Applying deep-learning techniques to detect freezing of gait episodes in Parkinson's disease patients

Facultat d'informàtica de Barcelona | Facultat de matemàtiques | Escola tècnica superior d'enginyeria

Julià Camps Sereix
julia.camps.sereix@est.fib.upc.edu

Dr Albert Samà[1] and Dr Mario Martín[2]

[1]Technical Research Centre for Dependency Care and Autonomous Living, CETPD, Universitat Politècnica de Catalunya, Barcelona Tech., Rambla de l'Exposició 59-69, Vilanova i la Geltrú 08800, Spain,

[2]Knowledge Engineering and Machine Learning Group, Universitat Politècnica de Catalunya, Barcelona Tech., C/ Jordi Girona 1-3, Barcelona, 08034, Spain,

A thesis submitted for the

Master in Artificial Intelligence

June 30, 2017

Acknowledgements

Part of this project has been performed within the framework of the Freezing in Parkinson's Disease: Improving Quality of Life with an Automatic Control System (MASPARK) [3] project which is funded by La Fundació La Marató de TV3 20140431. This work also forms part of the framework of the FP7 Personal Health Device for the Remote and Autonomous Management of Parkinsons Disease (REMPARK) [7] project ICT-287677, which is funded by the European Community. The author, thus, would like to acknowledge the contributions of the members from MASPARK and REMPARK consortium, and to his colleagues from the Technical Research Centre for Dependency Care and Autonomous Living for sharing their facilities and resources.

Abstract

Freezing of gait (FOG) is one of the most incapacitating symptoms among the motor alterations of Parkinson's disease (PD). Manifesting FOG episodes reduces the quality of life of patients and their autonomy to perform daily living activities, while it may provoke falls. Accurate ambulatory FOG assessment would enable non-pharmacologic support based on cues and would provide relevant information to neurologists on the disease evolution.

This master thesis presents a method for FOG detection based on deep learning and signal processing techniques. This thesis is, to the best of the author's knowledge, the first study to address FOG detection with deep learning strategies. The evaluation of the model has been done based on the data from 21 PD patients who manifested FOG. The data employed throughout this study were recorded using an inertial measurement unit placed at the left side of the patients' waist. These data were composed of 3 tri-axial signals corresponding to measurements from accelerometer, gyroscope and magnetometer.

The results obtained by our approach in detecting FOG outperform the state-of-the-art ones, achieving performances higher than 90% for the geometric mean between test sensitivity and test specificity. Furthermore, two recurrent extensions of our original approach were implemented and compared to assess its temporal representation capacity.

This work also reproduces four feature extraction methodologies from those composing the state-of-the-art for comparing them to our approaches. Concretely, suitable machine learning algorithms for binary classification tasks were trained with these features and later compared to the deep learning methods. From this comparison, it was possible to reaffirm that the presented approaches are, at the moment, the best strategy for automatic FOG detection.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Parkinson's disease (PD) is a long-term progressive neurological condition, resulting from the degeneration of dopamine-producing neurones. PD is the second most common neurodegenerative disorder after Alzheimer's disease. The prevalence of PD is approximately of 1% among people of age above 65 [135, 99, 88, 32, 92, 100, 101]. Although, there are indicators that relate the disease with genetic factors, the cause of PD is still unknown [118, 64]. There is no cure for PD, however, its average life expectancy is about 12 years, after being diagnosed [118, 64, 131].

Dopamine is a neurotransmitter involved in movement control [65, 61, 11]. A deficit of neurones responsible for producing this neurotransmitter typifies PD's pathology. As a result, PD patients manifest several motor and non-motor symptoms, such as Parkinsonian tremors. PD's tremor is, however, one of the symptoms with lower impact on patients' quality of life (QOL). Concretely, this symptom only affects patients' limbs at rest and disappears during involuntary movements. On the other hand, bradykinesia (slowness of movements), muscles stiffness, postural alteration and freezing of gait (FOG) symptoms manifestations may difficult the performance of activities of daily living (ADL), and, thus, reduce the patients' autonomy [125, 118, 100, 80, 61, 76].

Current clinical methods to assess PD's patients' symptoms are commonly based on in-lab examinations, symptoms questionnaires and symptoms diaries. These methods, as described in Section 1.2, provide only partial information regarding the patients' condition.

Some aspects of PD, such as its genetic representation towards diagnosis and complex symptoms, such as FOG, are still poorly understood [45, 50]. The disease is diagnosed in practice by observation of indicative known symptoms plus having a

positive response to the current medication therapies [61, 64]. On the other hand, as described in Section 1.2, the assessment of FOG is mainly performed with freezing of gait questionnaire (FOG-Q) and in-lab periodic tests [62, 47, 63, 108].

Assigning human-specialist assistants to control all PD patients is infeasible due to its prevalence and to the difficult social acceptance of this solution. Therefore, alternative non-intrusive techniques, such as wearable technologies [75], have recently gained the attention of the research community towards solving these problems by performing an automatic quantitative assessment of the disease's signs.

Especially, automatic FOG detection may permit to increase the understanding of this symptom, which may lead to discovering effective treatments for it. Furthermore, performing this detection in a real-time manner [79], would enable to provide online support to patients through rhythmic auditory cues, which may significantly enhance the patients' autonomy during their ADL [15, 148].

Wearable sensors are increasingly becoming a common practice for detecting PD's motor symptoms in the research community due to the increase of computation power in wearable sensing devices and the adoption of these technologies for biomedical research [81, 16, 75]. The state-of-the-art on algorithms for automatic FOG detection are shallow machine learning (ML) algorithms applied to signals acquired from inertial wearable sensors [79, 141, 149, 31, 12, 77, 110]. The state-of-the-art performance for FOG detection is defined by performances about 85% [31, 110] for the geometric mean (GM) between sensitivity and specificity. However, the complexity in designing handcrafted features and the scarcity of data from PD patients collected under real-life-like conditions for developing reliable solutions are the major impediments preventing the research community from mastering the problem.

Feature learning is a set of methods that learns a transformation of raw data input to a representation that can be exploited by ML methods. Deep learning (DL) methods are feature learning methods with multiple levels of representation [72]. DL models can learn feature extractions that can easily handle multimodal data, missing information and high dimensional feature's spaces [72, 53]. Thus, when working with DL methods, the manual feature engineering can be obviated, which is otherwise necessary for traditional ML methods. Furthermore, DL models can outperform shallow ML algorithms when enough data to represent the complexity of a target problem are provided adequately. Concretely, DL models have recently exhibited a breakthrough in several complex problems, even outperforming humans in complex tasks such as image classification [59] and playing games [128]. Some other remarkable applications which are currently possible to implement with DL techniques are: face

detection [134], image generation [142], video-frame prediction [37], speech recognition [90] and audio generation [90] and question answering [146, 54]. In this master thesis, it is hypothesised that DL methods will improve the state-of-the-art results in FoG detection through wearable sensors.

1.2 Background

Accurate automatic symptoms detection in PD patients has the potential of providing relevant indicators about their condition [34], enabling clinicians to maintain their regimens updated, without the workload overhead of current gold-standard techniques. These features may improve both: the patients' QOL by objectively adjusting their treatments, and the clinicians' efficiency by providing them with relevant and objective information about the patients' motor states.

Among PD's motor symptoms there are few symptoms, such as FOG, which may manifest even when patients are under the medication's influence, while most of them, such as bradykinesia, stiffness and postural alteration, can be effectively treated by the current medication-based therapies. The severity of these medication-responsive symptoms can be reduced by increasing the patients' dopamine concentration [139, 61, 11]. Directly administrating dopamine to PD patients was, however, found useless to combat the disease due to the protective bloodbrain barrier, which prevents dopamine from entering the required brain areas [11]. Levodopa (L-DOPA) is a precursor of dopamine which can overcome this limitation [11]. L-DOPA-based therapies are the most widely adopted medication treatments in PD [11, 94].

Long-term usage of L-DOPA in PD patients has, however, two major side effects: motor fluctuations and dyskinesias [120]. Motor fluctuations are transitions between ON and OFF states, which are described as the presence and absence of PD's symptoms, respectively. These symptomatic motor states are referred as the ON-state when medication is effective, and, thus, the symptoms have a minimum presence, and OFF-state when the symptoms regain presence [120]. The transitions between these motor states are referred as motor fluctuations. Typically, after some years of being administrated L-DOPA-based therapies, patients start to experience more frequently OFF-states even when having taken their medications. In these cases, motor fluctuations may occur from three to five times per day [140, 120]. Within the same period, L-DOPA-induced dyskinesias (choreic movements) may appear.

Monotonic L-DOPA-based therapies are, therefore, insufficient to deal with the whole life-cycle of the disease [74, 11]. Other non-dopaminergic alternatives, such

as apomorphine [98, 39], which aim to increase the dopamine levels by reducing its destruction in the human body instead of increasing its production rate, have also been included in PD's therapies to reduce L-DOPA's side effects. Dopamine agonists' proclivity to cause psychotoxicity has, however, limited their adoption as L-DOPA substitutes for the entire PD's duration [74, 39].

The course of the disease and medication treatments associated are typically composed of the following stages:

- Pre-L-DOPA: During the first months after the diagnosis, dopamine agonists or MAO-B inhibitors monotherapies are prescribed to delay the start of the L-DOPA administration [74, 120].

- L-DOPA 'honeymoon' period: Due to PD's symptoms worsening, dopamine agonists or MAO-B inhibitors monotherapies become insufficient, L-DOPA based therapies are employed [74]. Due to its long-term usage side effects, L-DOPA-based therapies are designed to minimise the quantity of L-DOPA administrated, for example, by combining L-DOPA with peripheral dopa decarboxylase inhibitors, such as carbidopa and benserazide, which increase L-DOPA's bioavailability [74, 100, 120]. This stage typically lasts from 2 to 6 years, although the disease experience is variable to each patient.

- L-DOPA with motor fluctuations: Once the side effects from long-term L-DOPA administration appear, there are several strategies, such as occasional apomorphine subcutaneous injections, which may be employed before adopting any of the final intrusive solutions [107, 89, 119, 114]. Apomorphine is a dopamine receptor agonist administrated in PD by subcutaneous injections due to its potential for treating PD's fluctuations [14, 98]. However, there is no gold standard in how to adapt the L-DOPA-based therapies in PD once these treatments become less effective. One of the common strategies is to fractionate the oral L-DOPA doses to reduce the changes in its in-serum levels. However, other techniques may vary depending on the criteria from the local clinicians and the patients' condition, for example, its a common practice to prescribe apomorphine injections to patients that suffer severe OFF-states, which should occasionally be used, due to its proclivity to cause psychotoxicity.

- Advanced PD: Advanced stages of the disease tend to present severe disabling motor and non-motor complications that cannot be effectively managed by pulsatile L-DOPA-based therapies. Thus, at this time, more complex and invasive

techniques such as drug infusion pumps or deep brain stimulation (DBS) are employed [48, 95, 120, 114, 94].

DBS is a neurological procedure which consists of placing electrodes in the region of the ventral intermediate nucleus of the thalamus and sending electrical impulses [95]. These intervention, although it can help to improve the patients' QOL, has dangerous side effects, including intracranial hemorrhage and infection [38, 95, 35, 138, 44, 116, 117].

Currently, the main infusion techniques are subcutaneous apomorphine and intraduodenal L-DOPA [114, 33]. Intraduodenal L-DOPA is the natural continuation of the carbidopa/L-DOPA oral therapy [119, 84], which increases the stability of the dopamine levels compared to oral supply. However, in some cases, patients may become untreatable by L-DOPA-based therapies. Thus, other drugs, such as apomorphine, have been evaluated to replace L-DOPA [14, 40, 98, 74, 84]. Subcutaneous apomorphine infusions are still employed as default treatment in substitution of carbidopa/L-DOPA oral treatment in some cases that it would be feasible to employ intraduodenal L-DOPA. This tendency is, however, decreasing due to apomorphine's proclivity to cause psychotoxicity and addiction [98, 48, 95, 136, 140, 120, 145, 114, 33, 84], such as the dopamine dysregulation syndrome [93] .

Although several studies have been undertaken to compare the pros and cons of these techniques, the selection criteria between these alternatives may differ depending on the health regulations and medical centre.

Effective PD's therapies should prevent its progression while abolishing motor and cognitive handicaps. However, none of the existing therapies meets all these needs [74]. Therefore, therapies require several adjustments and changes throughout the course of the disease, which need to be accurate therapy personalisation for each patient from continuous controls in their state and response to the designed therapy.

In the current clinical practice, most patients are assessed by the neurologist every 3 to 6 months [6]. Current clinical practice for controlling and assessing the PD patients' states and therapies is time-consuming. Periodical in-lab tests, symptoms' questionnaires and symptoms' diaries are some of the gold-standard PD's regimen assessment techniques. On the one hand, in-lab tests serve to assess the patients' movement capabilities from in-lab observations from the neurologists. However, these tests tend to be insufficient to represent the patients' daily experiences. Concretely,

similar factors than the ones triggering the 'white coat' effect [96], such as being under observation and performing the tests in controlled environments, may produce this lack of generalisation of in-lab tests' observations. While, on the other hand, symptoms' diaries and questionnaires serve to inform the neurologists about the disease evolution by identifying indicators, such as motor fluctuations, in the patients' descriptions. Regarding symptoms' diaries, the patients' role is to maintain a record of the symptoms that they experience while their ADL, while in questionnaires they are required to answer questions about their condition and recent experiences. However, the reliability of these two written assessment techniques is subject to each patients' mental capabilities, which may be affected by other PD's non-motor symptoms, such as dementia. Therefore, only techniques based on continued observation and evaluation mechanisms can ensure optimal therapy control for all PD patients [6].

This master thesis focuses on FOG, which is a poorly understood symptom associated with PD condition. This symptom is usually manifested in episodes shorter than 10 seconds (s) [125]. Suffering these episodes may provoke falling accidents when patients are willing to perform walking related actions [20]. According to Nieuwboer and Giladi [86], FOG might be defined as an inability to deal with concurrent cognitive, limbic, and motor inputs, causing an interruption of locomotion. Administrating drugs has proven to diminish most of PD's symptoms successfully, however, FOG episodes may manifest regardless on the patient's medication state [46, 125]. FOG's frequency is, furthermore, influenced by the patient's mental state and other environmental factors, such as walking through narrow spaces or shifting obstacles [20, 80]. As already mentioned, in-lab observations may differ from the patient's daily experiences [96]. Moreover, FOG's episodic and environment sensitive nature may even introduce additional bias to the results of these tests, converting FOG assessment into one of the most difficult to accurately assess among PD's motor symptoms [47, 48, 125]. The gold-standard strategy to assess FOG is the FOG-Q. The FOG-Q has proved to provide relevant indicators for the identification and characterisation of this symptom [47, 87, 49]; however, the informative power of this technique is insufficient for research purposes towards gaining a better understanding of the symptom [87]. Many medical research studies have been carried out to discover strategies to combat this symptom. These studies have proved that techniques such as to induce an acoustic or visual external rhythm to PD patients improve their walking capacity while minimising FOG's incisive frequency [15].

1.3 Objectives

Due to all these characteristics, this master thesis aims to present a novel DL-based approach capable of outperforming the state-of-the-art methods for FOG detection using inertial wearable sensors data acquired from PD patients at their homes while performing various ADL. Furthermore, this is, to the best of the author's knowledge, the first study to adopt DL models for addressing FOG detection in PD patients.

Several combinations of one-dimensional (1D) convolutional neural networks (ConvNets) architectures and data representations will be evaluated and optimised to achieve this goal.

The temporal representation capacity of our approach is evaluated by comparing its performance against two extensions which replace the hidden dense (fully connected) layers by recurrent layers. Our hypothesis is that if the 1D-ConvNet properly exploits the temporal information, none of its recurrent extensions should outperform our former approach. Note, however, that the only model being properly tuned is the 1D-ConvNet, since its architecture (i.e. the number of layers and neurones per layer) will be unchanged in the extensions, for simplicity and time constraints.

Concretely, the two recurrent extensions of our approach will be implemented. Concretely, the 1D-ConvNets models hidden layers are firstly interchanged by long-short-term-memory (LSTM) [60] layers producing a 1D-convolutional-long-short-term-memory network (1D-ConvLSTM), and, secondly, by gated-recurrent-unit (GRU) [25, 27] layers, producing a 1D-convolutional-gated-recurrent-unit network (1D-ConvGRU). These three DL-based approaches will be evaluated on the same data to assess the temporal representation capacity of the feed-forward approach.

To objectively assess the performances achieved by the DL methods implemented, these will, moreover, be compared to the current state-of-the-art methodologies for FOG detection from inertial sensors data in the literature. Within this second comparative study, the state-of-the-art feature extraction approaches will be fairly reproduced on the same data employed within this master thesis.

Other authors' feature extraction approaches selected for being reproduced are: I) the approach presented by Bächlin et al. in 2009 [16], which is hereinafter referred as the Moore-Bächlin FOG Algorithm (MBFA); II) the approach presented by Mazilu et al. in 2012 [79], which is an extension of the MBFA; III) the approach presented by Tripoliti et al. in 2013 [141]; and IV) the approach presented by Rodríguez et al. in 2016 [31]. Further details on their work are reviewed in Section 1.5, where the major

accomplishments of these studies are commented, and in Chapter 5, which exposes the details of the reproduction of these feature extraction approaches.

The resulting features from these approaches will be used for training powerful binary classification ML shallow algorithms, such as tree bagging [22], adaptive boosting (AdaBoost) [42], adaptive logistic regression boosting (LogitBoost) [43], random undersampling boosting (RUSBoost), robust adaptive boosting (RobustBoost) [41] and support vector machines (SVMs) [30], by leave-one-patient-out cross-validation on the overall training data used for the DL approaches. Finally, the resulting models from the hyperparameters exploration will be retrained on the overall training data and tested on the testing data selected for the DL methods to produce a fair comparison between both methodologies.

1.4 Framework

This study was conducted in the Technical Research Centre for Dependency Care and Autonomous Living (CETpD) [9], a research centre composed by researchers from the Department of Electronic Engineering and the Department of Automatic Control at the Universitat Politècnica de Catalunya (UPC).

The CETpD's researchers have conducted numerous different studies. However, their expertise areas are the following:

- Designing and implementing wearable devices for human activity recognition (HAR) for multiple purposes such as fall detection and online PD's motor symptoms monitoring [109, 112, 110].

- Inertial signal data analysis [123], which they perform over the data recorded by specific wearable devices in numerous areas, such as sports activity monitoring in professional athletes and HAR from smartphones [106, 85, 105].

- Machine learning applications for pattern detection in time series data [121, 111].

- Medical data analysis and projects, mainly related studies targeting elder people support systems and PD's symptoms monitoring [122, 113, 120, 115].

The research centre has a strong cooperation with medical staff of several hospitals, such as Hospital Sant Antoni Abat [2] in Vilanova i la Geltrú and Centro Médico Teknon [1] in Barcelona.

Data employed in this master thesis were collected within a European project research project, the Personal Health Device for the Remote and Autonomous Management of Parkinson's Disease (REMPARK) project [7], within which its members generated a great database of PD patients' inertial signal recordings while performing ADL. The REMPARK project main aims were the identification of OFF-states and L-DOPA-induced dyskinesia episodes in PD patients. These data are further described in Section 3.1.

1.5 Related work

Automatic FOG detection is still an open research issue despite having been widely addressed by several combinations of devices and algorithms. This section reviews some of these approaches.

Moore et al. in 2008 [81] made the first attempt to detect FOG automatically. They presented a novel method for automatic FOG detection in PD patients. The data for performing their study were composed of inertial signals recorded by microelectromechanical systems (MEMS) placed at the left shank of 11 PD patients who manifested FOG episodes. Their approach consists of a freeze index (FI) threshold, where FI is defined as the ratio between the power spectral density in the gait freezing band (FB) (i.e. 3–8 hertz (Hz)) and in the locomotion band (LB) (i.e. 0.5–3 Hz), which are applied window-wisely over tri-axial accelerometer recorded data, which is windowed into splits of 6 s. Considering the simplicity of the method, they were able to achieve highly accurate results. Concretely, they were able to detect 78% of FOG events correctly employing the same threshold for all patients' data. Furthermore, calibrating the threshold to each patient improved the method's performance to an 89%. However, they perform a weak evaluation on data containing only 46 FOG events, while not reporting the exact evaluation methodology employed. Furthermore, the data acquisition was performed twice per patient, first in OFF-state (without medication) and later in ON-state (while being under the drug's influence).

Later in 2009, Bächlin et al. [16] illustrated an extension of the method designed by Moore et al. [81] referred as MBFA, as mentioned in Section 1.3, which targeted online FOG monitoring. They introduced two main changes: I) they introduced an additional power index (PI) threshold for signal within the 0.5–8 Hz to discard standing-data as FOG episodes candidates; II) they employed a window duration of 4 s. They reported 73.1% and 81.6% for sensitivity and specificity, respectively, implementing a general threshold, whereas, with personalised thresholds, they achieved

88.6% for sensibility and 92.8% for specificity, in the task of online FOG monitoring. These results were, however, computed permitting an offset margin of 2 s of error for the predictions, which enhanced the results for both, sensitivity and specificity. Due to its simplicity, the MBFA method has been adopted by the research community as the basic performance reference for automatic FOG detection in PD patients inertial sensors data. The data employed throughout their study were the Daphnet dataset [17]. These data were collected in the Dynamic Analysis of Physiological Networks (DAPHNet) project, a European project. The data were composed by one hour of tri-axial accelerometer measurements recorded with three MEMS, placed to the shank (above the ankle), the thigh (above the knee) and the lower back. The collection was performed across 8 PD patients who manifested FOG episodes. The data collection was performed in-lab settings and under controlled conditions. Patients were required to complete three walking tasks: I) walking back and forth in a straight line, including several 180 degrees turns; II) randomly walking following the instructions of the therapists, which included stops and turns; III) and walking to another room and coming back with a cup of water. The patients and therapists agreed that the lower back was the most acceptable of the three sensors' placements employed. The Daphnet dataset was designed to permit the design of automatic FOG detection methods; however, the conditions defining the data collection protocol oversimplify the FOG detection's task, permitting to design highly accurate models, in test data, which would be useless in real environments.

The work from Bächlin et al. [16] was continued by Mazilu et al. in 2012 [79]. They presented a novel approach for monitoring FOG in PD patients, which combines the usage of smartphones and wearable accelerometers for data-collection. They employed, for the first time, ML algorithms for the online FOG detection task. Some of the ML algorithms they tested were: random forests, decision trees, naive Bayes and k-nearest neighbours (k-NN). They reported top results of 66.25% and 95.38% for sensitivity and specificity, respectively, using user-independent settings and random forests as classifier algorithm. The data employed throughout their study were the Daphnet dataset [17].

