traumatic brain injury». In: *J Head Trauma Rehabil* 30 (2015), E13 (cit. on pp. 29, 30).

[84] Wang Z. «Body sensor network-based gait quality assessment for clinical decision-support via multi-sensor fusion». In: *IEEE Access Spec Sect Wireless Body Area Netw* 7 (2019), p. 59884 (cit. on p. 29).

[85] R Huang, Z Peng, H Cheng, et al. «Learning-based walking assistance control strategy for a lower limb exoskeleton with hemiplegia patients». In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), pp. 2280–2285 (cit. on pp. 29, 36).

[86] John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. «A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence». In: *AI Magazine* 27 (2006) (cit. on p. 29).

[87] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2005. ISBN: 9780262256834. DOI: 10.7551/mitpress/3206.001.0001. URL: https://doi.org/10.7551/mitpress/3206.001.0001 (cit. on pp. 30, 167–169).

[88] Andrea Mannini and Angelo Sabatini. «Gait phase detection and discrimination between walking-jogging activities using hidden Markov models applied to foot motion data from a gyroscope». In: *Gait and posture* 36 (July 2012), pp. 657–61. DOI: 10.1016/j.gaitpost.2012.06.017 (cit. on p. 31).

[89] Nandi GC. «Less computationally intensive fuzzy logic (type-1)-based controller for humanoid push recovery». In: *Rob Auton Syst* 63 (2015), p. 122 (cit. on p. 31).

[90] Gabriella Balestra. *Slide del corso di Intelligenza Artificiale in Medicina*. University Lecture. 2021 (cit. on p. 34).

[91] Filippo Molinari. *Slide del corso di Elaborazione di IMmagini Mediche*. University Lecture. 2022 (cit. on pp. 34, 35).

[92] Vectornav. *Inertial Navigation Primer*. 2022. URL: https://www.vectornav.com/resources/inertial-navigation-primer/theory-of-operation/theory-inertial (cit. on p. 37).

[93] Zakriya Mohammed, Ibrahim (Abe) M. Elfadel, and Mahmoud Rasras. «Monolithic Multi Degree of Freedom (MDoF) Capacitive MEMS Accelerometers». In: *Micromachines* 9 (2018) (cit. on pp. 39–42).

[94] Riccardo Antonello and Roberto Oboe. «MEMS Gyroscopes for Consumers and Industrial Applications». In: June 2011. ISBN: 978-953-307-170-1. DOI: 10.5772/17689 (cit. on pp. 42, 43).

[95] Stefano Bertuletti, Andrea Cereatti, Daniele Comotti, Michele Caldara, and Ugo Della Croce. «Static and Dynamic Accuracy of an Innovative Miniaturized Wearable Platform for Short Range Distance Measurements for Human Movement Applications». In: *Sensors* 17.7 (2017). ISSN: 1424-8220. DOI: 10.3390/s17071492. URL: https://www.mdpi.com/1424-8220/17/7/1492 (cit. on pp. 45, 47, 48).

[96] F. Salis, S. Bertuletti, K. Scott, M. Caruso, T. Bonci, E. Buckley, U. Della Croce, C. Mazzà, and A. Cereatti. «A wearable multi-sensor system for real world gait analysis.» In: *Annu Int Conf IEEE Eng Med Biol Soc.* (2019) (cit. on pp. 47, 48, 50, 51, 149, 154).

[97] *STMicroelectronics VL6180X Official Web Page*. https://www.st.com/en/imaging-and-photonics-solutions/vl6180x.html. Accessed: 2022-10-20 (cit. on pp. 48, 50).

[98] *Medium SEQ2SEQ LEARNING*. https://medium.com/deep-learning-with-keras/part-a-introduction-to-seq2seq-learning-a-sample-solution-with-mlp-network-95dc0bcb9c83. Accessed: 2022-11-05 (cit. on p. 64).

[99] MATLAB. *version 9.12.0.1927505 (R2022a)*. Natick, Massachusetts: The MathWorks Inc., 2022 (cit. on pp. 65, 76, 80, 87, 91, 94–96, 105, 106, 167, 169, 172–175).

[100] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. 2018. DOI: 10.48550/ARXIV.1803.01271. URL: https://arxiv.org/abs/1803.01271 (cit. on pp. 65, 167).

[101] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. 2018. DOI: 10.48550/ARXIV.1803.01271. URL: https://arxiv.org/abs/1803.01271 (cit. on pp. 65, 167, 181–183, 191, 193).

[102] Matteo Gadaleta, Giulia Cisotto, Michele Rossi, Rana zia ur Rehman, Lynn Rochester, and Silvia Din. «Deep Learning Techniques for Improving Digital Gait Segmentation». In: (July 2019) (cit. on pp. 67, 147, 149).

[103] Xavier Glorot and Yoshua Bengio. «Understanding the difficulty of training deep feedforward neural networks». In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html (cit. on p. 76).

[104] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. *Exact solutions to the nonlinear dynamics of learning in deep linear neural networks*. 2013. DOI: 10.48550/ARXIV.1312.6120. URL: https://arxiv.org/abs/1312.6120 (cit. on p. 77).

[105] C. M. Bishop. *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006 (cit. on pp. 77, 78, 81).

