UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MSc IN SIGNAL THEORY AND COMMUNICATIONS - SIGNAL PROCESSING AND MACHINE LEARNING FOR BIG DATA

MASTER THESIS

DESIGN AND IMPLEMENTATION OF DEEP LEARNING MODELS FOR TIME SERIES FORECASTING IN PATHOLOGICAL TREMOR SIGNALS

ALBERTO JOSÉ BELTRÁN CARRERO

2023

UNIVERSIDAD POLITÉCNICA DE MADRID

MASTER THESIS

Design and Implementation of Deep Learning Models for Time Series Forecasting in Pathological Tremor Signals

---

Author:
Alberto José BELTRÁN CARRERO

Tutor:
Dr. Alejandro PASCUAL-VALDUNCIEL

Co-tutor:
Dr. Álvaro GUTIÉRREZ MARTIN

Degree:

MSc in Signal Theory and Communications
Signal Processing and Machine Learning for Big Data Track

by

Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT)

2023

ABSTRACT

Essential Tremor (ET) and Parkinson's disease (PD) are the two most common involuntary movement disorders characterized by pathological tremor. Patients who suffer from these conditions might be unable to perform even the simplest daily living activities, leading to disabilities and poor quality of living. Today, most of the state-of-the-art treatments consist in surgery and pharmacological planning, however, these approaches have several inconveniences including limited eligibility of patients, highly variable efficacy and side-effects. In recent years, some studies have proposed novel tremor management techniques based on minimum invasive electrical stimulation of muscles. This kind of approaches can implement various stimulation strategies, e.g.: out-of-phase, selective adaptive timely stimulation (SATS), which analyse tremor activity and estimate future tremor periods. They are based on traditional techniques of signal analysis and suffer from the lack of accurate and reliable algorithms for long tremor prediction horizons. Consequently, there exists an urgent demand of trustworthy methods for tremor signal prediction, in order to create advanced tremor management systems. In this context, some studies have presented machine learning (ML) and deep learning (DL) algorithms for tremor-associated signals forecasting, aiming to tackle the limitations of current techniques. However, the majority of them are focused on one disease, analyse a single type of model or the obtained performance is still insufficient. Thus, there exists a wide room for improvement in this field.

This Master thesis is an in-depth study on the feasibility to apply DL for prediction of tremor-associated EMG signals with minimal preprocessing. Data from twelve ET and fourteen PD patients were collected to create the working databases. Then, five distinct models were designed and implemented, based on recurrent (RNN), convolutional (CNN) and Transformer neural networks. Performance analyses were conducted using different training windows (300, 600, 1000 ms) and prediction horizons (100, 200, 300, 600, 1000 ms). Models were trained using signals from ET and PD patients separately and, also, in a combined database. Then, for each scenario, a global comparison between all models was conducted, based on Pearson's correlation coefficient. Additionally, models trained only with ET or PD data were tested on the other disease to analyse their generalization capabilities. Finally, an experiment on the usage of joint kinematic and EMG signals was performed, in order to assess the effects of employing both information.

Results show that a CNN-based architecture is the overall best performing model. Mean correlations obtained by the best models for each combination of training window, prediction horizon and database were always between 0.7 and 1. Moreover, a mean correlation over 0.8 for a prediction window of 1000 ms was obtained, using a 1000 ms input length and the best model. Results also show that the models can generalize the prediction of ET and PD tremor signals, with independence from the training database. Finally, it was concluded that the addition of kinematic information during training does not improve the results.

The work of this Master thesis serves as a reference in the designing of DL models for tremor signal prediction and provides a baseline to create novel tremor suppression devices.

Keywords: tremor, essential tremor, Parkinson's disease, electrical stimulation, tremor prediction, deep learning, EMG signals, kinematic signals.

Acknowledgements

For the record.

"Be yourself; everyone else is already taken."

— Oscar Wilde

Ciudad Real, 27 de junio de 2023.

Este trabajo, todas las horas dedicadas a hacer de él la mejor versión que pudiera darle, todas las veces que anhelaba poder rendirme y no lo hice, todas las ocasiones en que tuve miedo y me arriesgué, toda la superación que sustenta este trabajo, todas sus páginas van por quienes me ayudaron a conseguirlo y, aún más, quienes no dudaron en que lo haría.

Por mis abuelos, cuyo esfuerzo y sacrificio son la condición necesaria y suficiente para que esté hoy aquí. De su sudor y su sangre han nacido mi alegría y disfrute, también mi motivación para luchar y aprender, entre otras muchas cosas, a ser honrado y bueno. Desde donde estéis, quiero que miréis con orgullo el fruto de vuestro trabajo.

Por mi madre, mi padre y mi hermano, porque uno es en su mayor parte lo que ha visto y desde niño mis ojos han visto cariño, bondad, esfuerzo, determinación, constancia, ilusión, esperanza y valentía. Siento que muchos de esos valores aún se me escapan, pero teneros como referencia me conducirá siempre a encontrar el camino correcto, estoy seguro. Ese es el mayor regalo que me habéis dado, de precio incalculable y valor infinito.

Por quienes son mi familia y mis amigos, a veces lo mismo. Porque el miedo a la soledad y ella en sí misma nos arrebatan la vida en silencio y solo el amor de vuestra buena compañía me puede rescatar de ese pozo.