In 2013, Moore et al. [82] published a comparative study of different configurations of sensors and placements and signal processing parameters in PD patients for FOG monitoring with the MBFA. The data for their study were composed of inertial signals recorded from seven sensors in different body placements, which were collected from 25 PD patients. Thus, they were able to test all possible combinations of sensors' recorded data to assess the best locations for FOG monitoring objectively. They

showed that sensitivity performance was highly bound to the window size, retrieving results above 70% with all sensor combinations with window times ranging from 2.5 s to 5 s. Not surprisingly, the best sensors configuration was to employ all seven sensors simultaneously. Nevertheless, they recap with a recommendation of three possible configurations for FOG monitoring based on the performance obtained and some convenience criteria. The first recommended setting suggest implementing all seven sensors, though they claim this one to be taken as a reference configuration while focusing more attention on single shank sensor and single back sensor configurations, which although being suboptimal configurations also exhibited good results.

Within the same year, Tripoliti et al. [141] proposed an automatic FOG detection methodology. Their methodology consisted of four stages: data cleaning, filtering, feature extraction and classification. As classifiers, they tested the following ML algorithms: naive Bayes, random forests, decision trees and random trees. They reported best results by applying leave-one-patient-out cross-validation with the random forests algorithm, for which they achieved 89.3% and 79.15% for sensitivity and specificity, respectively, when considering only the results associated with PD patients who suffered the FOG symptom. The data they employed throughout the study were collected from 16 people, of whom 5 were healthy subjects, 5 were PD patients who manifested the FOG symptom, and the remaining 6 were PD patients who suffered from other symptoms. The devices for collecting the data were 6 accelerometers and 2 gyroscopes attached to different positions of the subject's body.

In 2016, Mazilu et al. [77] presented a study for assessing the representation power of wrist-worn sensing data compared to lower-limb sensing, which is the state-of-the-art for FOG monitoring in PD patients. Their approach implemented the C4.5 ML algorithm. Although they employed a more relaxed evaluation mechanism than in their previous work [79], they were unable to reach state-of-the-art performance in the target task. However, they still suggested that wrist sensing could be a feasible and comfortable alternative for FOG monitoring in PD patients.

Within the same year, Rodríguez et al. [31] presented a study aiming at FOG detection in PD patients during their ADL, and adopting the support vector machines (SVMs) for the FOG binary classification task. They proposed an innovative feature extraction which is designed to be implementable in low-power consumption wearables for online FOG detection. The data, which was composed of inertial signal recordings at 40 Hz from a single inertial measurement unit (IMU) placed at the left side of the waist, was acquired following the same conventions reviewed in [81]. However, laboratory data acquisition biases the data with information related to the experiment

characteristics as in [81] and [79]. Thus, they collected the data at the patients' homes, configuring each test to adapt to the real activities in which the patient would experience FOG, rather than employing homogeneous lab settings to force patients to trigger FOG events. Although they have not reported test error results in this work, they performed a comparative study of the state-of-the-art feature extraction techniques for FOG detection, while reporting cross-validation error when training a model for each combination of ML algorithm (e.g. k-NN, random forests, naive Bayes, logistic regression and SVM) and feature extraction strategy. Furthermore, they considered different window sizes (i.e. ranging from 0.8 s to 6.4 s) to maximise the representation power of each configuration. Their results suggested that SVMs with their proposed feature generation are powerful strategies for FOG detection since the cross-validation performance for this configurations were the most accurate among all regardless of the window size. Concretely, highest results were achieved by using a window duration of 1.6 s, for which they reported 89.77% as the GM between sensitivity and specificity.

Recently, in 2017, Rodríguez et al. [110] presented an extension of [31] and [112]. They present a complete review of the state-of-the-art for automatic FOG monitoring while giving specific details on their methodologies for performing FOG monitoring with SVMs and their evaluation strategy. Concretely, they propse a new assessment method denoted as episodic evaluation. This episodic evaluation is motivated by the idea of avoiding to overestimate the models' specificity. Thus, the model's evaluation is performed episode-wisely, rather than the window-wisely, which is the implemented technique by other authors. In their study, they showed performance results for the GM between sensitivity and specificity of 76.8% (i.e. 74.7% and 79% for sensitivity and specificity, respectively), which was determined by using leave-one-patient-out with an episode-based evaluation strategy, instead of the window-based strategy. Furthermore, when applying personalisation techniques to the model, they were able to achieve 84% for the same metric.

As mentioned in Section 1.3, from the previous work described, approaches presented by Bächlin et al. in 2009 [16], Mazilu et al. in 2012 [79], Tripoliti et al. in 2013 [141] and Rodríguez et al. in 2016 [31], are reproduced to train powerful biclassification shallow ML algorithms to compare the state-of-the-art approaches to ours. The detailed discussion on the reproduction and training procedures are exposed in Section 5.4.

Chapter 2

Deep learning

2.1 Convolutional neural networks

ConvNets [73] are a type of feed-forward deep neural network (DNN), which typically combines convolutional layers with traditional dense layers to reduce the number of weights composing the model. Convolutional layers enforce local connectivity between neurones of adjacent layers to exploit spatially local correlation. Concretely, convolutional layers are formed by kernels that share weights and, thus, permit to learn position invariant features from the input data.

Therefore, convolutional layers can extract features from data that have underlying spatial or temporal patterns, such as images and signal data. Furthermore, stacking these layers permits to extract progressively more abstract patterns.

While traditional DL models are composed of stacked dense layers, which lead to an overwhelming number of weights, ConvNets implement a powerful and efficient alternative if the target data present underlying spatial patterns.

2.1.1 ConvNets for image recognition

Image data problems are the main application field of ConvNets, due to the perfect matching between images characteristics and ConvNets properties. Although, ConvNets are the most powerful and efficient location invariant feature extractors, the key strategies when training DL models are to employ sample normalisation and augmentation techniques. ConvNets can usually are trained on image data; however, this may be one-dimensional (1D), two-dimensional (2D) or three-dimensional (3D) image data. Commonly, patterns in images, such as target elements to recognise, are invariant to rotation, scale, orientation, size, position, resolution and illumination.

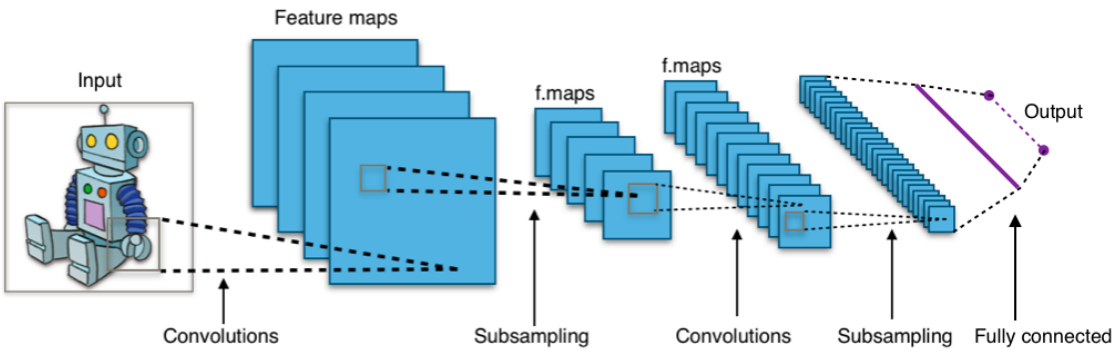


Figure 2.1: Typical ConvNet architecture for 2D image recognition [29]. This figure illustrates a typical seven-layer ConvNet architecture. Concretely, the architecture is composed by: I) an input layer, II) a first convolutional layer, III) a first pooling (or subsampling) layer, IV) a second convolutional layer, V) a second pooling layer, VI) a dense layer and VII) an output dense layer.

For achieving optimal performance, samples are first normalised to reduce the intensity and illumination variance between instances of similar meaning, and finally, the data are artificially replicating by applying random rotations, shifts, flips, distortions, rescales and crops. Therefore, in these circumstances, one may obtain a model capable of extracting the original data patterns in their maximal expressively form from each sample, such that if the same pattern appears in any different sample may be successfully identified [53].

ConvNets may include pooling layers between consecutive convolutional layers, as illustrated in Figure 2.1. Pooling layers down-sample the output of their previous layer by operating small neighbours areas (see Figure 2.2). The motivation for this strategy is that the relative approximate locations between patterns are more relevant than their exact positions [69, 97, 132]. This strategy may aid the model to generalise and to control overfitting by forcing it to learn more abstract representations with lower parameters as the deeper the information flows [132].

DL models are composed of large structures of learning units interconnected, which given a new sample should propagate the information of it through the network and, finally, perform a prediction on it. Activation functions control this information propagation in the units conforming this structure. The hyperbolic tangent function (Tanh), $f(x) = \tanh(x)$, the logistic sigmoid function (Sigmoid), $f(x) = (1 + e^{-x})^{-1}$ and the rectified linear unit (ReLU) $f(x) = max(x, 0)$, are some of the popular activation functions for ConvNets. ReLUs have shown, however, to be more time efficient than other alternatives. Since DL strategies have a computational bottleneck

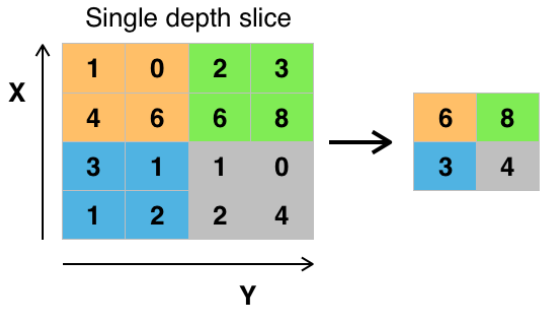


Figure 2.2: Max pooling illustration [29]. Max pooling operator always takes two arguments. Its first one, 2x2 in this case, determines the shape of the rectangle in the image to be operated by the pooling operator each time. Whereas the second parameter is named 'stride', which was set to 2 in this example, referrers to the number of values to skip before applying the pooling operator again. This figure, hence, illustrates an example case of applying '2x2 max pooling with stride 2'.

this last function has become the most widely exploited one when implementing ConvNets [69, 133].

2.1.2 ConvNets for sequence data

Whereas images maintain a set of invariance properties, which compose ConvNets' ideal conditions, other data formats, such as videos and signals, may also benefit from them. These other data types are denoted as time series data since including temporal dependencies between instances.

The most common technique to deal with classification tasks in time series data is to use a windowing strategy. Windowing consists of splitting the data into equally-sized consecutive parts to address the classification task window-wisely instead of instance-wisely.

When applying these techniques to signal data, the input is reshaped to 1D-image-like data samples, while in the case of video data, the input is reshaped to 3D-image-like samples. However, the temporal information restricts the types of invariance that should be considered. Their essential difference is that the augmentation strategies allowed on time series data may be drastically constrained (e.g. the flip operator may be incoherent since time flows only in one direction). Thus, all augmentation techniques should be designed to be coherent with the domain's knowledge.

2.1.3 Application of ConvNets to biomedical data

Recently, a trend in biomedical research has arisen towards adopting DL techniques for tackling some biomedical analysis problems [4, 104]. Next, some of the most recent work related to DL strategies for biomedical data analysis are commented.

ConvNets provide very powerful representations on image-data problems [55]. Other recent applications of ConvNets in biomedical research are to diagnose mild cognitive impairment from resting state functional MRI (fMRI) data [130], and to segment 3D biomedical image, such as MRI, fMRI and computed tomography [24]. ConvNets received exceptional acceptance for dealing with biomedical image data. The reasons for it were that ConvNets have a high capacity of exploiting image data, while biomedical data samples belonging to the same task (e.g. magnetic resonance imaging (MRI) brain data for stroke detection in young people) have low variance. Thus, ConvNets could deal with the scarcity of data, which is characteristic from biomedical datasets.

2.2 Recurrent neural networks

As ConvNets, recurrent neural networks (RNNs) are an extension of feed-forward DNNs, which can learn relations between data samples structured in sequences by having a recurrent hidden state which can maintain information from the previous samples.

Given a temporal input sequence $\boldsymbol{x} = (x_1, x_2, ..., x_r)$, the RNN updates its recurrent hidden state h_t by

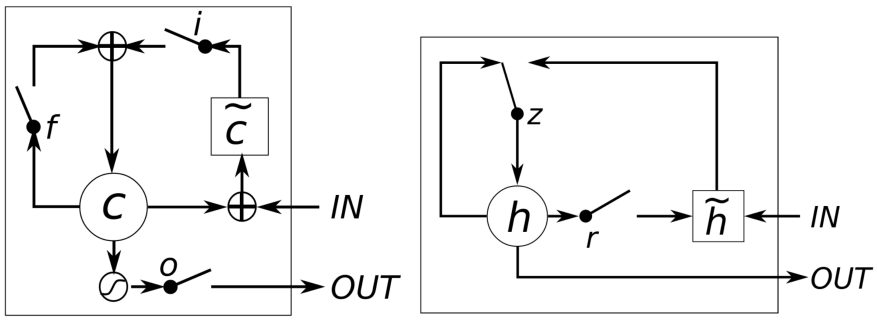
$$\boldsymbol{h}_t = g(\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{U}\boldsymbol{h}_{t-1}) \quad , \tag{2.1}$$

where g is an activation function such as the Sigmoid or the Tanh, $\boldsymbol{W}$ is the input-hidden matrix and $\boldsymbol{U}$ represents the hidden-hidden weight matrix.

Although having Turing capabilities [127], according to Bengio et al. (1994) [19], it is hard to train long-term dependencies by these architectures.

LSTMs [60]. LSTMs extend RNNs by implementing memory blocks instead of recurrent units. These blocks implement additional gating mechanisms by which the network can learn when to store, retrieve and flush previous information. These functionalities allow the network to learn long-term relationships.

GRUs [25, 27]. GRUs simplify LSTMs by setting a single gating unit that simultaneously controls the flush and store actions, which were controlled by two independent gates in the former. Although being simpler than LSTMs (see Figure 2.3), this approach has reached comparable results in several tasks [27]. GRUs tend, moreover, to be more computationally efficient and to converge in fewer epochs than LSTMs.



(a) LSTM block. (b) GRU block.

Figure 2.3: Figures 2.3a and 2.3b illustrate an LSTM and a GRU blocks, respectively. In Figure 2.3a i, f and o are the input, forget and output gates, respectively. c and $\tilde{c}$ denote the memory cell and the new memory cell content. While in Figure 2.3b r and z are the reset and update gates, and h and $\tilde{h}$ are the activation and the candidate activation. These figures were extracted from [27].

2.2.1 Application of RNNs to biomedical data

The latest advances in DL strategies specially designed to tackle time series data problems, such as Memory Networks [146] and Differential Neural Computers [54], have recently become the state-of-the-art on several complex problems, such as to answer questions about a text or dialogue, to find the shortest path and to infer missing links in graphs [54]. However, these architectures intend to address reasoning-like problems, while most of time series biomedical data problems can be defined as classification or regression problems with relevant underlying patterns across sequences of data samples. These problems can, thus, be efficiently dealt by simpler recurrent DL strategies, such as RNNs, LSTMs and GRUs. However, whereas ConvNets could handle scarcity of data due to the low variance between data samples, in time series data the inter-samples differences increase, even when all collection conditions are equally setup.

Biomedical signals' extreme complexity, its susceptibility to noise and the intersubject physical differences, which may introduce high variance in the data, are the major issues preventing these problems from being solved. Furthermore, the data available

for addressing these problems is limited, while designing appropriate data augmentation strategies is usually challenging due to the data's correctness fragility. Some recent work on recurrent DL models in biomedical research are electrocardiography signal classification [102] and to learn representations from electroencephalography data [18].

2.2.2 Application of RNNs to human activity recognition using wearable sensors

The increasing interest in human activity contextualisation to produce adaptable support systems and the growing computational capacity of wearable devices have recently encouraged several research studies towards designing human activity recognition (HAR) systems using wearable sensors. The technologies and strategies devoted to solving these problems commonly match to the ones employed within biomedical research field, such as posture contextualisation [112] and FOG detection [58, 103, 110].

Some recent studies on DL models for HAR are: Ordóñez et al. [91] who present a ConvLSTM for addressing generic time series problems, which was able to outperform the state-of-the-art shallow ML algorithms in several benchmark datasets, while demonstrating that introducing LSTM layers in the model improved its representation power; and Ravi et al. [103] who presented a novel approach for real-time HAR in low-power devices. Their approach used sums of convolutions of spectral data, which was previously windowed in the temporal domain. Their approach outperformed other handcrafted feature extraction methods.

Despite the fact that considerable research is being undertaken to approach the biomedical data analysis problems with DL strategies, shallow ML algorithms still compose the state-of-the-art in PD's motor symptoms detection [94, 110]. Nevertheless, problems in which DL techniques have become the state-of-the-art are rapidly increasing, as can be observed in the cardiac disease research field [13, 67, 83].

2.3 Regularization for DL models

Regularisation techniques are those that modify a learning algorithm to reduce its generalisation error while preserving training accuracy [53] (see Figure 2.4). Regularisation techniques, such as the L2-norm regularisation and L1-norm regularisation, are frequently applied to in ML algorithms, e.g., SVMs, to control overfitting. This

section is devoted to providing a general overview on some of the most common regularisation techniques for DL. Concretely, the approaches described will be: I) parameter norm penalties, such as weight decay; II) early stopping; III) data augmentation; IV) dropout; and V) zoneout.

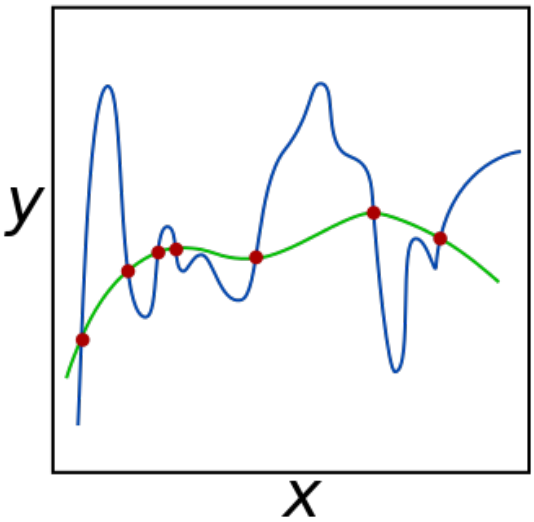


Figure 2.4: Regularization example [29]. This figure contains three main elements: I) the red dots representing the train data of a regression problem; II) a blue line representing a solution without regularization for the regression problem; and III) a green line describing the same solution, but with L2 weight regularization.

2.3.1 Parameter norm penalties

The regularisation techniques that limit the representation power of learning models are hereafter described. These approaches are directly added as a penalisation cost $\Omega(\boldsymbol{\theta})$ to the models' objective function $J(\boldsymbol{Y}_t, \boldsymbol{Y}_p)$ such that the function to be optimised is redefined as

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{Y}_t, \boldsymbol{Y}_p) = J(\boldsymbol{Y}_t, \boldsymbol{Y}_p) + \lambda\Omega(\boldsymbol{\theta}) \quad , \tag{2.2}$$

where λ is an hyperparameter that balances between the model's capacity and its training error, $\boldsymbol{\theta}$ represents the model's parameters, which would correspond to DL's weights and/or activations, and $\boldsymbol{Y}_t$ are the ground truth labels of the data samples, while $\boldsymbol{Y}_p$ represent the model's predictions.

L2 parameter regularisation. This technique penalises the parameters' norm, with emphasis on abnormally high values. Concretely, this technique is defined as

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|_2^2 = \frac{1}{2}\sum_i \theta_i^2 \quad . \tag{2.3}$$

Thus, it will positively reward the model's objective function when the representation responsibility is distributed among all patterns, compared to when it is concentrated on a set of them. Among all traditional ML regularisation techniques, the L2-norm regularisation is the most widely implemented in DL. Concretely, it is usually applied to the models' weights and referred as weight decay.

L1 parameter regularisation. Similarly to the L2-norm, this technique penalises the parameters' norm; however, it lacks any special attention, by penalising only the sum of values. Concretely, this technique is defined as

$$\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1 = \sum_i |\theta_i| \quad , \tag{2.4}$$

which will positively reward the model's objective function when the representation responsibility is sparse among all patterns, thus, only a set of parameters are non-zero.

2.3.2 Early stopping

DL models are trained iteratively aiming to approach relevant local optima in the solutions space, which can successfully represent the target problem. These iterations are denoted as epochs. Correctly establishing the number of epochs is important to prevent the model from overfitting to the training data, while avoiding useless computation by detecting if the model has converged.

The number of epochs by which a model may reach convergence will usually be correlated to several other characteristics, such as the model's architecture, the data, the optimisation method and its internal parameters, such as the learning rate, and the regularisation strategies being employed.

Due to its importance and complexity, DL models are usually trained using an early stopping strategy. Early stopping is the term for referring that the train will stop when some criteria, which are related to the algorithm's training performance, is accomplished, instead of running indefinitely or fixing the number of epochs.

2.3.3 Data augmentation

The best strategy for enhancing ML models' generalisation is to increase the training data available [53]. Data augmentation approaches are those that artificially increment the training data available, which, as mentioned in Section 2.1, are a common practice in DL. Furthermore, in some problems, it may be possible to artificially generate new coherent data, such as generating in-car conversation data by adding in-car noise over conversation data.

2.3.4 Dropout

Dropout is an efficient and effective regularisation strategy [129, 133], which provides an approximation to implementing the bagging ensemble method over numerous DL models while preserving inexpensive computational costs. The main intuition behind dropout is to employ only a random subset of the network each time a new instance is fed to the model. This approach takes only one parameter, which corresponds to the probability by which a neurone is dropped.

Implementing dropout leads the model to learn more robust features since several parts of the model must be able to perform the target task independently successfully. In evaluation time, dropout is deactivated producing, thus, this bagging-like result.

2.3.5 Zoneout

Zoneout [70] is a regularization strategy especially designed for recurrent architectures. Its main intuition is, similarly to dropout, which randomly drops parts of the network, to randomly preserve the values of the network. Concretely, zoneout introduces noise to the training by randomly replacing some activations of the network's units with their previous timestep activation values.

2.4 Training DL models

Training shallow ML algorithms are typically performed employing strategies, such as cross-validation and leave-one-out, which intend to perform an exhaustive evaluation of each hyperparameters' configuration before training the final model. These strategies imply that the number of models to train is equal to the total number of possible configurations, times the number of folds for the cross-validation. This procedure is, moreover, commonly repeated several times to ensure robustness of the configuration selected. DL models, however, are defined by overwhelming amounts of

hyperparameters, while being non-deterministic due to randomness in several operations, such as weights initialization and dropout. Consequently, these models may require being retrained several additional times compared to traditional ML algorithms to ensure robustness. Running blind hyperparameters exhaustive exploration with cross-validation would imply an overhead of factor 10, plus having to explore the entire hyperparameters space, which is usually avoided.