[106] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. «Dropout: A Simple Way to Prevent Neural Networks from Overfitting». In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html (cit. on pp. 79, 167).

[107] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (cit. on pp. 79, 167).

[108] Diederik Kingma and Jimmy Ba. «Adam: A Method for Stochastic Optimization». In: *International Conference on Learning Representations* (Dec. 2014) (cit. on p. 80).

[109] K. P. Murphy. *Machine Learning: A Probabilistic Perspective.* Cambridge, MA: The MIT Press, 2012 (cit. on p. 81).

[110] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. «On the difficulty of training recurrent neural networks». In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, June 2013, pp. 1310–1318. URL: https://proceedings.mlr.press/v28/pascanu13.html (cit. on p. 81).

[111] Lawrence R. Rabiner and Bernard Gold. *Theory and application of digital signal processing / Lawrence R. Rabiner, Bernard Gold.* eng. Englewood Cliffs, N.J.: Prentice-Hall, 1975. ISBN: 0139141014 (cit. on p. 90).

[112] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. 2e. New York, NY, USA: Springer, 2006 (cit. on p. 95).

[113] Tom Sharp. *Medium An Introduction to Support Vector Regression (SVR)*. 2020. URL: https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2 (visited on 11/12/2022) (cit. on pp. 96, 172, 173).

[114] Stephen V. Stehman. «Selecting and interpreting measures of thematic classification accuracy». In: *Remote Sensing of Environment* 62.1 (1997), pp. 77–89. ISSN: 0034-4257. DOI: https://doi.org/10.1016/S0034-4257(97)00083-7. URL: https://www.sciencedirect.com/science/article/pii/S0034425797000837 (cit. on p. 98).

[115] David Powers. «Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation». In: *Mach. Learn. Technol.* 2 (Jan. 2008) (cit. on p. 98).

[116] Working with Planning. *MAPE(Mean Absolute Percentage Error)*. URL: https://docs.oracle.com/en/cloud/saas/planning-budgeting-cloud/pfusu/insights_metrics_MAPE.html (visited on 11/14/2022) (cit. on p. 106).

[117] Tong-Hun Hwang, Julia Reh, Alfred Effenberg, and Holger Blume. «Validation of Real Time Gait Analysis Using a Single Head-Worn IMU». In: Jan. 2021, pp. 87–97. ISBN: 978-981-15-8349-0. DOI: 10.1007/978-981-15-8350-6_8 (cit. on p. 147).

[118] Javier Marin. *Explainable Deep Neural Networks*. 2021. URL: https://towardsdatascience.com/explainable-deep-neural-networks-2f40b89d4d6f (visited on 11/26/2022) (cit. on p. 151).

[119] Jiayuan Ding et al. «DANCE: A Deep Learning Library and Benchmark for Single-Cell Analysis». In: *bioRxiv* (2022). DOI: 10.1101/2022.10.19.512741. eprint: https://www.biorxiv.org/content/early/2022/10/24/2022.10.19.512741.full.pdf. URL: https://www.biorxiv.org/content/early/2022/10/24/2022.10.19.512741 (cit. on p. 151).

[120] *Medium LSTM: Understanding Output Types*. https://medium.com/deep-learning-with-keras/lstm-understanding-output-types-e93d2fb57c77. Accessed: 2022-11-04 (cit. on pp. 167, 175).

[121] *Unit8 Temporal Convolutional Networks and Forecasting*. https://unit8.com/resources/temporal-convolutional-networks-and-forecasting/. Accessed: 2022-11-06 (cit. on pp. 167, 182–193).

[122] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385 (cit. on pp. 167, 191).

[123] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2014. DOI: 10.48550/ARXIV.1411.4038. URL: https://arxiv.org/abs/1411.4038 (cit. on pp. 167, 181).

[124] R. M. Neal. *Bayesian Learning for Neural Networks, Vol. 118 of Lecture Notes in Statistics*. Springer-Verlag, 1996 (cit. on p. 170).

[125] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995. ISBN: 0-387-94559-8 (cit. on p. 171).

[126] Wikipedia. *Flat function — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Flat%20function&oldid=1109025563. [Online; accessed 12-November-2022]. 2022 (cit. on p. 172).

[127] John Platt. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*. Tech. rep. MSR-TR-98-14. Microsoft, Apr. 1998. URL: https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/ (cit. on p. 174).

[128] Pai-Hsuen Chen, Rong-En Fan, and Chih-Jen Lin. «A Study on SMO-Type Decomposition Methods for Support Vector Machines». In: *Trans. Neur. Netw.* 17.4 (July 2006), pp. 893–908. ISSN: 1045-9227. DOI: 10.1109/TNN.2006.875973. URL: https://doi.org/10.1109/TNN.2006.875973 (cit. on p. 174).

[129] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. «Working Set Selection Using Second Order Information for Training Support Vector Machines». In: *J. Mach. Learn. Res.* 6 (Dec. 2005), pp. 1889–1918. ISSN: 1532-4435 (cit. on p. 174).

[130] Pathmind. *A Beginner's Guide to LSTMs and Recurrent Neural Networks*. 2020. URL: https://wiki.pathmind.com/lstm#two (visited on 03/11/2022) (cit. on pp. 177, 194).

[131] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-Term Memory». In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (cit. on p. 179).