DL models are, thus, trained and evaluated on different data to avoid having to repeat the training process. Concretely, the available data are split into training data and testing data. Most data are usually assigned to the training partition (e.g. 70%), while a still representative part (e.g. 30%) is kept for testing purposes. This split is performed randomly on data that follows the same distribution.

The main intuition, hence is that, from all models trained only one is selected and tested on the testing data. Therefore, the final performance will be the one reported on the testing data. However, as mentioned in Section 2.3, the DL model, may overfit on the training data, and, thus, fail in the testing data. Therefore, to control the overfitting, the training data are further split into the training-train data (e.g. 70%) and training-validation data (e.g. 30%), which will sever for training the model and for evaluating its generalisation capacity, respectively. The main drawback of this strategy is that the finally amount of data employed for training the model is considerably reduced (e.g. $0.7 * 0.7 * 100$ =49%).

Optimization algorithms for DL DL models implement stochastic gradient-based algorithms to optimise the error loss after each batch is processed. These stochastic training methods perform the optimisation in minibatches, which are partitions of the entire training-train data. Thus, models compute the gradients and carry out the weight corrections per each minibatch, rather than performing a single update per epoch. The optimal magnitude of these corrections is difficult to match, however, if these corrections are small enough, the model will eventually converge. This magnitude is referred as the learning rate and is commonly set to low values.

Recent optimisation algorithms, such as adaptive gradient algorithm (AdaGrad) [36], implement adaptive learning rate techniques. Algorithms with adaptive learning rates permitting to set the learning rate to high values to accelerate the training procedure. Other popular stochastic gradient-based algorithms adopted in DL are: stochastic gradient descent (SGD) [21], which, although having a fixed learning rate, is the most widely used optimisation algorithm in DL models [53]; root mean square propagation (RMSProp) [137], which extends the AdaGrad algorithm;

AdaDelta [150], which is another extension of AdaGrad; and adaptive momentum (Adam) [66].

Weight initialization. Setting proper initial weights is a must for successfully training DL models. However, according to Goodfellow et al. [53], several initializations will usually allow a DL model to train in a proper way [53]. Furthermore, there are conflicting perspectives about which initialization approaches may perform better in practice due to several factors being affected, such as regularisation and optimisation, which may encourage weights to be small and large, respectively.

2.5 Architecture notation

The notation employed throughout this master thesis for describing DL architectures, which extends the nomenclature presented by Pigou et al. [97], is hereafter defined:

- Convolutional layer $\rightarrow C(x_1, x_2, ..., x_d|k)$ were x_i refers to the size on the ith dimension of the kernels in the layer, and k denotes the number of kernels.
- Dense layer $\rightarrow F(n)$ were n is the number of neurones of the layer.
- LSTM layer $\rightarrow LSTM(n)$ were n is the number of neurones of the layer.
- GRU layer $\rightarrow GRU(n)$ were n is the number of neurones of the layer.

Chapter 3

Data collection and processing

3.1 Data collection

The data [110] employed were composed of inertial signals from 21 PD patients recorded by a single IMU placed at the left side of the patient's waist (see figure 3.1). This IMU generated 9 signals sampled at 200 Hz as output. The 9 signals represented the measurements of 3 tri-axial sensors: gyroscope, accelerometer and magnetometer.

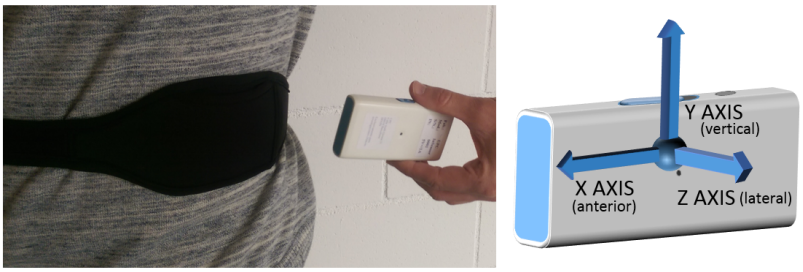


Figure 3.1: The data collector IMU and its location on patients' body (i.e. left side of their waist) [31].

As mentioned in Section 1.4, the data were acquired within the scope of REMPARK [124]. REMPARK's experimental protocol collected data from 92 PD patients (i.e. 36 women and 56 men), which were selected according to the following criteria:

1. To be diagnosed with PD according to the UK Brain Bank.
2. To have Hoehn & Yahr stage above 2 in OFF-state.
3. To not have dementia according to DSM-IV criteria.

4. To give their written informed consent for using the collected data in the research carried out within the REMPARK project, and other studies conducted by the same intuitions, which is the case of this study.

REMPARK's data aimed to be suitable for the creation of algorithms to analyse PD's motor characteristics in uncontrolled environments. Collection protocols, thus, were performed at patients' ADL locations (i.e. homes and surroundings) while performing several activities, such as showing their homes, brushing their teeth, carrying shopping bags and entering to their building (e.g. climbing stairs or using their elevator). All data collection trials were video-recorded. Medical experts designed this collection protocol, which, moreover, complied with the ethical approval. The protocol aimed to provide data on patients performing similar ADL to ensure reproducibility of results of learning algorithms which intended to analyse PD's motor symptoms data and information, such as FOG, tremor, gait parameters (e.g. speed and step length), bradykinesia of lower and upper limbs, and dyskinesia.

Data were labelled patient independently using the patient's motor characteristics (e.g. symptoms' manifestation frequency and walking patterns). These features were obtained from analysing every patient's performance of specific activities, such as walking through narrow spaces and performing several turns. When labelling the data these patients' motor characteristics information served to accurately determine each patient's symptoms and activities from the video recordings and clinician annotations, individually. This strategy aimed to capture any inter-patient variability by avoiding to compare them during the labelling tasks, even if differing from the standard symptoms manifestation patterns.

The collection trials were structured in two main parts: the first part of the collection was performed with the patients in OFF-state; before the second part, patients took their medications, and, thus, it was conducted again with patients in ON-state. Each of these protocol parts was, moreover, divided into two main subparts: I) to perform controlled and triggering activities, which lasted for few minutes (min); II) free activity monitoring of the patient so that natural symptoms and activities are recorded, which composes most of the data recorded per patient.

During the free activities monitoring parts of the collection protocol, the patients performed a set of activities more frequently recorded ADL [16], such carrying an object from one room to another, while other ADL were specially introduced to difficult the symptoms analysis tasks, such as teeth brushing, drawing, painting and erasing in a sheet of paper. The characteristics of this reach dataset intend to force

models trained on it to learn robust representations, to produce reliable and useful systems for PD patients daily support.

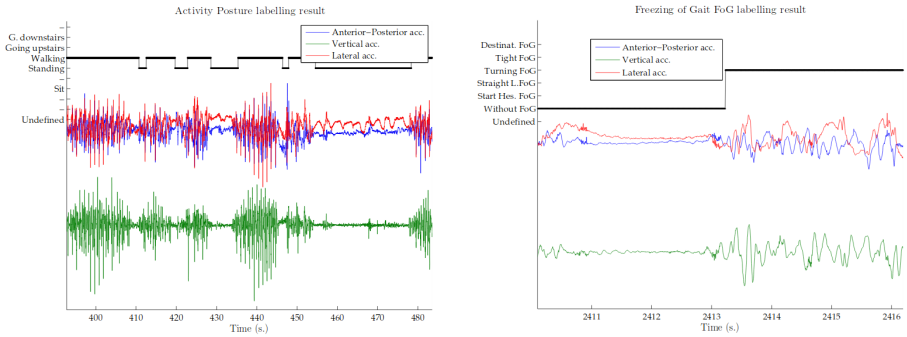


Figure 3.2: This figure is illustrates two slices of the data composing the REMPARK's database. On the left side, the figure illustrates a part of the accelerometer signals data with its associated activity and posture label, while on the right, a different part of accelerometer data is shown with its FOG labels associated [124].

The data collected included: I) signal data from an IMU placed at the left side of the waist; II) signal data recorded from a tri-axial accelerometer placed on the wrist for assessing tremor and bradykinesia; III) video recordings recorded from a mobile phone camera (i.e. Nexus S Google), which was chosen to reduce the intrusiveness perception of the patients; IV) the clinicians annotation made on the video after the data capturing (i.e. symptoms and activities); and V) the patients results from all the inclusion criteria and additional tests performed, such as the FOG-Q. From these data, only the signals recorded from the patients' waists were employed.

Concretely, this master thesis has contributed to an ongoing project named 'Improving Quality of Life with an Automatic Control System' (MASPARK), which is especially focused on the study of FOG. For the MASPARK project, 21 patients were selected from the REMPARK database [124]. This second selection protocol involved evaluating the relevance of patients for participating in FOG detection studies [110]. These inclusion criteria were:

1. To manifest at least 1 min of FOG labelled in their data recordings.

2. To have reached at least a score of 6 in the FOG-Q.

Within the scope of the MASPARK, the data from the overall 21 PD patients (i.e. 3 women and 18 men) accomplishing these prerequisites were relabelled by clinicians relying only on the video recording associated with the waists' signals data.

Most of the previous research targeting automatic FOG detection employed datasets acquired by using uniform in-lab setups [81, 16, 79, 78]. Thus, the data utilised in this study are significantly more complex and complete than most of other datasets considered in previous research approaching the same issue.

3.2 Data overview

The data were composed by, approximately, 40 min (i.e. 20 min in OFF-state and 20 min in ON-state) of 9-channel signal data sampled at 200 Hz per patient. The overall data instances available could, thus, be computed as 40 min $\times$ 60 s $\times$ 200 Hz $\times$ 21 patients $\approx 10^7$ instances.

As mentioned in Section 3.1, the data were collected by the medical institutions participating in the REMPARK project. Consequently, these data were collected by 4 different medical institutions. Furthermore, only 3 out of the 21 patients were women. A major rule in ML is to ensure that training and testing data represent the same distribution. Therefore, training and test patients were split randomly within the following constraints: I) both, the training and the testing patients sets, should contain at least one patient from each medical institution, as well as, at least one women; II) the relative difference between FOG percentages of both sets should be less than 50% (e.g. if the training set had a 15% of FOG instances, the test set should have a FOG percentage within the range [7.5%, 22.5%], which corresponds to $15 \pm \frac{15}{2}$); III) the number of patients assigned to the test set should be less than 7. Fortunately, this process concluded with a split such that 4 patients were assigned to the test set, while 17 were assigned to the training set.

Table 3.1: Training and test data properties. Columns naming description: '# patients' $\rightarrow$ number of patients in the set; '# Instances' $\rightarrow$ number of data instances recorded by the IMU; 'FOG %' $\rightarrow$ percentage of FOG instances (i.e. those with label equals 1) with respect to the '# Instances' value; 'Undefined %' $\rightarrow$ percentage of undefined instances (i.e. those with label equals 0) with respect to the '# Instances' value.

set of data	# patients	# Instances	FOG %	Unlabelled %
Train	17	10755200	11.04	27.23
Test	4	2666600	8.14	37.67

As can be observed from table 3.1, the percentage of FOG instances was similar for training and testing. However, both percentages were small, which could lead to class

imbalance issues. The percentage of undefined data were, moreover, large enough to produce patient imbalance problems. Furthermore, processing these data in train time would imply significant useless computation while solving the bi-classification task.

Table 3.2: Training data properties per patient. Columns naming description: 'Patient ID' → patient reference employed throughout this document; '# Instances' → number of data instances recorded by the IMU; 'FOG %' → percentage of FOG instances with respect to the '# Instances' value; 'Undefined %' → percentage of undefined instances with respect to the '# Instances' value; and 'Relevance %' → proportion of defined (i.e. not undefined) instances of this patient with respect to the overall defined instances in the patients group (i.e. train or test) being analysed.

Patient ID	# Instances	FOG %	Undefined %	Relevance %
0	827400	8.05	38.69	6.48
1	414600	2.66	34.24	3.48
2	696200	7.7	24.84	6.69
3	182800	10.72	27.88	1.68
4	550400	15.04	49.83	3.53
5	472600	15.65	18.56	4.92
6	479800	14.76	18.75	4.98
7	570000	5.1	19.11	5.89
8	877800	5.38	25.14	8.4
9	897000	12.85	24.1	8.7
10	638600	9.53	43.88	4.58
11	489600	26.99	16.85	5.2
12	700400	5.74	28.98	6.36
13	328400	8.38	25.46	3.13
14	790600	4.42	26.43	7.43
15	1080800	26.68	17.91	11.34
16	758200	4.38	25.4	7.23

Table 3.2 and Table 3.3 illustrate the per patient characteristics for the train and test data, respectively. From them, it can be observed that patients have different influence in their sets. These differences may produce patient patterns imbalance.

On the one hand, from Table 3.2 it can be observed that there was a significant patient imbalance problem on the original data, which was worsened by the undefined instances. Training the model with the original data would lead to overspecialising the model on a subset of patients while ignoring the rest. On the other hand, from

Table 3.3: Testing data properties per patient. Columns naming description: 'Patient ID' → patient reference employed throughout this document; '# Instances' → number of data instances recorded by the IMU; 'FOG %' → percentage of FOG instances with respect to the '# Instances' value; 'Undefined %' → percentage of undefined instances with respect to the '# Instances' value; and 'Relevance %' → proportion of defined instances of this patient with respect to the overall defined instances in the patients group being analysed.

Patient ID	# Instances	FOG %	Undefined %	Relevance %
17	890800	5.58	49.51	27.06
18	606800	5.98	25.71	27.12
19	457400	10.38	23.78	20.98
20	711600	11.73	41.96	24.85

Table 3.3 it can be observed, however, that this problem was less severe in the testing data.

3.3 Offline data cleansing and signal processing

This section describes and discusses, step by step, the overall data processing process performed before training the DL models.

3.3.1 Data cleansing

Cleaning the data was the first action to take to solve the missing values and unlabelled data issues.

Missing values. If the missing values segment was shorter than 0.1 s in a row, these were linearly interpolated. Otherwise, the file was split leaving out the massaging values to maintain temporal coherence in the data. These missing values corresponded to microSD errors during the data acquisition process.

Unlabelled data. The presence of unlabelled data by itself did not suppose a major issue since it was still possible to train the algorithms by skipping the unlabeled instances in training time. It was noted, however, that some patients were nearly ignored due to having several unlabelled data in their files. Furthermore, this supposed a continuous waste of computational power. The implemented solution consisted to splitting the patients' files leaving the unlabelled data out if these last more than 5 s.

3.3.2 Signal processing

The data were recorded at 200 Hz, which is a very high frequency to mapping human movements. In addition, Rodríguez et al. [110] performed FOG detection successfully on the same data down-sampled at 40 Hz. According to the NyquistShannon sampling theorem, thus, the data employed by Rodríguez et al. [110] could only represent appropriately frequencies up to 20 Hz, confirming the excess of resolution in the original signal, which may be representing frequencies up to 100 Hz according to the same theorem.

To remove irrelevant noise from the data, such as sensor's generated one, the data were filtered using a lowpass filter. This filter was implemented by a Butterworth filter, with cut-off frequency set to 20 Hz and 8th order. The Gustaffson's method [57] was implemented to handle the filter's initial conditions.

Following the intuition of the sampling theorem, the data were down-sampled to 50 Hz to avoid unnecessary computation due to reducing the number of samples in the data by a factor of 4.

3.3.3 Patient data balancing

At this stage of the offline data preprocessing, the original imbalance problem between patients prevailed. This problem would give more relevance to some patients compared to others, which, with only 17 patients for the training process and 4 for the testing, could lead to misleading results. Note that most authors in the FOG detection literature present personalised methods, which may outperform their generic approaches by more than 10% (e.g. the improvement achieved by Mazilu et al. in 2012 [79] by using a personalised strategy was from 79.4% to 99.83% for the GM between sensitivity and specificity, which essentially is overfitting); thus, it can be assumed that patients may present the same patterns repeated in their data.

Furthermore, towards training recurrent DL models, it is useful to have equally sized files to control when to flush the model's memories. The strategy to correct the patient imbalance should, thus, produce equally sized files for implementation convenience.

Recurrent DL models allow representing temporal patterns across samples. However, even the gated techniques, such as LSTMs and GRUs, are unlikely to learn long-term dependencies (e.g. from more than 50 of samples before) in practice. Additionally, even by doing it, due to the lack of data, undesired patterns could be introduced, which were arbitrarily produced. Therefore, from the intuition that events

taking place more than 2 minutes previous to the current state may have an insignificant influence towards triggering FOG events, data were, thus, split into files of 2.2 minutes each with overlapping of 50% to capture all temporal patterns in the original data.

Concretely, the file size was adjusted to reduce the amount of data discarded. As described in Chapter 5, the batch size for training the models was set to 16, while, as described in Section 3.4, data were split into windows of 2.56 s; thus, setting the file size to 2.2 minutes would allow to generate 3.22 batches per file, which will permit to ensure training at least 3 batches per file in the recurrent DL models, which require that batches are fully generated by the same file. Finally, note that due to the augmentation strategies defined in Section 3.5, the 0.22 remaining part of the file will allow applying window shifting strategies while still ensuring that the number of batches will be of 3.

Once all patients data were represented by files of 2.2 minutes, to ensure similar relevance between them it was enough to replicate files from patients with lower representation randomly. In Figure 3.3 the file shape is shown together with the window's one; moreover, Algorithm 1 provides an overall overview of the data preprocessing procedure.

3.4 Data representation

This section describes the techniques implemented to transform the preprocessed data to a proper representation for feeding the DL models.

3.4.1 Windowing

The most common technique to deal with classification tasks in time-series data is to use a windowing strategy. To window the data implies to split it into equally-sized consecutive parts to address the classification task window-wisely instead of instance-wisely.

According to Moore et al. [82] window sizes for FOG detection should be at least of 2.5 s, while Goodfellow et al. [53] mentions that in practice, GPUs are more efficient when input sizes are powers of two. The windows' length were, therefore, set to 2.56 s, since this value produced inputs of size 2^7 (i.e. 2.56 s $\times$50 Hz = 2^7 instances). Figure 3.3 illustrates how windows are arranged in each data file.

The labels of the original data were, however, per instance; thus, each windowed was relabelled according to the following criteria:

Algorithm 1 Data preprocessing

```
procedure PREPROCESS(raw_data)
    cut ← 20                                                    ▷ filter's cut-off
    order ← 8                                                   ▷ filter's order
    type ← Butterworth                                          ▷ filter's type
    freq ← 50                                                   ▷ subsampling frequency
    size ← 132                                                  ▷ files' sizes
    data ← [ ]                                                  ▷ processed data
    for <patient in raw_data> do
        patient_data ← [ ]
        for <raw_file in patient> do
            clean_list = CLEAN(raw_file)                        ▷ data cleansing
            for <clean_file in clean_list> do
                file_data = FILTER(clean_file,cut,order,type)   ▷ data filtering
                file_data = SAMPLE(file_data,freq)              ▷ data subsampling
                file_list = SPLIT(file_data,size)               ▷ rearrange files' sizes
                for <file in file_list> do
                    APPEND(patient_data,file)
                end for
            end for
            APPEND(data,patient_data)
        end for
    end for
    patient_list ← data.patients                                ▷ list of files grouped by patients
    file_count ← MAX_SIZE(patient_list))                        ▷ maximum number of files
    for <patient in data> do                                    ▷ set all patients to file_count
        while SIZE(patient) < file_count do
            file ← RANDOM_SELECT(patient)                       ▷ select a random patien's file
            file_copy ←COPY(file)
            APPEND(patient,file_copy)
        end while
    end for
    return data
end procedure
```

1. If the window contained FOG symptom instances:
   (a) If the FOG instances composed at least the 50% of the in-window data, then this window was labelled as a FOG window (i.e. label equals 1).
   (b) Otherwise, the window was labelled as an undefined window (i.e. label equals 0).
2. If the window contained non-FOG symptom instances, but, it contained unlabeled instances, the window was set to be unlabelled.
3. Otherwise, the window was set to be a non-FOG window (i.e. label equals -1).

Finally, the data were normalised by diving by the precomputed sample standard deviation from the overall training dataset to enhance DL models' learning quality.

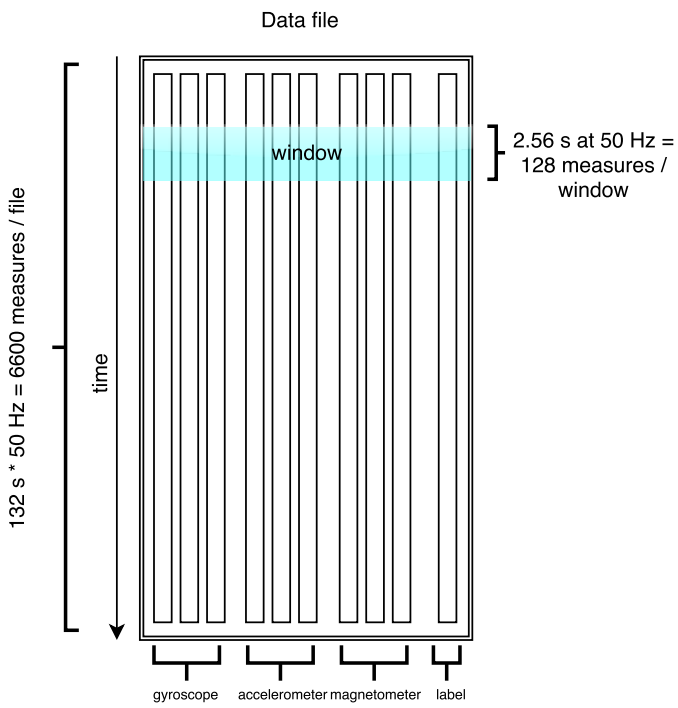


Figure 3.3: Preprocessed file being windowed.

3.4.2 Spectral window stacking

DL models are powerful feature extractors, however, if being provided with insufficient information, these models, as any ML algorithm, may fail to solve the task. Part of the features computed in the approach proposed by Rodríguez et al. in 2017 [110] involve operating with the previous window. This window transitional information usage

suggests interwindow information may be necessary succeed in the FOG detection task on the target data.

Following this intuition, the presented approach is trained with a data representation that allows the DL algorithms to extract interwindow dependencies. This representation strategy is from now on referred as spectral window stacking.

The spectral window stacking $SWS(W_t, W_{t-1})$ is a function with two arguments: W_t and W_{t-1}, which refer to the window to be analysed at time t and its previous one, respectively. Concretely, the process is composed of the following steps: I) the fast Fourier transform (FFT) is computed for both windows, resulting in two new windows of the same size; II) these windows are rescaled by dividing by the window's size; III) taking advantage of the symmetry properties of the Fourier transform (FT), only the first half of each window is kept; and VI) finally, both windows are joined, one alongside the other, forming a unique representation of half the window original length and twice its width. The overall process is illustrated in Figure 3.4.

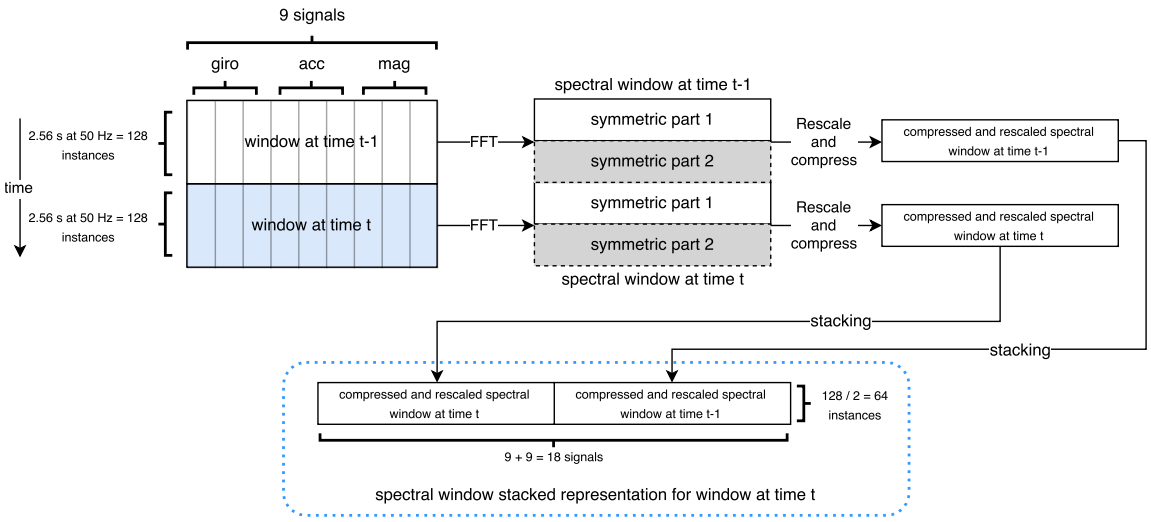


Figure 3.4: Spectral window stacking process diagram.

3.5 Data augmentation

After preprocessing the data and implementing the windowing strategy, the data available were characterised as shown in Table 3.4. From it, it can be observed that after preprocessing the data, the patient imbalance issue was solved.

As already mentioned in Section 3.2, from these patients, 4 of them were set aside for testing purposes, leaving only 17 for training the learning algorithms. Furthermore, as discussed in Section 2.4, the training data were split into a training-train

dataset and a training-validation dataset, which were composed by 13 and 4 patients, respectively. This partition was performed following the same guidelines than for splitting the data between training and testing as described in Section 3.2. These data are windowed, as already described, and grouped in batches of 16 windows per batch. Further details on the data partitioning and minibatch strategies, which were introduced in Section 2.4, the data partitioning procedure, and the implementations of the feeding strategies for the DL models are further discussed in Subsection 5.2.8.

Table 3.5 presents the exact number of samples available for each of the mentioned partitions. This table presents two different rows, one for the data available for training the 1D-ConvNet, and the other for both recurrent models (i.e. 1D-ConvLSTM and 1D-ConvGRU). This differentiation is caused by the restriction of recurrent DL models, which for training require that the time order is preserved in the data within each batch; thus, each batch can contain only data from one patient at a time. However, the values for the Training-validation and the Testing datasets match between both feeding strategies due to the evaluation strategy, which is performed per patient, see Section 5.3 for the full description.

From Table 3.5 it can be observed that at this stage, the problem may be infeasible to be addressed by conventional DL strategies due to lack of Training-train data. The most popular strategies to overcome data scarcity in DL are the data augmentation techniques. Data augmentation techniques permit to increase the knowledge extracted from data by performing replications on the data that are consistent with the task's domain. Data augmentation strategies implemented were:

- To shift the first window starting instance by sampling random values from a uniform distribution defined by the range [0, *window size*].
- To rotate each windowed signal by simulating a rotation on the waist-sensor through a rotation matrix generated by sampling angles (see Figure 3.1 for the axis reference) over a normal distribution defined as:
  - **X-axis**
    - Range → [-30, +30].
    - Sampled standard deviation (ST) → 10.
    - Mean → 0.
  - **Y-axis**
    - Range → [-40, +40].

    - ST → 15.
    - Mean → 0.
  - **Z-axis**
    - Range → [-10, +10].
    - ST → 2.5.
    - Mean → 0.

  These rotation distributions were designed to resemble naturally introduced rotations due to the patient's waist form and movements.

The parameters of these strategies are stochastic and may vary between epochs. Concretely, each file is shifted differently at each epoch, while it may be randomly rotated with a probability of 0.5. This approach introduced stochastic noise in the training process, which actuated as having more data to train the model while regularising it, and considerably reducing the model's overfitting. This strategy was repeated 4 times per epoch, proving the DL models with about 55000 samples for training.

Algorithm 2, presented in Subsection 5.2.7, provides an overall overview of the data representation procedure, which as already mentioned is composed of steps: I) data augmentation; II) windowing; and II) spectral window stacking.

Table 3.4: Data properties per dataset and patient in it. Columns naming description: '# Instances' → number of data instances recorded by the IMU; 'FOG %' → percentage of FOG instances with respect to the '# Instances' value; 'Undefined %' → percentage of undefined instances with respect to the '# Instances' value; and 'Relevance %' → proportion of defined instances of this patient with respect to the overall defined instances in the patients group being analysed.

Dataset	Patient ID	# Instances	FOG %	Undefined %	Relevance %
Training	1	171600	32.85	0.45	7.71
	2	171600	5.34	0.2	7.73
	3	171600	18.48	0.35	7.72
	4	171600	24.2	1.19	7.66
	5	171600	21.23	0.18	7.73
	6	171600	5.61	0.5	7.71
	7	171600	15.41	0.85	7.68
	8	171600	8.73	0.56	7.7
	9	171600	8.28	0.69	7.69
	10	171600	15.13	0	7.75
	11	171600	10.98	2.25	7.57
	12	171600	6.78	0.54	7.71
	13	171600	38.51	1.6	7.62
	Total-train	2230800	16.27	0.72	100
Validation	14	204600	26.83	0.61	24.93
	15	204600	32.92	0.78	24.89
	16	204600	2.69	0	25.09
	17	204600	9.7	0	25.09
	Total-val	818400	18.04	0.35	100
Testing	18	105600	5.16	0.33	25.07
	19	105600	26.13	0.74	24.97
	20	105600	9.92	0.53	25.02
	21	105600	10.96	0.84	24.94
	Total-test	422400	13.04	0.61	100
	Total	3471600	15.63	0.57	100

Table 3.5: Data properties per patient. Columns naming description: 'Train' → Training-train dataset; 'Validation' → Training-validation dataset; 'Test' → Testing dataset; 'strategy' → implementation of the feeding strategy for DL models, which can be feed-forward or recurrent; '# samples' → number samples (i.e. spectral windows stacked) available in the dataset; 'FOG %' → percentage of FOG samples with respect to the '# samples' value.

	Train		Validation		Test	
strategy	# samples	FOG %	# samples	FOG %	# samples	FOG %
feed-forward	15568	16.12	5040	17.38	2656	12.65
recurrent	13648	14.23	5040	17.38	2656	12.65

Chapter 4

Architecture and training parameters

This chapter describes and discusses the DL architectures presented in this thesis.

4.1 1D-ConvNet

This approach constitutes the major contribution of this master thesis since other presented approaches (i.e. the 1D-ConvLSTM and the 1D-ConvGRU) are its extensions. The 1D-ConvNet is composed of eight layers: I) an input layer; II) four convolutional layers; III) two dense layers; and IV) an output layer. According to the notation defined in Section 2.5, this approach is described as

$$C(3|16) - C(3|16) - C(3|16) - C(3|16) - F(32) - F(32) - F(1) \quad . \tag{4.1}$$

Furthermore, following a similar notation than LeCun et al. [73] for describing the LeNet-5, Figure 4.1 illustrates the architecture of our 1D-ConvNet.

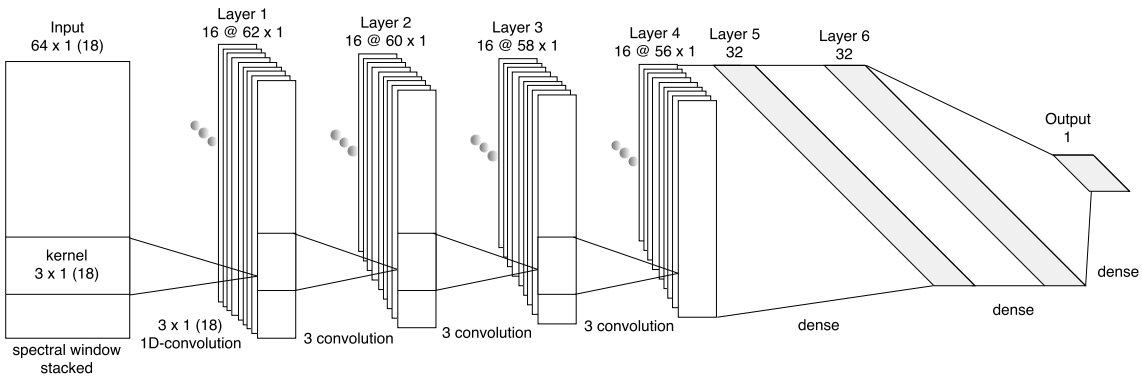


Figure 4.1: Diagram of the 1D-ConvNet's architecture.

4.1.1 Convolutional layers

This subsection described the properties of the convolutional layers implemented in the 1D-ConvNet model.

Kernel size. The kernel sizes were set to 3 following the intuition that larger patterns in the input would be handled by the network's depth, rather than the kernels width. This intuition is illustrated in Figure 4.2.

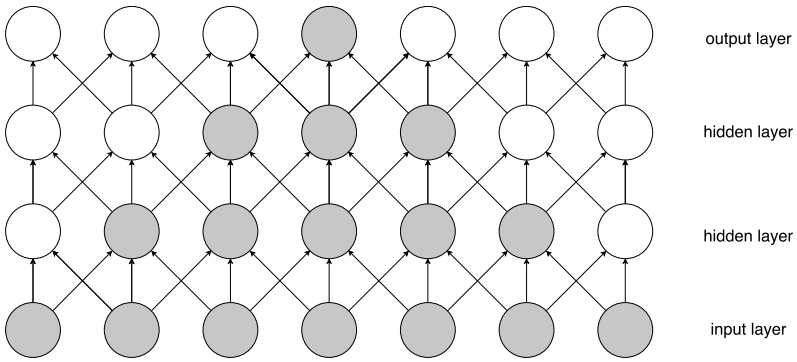


Figure 4.2: Diagram of the receptive fields structure of a 4-layer ConvNet with kernels of size 3. In this figure, all circles represent cells of a network with sparse connectivity, which is only connected to three output cells of the next layer. Therefore, as can be observed, the input layer being larger than 3, this can still be 'seen' by one cell, although not in the first layer. Concretely, it illustrates the effect of depth in ConvNets' connectivity structures, which permits to extract large and abstract patterns.

Number of convolutional layers and kernels. The number of convolutional layers and the number of kernels per layer were both set to minimum numbers, 4 and 16, respectively, which allowed the models to train properly (i.e. reaching train performances above 90%) with and without regularisation.

Activations. As mentioned in Section 2.4, ReLUs are the most widely exploited activation function for DL models. Moreover, following the recommendations in Goodfellow et al. (2016) [53], the activation functions of the convolutional layers were all implemented by ReLUs.

4.1.2 Hidden dense layers

This subsection described the properties of the hidden dense layers implemented in the 1D-ConvNet model.

Number of dense layers and neurones per layer. The number of hidden dense layers was fixed to be 2, similarly to the architecture presented by Ordóñez et al. in 2016 [91]. The number of neurones per dense layer was always set to be two times the number of kernels in the convolutional layers; thus, the number of neurones per layer in the 1D-ConvNet was configured to be 32. However, the number of dense layers was fixed to be 2.

Activations. The activation functions were set to be ReLUs, as in the convolutional layers.

4.1.3 Output layer

This subsection described the properties of the output dense layer implemented in the 1D-ConvNet model.

Number of neurones. This thesis defines the FOG events detection problem as a binary classification task, such that FOG instances are labelled as positive values (i.e. 1), whereas non-FOG instances are labelled as negative values (i.e. -1). This binary output can be handled by a single neurone connected to all the neurones in the previous layer. Therefore, the number of neurones in the output layer was one.

Activations. The activation function of this neurone was implemented by a linear function with weight decay.

4.2 1D-ConvLSTM

This approach implements an 8-layer 1D-ConvLSTM composed by: I) an input layer; II) four convolutional layers; III) two LSTM layers; and IV) an output layer. According to the notation defined in Section 2.5, this approach is described as

$$C(3|16) - C(3|16) - C(3|16) - C(3|16) - LSTM(32) - LSTM(32) - F(1) \quad . \quad (4.2)$$

Furthermore, following a similar notation than LeCun et al. [73] for describing the LeNet-5, Figure 4.1 illustrates the architecture of our 1D-ConvLSTM.

As mentioned in Section 1.3, this model was an extension of the 1D-ConvNet; thus, most of its predecessor's properties were maintained. Concretely, only properties concerning the model's new layers were modified. Hereafter these differing properties concerning the LSTM layers are commented.

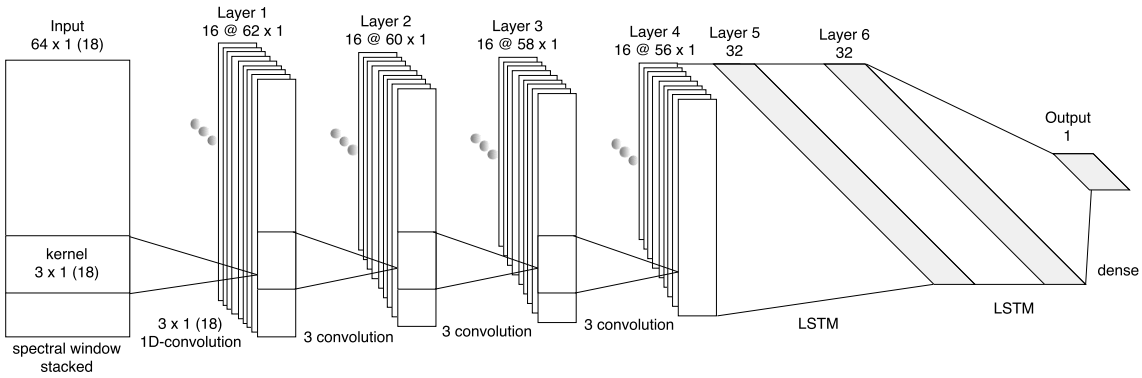


Figure 4.3: Diagram of the 1D-ConvLSTM's architecture.

Number of LSTM layers and neurones per layer. The number of LSTM layers was fixed to be 2 to maximise the model's resemblance to our original approach for comparability purposes. Following the same intuition, the number of neurones per LSTM layer was also set to 32, as in the 1D-ConvNet's dense layers.

Activations. Dealing with vanishing gradient problem in recurrent DL models is different than in feed-forward ones. To perform a fear comparison between our feed-forward and recurrent DL approaches, each model was implemented adopting the best practices for it. As mentioned in Section 2.1, the literature recommends ReLUs as the best practice for feed-forward models' activation functions, however, in recurrent models the commonly implemented activation function is Tanh [25, 27, 28]. The activation functions for the LSTM layers were, therefore, set to the Tanh, while its gating functions were implemented by the hard sigmoid function (i.e. an approximation of Sigmoid) since it fits perfectly with the gating requirement to retrieve values within the range [0, 1].

4.3 1D-ConvGRU

This approach implements an 8-layer 1D-ConvGRU composed by: I) an input layer; II) four convolutional layers; III) two GRU layers; and IV) an output layer. According to the notation defined in Section 2.5, this approach is described as

$$C(3|16) - C(3|16) - C(3|16) - C(3|16) - GRU(32) - GRU(32) - F(1) \quad . \quad (4.3)$$

Furthermore, following a similar notation than LeCun et al. [73] for describing the LeNet-5, Figure 4.3 illustrates the architecture of our 1D-ConvGRU.

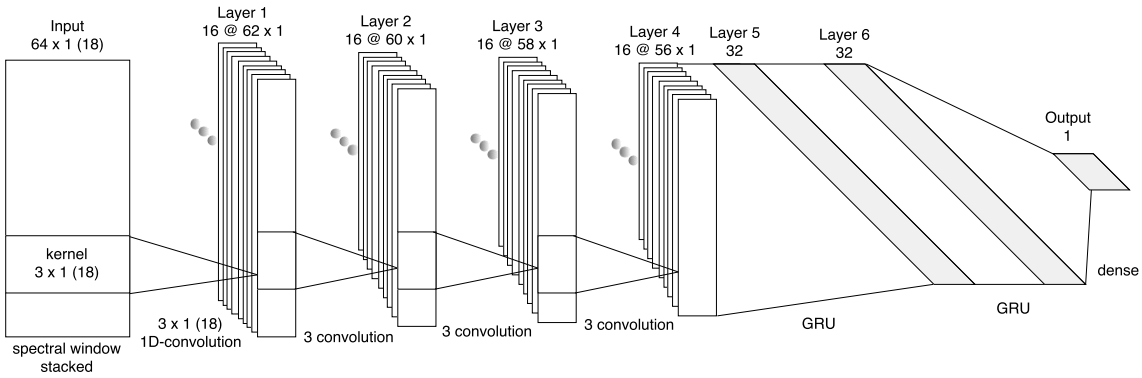


Figure 4.4: Diagram of the 1D-ConvGRU's architecture.

Due to GRUs similarities with LSTMs, all the architectural configurations defined for LSTMs in Section 4.2 were adopted for implementing the 1D-ConvGRU extension of our feed-forward approach.

4.4 Structures comparison

Compared to other DL approaches, ConvNets are usually faster to train. These models, however, are usually composed by large amounts of parameters, which are concentrated in the last dense layers.

Strategies, as pooling, may reduce the impact of this issue in the transition between the last convolutional layer and the first dense layer, where the input is flattened, and, thus, the number of layers' parameters will be proportional to the inputs' size. However, this problem will prevail between the dense hidden layers, whose number of parameters is the number of neurones squared.

The number of parameters p in a DL model are calculated according to the following equations:

- Number of parameters of a convolutional layer:

$$p = c_n * k_n * k_s + k_b \quad , \tag{4.4}$$

  where c_n is the number of channels in the layer's input, k_n is the number of kernels, k_s is the kernel's size, and k_b is the number of biases, which will match with the number of kernels.

- Number of parameters of a dense layer:

$$p = i_s * n_n + n_b \quad , \tag{4.5}$$

where i_s is the size of the flattened layers' input, n_n is the number of neurones, and n_b is the number of biases, which will match with the number of neurones.

- Number of parameters of a gated recurrent layer (i.e. LSTM or GRU):

$$p = a_n * ((c_n + 1) * o_c + o_c^2) \quad , \tag{4.6}$$

  where a_n is the number of gating and activation functions in a cell of the layer (e.g. for an LSTM, this would be 3 Sigmoids and a Tanh, resulting to 4 functions per cell), c_n is the number of channels in the layer's input, which will match with the number of neurones, kernels or cells, in the previous layer, and o_c is layer's channel outputs number.

Table 4.1 presents the details of the number of parameters composing each approach while specifying the structure properties employed for calculating them. From it, it can be observed that, as mentioned in the previous sections, most layers composing the 1D-ConvLSTM and the 1D-ConvGRU have identical structures than the ones in the 1D-ConvNet. The effect of implementing dense layers in a ConvNet without any pooling strategy manifests in the fifth layer of the 1D-ConvNet, where the number of parameters is calculated on the layers' input size resulting into a layer with the 77% of the model's parameters. From comparing the models' complexity, it can be noted that, while the complexity of the connections in the 1D-ConvNet is lower than its recurrent extensions, these have less to be trained parameters.

Table 4.1: Approaches structures and parameters. Columns naming description: '1D-ConvNet', '1D-ConvLSTM' or '1D-ConvGRU' → approach to which the information in the columns under this one belongs; 'Layer' → layer's number, according to Figure 4.1, Figure 4.3 and Figure 4.4 for the 1D-ConvNet, the 1D-ConvLSTM and the 1D-ConvGRU, respectively; 'Properties' → layers' properties affecting the number of parameters; '# Param' → number of parameters in the 'Layer'-th layer; 'c_n' → layer's number of input channels in Equation (4.4) and Equation (4.6); 'k_n' → number of kernels in Equation (4.4); 'k_s' → kernels' size in Equation (4.4); 'k_b' → number of kernels' biases in Equation (4.4); 'i_s' → layer's input size in Equation (4.5); 'n_n' → layer's number of neurones in Equation (4.5); 'n_b' → number of neurones' biases in Equation (4.5); 'a_n' → number of functions in a layer's memory block in Equation (4.6); 'o_c' → layer's output number of channels in Equation (4.6); and row 'Total' → total number of parameters of the approach.

	1D-ConvNet		1D-ConvLSTM		1D-ConvGRU	
Layer	Properties	# Param	Properties	# Param	Properties	# Param
1	$c_n = 1$ $k_n = 16$ $k_s = 3 \times 18$ $k_b = 16$	880	$c_n = 1$ $k_n = 16$ $k_s = 3 \times 18$ $k_b = 16$	880	$c_n = 1$ $k_n = 16$ $k_s = 3 \times 18$ $k_b = 16$	880
2	$c_n = 16$ $k_n = 16$ $k_s = 3 \times 18$ $k_b = 16$	880	$c_n = 1$ $k_n = 16$ $k_s = 3 \times 18$ $k_b = 16$	880	$c_n = 1$ $k_n = 16$ $k_s = 3 \times 18$ $k_b = 16$	880
3	$c_n = 16$ $k_n = 16$ $k_s = 3 \times 18$ $k_b = 16$	880	$c_n = 1$ $k_n = 16$ $k_s = 3 \times 18$ $k_b = 16$	880	$c_n = 1$ $k_n = 16$ $k_s = 3 \times 18$ $k_b = 16$	880
4	$c_n = 16$ $k_n = 16$ $k_s = 3 \times 18$ $k_b = 16$	880	$c_n = 1$ $k_n = 16$ $k_s = 3 \times 18$ $k_b = 16$	880	$c_n = 1$ $k_n = 16$ $k_s = 3 \times 18$ $k_b = 16$	880
5	$i_s = 16 \times 56$ $n_n = 32$ $n_b = 32$	28704	$a_n = 4$ $c_n = 16$ $o_c = 32$	6272	$a_n = 3$ $c_n = 16$ $o_c = 32$	4704
6	$i_s = 32$ $n_n = 32$ $n_b = 32$	1056	$a_n = 4$ $c_n = 32$ $o_c = 32$	8320	$a_n = 3$ $c_n = 32$ $o_c = 32$	6240
7	$i_s = 32$ $n_n = 1$ $n_b = 1$	33	$i_s = 32$ $n_n = 1$ $n_b = 1$	33	$i_s = 32$ $n_n = 1$ $n_b = 1$	33
Total		37121		17857		14209

Chapter 5

Experiments

5.1 Implementation and technologies

This section describes the software and hardware used for deploying the experiments of the studies conducted within this thesis.

5.1.1 Machines' specs

The experiments in this work were executed using the five following machines:

- **PC-1**
  - CPU → Intel Core i7-4770 (8 cores at 3.40 GHz)
  - Memory → 16 GB DDR3
  - GPU → GTX970 (4 GB DDR5)
  - Operative system (OS) → Linux
  - Dist → Ubuntu 14.04
- **PC-2**
  - CPU → Intel Core i7-7700 (8 cores at 3.60 GHz)
  - Memory → 32 GB DDR3
  - GPU → GTX 1060 (6GB DDR5)
  - OS → Linux
  - Dist → Ubuntu 14.04
- **PC-3**

  - CPU → Intel Core i7-7700 (8 cores at 3.60 GHz)
  - Memory → 32 GB DDR3
  - GPU → GTX 1060 (6GB DDR5)
  - OS → Linux
  - Dist → Ubuntu 14.04

- **PC-4**
  - CPU → Intel Core i7-4770 (8 cores at 3.40 GHz)
  - Memory → 16 GB DDR3
  - OS → Microsoft Windows
  - Dist → Windows 7

- **PC-5**
  - CPU → Intel Core i7-7700 (8 cores at 3.60 GHz)
  - Memory → 32 GB DDR3
  - GPU → GTX 1060 (6GB DDR5)
  - OS → Microsoft Windows
  - Dist → Windows 10

From these machines, PC-1, PC-2 and PC-3 were employed for training the DL models, while PC-4 and PC-5 were devoted to the shallow ML algorithms.

5.1.2 Programming tools

Technologies adopted for training DL models for FOG detection. The code for training the DL models implemented was written in Python (version 3.4), using Keras library (version 1.2.2) [26] running on top of TensorFlow (version tensorflow-gpu 1.0.1) [10]. Furthermore, the code implemented for the training and processing algorithms is publicly available at [8].

Technologies adopted for training shallow ML models for FOG detection. The code for training the shallow ML models implemented was written in Matlab (version R2017a), using its Statistics and Machine Learning Toolbox.

5.2 DL training and evaluation settings

This section explains the configurations of the DL training employed.

5.2.1 Weights initialisation

As mentioned in Section 2.4, several different approaches may function for our problem.

For all the DL models trained, their weights were, thus, initialized using the method presented by Glorot et al. in 2010 [51], which, indeed, allowed the model to learn the FOG representations successfully in the data.

5.2.2 Activations

As discussed in Chapter 4, the activation functions of the models were set per each layer as follows:

- Convolutional layers → ReLU.
- Recurrent layers → Tanh for the activations and hard sigmoid for the gates.
- Hidden dense layers → ReLU.
- Output layer → linear function, for providing the hinge loss with an expressive output representation.

5.2.3 Error loss

Since FOG detection was addressed as a binary classification task, hinge loss (5.1) algorithm was implemented as the error loss function. Hinge loss algorithm is a loss error method specialised for binary classification problems, which is defined as

$$l_h = mean(max(0, 1 - \boldsymbol{y}_{true} * \boldsymbol{y}_{pred})) \quad , \tag{5.1}$$

where $\boldsymbol{y}_{true}$ are the real labels of the data, which can either be -1 or 1, while $\boldsymbol{y}_{pred}$ are the model label predictions which can adopt real values in the range $[-\infty, +\infty]$.

However, the class imbalance in the training data prevented the model from learning strong FOG representations. Therefore, a balanced extension of the hinge loss

function was considered. This function was defined as

$$\boldsymbol{l}_b^+ = max(0, 1 - \boldsymbol{y}_{true}^+ * \boldsymbol{y}_{pred}) * (1 - \rho) \tag{5.2}$$

$$\boldsymbol{l}_b^- = max(0, 1 - \boldsymbol{y}_{true}^- * \boldsymbol{y}_{pred}) * \rho \tag{5.3}$$

$$l_b = mean(\boldsymbol{l}_b^+ + \boldsymbol{l}_b^-) \quad , \tag{5.4}$$

where $\boldsymbol{y}_{true}^+$ defines the real positive samples in the data while $\boldsymbol{y}_{true}^-$ defines the negative ones, ρ is the prior of FOG samples in the training data (i.e. the percentage of FOG samples in the training dataset) and, thus, $1 - \rho$ is the prior of non-FOG samples in the training data.

This class balancing strategy permitted to train our models properly, however, it overweighted the FOG labelled samples leading to models with low accuracy. The Equation 5.4 was, thus, redefined to regulate the models' attention on the specificity metric while still considering the class imbalance factor on the data. Concretely, the weighted hinge loss function implemented was defined as

$$l_w = mean(\boldsymbol{l}_b^+ * w^{-1} + \boldsymbol{l}_b^- * w) \quad , \tag{5.5}$$

where w is a class weighting coefficient and $1 < w < \frac{(1-\rho)}{\rho}$.

This new weighting coefficient was included in the hyperparameters tuning exploration. From which, in the presented approaches this parameter w was set to 1.5 for the 1D-ConvNet, 2 for the 1D-ConvLSTM, and 1.5 for the 1D-ConvGRU.

Furthermore, gradient clipping with clip value set to 1 was implemented to control the exploding gradient phenomenon, such as cliffs, while training the recurrent extensions models.

5.2.4 Optimiser

Adam [66] has been widely implemented in several DL approaches [143, 71, 68, 147, 56], moreover, Goodfellow et al. in 2016 [53] recommended Adam as a good optimiser for DL models. The models presented were, thus, trained via backpropagation and Adam algorithm as the optimisation method.

The learning rate for training the models of the reported results was set to $5 \cdot 10^{-5}/batch - size$.

5.2.5 Minibatch training

According to Goodfellow et al. (2016) [53], generalisation error is often best for a batch size of 1. However, this strategy is time-consuming. Therefore, the DL models were trained by minibatches of 16 samples.

5.2.6 Regularization

The 1D-ConvNets trained initially were prone to overfit on the training data; thus, the following strategies were implemented:

1. Dropout [129] with the probability parameter set to 0.5 for the convolutional layers and 0.25 for the hidden dense ones.

2. L2-norm weight regularisation with the penalty parameter set to 10^{-5}, in all layers except the output layer. This last one was trained using the L2-norm weight regularisation, but, with 10^{-2} as penalty parameter.

3. Early stopping such that convergence was redefined to take place when a model's best metric increased less than a threshold during a large number of epochs. Concretely, the metric evaluated within this process was the minimum between the GM of the Training-train sensitivity and the Training-train specificity, and the GM of the Training-validation sensitivity and the Training-validation specificity, while the change threshold was set to 0.005 and the number epochs to wait before determining convergence was set to 300. Therefore, the model corresponding to the epoch with highest metric value was set as the final model; however, only those that trained for at least 100 epochs before converging were further considered to control premature convergence.

4. Data augmentation which increased the Training-train dataset size by a factor of four, while introducing coherent stochastic noise in the data.

When implementing our approach's extensions, it was noted that none of the recurrent could train with all these regularisation strategies. From the regularization strategies implemented for training the feed-forward models, the 1D-ConvLSTM and the 1D-ConvGRU only implemented the following: I) L2 norm weight regularization with the penalty parameter set to 10^{-2} only in the model's output layer; II) early stopping, but raising the number of epochs for the convergence criteria to 700 and 1000 for the 1D-ConvGRU and the 1D-ConvLSTM, respectively; and III) the same data augmentation strategies than the feed-forward models.

5.2.7 Training data feeding strategies

This subsection is devoted to describing the strategies designed for training our DL approaches.

Feed-forward feeding strategy. This strategy is presented in Algorithm 2. This algorithm first generated the random parameters for the data augmentation strategies. These parameters were the starting random shifts, which were generated one for each pair file and augmentation iteration, and the random rotations, which were generated $1000 \times augmentation\_factor$ random rotation matrices, since generating a different rotation for each sample would be inefficient. Next, the algorithm iterated $augmentation\_factor$ times over the data, augmenting it differently. Within each iteration the algorithm performed a stochastic strategy to train samples in an absolute random order.

Algorithm 2 Feed-forward feeding strategy

```
 1: procedure TRAIN_EPOCH(model,data)        ▷ Feeds the model to train one epoch
 2:     num_files ← COUNT_FILES(data)
 3:     augment_factor ← 4
 4:     num_rotations ← 1000 × augment_factor
 5:     augment_shifts = GET_SHIFTS(num_files,augment_factor)
 6:     augment_rotations = GET_ROTATIONS(num_rotations)
 7:     batch ← [ ]
 8:     for i = 1, i++, i < augment_factor do
 9:         shift ← augment_shifts[i]
10:         rotation ← RANDOM_SELECT(augment_rotations)
11:         new_data = AUGMENTATION(data,shift,rotate)
12:         windowed_data = WINDOW(new_data)
13:         sample_list = SWS(windowed_data)          ▷ spectral window stacking
14:         sample_list = SHUFFLE(sample_list)
15:         for <sample in sample_list> do
16:             APPEND(batch,sample)
17:             if SIZE(batch) = 16 then
18:                 TRAIN(model,batch)
19:                 batch ← [ ]
20:             end if
21:         end for
22:     end for
23:     return model                              ▷ The model has trained one epoch
24: end procedure
```

Recurrent feeding strategy. The recurrent version of the feeding strategy slightly differs from the feed-forward one due to the need of maintaining temporal dependence between consecutive samples being feed. As before, this process is presented in Algorithm 3, while the steps composing it are hereafter outlined. This algorithm first

generated the random parameters for the data augmentation strategies. These parameters were the starting random shifts and random rotations, which were both generated one for each pair file and augmentation iteration, since files will be feed in a row and should, thus, have the same augmentation operations applied. Next, the algorithm iterated *augmentation_factor* times over the data, augmenting it differently. Within each iteration, the algorithm performed a stochastic strategy to train files in random order. Furthermore, after each file is feed to the model for training, the recurrent memories of it are reset before feeding the next file.

Algorithm 3 Recurrent feeding strategy

```
 1: procedure TRAIN_EPOCH(model,data)        ▷ Feeds the model to train one epoch
 2:     data = SHUFFLE(data)
 3:     for <file in data> do
 4:         size ← SIZE(file)
 5:         augment_factor ← 4
 6:         augment_shifts = GET_SHIFTS(size,augment_factor)
 7:         augment_rotations = GET_ROTATIONS(size,augment_factor)
 8:         for i = 1, i++, i < augment_factor do
 9:             shift ← augment_shifts[i]
10:             rotation ← augment_rotations[i]
11:             new_data = AUGMENTATION(file,shift,rotate)
12:             windowed_data = WINDOW(new_data)
13:             sample_list = SWS(windowed_data)        ▷ Spectral Window Stacking
14:             batch ← [ ]
15:             for <sample in sample_list> do
16:                 APPEND(batch,sample)
17:                 if SIZE(batch) = 16 then
18:                     TRAIN(model,batch)
19:                     batch ← [ ]
20:                 end if
21:             end for
22:             RESET(model)                        ▷ Reset model's memories
23:         end for
24:     end for
25:     return model                        ▷ The model has trained one epoch
26: end procedure
```

5.2.8 Evaluation data feeding strategies

The evaluation for the DL approaches, which is deeply reviewed in Section 5.3, was performed patient-wisely without data augmentation strategies or dropout. There-

fore, for simplicity, all DL models were evaluated using the same feeding strategy which is presented in Algorithm 4, which was executed once per patient, setting as the algorithm's second argument *patient* to a patient's data (i.e. a list of files of signals). This strategy was implemented by the following steps for each patient. First, the algorithm iterates over the patient's files and, within each iteration, the algorithm evaluates the data in batches and accumulates the results, which will be later employed for calculating the model's metrics. Furthermore, after each file is fed to the model, its recurrent memories of it are reset before feeding the next file; otherwise, the process continues normally.

Algorithm 4 Evaluation feeding strategy

```
procedure EVALUATE_MODEL(model,patient)        ▷ Feeds one patient's data to evaluate
    patient_eval ← [ ]
    for <file in patient> do
        windowed_data = WINDOW(file)
        sample_list = SWS(windowed_data)        ▷ Spectral Window Stacking
        batch ← [ ]
        for <sample in sample_list> do
            APPEND(batch,sample)
            if SIZE(batch) = 16 then
                batch_eval ← EVALUATE(model,batch)
                APPEND(patient_eval,batch_eval)
                batch ← [ ]
            end if
        end for
        if IS_RECURRENT(model) then
            RESET(model)
        end if
    end for
    return patient_eval                         ▷ Patient's evaluation
end procedure
```

This feeding strategy was employed for both, validation and testing processes. Note that compared to the feed-forward strategy more data is lost due to restricting that in a batch data should belong to the same patient. However, since the evaluation was performed in batches, to obtain the algorithm performance per patient, it was necessary to implement this constraint for all DL models' evaluation.

5.3 Evaluation

As already introduced in Subsection 5.2.8, the evaluation of all algorithms implemented was performed per patient. This strategy implied that the metrics, such as the GM between sensitivity and specificity, of the models were computed for each patient's data independently, and, finally, averaged among all patients' results. The motivation of this evaluation approach is that models that learn only the patterns of some patients may compensate their performance in others, hence outlier patients will have a similar negative effect in all learning algorithms. However, models will be unable to compensate their performance when representing all patients wrong. Concretely, models that can distinguish FOG from non-FOG for a particular patient can reach a score greater than 0 for the GM between sensitivity and specificity for that patient.

Following the trend in the related literature [79, 141, 31, 110], the metric employed for selecting the models in this thesis was the average among the patients' GM between the sensitivity and specificity. Other metrics were computed following the same per patient intuition, and are reported in Chapter 6 for comparability purposes.

This study aimed to outperform the state-of-the-art for automatic FOG detection using DL while providing other researchers with a complete and reproducible basis work from where to start future studies. To ensure these characteristics, all procedures in this work were neatly and justly performed. The models were trained with several hundreds of hyperparameter's configurations until reaching the presented ones. The approaches implemented were trained uniquely with Training-train data and assessed with the Training-validation data before being tested. However, the models were only tested once, and all at the same time. Their results were ranked according to the training selection criteria for the early stopping, which was the minimum between the GM of sensitivity and specificity, for train and validation. From the overall results, only the top-5 models according to these criteria were tested, however, only the best training model will be used for comparison and considered for discussion throughout this thesis, while the remaining 4 models were only tested and reported to ensure robustness and reproducibility of our work.

The evaluation strategy that was employed for in this study is hereafter outlined, while in Chapter 6, the results and discussing derived from applying this strategy are presented.

Comparison among DL approaches. To report the performances achieved by 1D-ConvNet and its recurrent extensions, and compare them.

Comparison among shallow ML approaches. To report the performances obtained by reproductions of other authors' work when being trained using the most suitable shallow ML algorithms, and compare them to the results retrieved by our DL models.

Comparison between DL and shallow ML approaches. To compare and discuss the results reported from our DL approaches and our reproduction of the state-of-the-art strategies used for training shallow ML algorithms.

5.4 Reproduction of the state-of-the-art approaches

This section describes and discusses the feature extractions, and their settings reproduced within this thesis.

5.4.1 Data representation and preprocessing for reproducing the approaches

This subsection discusses the preprocessing and representation techniques and settings implemented for reproducing the state-of-the-art approaches.

As mentioned in Section 1.5, many of these authors employed different window sizes, labelling strategies, and evaluation methods. The data they employed were, however, different than the one used for this study; thus, to perform a fair comparison, these characteristics should be optimised as hyperparameters for each approach. Rodríguez et al. [31] replicated these same approaches on a subset of 6 patients of the dataset being employed. In their study, moreover, these approaches were compared using different window sizes on the data sampled at 40 Hz, by which high validation performances were achieved. According to Rodríguez et al. [31], furthermore, there were no significant differences when changing the window's size. Note that the best performances reported by Rodríguez et al. [31] were using window sizes of 0.8 s and 1.6 s, however, according to Moore et al. [80] the window's size should be higher than 2.5 s. Thus, the window size was set to 3.2 s, to keep in concordance with the literature, while maintaining the others authors' approaches as close as their original work as possible, since they all (except Tripoliti et al. in 2013 [141]) employed windows larger than 2 s for FOG detection.

The data preprocessing strategies employed were the same that for preparing the data for training the DL models, as described in Section 3.3. The only differences for these feature extraction methods and later training the shallow ML algorithms were:

1. The data were sampled at 40 Hz, instead of 50 Hz.
2. The acceleration signals data were filtered using a 2nd order lowpass Butterworth filter with the cut-off set to 15 Hz and its initial conditions handled by setting them to the mean of the signal.
3. The gyroscope signals data were filtered using a 2nd order highpass Butterworth filter with the cut-off set to 0.2 Hz and its initial conditions handled by setting them to the mean of the signal.
4. The magnetometer signals data were discarded since none of the reproduced approaches employed these signals.
5. Since these handcrafted feature extraction methods worked without any data augmentation strategy; data were directly windowed using windows of 3.2 s and 50% of window overlapping to avoid losing interwindow information due to the splitting points.

5.4.2 Implementation of the feature extractions

Although some of the following authors presented extensions of their work, these were not taken into account due to the ambiguity of their performances achieved in their literature sources (e.g. Mazilu et al. in 2013 [78] and Mazilu et al. in 2016 [77]).

Notation for the feature extraction methods. Let the following notation be introduced and later adopted for describing the feature extraction operations in this section.

- Accelerometer signals from the current window being analysed $\rightarrow \boldsymbol{acc}$ or $\boldsymbol{acc}_t$, hence equations including the $\boldsymbol{acc}_{t-1}$ term will necessarily refer to the current window as $\boldsymbol{acc}_t$ rather than $\boldsymbol{acc}$.
- Accelerometer signals from the previous window to the current one being analysed $\rightarrow \boldsymbol{acc}_{t-1}$.
- Gyroscope signals from the current window being analysed $\rightarrow \boldsymbol{gyro}$.

- Accelerometer X-axis signal from the current window being analysed → $\boldsymbol{acc}_x$ or $\boldsymbol{acc}_{xt}$.
- Accelerometer X-axis signal from the previous window to the current one being analysed → $\boldsymbol{acc}_{x(t-1)}$.
- Accelerometer Y-axis signal from the current window being analysed → $\boldsymbol{acc}_y$ or $\boldsymbol{acc}_{yt}$.
- Accelerometer Y-axis signal from the previous window to the current one being analysed → $\boldsymbol{acc}_{y(t-1)}$.
- Accelerometer Z-axis signal from the current window being analysed → $\boldsymbol{acc}_z$ or $\boldsymbol{acc}_{zt}$.
- Accelerometer Z-axis signal from the previous window to the current one being analysed → $\boldsymbol{acc}_{z(t-1)}$.
- Gyroscope X-axis signal from the current window being analysed → $\boldsymbol{gyro}_x$.
- Gyroscope Y-axis signal from the current window being analysed → $\boldsymbol{gyro}_y$.
- Gyroscope Z-axis signal from the current window being analysed → $\boldsymbol{gyro}_z$.

The following paragraphs detail the features composing the approaches reproduced in this work. These approaches extract the described features for each window in the data, which were arranged as described in Subsection 5.4.1.

MBFA presented by Bächlin et al. in 2009 [16]. This feature extraction was applied for each window in the data, specifically using the $\boldsymbol{acc}$. The features composing the MBFA are

1. The FI_y, which corresponds to the FI of the $\boldsymbol{acc}_y$, defined as

$$FI_y = \frac{FB_y}{LB_y} \quad , \tag{5.6}$$

where FB_y refers to the FB of the $\boldsymbol{acc}_y$ and LB_y is the LB of the $\boldsymbol{acc}_y$. Concretely, the FB_y is defined as

$$FB_y = \sum_{f_i=3}^{8} A_{y,f_i}^2 \quad , \tag{5.7}$$

where, using a similar notation that the one employed by Rodríguez et al. (2017) [110], f_i is the iterator over the frequencies represented, while 8 is the upper frequency amplitude bound, which together with the initial value of f_i define the band to be summed. Concretely, in Equation (5.7), the amplitude of all harmonics in spectral representation of the $\boldsymbol{acc}_y$ within the frequencies rage [0.5 Hz, 3 Hz] are summed. The $\boldsymbol{A}_y$ vector is composed of the harmonic amplitudes obtained by applying the FFT to $\boldsymbol{acc}_y$, performing its absolute value and keeping only its first symmetric half. Thus, the elements of the amplitude vector $\boldsymbol{A}_y$ are defined by

$$A_{y,f_i} = 2 * |A^F_{y,f_i}| \quad , \tag{5.8}$$

where A^F_{y,f_i} is the harmonic corresponding to frequency $f_i \in \left[0, \frac{f_N}{2}\right]$, where f_N is the sampling frequency, and $\boldsymbol{A}^F_y$ is the vector of the complex values corresponding to the harmonics from applying the FFT function to $\boldsymbol{acc}_y$, which is defined as $\boldsymbol{A}^F_y = FFT(\boldsymbol{acc}_y)$.

The LB_y from Equation (5.6) is defined as

$$LB_y = \sum_{f_i=0.5}^{3} A^2_{y,f_i} \quad . \tag{5.9}$$

2. The PI of the $\boldsymbol{acc}_y$ defined as

$$PI_y = \sum_{f_i=f_2}^{\frac{f_N}{2}} A^2_{yi} \quad , \tag{5.10}$$

where f_2 is the fist frequency after the continuous component.

Online FOG detection presented by Mazilu et al. in 2012 [79]. This feature extraction was applied for each window in the data, specifically using the $\boldsymbol{acc}$ signals. The features composing this approach are

1. The FI_y implemented as in the MBFA.
2. The PI_y implemented as in the MBFA.
3. The mean of the $\boldsymbol{acc}_x$.
4. The mean of the $\boldsymbol{acc}_y$.
5. The mean of the $\boldsymbol{acc}_z$.

6. The ST of the $\boldsymbol{acc}_x$.

7. The ST of the $\boldsymbol{acc}_y$.

8. The ST of the $\boldsymbol{acc}_z$.

9. The variance (VAR) of the $\boldsymbol{acc}_x$.

10. The VAR of the $\boldsymbol{acc}_y$.

11. The VAR of the $\boldsymbol{acc}_z$.

12. The entropy S of the amplitudes of $\boldsymbol{acc}_y$. The detailed implementation employed is described in Gonzalez et al. [52]. Concretely, the entropy of the amplitudes of $\boldsymbol{acc}_y$ was defined as

$$S(\boldsymbol{A}_y) = -\sum_{f_i=f_2}^{\frac{f_N}{2}} h(\boldsymbol{A}_y) * log_2(h(\boldsymbol{A}_y)) \quad , \tag{5.11}$$

where h is the histogram function and log_2 is the logarithm function with basis equals 2.

13. The energy of $\boldsymbol{acc}_x$ defined as

$$E_x = \frac{2}{f_N} * \sum_{f_i=f_2}^{\frac{f_N}{2}} A_{xi}^2 \quad , \tag{5.12}$$

where $\boldsymbol{A}_x$ is defined as in Equation (5.8), but using the $\boldsymbol{acc}_x$ instead.

14. The energy of $\boldsymbol{acc}_y$ defined as in Equation (5.12).

15. The energy of $\boldsymbol{acc}_z$ defined as in Equation (5.12).

Four-stage FOG detection presented by Tripoliti et al. in 2013 [141]. This feature extraction was applied for each window in the data using the $\boldsymbol{acc}$ and $\boldsymbol{gyro}$. The features composing this approach are

1. The entropy of the amplitudes of $\boldsymbol{acc}_x$, defined as in Equation (5.11).

2. The entropy of the amplitudes of $\boldsymbol{acc}_y$.

3. The entropy of the amplitudes of $\boldsymbol{acc}_z$.

4. The entropy of the amplitudes of $\boldsymbol{gyro}_x$.

5. The entropy of the amplitudes of $\boldsymbol{gyro}_y$.

6. The entropy of the amplitudes of $\boldsymbol{gyro}_z$.

Uncontrolled environments FOG detection presented by Rodríguez et al. in 2016 [31]. This feature extraction, which was also used throughout the study presented by Rodríguez et al. in 2017 [110], was applied for each window in the data using the $\boldsymbol{acc}_t$ and the $\boldsymbol{acc}_{t-1}$. The features composing this approach are

1. The mean of the $\boldsymbol{acc}_x$.

2. The mean of the $\boldsymbol{acc}_y$.

3. The mean of the $\boldsymbol{acc}_z$.

4. The ST of the $\boldsymbol{acc}_x$.

5. The ST of the $\boldsymbol{acc}_y$.

6. The ST of the $\boldsymbol{acc}_z$.

7. The difference between Item 1 and Item 3.

8. The difference between Item 2 and Item 7.

9. The difference between Item 1 and the same operation but on $\boldsymbol{acc}_{x(t-1)}$.

10. The difference between Item 7 and the same operation but on $\boldsymbol{acc}_{x(t-1)}$ and $\boldsymbol{acc}_{z(t-1)}$.

11. The difference between Item 8 and the same operation but on $\boldsymbol{acc}_{x(t-1)}$, $\boldsymbol{acc}_{z(t-1)}$ and $\boldsymbol{acc}_{y(t-1)}$.

12. The skewness of $\boldsymbol{acc}_x$, defined as the third central moment of the signal divided by the cube of its ST.

13. The skewness of $\boldsymbol{acc}_{yt}$.

14. The skewness of $\boldsymbol{acc}_{zt}$.

15. The skewness of the $\boldsymbol{acc}$'s L2-norm in the spectral domain defined as

$$Skew_{L2} = Skew(\|\boldsymbol{acc}\|_2) = Skew(\sqrt{\boldsymbol{acc}_x^2 + \boldsymbol{acc}_y^2 + \boldsymbol{acc}_z^2}) \quad , \tag{5.13}$$

where $Skew$ is the skewness function.

16. The skewness of the LB amplitudes defined as

$$Skew_{LB} = Skew([A_{y,0.5}^2, \ldots, A_{y,3}^2]) \quad . \tag{5.14}$$

17. The skewness of the FB amplitudes defined as

$$Skew_{FB} = Skew([A_{y,3}^2, \ldots, A_{y,8}^2]) \quad . \tag{5.15}$$

18. The skewness of both bands, the LB and the FB amplitudes defined as

$$Skew_{FLB} = Skew_{(}[A_{y,0.5}^2, \ldots, A_{y,8}^2]) \quad . \tag{5.16}$$

19. The ST of the posture transition band (PT), which considers frequencies from 0.1 to 0.68 Hz.

20. The ST of the FB amplitudes.

21. The ST of the LB amplitudes.

22. The ST of the FB and LB amplitudes (i.e. from 0.5 to 8 Hz).

23. The ST of amplitudes corresponding to frequencies above the LB; thus, the range considered is from 8 to $\frac{f_N}{2}$ Hz.

24. The frequency of the centre of mass, defined as a weighted sum of frequencies, where each frequency is weighted by its amplitude using the $\boldsymbol{acc}_y$.

25. The correlation coefficient between $\boldsymbol{acc}_x$ and $\boldsymbol{acc}_y$.

26. The correlation coefficient between $\boldsymbol{acc}_x$ and $\boldsymbol{acc}_z$.

27. The correlation coefficient between $\boldsymbol{acc}_y$ and $\boldsymbol{acc}_z$.

28. The frequency with maximum amplitude in $\boldsymbol{acc}_y$.

29. The second frequency with maximum amplitude in $\boldsymbol{acc}_y$.

30. The first value of applying the principal components analysis' (PCA) of the spectral representation of $\boldsymbol{acc}_y$.

31. The second value of applying the PCA as defined in Item 30. Note that, only the first two values from the PCA were employed following the intuition from the 'elbow rule'. The 'elbow rule' establishes that the information gain of using the n-th eigenvector of the PCA decomposition should be significantly greater than later taking the $n+1$-th one. Thus, the process is interrupted when an 'elbow' appears in the information gain curve.

5.5 Shallow ML experiments

This section explains the shallow ML algorithms and the training and evaluation strategies implemented for them.

5.5.1 Shallow ML algorithms implemented

On the one hand, our approaches implement some of the most novelty techniques in ML, and, moreover, were laboriously designed and tuned. Whereas, on the other hand, most of the shallow ML algorithms reviewed in the automatic FOG detection's state-of-the-art are unrecommended algorithms for binary classification according to Caruana et al. in 2006 [23]. More precisely, in their previous works:

- Bächlin et al.'s (2009) [16] approach used only threshold-based techniques for solving the binary classification task.

- Mazilu et al.'s (2012) [79] approach presents results using several algorithms, such as RF and AdaBoost, which indeed according to Caruana et al. in 2006 [23] may be a powerful algorithm on this task. However, they include results from algorithms such as K-NN and NB, which are known to be weak for binary classification [23].

- Tripoliti et al.'s (2013) [141] approach presents results using several algorithms, from which only RF, according to Caruana et al. in 2006 [23], may be a powerful algorithm on this task.

- Rodríguez et al.'s (2016) [31] approach presents results using SVMs. However, in the comparison section, only RF is another powerful algorithm for binary classification according to Caruana et al. in 2006 [23], while K-NN, NB and

logistic regression are weak ones; moreover, logistic regression was not even implemented in the original sources of the other feature extraction methods.

To perform a fair comparison between the state-of-the-art for FOG detection and our approaches, the feature extractions were reproduced and used for training strong classifiers for two-class problems. As mentioned in Section 1.3, the shallow ML algorithms implemented were the tree bagging [22], the AdaBoost [42], the LogitBoost [43], the RUSBoost [126], the RobustBoost [41] and the SVM [30]. The motivation for selecting these algorithms is hereafter discussed while presenting the hyperparameters considered for the exploration and the final configurations chosen for each feature extraction method.

Tree bagging described in Breiman et al. (1996) [22]. This ML algorithm trains an ensemble of decision trees using subsets from the training data, which are generated by sampling N instances with replacement, where N is the number of samples in the training set. The prediction is performed by majority voting from all the trees in the ensemble.

The algorithm's hyperparameters' were tuned independently for each feature extraction by performing an exhaustive exploration considering the following values for each parameter:

1. Number of trees: 128, 256, 512 and 1024.
2. Tree shapes:
    - Decision stumps (i.e. one level decision trees).
    - Decision trees with minimum leaf size set to 3.
    - Decision trees with minimum leaf size set to 5% of training data.
    - Decision trees with minimum leaf size set to 10% of training data.

The implemented hyperparameters' configuration implemented for this algorithm are shown in Table 5.1.

AdaBoost described in Freund et al. (1995) [42], using decision trees as weak learners. This algorithm trains an ensemble of decision trees sequentially, such that the new trees aggregated to the ensemble are focused on the previously misclassified samples. Finally, this algorithm will predict the label for new samples by performing a weighted average on over all the predictions of the trees in the ensemble.

Table 5.1: Tree bagging configurations. Columns naming description: '# trees' → number of decision trees forming the ensemble method; and 'tree type' → properties of the trees implemented. The values in the column 'tree type' may adopt one of the following values: 'tree' → traditional decision tree; 'min_x' → tree composed by leafs with minimum size x; or 'max_x' → tree of maximum depth x, note that 'max_1' will be the decision stump. Additionally, next to the 'min_x' or 'max_x' values, there might appear a percentage % symbol, which indicates that the value of x is a percentage over the training data.

Feature extraction method	# Trees	Tree type
Bächlin et al. (2009) [16]	1024	min_3
Mazilu et al.'s (2012) [79]	1024	min_5%
Tripoliti et al.'s (2013) [141]	1024	min_5%
Rodríguez et al.'s (2016) [31]	1024	min_5%

This algorithm was chosen since it has been recognised as a strong approach for binary classification tasks [23].

The algorithm's hyperparameters' were tuned independently for each feature extraction by performing an exhaustive exploration considering the following values for each parameter:

1. Number of trees: 128, 256, 512 and 1024.
2. Tree shapes:
   - Decision stumps (i.e. one level decision trees).
   - Decision trees with minimum leaf size set to 3.
   - Decision trees with minimum leaf size set to 5% of training data.
   - Decision trees with minimum leaf size set to 10% of training data.
3. Learning rate: 0.1, 0.5 and 1.

The implemented hyperparameters' configuration implemented for this algorithm are shown in Table 5.2.

LogitBoost described in Friedman et al. (2000) [43], using decision trees as weak learners. This algorithm extends AdaBoost by reducing the weight assigned to badly misclassified samples; thus, it may outperform AdaBoost in classifying poorly separable data.

Table 5.2: AdaBoost configurations. Columns naming description: '# trees' → number of decision trees forming the ensemble method; 'tree type' → properties of the trees implemented; and 'learning rate' → the learning rate for training the algorithm, note that values lower than 1 may have an shrinkage effect. The values in the column 'tree type' may adopt one of the following values: 'tree' → traditional decision tree; 'min_x' → tree composed by leafs with minimum size x; or 'max_x' → tree of maximum depth x, note that 'max_1' will be the decision stump. Additionally, next to the 'min_x' or 'max_x' values, there might appear a percentage % symbol, which indicates that the value of x is a percentage over the training data.

Feature extraction method	# Trees	Tree type	Learning rate
Bächlin et al. (2009) [16]	1024	min_3	0.1
Mazilu et al.'s (2012) [79]	1024	min_3	0.1
Tripoliti et al.'s (2013) [141]	1024	min_3	0.1
Rodríguez et al.'s (2016) [31]	1024	min_10%	0.1

The algorithm's hyperparameters' were tuned independently for each feature extraction by performing an exhaustive exploration considering the following values for each parameter:

1. Number of trees: 128, 256, 512 and 1024.
2. Tree shapes:
   - Decision stumps (i.e. one level decision trees).
   - Decision trees with minimum leaf size set to 3.
   - Decision trees with minimum leaf size set to 5% of training data.
   - Decision trees with minimum leaf size set to 10% of training data.
3. Learning rate: 0.1, 0.5 and 1.

The implemented hyperparameters' configuration implemented for this algorithm are shown in Table 5.3.

RUSBoost [126], using decision trees as weak learners. This algorithm extends AdaBoost by training the learners with class balanced subsets of the training data; thus, it may outperform AdaBoost in classifying class imbalance data.

The algorithm's hyperparameters' were tuned independently for each feature extraction by performing an exhaustive exploration considering the following values for each parameter:

Table 5.3: LogitBoost configurations. Columns naming description: '# trees' → number of decision trees forming the ensemble method; 'tree type' → properties of the trees implemented; and 'learning rate' → the learning rate for training the algorithm. The values in the column 'tree type' may adopt one of the following values: 'tree' → traditional decision tree; 'min_x' → tree composed by leafs with minimum size x; or 'max_x' → tree of maximum depth x, note that 'max_1' will be the decision stump. Additionally, next to the 'min_x' or 'max_x' values, there might appear a percentage % symbol, which indicates that the value of x is a percentage over the training data.

Feature extraction method	# Trees	Tree type	Learning rate
Bächlin et al. (2009) [16]	1024	min_3	0.1
Mazilu et al.'s (2012) [79]	1024	min_5%	0.1
Tripoliti et al.'s (2013) [141]	1024	min_5%	0.1
Rodríguez et al.'s (2016) [31]	1024	min_5%	0.1

1. Number of trees: 128, 256, 512 and 1024.
2. Tree shapes:
   - Decision stumps (i.e. one level decision trees).
   - Decision trees with minimum leaf size set to 3.
   - Decision trees with minimum leaf size set to 5% of training data.
   - Decision trees with minimum leaf size set to 10% of training data.
3. Learning rate: 0.1, 0.5 and 1.

The implemented hyperparameters' configuration implemented for this algorithm are shown in Table 5.4.

RobustBoost [41], using decision trees as weak learners. The traditional AdaBoost focuses each iteration on classifying previously misclassified samples. This behaviour may lower the average accuracy of the classifier if there are incorrect labels in the data. The RobustBoost algorithm extends AdaBoost by maximising the number of samples undoubtedly (i.e. above a certain threshold) well classified, instead of minimising the models' train error. Furthermore, this algorithm allows the usage of an error tolerance, which is used to figure optimal margins and prevent it from overfitting.

Table 5.4: RUSBoost configurations. Columns naming description: '# trees' → number of decision trees forming the ensemble method; 'tree type' → properties of the trees implemented; and 'learning rate' → the learning rate for training the algorithm. The values in the column 'tree type' may adopt one of the following values: 'tree' → traditional decision tree; 'min_x' → tree composed by leafs with minimum size x; or 'max_x' → tree of maximum depth x, note that 'max_1' will be the decision stump. Additionally, next to the 'min_x' or 'max_x' values, there might appear a percentage % symbol, which indicates that the value of x is a percentage over the training data.

Feature extraction method	# Trees	Tree type	Learning rate
Bächlin et al. (2009) [16]	1024	max_1	0.1
Mazilu et al.'s (2012) [79]	1024	max_1	0.1
Tripoliti et al.'s (2013) [141]	1024	max_1	0.1
Rodríguez et al.'s (2016) [31]	1024	max_1	0.1

The algorithm's hyperparameters' were tuned independently for each feature extraction by performing an exhaustive exploration considering the following values for each parameter:

1. Number of trees: 128, 256, 512 and 1024.
2. Tree shapes:
   - Decision stumps (i.e. one level decision trees).
   - Decision trees with minimum leaf size set to 3.
   - Decision trees with minimum leaf size set to 5% of training data.
   - Decision trees with minimum leaf size set to 10% of training data.
3. Error goal: 0.1, 0.05 and 0.1.

The implemented hyperparameters' configuration implemented for this algorithm are shown in Table 5.5.

SVM [30], using the radial basis function (RBF) kernel [144] (SVM-RBF). SVMs are a a powerful shallow ML algorithm, specially in binary classification tasks. SVMs use the 'kernel trick', which implies to switch the inner product of the data ($\boldsymbol{x}_i^T \boldsymbol{x}_j$) for a kernel K, which is defined as $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$. Concretely, the kernel implemented was the RBF kernel, which is defined as $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma^2}\right)$

Table 5.5: RobustBoost configurations. Columns naming description: '# trees' → number of decision trees forming the ensemble method; 'tree type' → properties of the trees implemented; and 'error goal' → the error tolerance percentage by which the algorithm will stop training and, thus, end with larger confidence margins. The values in the column 'tree type' may adopt one of the following values: 'tree' → traditional decision tree; 'min_x' → tree composed by leafs with minimum size x; or 'max_x' → tree of maximum depth x, note that 'max_1' will be the decision stump. Additionally, next to the 'min_x' or 'max_x' values, there might appear a percentage % symbol, which indicates that the value of x is a percentage over the training data.

Feature extraction method	# Trees	Tree type	Error goal
Bächlin et al. (2009) [16]	1024	min_3	5
Mazilu et al.'s (2012) [79]	1024	min_5%	10
Tripoliti et al.'s (2013) [141]	1024	min_3	5
Rodríguez et al.'s (2016) [31]	1024	min_10%	5

or $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(\gamma(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)\right)$, which are equivalent notations. Therefore, the SVM-RBF was implemented as

$$\textbf{maximise:} \quad W(\boldsymbol{\nu}) = \sum_{i=1}^{n} \nu_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \nu_i \nu_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad , \tag{5.17}$$

$$\textbf{subject to:} \quad \sum_{i=1}^{n} \nu_i y_i = 0, \text{ and } 0 \leq \nu_i \leq \lambda \; \forall i \quad ,$$

where ν_i are the Lagrange multiplers and λ is a scalar value to trade-off between the empirical error and the margin.

The algorithm's hyperparameters' were tuned independently for each feature extraction by performing an exhaustive exploration considering the following values for each parameter:

1. λ: 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2} and 10^{3}.

2. γ: 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2} and 10^{3}.

The implemented hyperparameters' configuration implemented for this algorithm are shown in Table 5.6.

5.5.2 Shallow ML training and evaluation

The shallow ML algorithms implemented were tuned by performing an exhaustive hyperparameters exploration on the configurations, as discussed in Subsection 5.5.1.

Table 5.6: SVM-RBF configurations. Columns naming description: '# SV' → number of support vectors composing the model.

Feature extraction method	λ	γ	# SV
Bächlin et al. (2009) [16]	10^3	10^3	18873
Mazilu et al.'s (2012) [79]	10^2	10^{-3}	19707
Tripoliti et al.'s (2013) [141]	10^3	1	17258
Rodríguez et al.'s (2016) [31]	10	10^-1	15947

Concretely, the training strategy implemented the leave-one-patient-out technique over the features extracted from the training data, which, as detailed in Section 3.2, were composed by 17 PD patients' data.

Next, each algorithm was retrained on the entire features training data using the optimal hyperparameters identified from the leave-one-patient-out process.

Finally, these models were tested on the features corresponding to the same testing data employed for the evaluating the DL approaches. This strategy was motivated by the risk of assessing a model on only 4 patients' data. Thus, if these patients, which were randomly selected, implied a simplification of the FOG detection task, this simplification would affect all topologies evaluated.

Chapter 6

Results and discussion

6.1 Comparison among DL approaches

This section presents and discusses the results obtained. These results are structured as: I) training performances and justification of the data representation adopted; II) performance of the 1D-ConvNet; III) performance of the 1D-ConvLSTM; IV) performance of the 1D-ConvGRU; V) and discussion of the DL approaches' performance.

6.1.1 Data representation

The data representation strategy adopted was the spectral window stacking. This decision was performed according to the training performance of the 1D-ConvNet, our first approach, on diverse representation strategies. This subsection presents the training results from the spectral window stacking, which is described in Subsection 3.4.2, compared to training the 1D-ConvNet model using two similar temporal representations. Therefore, these results are a reproduction of the initial experiments endeavoured to select the representation strategy, which was undertaken using only the training data. Concretely, the representation strategies hereafter compared are:

1. **Temporal single window.** In this strategy the models were trained using the window representation as described in Subsection 3.4.1; thus, the input size of a sample was of 128x9 (measures times signal channels). Hence that from this representation no window transition characteristics could be extracted by our feed-forward models.

2. **Temporal window stacking.** This strategy extended the temporal single window one, by concatenating windows in pairs on their temporal dimension. The sample dimensions were, thus, 256x9. Hence this representation allowed models to extract window transition information.

3. **Spectral window stacking.** This representation, which is described in Subsection 3.4.2, was implemented for training the presented DL-based approaches.

As can be seen in Table 6.1, the top-3 model trained using spectral window stacking strategy achieved bets performance regardless presents the top-5 results from performing the training hyperparameters exploration using each of these 3 representations.

Table 6.1: Top-3 training models' results from the representation strategies announced for comparison in Subsection 6.1.1, sorted according to the stopping criteria described in Subsection 5.2.6.

Data	Training-train			Training-validation		
representation	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
Temporal	91.1	90.3	90.3	82.9	90.2	82.2
single	91.1	95.2	90.0	82.7	88.3	82.9
window	88.5	94.9	86.9	80.0	90.9	79.4
Temporal	70.4	87.16	77.82	75.3	69.32	87.29
window	68.4	82.8	63.4	72.8	77.1	72.0
stacking	71.5	86.56	63.79	68.1	78.37	65.96
Spectral	93.9	**94.5**	**93.5**	87.9	**92.6**	**88.7**
window	93.2	94.8	92.7	88.3	91.2	89.5
stacking	93.3	91.9	93.0	88.7	90.5	89.8

From Table 6.1, it can be observed that the best model was trained using the spectral window stacking representation strategy. The training process included an hyperparameters exploration phase, which concluded to lowering the regularisation for models trained on the temporal single window representation. Concretely, the regularisation strategy selected was equivalent to the one employed for training the recurrent models, as described in Subsection 5.2.6. However, models trained using the temporal window stacking representation achieved higher training results when using the same regularisation configuration than the one set for other feed-forward DL models trained; thus, the same strategy that when using the spectral window stacking strategy.

The scoring metric (i.e. same as employed for the early stopping strategy) for each of the representations' top models in Table 6.1 were: 86.11 for the temporal single

window, 77.79 for the temporal window stacking and 90.2 for the spectral window stacking.

From the training performances, it can be observed that the temporal window stacking is worsening the performance of the models compared to its former temporal single window approach. This fact was due to the labelling strategy plus the weights sharing properties of ConvNets. Concretely, each label is labelled according to half of its data in this approach, while the other half may be from a different class, which should provide information from trigger events that lead to the current state. However, ConvNets extract features in a spatial invariant manner within an input sample; thus, being unable to distinguish current from previous patterns when the data are scarce. Consequently, this representation may confuse the models rather than informing them.

Furthermore, Table 6.2 presents the test performances of these strategies; thus, will serve to confirm the generalisation capacity of each approach.

Table 6.2: Top training models' test results from the representation strategies announced for comparison in Subsection 6.1.1.

Representation	Accuracy	Sensitivity	Specificity	GM
Temporal single window	82.3	82.0	83.8	82.6
Temporal window stacking	58.1	66.1	62.7	57.3
Spectral window stacking	89.0	**91.9**	**89.5**	**90.6**

Table 6.2 confirms the robustness of our approach compared to simple temporal representations since the spectral windowing stacking was the only representation that allowed the trained models to generalise to the test data. Furthermore, it can be observed that whereas in Table 6.1 the temporal single window strategy reached acceptable train performances, its generalisation capacity is significantly lower than our approach's one. This fact may be related to the regularisation configurations in models trained for each approach; however, repeating the experiments using the same

regularisation strategy that for our approach confirmed that the hyperparameters selection was correct since these last models achieved test performances about 20% lower for the GM of the test sensitivity and test specificity.

6.1.2 1D-ConvNet

Training results. The approach selected from the trained 1D-ConvNets trained for 456 epochs, and as can be observed from Table 6.3, its selection metric (i.e. same as employed for the early stopping strategy) achieved 90.2%. Furthermore, from other top models shown in the table, it can be appreciated that the results reported by our approach were consistent with other similar models trained.

Table 6.3: 1D-ConvNet top-5 models' train performance. Columns naming description: 'GM' → GM of the test sensitivity and test specificity.

Training-train				Training-validation			
Accuracy	Sensitivity	Specificity	GM	Accuracy	Sensitivity	Specificity	GM
93.9	**94.5**	**93.5**	**93.9**	87.9	**92.6**	**88.7**	**90.2**
93.2	94.8	92.7	93.7	88.3	91.2	89.5	90.0
93.3	91.9	93.0	92.4	88.7	90.5	89.8	89.8
92.9	94.8	92.1	93.4	87.2	91.9	87.9	89.4
92.3	95.1	91.5	93.2	86.6	92.9	86.6	89.4

Test results. The test performances for the models included in Table 6.3 are presented in Table 6.4. These results corroborate that the training process has allowed our approach to generalise its learnt representations for automatic FOG detection successfully. Furthermore, it can be noted that these results already outperformed the state-of-the-art for automatic FOG detection using patient-independent settings.

6.1.3 1D-ConvLSTM

Train performance. The approach selected from the trained 1D-ConvLSTM, whose training performances are presented in Table 6.5, trained for only 379 epochs. Note that, usually, models implementing LSTM layers are slower converging than feed-forward models and those using GRUs instead [27, 28, 53]. Not surprisingly, in our experiments, indeed, most of the 1D-ConvLSTMs training executions lasted for at least 1000 of epochs before converging. However, detailed results are only reported

Table 6.4: 1D-ConvNet top-5 models' test performance. Columns naming description: 'GM' → GM of the test sensitivity and test specificity.

Accuracy	Sensitivity	Specificity	GM
89.0	**91.9**	**89.5**	**90.6**
89.5	88.6	91.4	89.6
86.1	90.5	87.1	88.5
90.2	88.8	91.9	90.0
88.5	89.3	90.2	89.3

for the selected models, which in this case fulfilled the stopping criteria while being ranked as the top-1 training model in few epochs.

Table 6.5: 1D-ConvLSTM top-5 models' train performance. Columns naming description: 'GM' → GM of the test sensitivity and test specificity.

Training-train				Training-validation			
Accuracy	Sensitivity	Specificity	GM	Accuracy	Sensitivity	Specificity	GM
91.6	**87.6**	**91.9**	**89.3**	88.0	**91.6**	**89.4**	**90.0**
86.9	90.6	85.8	87.8	83.6	92.4	83.7	87.5
86.6	90.6	85.2	87.4	84.2	93.9	84.1	88.4
88.0	88.0	87.5	87.3	84.4	91.3	84.6	87.5
90.5	87.0	90.5	88.1	83.9	90.8	85.1	87.3

Test performance. The test performances for the models included in Table 6.5 are presented in Table 6.6.

6.1.4 1D-ConvGRU

Train performance. The top-5 training 1D-ConvGRUs' results are presented in Table 6.7. From it, it can be observed that implementing GRUs instead of LSTMs for extending our feed-forward approach, also permitted to train promising models. The approach selected trained for 1147 epochs.

Test performance. The test performances for the models included in Table 6.7 are presented in Table 6.8.

Table 6.6: 1D-ConvLSTM top-5 models' test error. Columns naming description: 'GM' → GM of the test sensitivity and test specificity.

Accuracy	Sensitivity	Specificity	GM
87.8	**88.1**	**89.1**	**88.4**
84.6	94.3	83.6	88.8
82.8	95.5	81.4	88.1
87.0	92.0	86.8	89.4
87.4	91.7	87.1	89.4

Table 6.7: 1D-ConvGRU top-5 models' train performance. Columns naming description: 'GM' → GM of the test sensitivity and test specificity.

Training-train				Training-validation			
Accuracy	Sensitivity	Specificity	GM	Accuracy	Sensitivity	Specificity	GM
92.3	**93.0**	**91.7**	**92.2**	89.3	**92.9**	**90.5**	**91.3**
91.2	92.5	91.0	91.6	88.1	92.0	89.5	90.3
89.0	92.8	88.5	90.5	86.8	94.1	86.9	90.1
92.2	88.4	92.0	89.8	88.3	91.6	89.4	90.2
92.6	88.0	92.8	90.1	88.6	89.9	90.4	89.8

Table 6.8: 1D-ConvGRU top-5 models' test error. Columns naming description: 'GM' → GM of the test sensitivity and test specificity.

Accuracy	Sensitivity	Specificity	GM
85.4	**94.8**	**84.7**	**89.5**
87.6	93.2	87.9	90.4
86.5	94.0	86.4	90.0
85.3	87.1	86.2	86.5
86.2	89.8	87.2	88.2

6.1.5 Discussion

On the overall, all approaches demonstrated being capable for being properly trained. However, from comparing Table 6.5 to Table 6.3 and Table 6.7 it can be noted that models implementing LSTM layers were less promising than others. The reason for it was that the models' architectures were minimised regarding the number of

parameters on the feed-forward and later adopted by the recurrent extensions. This issue, as discussed in Subsection 5.2.6, moreover, lead to having to remove part of the regularisation strategies to train the recurrent extensions without altering the models' architectures.

Table 6.9 summarises the results reported from our DL experiments. From it, it can be concluded that our approach for FOG detection, which was composed by spectral window stacking data representation and 1D-ConvNets, was able to capture temporal dependencies and achieve higher performances than its recurrent extensions. However, from observing the number of parameters in these approaches, it can be noted that the implementations of the recurrent extensions contain, nearly, half of the parameters in their predecessor.

The results of these experiments indicate that our feed-forward approach could be missing some temporal information since its results are nearly equivalent to the 1D-ConvGRU, who is composed of less than half the 1D-ConvNets number of parameters. The 1D-ConvGRU presents, however, other less attractive characteristics

Table 6.9: DL approaches results. Columns naming description: '# Param' → number of model's parameters as shown in Table 4.1; '# Epoch' → number of epochs that lasted the model's training process; and 'GM' → GM of the test sensitivity and test specificity.

	Training configuration		Testing performance			
Model	# Param	# Epoch	Accuracy	Sensitivity	Specificity	GM
1D-ConvNet	37121	456	89.0	91.9	89.5	**90.6**
1D-ConvLSTM	17857	379	87.8	88.1	89.1	88.4
1D-ConvGRU	14209	1147	85.4	94.8	84.7	89.5

6.2 Comparison among shallow ML

Table 6.10 presents the results obtained from reproducing the state-of-the-art approaches and training shallow ML algorithms using leave-one-patient-out cross-validation.

Feature extraction results. Regarding the feature extraction approaches composing the state-of-the-art for automatic FOG detection, on the one hand, the best results from the approaches reproduced were achieved by the approach proposed by Rodríguez et al. (2016) [31]. Concretely, 3 out of the 6 trained algorithms on this features surpassed the 80% for the GM between the test sensitivity and test specificity.

Table 6.10: Comparative table of the ML approaches' test results. Columns naming description: 'GM' → GM of the test sensitivity and test specificity.

Features	Algorithm	Accuracy	Sensitivity	Specificity	GM
Bächlin et al. (2009)	Tree bagging	82.90	30.00	91.89	52.50
	AdaBoost	80.56	36.25	88.10	56.51
	LogitBoost	80.46	34.64	88.25	55.29
	RUSBoost	66.05	96.96	60.80	76.78
	RobutBoost	79.60	36.96	86.85	56.66
	SVM-RBF	62.57	94.46	57.15	73.48
Mazilu et al. (2012)	Tree bagging	83.49	64.29	86.76	74.68
	AdaBoost	81.88	59.82	85.64	71.57
	LogitBoost	80.28	68.75	82.24	75.19
	RUSBoost	66.52	98.04	61.16	77.43
	RobutBoost	77.32	62.86	79.78	70.81
	SVM-RBF	64.81	97.50	59.25	76.00
Tripoliti et al. (2013)	Tree bagging	83.57	12.68	95.63	34.82
	AdaBoost	82.01	18.75	92.77	41.71
	LogitBoost	82.92	16.79	94.17	39.76
	RUSBoost	59.62	96.07	53.42	71.64
	RobutBoost	78.61	37.32	85.64	56.53
	SVM-RBF	59.77	97.50	53.36	72.13
Rodríguez et al. (2016)	Tree bagging	88.99	51.89	95.35	70.34
	AdaBoost	85.36	30.16	94.83	53.48
	LogitBoost	85.18	77.20	86.55	81.74
	RUSBoost	72.04	96.95	67.76	81.05
	RobutBoost	85.52	59.25	90.02	73.03
	SVM-RBF	75.19	96.23	71.58	**83.00**

On the other hand, the algorithms trained on the features proposed by Tripoliti et al. (2013) [141] achieved the poorest results among all.

From Table 6.10 it can be observed that Mazilu et al.'s (2012) [79] demonstrated being the most robust representation since all algorithms trained using this features achieved performances higher than 70% for the GM between the test sensitivity and test specificity.

Shallow ML results. Regarding the results from the shallow ML algorithms selected, it can be observed that, as already discussed in Subsection 5.5.1, all algorithms achieved performances higher than 70% for the GM between the test sensitivity and test specificity, at least for one feature extraction approach. Thus, confirming the suitability of these algorithms for dealing with binary classification tasks.

The algorithm which obtained the best performance was the SVM when combined with Rodríguez et al.'s (2016) [31] features. Furthermore, Rodríguez et al.'s (2016) [31] and Rodríguez et al.'s (2017) [110], both indicate their preference for SVMs for FOG detection; thus, these results agree with their ML algorithm selection.

Comparison between our work and the state-of-the-art in the literature. The state-of-the-art for automatic FOG detection in the literature using patient independent conditions (i.e. patients in the test data and train are different people) were mentioned in Section 1.5. Table 6.11 includes this results together with a summary of our reproduction of these approaches.

Table 6.11: Experiments' results comparison. Columns naming description: 'GM' → GM of the test sensitivity and test specificity.

	Data representation	Model	GM
Literature	Bächlin et al. (2009) [16]	Thresholds	77.23
	Mazilu et al. (2012) [79]	Random forests	79.49
	Tripoliti et al. (2013) [141]	Random forests	84.07
	Rodríguez et al. (2017) [110]	SVM-RBF	76.8
Reproduction	Bächlin et al. (2009)	RUSBoost	76.78
	Mazilu et al. (2012)	RUSBoost	77.43
	Tripoliti et al. (2013)	SVM-RBF	72.13
	Rodríguez et al. (2016)	SVM-RBF	83.00

From Table 6.11 it can be observed that some differences appear between our reproduction and the literature's results. However, these differences can be justified by the following observations in the data and evaluation strategies employed for each approach:

- Bächlin et al. (2009) [16] employed an evaluation strategy with 50% of window tolerance. Moreover, they employed the Daphnet dataset. Thus, as discussed in Section 1.5, their results were overestimated compared to our single sample evaluation strategy.

- Mazilu et al. (2012) [79] employed the Daphnet dataset, which also overestimated the performance of their approach.
- Tripoliti et al. (2013) [141] employed a simple data collection protocol similar to for the Daphnet dataset. Moreover, their results are only computed from the performance achieved by cross-validating over 5 PD patients, which will poorly represent the actual performance of the method.
- Rodríguez et al. (2017) [110] employed an episodic evaluation strategy, which lowers the methods' performance due to the easiness of detecting extended periods of similar data (e.g. straight walking periods, sitting and standing).

Therefore, on the overall, these results resemble the performances achieved by the original authors themselves when being accurately reproduced and applied to our data. Furthermore, all performance disagreements can be argued and justified; thus, it can be stated that results presented in Table 6.10 are equivalent to applying the state-of-the-art approaches to our data.

6.3 Comparison between DL and shallow ML approaches

Table 6.12 summarises the results reported from our DL experiments together with the best results from Table 6.10 to compare the results from our approach to state-of-the-art reproduction.

From it, it can be concluded that our approach for FOG detection, which was composed by spectral window stacking data representation and 1D-ConvNets, was able to capture temporal dependencies and achieve higher performances than its recurrent extensions. However, from observing the number of parameters in these approaches, it can be noted that the implementations of the recurrent extensions contain, nearly, half of the parameters in their predecessor.

The results of these experiments confirm that our DL-based approaches were able to outperform the state-of-the-art methodologies for automatic FOG.

Table 6.12: Experiments' results comparison. Columns naming description: 'GM' → GM of the test sensitivity and test specificity.

Data representation	Model	Accuracy	Sensitivity	Specificity	GM
spectral window stacking	1D-ConvNet	89.0	91.9	89.5	**90.6**
spectral window stacking	1D-ConvLSTM	87.8	88.1	89.1	88.4
spectral window stacking	1D-ConvGRU	85.4	94.8	84.7	89.5
Bächlin et al. (2009)	RUSBoost	66.05	96.96	60.80	76.78
Mazilu et al. (2012)	RUSBoost	66.52	98.04	61.16	77.43
Tripoliti et al. (2013)	SVM-RBF	59.77	97.50	53.36	72.13
Rodríguez et al. (2016)	SVM-RBF	75.19	96.23	71.58	83.00

Chapter 7

Conclusions

This thesis is, to the best of our knowledge, the first study to present a method for FOG detection on uncontrolled environments based on DL models. Furthermore, our approach was able to solve the committed task, while outperforming the state-of-the-art methodologies for automatic FOG detection in the literature.

The DL models presented were a feed-forward 1D-ConvNet, which achieved performances above 90% for the GM between test sensitivity and specificity, and two recurrent extensions from it, the 1D-ConvLSTM and the 1D-ConvGRU.

For comparison purposes to our approaches, the methodologies composing the state-of-the-art for automatic FOG detection were accurately reproduced on our data, generating 4 different sets of features. These features were used for training powerful binary classification shallow ML algorithms. The results reported by our reproduction of the state-of-the-art approaches were consistent with the results reported by the original authors of these feature extraction methods; thus, allowing our DL approaches to be compared to the newly trained models.

DL approaches outperformed all shallow ML models, demonstrating its superior generalisation and representation capacity for detecting FOG events from wearable inertial sensors' data recorded from PD patients in uncontrolled environments while performing ADL.

As described in Section 1.1, our approaches, which significantly improve the current performance of existing automatic FOG detection methods, may serve to the following:

- Improve the medical record about FOG evolution in PD patients to allow clinicians to dispose of accurate and objective symptomatic records of their patients. These records could serve to understand FOG in PD patients better and to improve the clinical control of the disease evolution.

- Allow clinicians to objectively evaluate the effect of drugs (e.g. during clinical trials) over the symptom's characteristics from automatically gathered indicators.

Future work. Interesting extensions of our work are:

- To perform an hyperparameter and architecture exploration focused for the 1D-ConvGRU, whose performances, presented in Table 6.9, indicate that this is a promising manner to improve our results further.

- Implementing and evaluating our approaches on real-time. Currently, at the CETpD, the authors are working with a real-time system for automatic FOG detection, which implements an SVM-RBF on a subset of the features described by Rodríguez et al.'s (2017) [110]. This model occupies 26 kilobytes (KB) (i.e. SVM with 27 features and 250 support vectors) in memory, while our current DL approaches occupy the following:

  - 1D-ConvNet: 37121 parameters, 145 KB.
  - 1D-ConvLSTM: 17857 parameters, 69 KB.
  - 1D-ConvGRU: 14209 parameters, 55 KB.

  Our device implements a microcontroller (μC) with architecture ARM-cortex M4, concretely, the STM32F415RG [5]. This μC disposes of 500 KB free memory available for the FOG detection model, which would allow the storage of our DL approach.

Publications. From the work undertaken within this study, the author has already submitted a conference paper for the 14th International Work-Conference on Artificial Neural Networks titeled 'Deep learning for detecting freezing of gait episodes in Parkinson's disease based on accelerometers'. This paper has been accepted for oral presentation on the 'Human activity recognition for health and well-being applications' special session. Furthermore, the results presented in this thesis will be employed for writing a journal paper.

Bibliography

[1] Centro Médico Teknon. http://www.teknon.es/es/unidad-parkinson-trastornos-movimiento. Accessed: 2017-04-11.

[2] Hospital Sant Antoni Abat. http://www.csg.cat/nosaltres/els-nostres-centres/hospital-sant-antoni-abat/. Accessed: 2017-04-11.

[3] Improving Quality of Life with an Automatic Control System (MASPARK). http://futur.upc.edu/15557508. Accessed: 2017-04-11.

[4] J-BHI Special Issue on 'Deep Learning for Biomedical and Health Informatics', Journal: Journal of Biomedical and Health Informatics; Editor-in-Chief: Guang-Zhong, Yang. http://jbhi.embs.org/special-issues/deep-learning-for-biomedical-and-health-informatics/. Accessed: 2017-04-11.

[5] micro controller stm32f415rg. http://www.st.com/en/microcontrollers/stm32f415rg.html. Accessed: 2017-04-20.

[6] PD patient self care blog. http://www.riggare.se/1-vs-8765/. Accessed: 2017-04-06.

[7] Remote and Autonomous Management of Parkinson's Disease (REMPARK). http://www.rempark.eu/. Accessed: 2017-04-11.

[8] source code github. https://github.com/juliacamps/FOG. Accessed: 2017-04-22.

[9] Technical Research Centre for Dependency Care and Autonomous Living (CETpD). http://www.epsevg.upc.edu/cetpd/index.php. Accessed: 2017-04-11.

[10] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[11] Alison Abbott. Levodopa: the story so far. *Nature*, 466(7310):S6–S7, 2010.

[12] Claas Ahlrichs, Albert Samà, Michael Lawo, Joan Cabestany, Daniel Rodríguez-Martín, Carlos Pérez-López, Dean Sweeney, Leo R Quinlan, Gearòid Ò Laighin, Timothy Counihan, et al. Detecting freezing of gait with a tri-axial accelerometer in parkinsons disease patients. *Medical & biological engineering & computing*, 54(1):223–233, 2016.

[13] MM Al Rahhal, Yakoub Bazi, Haikel AlHichri, Naif Alajlan, Farid Melgani, and RR Yager. Deep learning approach for active classification of electrocardiogram signals. *Information Sciences*, 345:340–354, 2016.

[14] Nils-Erik Andén, Allan Rubenson, Kjell Fuxe, and Tomas Hökfelt. Evidence for dopamine receptor stimulation by apomorphine. *Journal of Pharmacy and Pharmacology*, 19(9):627–629, 1967.

[15] Pablo Arias and Javier Cudeiro. Effect of rhythmic auditory stimulation on gait in parkinsonian patients with and without freezing of gait. *PloS one*, 5(3):e9675, 2010.

[16] Marc Bächlin, Jeffrey M Hausdorff, Daniel Roggen, Nir Giladi, Meir Plotnik, and Gerhard Tröster. Online detection of freezing of gait in parkinson's disease patients: A performance characterization. In *Proceedings of the Fourth International Conference on Body Area Networks*, page 11. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[17] Marc Bachlin, Meir Plotnik, Daniel Roggen, Inbal Maidan, Jeffrey M Hausdorff, Nir Giladi, and Gerhard Troster. Wearable assistant for parkinsons disease patients with the freezing of gait symptom. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):436–446, 2010.

[18] Pouya Bashivan, Irina Rish, Mohammed Yeasin, and Noel Codella. Learning representations from eeg with deep recurrent-convolutional neural networks. *arXiv preprint arXiv:1511.06448*, 2015.

[19] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[20] Bastiaan R Bloem, Jeffrey M Hausdorff, Jasper E Visser, and Nir Giladi. Falls and freezing of gait in parkinson's disease: a review of two interconnected, episodic phenomena. *Movement Disorders*, 19(8):871–884, 2004.

[21] Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

[22] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[23] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

[24] Jianxu Chen, Lin Yang, Yizhe Zhang, Mark Alber, and Danny Z Chen. Combining fully convolutional and recurrent neural networks for 3d biomedical image segmentation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3036–3044. Curran Associates, Inc., 2016.

[25] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[26] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[27] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[28] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015.

[29] Wikimedia Commons. Main page — wikimedia commons, the free media repository, 2017. [Online; accessed 17-April-2017].

[30] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[31] Carlos Pérez-López Andreu Català Berta Mestre Sheila Alcaine Àngels Bayès Daniel Rodríguez-Martín, Albert Samà. *Comparison of Features, Window Sizes and Classifiers in Detecting Freezing of Gait in Patients with Parkinson's Disease Through a Waist-Worn Accelerometer*, volume 288 of *Frontiers in Artificial Intelligence and Applications*. 2016.

[32] Lonneke ML De Lau and Monique MB Breteler. Epidemiology of parkinson's disease. *The Lancet Neurology*, 5(6):525–535, 2006.

[33] Anna De Rosa, Alessandro Tessitore, Leonilda Bilo, Silvio Peluso, and Giuseppe De Michele. Infusion treatments and deep brain stimulation in parkinson's disease: The role of nursing. *Geriatric Nursing*, 37(6):434–439, 2016.

[34] Silvia Del Din, Alan Godfrey, Claudia Mazzà, Sue Lord, and Lynn Rochester. Free-living monitoring of parkinson's disease: Lessons from the field. *Movement Disorders*, 31(9):1293–1313, 2016.

[35] PareshK Doshi. Long-term surgical and hardware-related complications of deep brain stimulation. *Stereotactic and functional neurosurgery*, 89(2):89–95, 2011.

[36] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[37] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 64–72. Curran Associates, Inc., 2016.

[38] Weaver FM, Follett K, Stern M, and et al. Bilateral deep brain stimulation vs best medical therapy for patients with advanced parkinson disease: A randomized controlled trial. *JAMA*, 301(1):63–73, 2009.

[39] Susan H Fox. Non-dopaminergic treatments for motor control in parkinsons disease. *Drugs*, 73(13):1405–1415, 2013.

[40] JP Frankel, AJ Lees, PA Kempster, and GM Stern. Subcutaneous apomorphine in the treatment of parkinson's disease. *Journal of Neurology, Neurosurgery & Psychiatry*, 53(2):96–101, 1990.

[41] Yoav Freund. A more robust boosting algorithm. *arXiv preprint arXiv:0905.2138*, 2009.

[42] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[43] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[44] Anders Fytagoridis, Tomas Heard, Jennifer Samuelsson, Peter Zsigmond, Elena Jiltsova, Simon Skyrman, Thomas Skoglund, Terry Coyne, Peter Silburn, and Patric Blomstedt. Surgical replacement of implantable pulse generators in deep brain stimulation: adverse events and risk factors in a multicenter cohort. *Stereotactic and Functional Neurosurgery*, 94(4):235–239, 2016.

[45] Sonia Gandhi and Helene Plun-Favreau. Mutations and mechanism: how pink1 may contribute to risk of sporadic parkinsons disease. *Brain*, 140(1):2–5, 2017.

[46] N Giladi, MP McDermott, S Fahn, S Przedborski, J Jankovic, M Stern, C Tanner, Parkinson Study Group, et al. Freezing of gait in pd prospective assessment in the datatop cohort. *Neurology*, 56(12):1712–1721, 2001.

[47] N Giladi, H Shabtai, ES Simon, S Biran, J Tal, and AD Korczyn. Construction of freezing of gait questionnaire for patients with parkinsonism. *Parkinsonism & related disorders*, 6(3):165–170, 2000.

[48] N Giladi, TA Treves, ES Simon, H Shabtai, Y Orlov, B Kandinov, D Paleacu, and AD Korczyn. Freezing of gait in patients with advanced parkinson's disease. *Journal of neural transmission*, 108(1):53–61, 2001.

[49] Nir Giladi, Joseph Tal, Tali Azulay, Oliver Rascol, David J Brooks, Eldad Melamed, Wolfgang Oertel, Werner H Poewe, Fabrizio Stocchi, and Eduardo Tolosa. Validation of the freezing of gait questionnaire in patients with parkinson's disease. *Movement Disorders*, 24(5):655–661, 2009.

[50] Anamika Giri, Kin Y Mok, Iris Jansen, Manu Sharma, Christelle Tesson, Graziella Mangone, Suzanne Lesage, José M Bras, Joshua M Shulman, Una-Marie Sheerin, et al. Lack of evidence for a role of genetic variation in tmem230 in the risk for parkinson's disease in the caucasian population. *Neurobiology of aging*, 50:167–e11, 2017.

[51] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[52] Rafael C Eddins Gonzalez, Steven L Woods, Richard E Richard Eugene Rafael C Gonzalez, Richard E Woods, and Steven L Eddins. *Digital image processing using MATLAB*. Number 04; TA1637, G6. 2004.

[53] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[54] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[55] Hayit Greenspan, Bram van Ginneken, and Ronald M Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, 2016.

[56] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[57] Fredrik Gustafsson. Determining the initial states in forward-backward filtering. *IEEE Transactions on Signal Processing*, 44(4):988–992, 1996.

[58] Nils Y Hammerla, Shane Halloran, and Thomas Ploetz. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880*, 2016.

[59] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[60] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[61] Joseph Jankovic. Parkinsons disease: clinical features and diagnosis. *Journal of Neurology, Neurosurgery & Psychiatry*, 79(4):368–376, 2008.

[62] C Jenkinson, V Peto, R Fitzpatrick, R Greenhall, and N Hyman. Self-reported functioning and well-being in patients with parkinson's disease: comparison of the short-form health survey (sf-36) and the parkinson's disease questionnaire (pdq-39). *Age and ageing*, 24(6):505–509, 1995.

[63] Crispin Jenkinson, Carl Clarke, Richard Gray, Paul Hewitson, Natalie Ives, David Morley, Caroline Rick, Keith Wheatley, and Adrian Williams. Comparing results from long and short form versions of the parkinson's disease questionnaire in a longitudinal study. *Parkinsonism & Related Disorders*, 21(11):1312–1316, 2015.

[64] Lorraine V Kalia and Anthony E Lang. Parkinson disease in 2015: Evolving basic, pathological and clinical concepts in pd. *Nature reviews Neurology*, 2016.

[65] John W Kebabian and Donald B Calne. Multiple receptors for dopamine. *Nature*, 277(5692):93–96, 1979.

[66] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[67] Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj. Real-time patient-specific ecg classification by 1-d convolutional neural networks. *IEEE Transactions on Biomedical Engineering*, 63(3):664–675, 2016.

[68] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances*

in Neural Information Processing Systems 28, pages 3294–3302. Curran Associates, Inc., 2015.

[69] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[70] David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, Aaron Courville, et al. Zoneout: Regularizing rnns by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*, 2016.

[71] Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285, 2015.

[72] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[73] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[74] Andrew Lees. Alternatives to levodopa in the initial treatment of early parkinson's disease. *Drugs & Aging*, 22(9):731–740, 2005.

[75] Walter Maetzler, Josefa Domingos, Karin Srulijes, Joaquim J Ferreira, and Bastiaan R Bloem. Quantitative wearable sensors for objective assessment of parkinson's disease. *Movement Disorders*, 28(12):1628–1637, 2013.

[76] Jaroslaw Marusiak, Katarzyna Kisiel-Sajewicz, Anna Jaskólska, and Artur Jaskólski. Higher muscle passive stiffness in parkinson's disease patients than in controls measured by myotonometry. *Archives of physical medicine and rehabilitation*, 91(5):800–802, 2010.

[77] Sinziana Mazilu, Ulf Blanke, Alberto Calatroni, Eran Gazit, Jeffrey M. Hausdorff, and Gerhard Trster. The role of wrist-mounted inertial sensors in detecting gait freeze episodes in parkinsons disease. *Pervasive and Mobile Computing*, 33:1 – 16, 2016.

[78] Sinziana Mazilu, Alberto Calatroni, Eran Gazit, Daniel Roggen, Jeffrey M Hausdorff, and Gerhard Tröster. Feature learning for detection and prediction of freezing of gait in parkinsons disease. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 144–158. Springer, 2013.

[79] Sinziana Mazilu, Michael Hardegger, Zack Zhu, Daniel Roggen, Gerhard Troster, Meir Plotnik, and Jeffrey M Hausdorff. Online detection of freezing of gait with smartphones and machine learning techniques. In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2012 6th International Conference on*, pages 123–130. IEEE, 2012.

[80] Orna Moore, Chava Peretz, and Nir Giladi. Freezing of gait affects quality of life of peoples with parkinson's disease beyond its relationships with mobility and gait. *Movement disorders*, 22(15):2192–2195, 2007.

[81] Steven T Moore, Hamish G MacDougall, and William G Ondo. Ambulatory monitoring of freezing of gait in parkinson's disease. *Journal of neuroscience methods*, 167(2):340–348, 2008.

[82] Steven T Moore, Don A Yungher, Tiffany R Morris, Valentina Dilda, Hamish G MacDougall, James M Shine, Sharon L Naismith, and Simon JG Lewis. Autonomous identification of freezing of gait in parkinson's disease from lower-body segmental accelerometry. *Journal of neuroengineering and rehabilitation*, 10(1):19, 2013.

[83] Priya Ranjan Muduli, Rakesh Reddy Gunukula, and Anirban Mukherjee. A deep learning approach to fetal-ecg signal reconstruction. In *Communication (NCC), 2016 Twenty Second National Conference on*, pages 1–6. IEEE, 2016.

[84] Giovanna Mulas, Elena Espa, Sandro Fenu, Saturnino Spiga, Giovanni Cossu, Elisabetta Pillai, Ezio Carboni, Gabriella Simbula, Dragana Jadi, Fabrizio Angius, Stefano Spolitu, Barbara Batetta, Daniela Lecca, Andrea Giuffrida, and Anna R. Carta. Differential induction of dyskinesia and neuroinflammation by pulsatile versus continuous l-dopa delivery in the 6-ohda model of parkinson's disease. *Experimental Neurology*, 286:83 – 92, 2016.

[85] Le Nguyen Ngu Nguyen, Daniel Rodríguez-Martín, Andreu Català, Carlos Pérez-López, Albert Samà, and Andrea Cavallaro. Basketball activity recognition using wearable inertial measurement units. In *Proceedings of the XVI International Conference on Human Computer Interaction*, page 60. ACM, 2015.

[86] Alice Nieuwboer and Nir Giladi. Characterizing freezing of gait in parkinson's disease: models of an episodic phenomenon. *Movement Disorders*, 28(11):1509–1519, 2013.

[87] Alice Nieuwboer, Lynn Rochester, Talia Herman, Wim Vandenberghe, George Ehab Emil, Tom Thomaes, and Nir Giladi. Reliability of the new freezing of gait questionnaire: agreement between patients with parkinson's disease and their carers. *Gait & posture*, 30(4):459–463, 2009.

[88] Robert L Nussbaum and Christopher E Ellis. Alzheimer's disease and parkinson's disease. *New england journal of medicine*, 348(14):1356–1364, 2003.

[89] John G. Nutt, William R. Woodward, John P. Hammerstad, Julie H. Carter, and John L. Anderson. The onoff phenomenon in parkinson's disease. *New England Journal of Medicine*, 310(8):483–488, 1984. PMID: 6694694.

[90] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[91] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[92] World Health Organization. *Neurological disorders: public health challenges*. World Health Organization, 2006.

[93] Sean S OSullivan, Andrew H Evans, and Andrew J Lees. Dopamine dysregulation syndrome. *CNS drugs*, 23(2):157–170, 2009.

[94] Carlos Pérez-López, Albert Samà, Daniel Rodríguez-Martín, Andreu Català, Joan Cabestany, Juan Manuel Moreno-Arostegui, Eva de Mingo, and Alejandro Rodríguez-Molinero. Assessing motor fluctuations in parkinsons disease patients based on a single inertial sensor. *Sensors*, 16(12):2132, 2016.

[95] Joel S Perlmutter and Jonathan W Mink. Deep brain stimulation. *Annu. Rev. Neurosci.*, 29:229–257, 2006.

[96] Thomas G Pickering, William Gerin, and Amy R Schwartz. What is the white-coat effect and how should it be measured? *Blood pressure monitoring*, 7(6):293–300, 2002.

[97] Lionel Pigou, Aäron van den Oord, Sander Dieleman, Mieke Van Herreweghe, and Joni Dambre. Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video. *arXiv preprint arXiv:1506.01911*, 2015.

[98] Werner Poewe and Gregor K Wenning. Apomorphine: an underutilized therapy for parkinson's disease. *Movement disorders*, 15(5):789–794, 2000.

[99] Mihael H Polymeropoulos, Christian Lavedan, Elisabeth Leroy, Susan E Ide, Anindya Dehejia, Amalia Dutra, Brian Pike, Holly Root, Jeffrey Rubenstein, Rebecca Boyer, et al. Mutation in the α-synuclein gene identified in families with parkinson's disease. *science*, 276(5321):2045–2047, 1997.

[100] Bart Post, Maruschka P Merkus, Rob J De Haan, and Johannes D Speelman. Prognostic factors for the progression of parkinson's disease: a systematic review. *Movement disorders*, 22(13):1839–1851, 2007.

[101] Tamara Pringsheim, Nathalie Jette, Alexandra Frolkis, and Thomas DL Steeves. The prevalence of parkinson's disease: A systematic review and meta-analysis. *Movement disorders*, 29(13):1583–1590, 2014.

[102] Rangaraj M Rangayyan. *Biomedical signal analysis*, volume 33. John Wiley & Sons, 2015.

[103] Daniele Ravi, Charence Wong, Benny Lo, and Guang-Zhong Yang. Deep learning for human activity recognition: A resource efficient implementation on low-power devices. In *Wearable and Implantable Body Sensor Networks (BSN), 2016 IEEE 13th International Conference on*, pages 71–76. IEEE, 2016.

[104] D. Rav, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G. Z. Yang. Deep learning for health informatics. *IEEE Journal of Biomedical and Health Informatics*, 21(1):4–21, Jan 2017.

[105] Jorge-L Reyes-Ortiz, Luca Oneto, Albert Sama, Xavier Parra, and Davide Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171:754–767, 2016.

[106] Jorge-Luis Reyes-Ortiz, Luca Oneto, Alessandro Ghio, Albert Samá, Davide Anguita, and Xavier Parra. Human activity recognition on smartphones with awareness of basic activities and postural transitions. In *International Conference on Artificial Neural Networks*, pages 177–184. Springer, 2014.

[107] L Rivera-Calimlim, CA Dujovne, JP Morgan, L Lasagna, and JR Bianchine. L-dopa treatment failure: explanation and correction. *Br Med J*, 4(5727):93–94, 1970.

[108] Alexandra Rizos, Pablo Martinez-Martin, Suvankar Pal, Camille Carroll, Davide Martino, Rani Sophia, Cristian Falup-Pecurariu, Belinda Kessel, Anna Sauerbier, Anne Martin, et al. The first parkinson's disease pain questionnaire (king's pd pain quest)–an interim analysis of a multicentre study of the patient's perspective. *Parkinsonism & Related Disorders*, 22:e41, 2016.

[109] Daniel Rodríguez-Martín, Carlos Pérez-López, Albert Samà, Joan Cabestany, and Andreu Català. A wearable inertial measurement unit for long-term monitoring in the dependency care area. *Sensors*, 13(10):14079–14104, 2013.

[110] Daniel Rodríguez-Martín, Albert Samà, Carlos Pérez-López, Andreu Català, Joan M Moreno Arostegui, Joan Cabestany, Àngels Bayés, Sheila Alcaine, Berta Mestre, Anna Prats, et al. Home detection of freezing of gait using support vector machines through a single waist-worn triaxial accelerometer. *PloS one*, 12(2):e0171764, 2017.

[111] Daniel Rodriguez-Martin, Albert Samà, Carlos Perez-Lopez, Andreu Català, Joan Cabestany, and Alejandro Rodriguez-Molinero. Svm-based posture identification with a single waist-located triaxial accelerometer. *Expert Systems with Applications*, 40(18):7203–7211, 2013.

[112] Daniel Rodríguez-Martína, Albert Samàa, Carlos Pérez-Lópeza, Andreu Catalàa, Joan Cabestanya, Patrick Browneb, and Alejandro Rodríguez-Molinerod. Posture detection based on a waist-worn accelerometer: an application to improve freezing of gait detection in parkinsons disease patients. *Recent Advances in Ambient Assisted Living-Bridging Assistive Technologies, E-Health and Personalized Health Care*, 20:3, 2015.

[113] A Rodriguez-Molinero, D Perez-Martinez, A Samá, P Sanz, M Calopa, A Galvez, C Perez-Lopez, J Romagosa, and C Catala. Detection of gait parameters, bradykinesia, and falls in patients with parkinson's disease by using a unique triaxial accelerometer. *Movement Disorders*, 25:S646, 2010.

[114] Alejandro Rodríguez-Molinero, David Andrés Pérez-Martínez, César Gálvez-Barrón, J Hernández-Vara, JC Martínez-Castrillo, R Álvarez, O de Fabregues, A Samà, Carlos Pérez-López, J Romagosa, et al. Remote control of apomorphine infusion rate in parkinson's disease: Real-time dose variations according to the patients' motor state. a proof of concept. *Parkinsonism & related disorders*, 21(8):996, 2015.

[115] Alejandro Rodríguez-Molinero, Albert Samà, David A Pérez-Martínez, Carlos Pérez López, Jaume Romagosa, Àngels Bayés, Pilar Sanz, Matilde Calopa, César Gálvez-Barrón, Eva de Mingo, et al. Validation of a portable device for mapping motor and gait disturbances in parkinsons disease. *JMIR mHealth and uHealth*, 3(1):e9, 2015.

[116] M. C. Rodriguez-Oroz, J. A. Obeso, A. E. Lang, J.-L. Houeto, P. Pollak, S. Rehncrona, J. Kulisevsky, A. Albanese, J. Volkmann, M. I. Hariz, N. P. Quinn, J. D. Speelman, J. Guridi, I. Zamarbide, A. Gironell, J. Molet, B. Pascual-Sedano, B. Pidoux, A. M. Bonnet, Y. Agid, J. Xie, A.-L. Benabid, A. M. Lozano, J. Saint-Cyr, L. Romito, M. F. Contarino, M. Scerrati, V. Fraix, and N. Van Blercom. Bilateral deep brain stimulation in parkinson's disease: a multicentre study with 4 years follow-up. *Brain*, 128(10):2240, 2005.

[117] Manuela Rosa, Emma Scelzo, Marco Locatelli, Giorgio Carrabba, Vincenzo Levi, Mattia Arlotti, Sergio Barbieri, Paolo Rampini, and Alberto Priori. Risk of infection after local field potential recording from externalized deep brain stimulation leads in parkinson's disease. *World Neurosurgery*, 97:64–69, 2017.

[118] G Webster Ross, Helen Petrovitch, Robert D Abbott, James Nelson, William Markesbery, Daron Davis, John Hardman, Lenore Launer, Kamal Masaki, Caroline M Tanner, et al. Parkinsonian signs and substantia nigra neuron density in decendents elders without pd. *Annals of neurology*, 56(4):532–539, 2004.

[119] JI Sage, L Schuh, RE Heikkila, and RC Duvoisin. Continuous duodenal infusions of levodopa: plasma concentrations and motor fluctuations in parkinson's disease. *Clinical neuropharmacology*, 11(1):36–44, 1988.

[120] A Samà, C Perez-Lopez, J Romagosa, D Rodriguez-Martin, A Catala, J Cabestany, DA Perez-Martinez, and A Rodriguez-Molinero. Dyskinesia and motor state detection in parkinson's disease patients with a single movement sensor. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 1194–1197. IEEE, 2012.

[121] Albert Sama, Cecilio Angulo, Diego Pardo, Andreu Català, and Joan Cabestany. Analyzing human gait and posture by combining feature selection and kernel methods. *Neurocomputing*, 74(16):2665–2674, 2011.

[122] Albert Sama and Andreu Catala. Extracting gait spatiotemporal properties from parkinson's disease patients.

[123] Albert Sama, Diego E Pardo-Ayala, Joan Cabestany, and Alejandro Rodríguez-Molinero. Time series analysis of inertial-body signals for the extraction of dynamic properties from human gait. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–5. IEEE, 2010.

[124] Albert Samà Monsonís, Carlos Pérez López, Daniel Manuel Rodríguez Martín, Joan Cabestany Moncusí, Juan Manuel Moreno Aróstegui, and Alejandro Rodríguez Molinero. A heterogeneous database for movement knowledge extraction in parkinson's disease. In *ESANN 2013 proceedings: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning: Bruges (Belgium), 24-26 April 2013*, pages 413–418, 2013.

[125] JD Schaafsma, Y Balash, T Gurevich, AL Bartels, Jeffrey M Hausdorff, and N Giladi. Characterization of freezing of gait subtypes and the response of each to levodopa in parkinson's disease. *European Journal of Neurology*, 10(4):391–398, 2003.

[126] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: Improving classification performance when training data is skewed. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[127] Hava T Siegelmann and Eduardo D Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77–80, 1991.

[128] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[129] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[130] Heung-Il Suk, Chong-Yaw Wee, Seong-Whan Lee, and Dinggang Shen. State-space model with deep learning for functional dynamics estimation in resting-state fmri. *NeuroImage*, 129:292–307, 2016.

[131] Sigurlaug Sveinbjornsdottir. The clinical symptoms of parkinson's disease. *Journal of Neurochemistry*, 139:318–324, 2016.

[132] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[133] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[134] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[135] Caroline M Tanner and Samuel M Goldman. Epidemiology of parkinson's disease. *Neurologic clinics*, 14(2):317–335, 1996.

[136] Carlos Téllez, M Leonor Bustamante, Pablo Toro, and Pablo Venegas. Addiction to apomorphine: a clinical case-centred discussion. *Addiction*, 101(11):1662–1665, 2006.

[137] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2), 2012.

[138] Hiroki TODA, Hidemoto SAIKI, Namiko NISHIDA, and Koichi IWASAKI. Update on deep brain stimulation for dyskinesia and dystonia: A literature review. *Neurologia medico-chirurgica*, 56(5):236–248, 2016.

[139] Eduardo Tolosa, Maria J Martí, Francesc Valldeoriola, and José L Molinuevo. History of levodopa and dopamine agonists in parkinson's disease treatment. *Neurology*, 50(6 Suppl 6):S2–S10, 1998.

[140] Claire L Tomlinson, Rebecca Stowe, Smitaa Patel, Caroline Rick, Richard Gray, and Carl E Clarke. Systematic review of levodopa dose equivalency reporting in parkinson's disease. *Movement disorders*, 25(15):2649–2653, 2010.

[141] Evanthia E Tripoliti, Alexandros T Tzallas, Markos G Tsipouras, George Rigas, Panagiota Bougia, Michael Leontiou, Spiros Konitsiotis, Maria Chondrogiorgi, Sofia Tsouli, and Dimitrios I Fotiadis. Automatic detection of freezing of gait events in patients with parkinson's disease. *Computer methods and programs in biomedicine*, 110(1):12–26, 2013.

[142] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4790–4798. Curran Associates, Inc., 2016.

[143] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence - video to text. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[144] Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. A primer on kernel methods. *Kernel Methods in Computational Biology*, pages 35–70, 2004.

[145] Jens Volkmann, Alberto Albanese, Angelo Antonini, K Ray Chaudhuri, Carl E Clarke, Rob MA de Bie, Günther Deuschl, Karla Eggert, Jean-Luc Houeto, Jaime Kulisevsky, et al. Selecting deep brain stimulation or infusion therapies in advanced parkinsons disease: an evidence-based review. *Journal of neurology*, 260(11):2701–2714, 2013.

[146] Jason E Weston. Dialog-based language learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 829–837. Curran Associates, Inc., 2016.

[147] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5, 2015.

[148] William R Young, Lauren Shreve, Emma Jane Quinn, Cathy Craig, and Helen Bronte-Stewart. Auditory cueing in parkinson's patients with freezing of gait. what matters most: Action-relevance or cue-continuity? *Neuropsychologia*, 87:54–62, 2016.

[149] Heidemarie Zach, Arno M Janssen, Anke H Snijders, Arnaud Delval, Murielle U Ferraye, Eduard Auff, Vivian Weerdesteyn, Bastiaan R Bloem, and Jorik Nonnekes. Identifying freezing of gait in parkinson's disease during freezing provoking tasks using waist-mounted accelerometry. *Parkinsonism & related disorders*, 21(11):1362–1366, 2015.

[150] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.