Por aquellos cuya confianza y tiempo han dedicado a mí, mis tutores: Álvaro y Alejandro. Quien llega nuevo a cualquier sitio solo tiene lo que dicen de él como aval. Gracias a vosotros, ahora pueden decir mucho más de mí.

Va por vosotros. Gracias.

Alberto José Beltrán Carrero

Contents

List of Figures

List of Tables

List of Acronyms

AI: Artificial Intelligence.

BPTT: Back Propagation Through Time.

CNN: Convolutional Neural Network.

DBS: Deep Brain Stimulation.

EMG: Electromyography.

sEMG: Surface Electromyography.

ET: Essential Tremor

ECR: Extensor Carpi Radialis..

FCR: Flexor Carpi Radialis.

ESAP: Electrical Stimulation of Afferent Pathways.

FES: Functional Electrical Stimulation.

LSTM: Long Short-Term Memory.

ML: Machine Learning.

MSE: Mean Squared Error.

MLP: Multilayer Perceptron.

PCC: Pearson correlation coefficient.

PD: Parkinson's Disease.

RMS: Root Mean Square.

SOTA: State-of-the-Art.

PSD: Power Spectral Density.

RNN: Recurrent Neural Network.

Chapter 1

Introduction

Tremor is part of a group of involuntary movements, in which tics, myoclonic jerks, chorea, athetosis, dystonia and hemiballism are also included. Namely, tremor is usually defined as involuntary, oscillatory and rhythmic movements which can affect to one or more parts of the body, including upper and lower limbs [1]. It is classified into two main categories: physiological and pathological. The first type corresponds to uncontrolled movements of body parts that are naturally present and arise in situations of anxiety, fear, physical exhaustion, hypoglycemia, hyperthyroidism or alcohol withdrawal, among others [2]. On the other hand, the main criterion to consider tremor as pathological is that it appears as the primary or most acute manifestation of a certain disease. However, the complete classification procedure for tremor types involves the analysis of the frequency, amplitude, the conditions in which tremor starts, the medical and family history, and a neurologic examination [3]. For instance, a critical factor is the amplitude, which is substantially lower for the case of physiological tremor. In clinical practice, pathological tremor is one of the most common involuntary movement disorders evaluated [1].

Due to the ageing of the nervous and motor systems, pathological tremor is prevalent in middle-aged and older adults. However, it can be experienced at any age. Major causes include: neurodegenerative diseases, stroke, head injury, drugs and toxins, demyelinating diseases, systemic diseases or metabolic disorders [4]. In the context of involuntary movement disorders, the two main diseases characterized by the presence of tremor are: essential tremor (ET) and Parkinson's disease (PD) [5]. Thus, the research that was developed for this Master thesis is focused on these two disorders.

1.1 Medical context

ET is a chronic neurodegenerative disease [6] which has as primary manifestation tremor at hands and arms, that could eventually spread to other parts of the body. This tremor has typically frequency components between 4 and 12 Hz [7]. On the other hand, PD is a progressive multisystem neurodegenerative disease that has a wide range of motor and non-motor manifestations, in which orthostatic hypotension, constipation, urinary disturbances, sleep disorders, impaired motor function or tremor are included [8]. In this case, patients who suffer from PD experience tremor episodes whose frequency components are typically between 3 to 6 Hz [9].

Regarding the impact of these disorders, some studies made in different countries since 1960 to 2019 showed that the prevalence of ET in the population older than 60 years is between 2.3 % and 14.3 % (median: 6.3 %) and increases with age [10]. Another study from North America concludes that the prevalence of PD is less than 1 % in men and women aged 45-54 years, while it rises to 4 % and 2 % in men and women aged 85 years or older [11]. Furthermore, this type of disorders does not only lead to motor dysfunction, but also to psychological issues, such as depression, which are associated with difficulties in performing activities of daily living [12]. Finally, it is known that the impact of motor dysfunctions caused by the aforementioned diseases will continue to grow in the forthcoming years.

Today, there are several techniques aimed at suppressing pathological tremor to a certain degree. They use different strategies to achieve the best possible results: from medication to surgery or direct stimulation of brain structures and motor pathways. However, most treatments have several drawbacks and side-effects for the patients, while the results are often uncertain and insufficient. Drugs are not wide-applicable, being effective for about 50 % of patients [13]. Their side-effects lead to withdrawal of treatment in 33 % of cases [14] and, also, patients can develop cognitive disorders [15]. On the other hand, surgical procedures, such as thalamotomy, High-Intensity Focused Ultrasounds (HIFU) and Deep Brain Stimulation (DBS), are invasive and rely on the removal of certain brain structures or in the usage of electronic implants. These approaches have important inconveniences regarding: patient eligibility, surgery side-effects and adverse events [16]–[18]. Apart from these

traditional techniques, in recent years, researchers have worked on other minimum invasive approaches to overcome many of the implicit drawbacks of the aforementioned methods. Those with most promising results are: Functional Electrical Stimulation (FES) using out-of-phase strategy and electrical stimulation of afferent pathways (ESAP) using out-of-phase or Selective Adaptive Timely Stimulation (SATS) approach. FES is based on using peripheral or intramuscular electrodes to apply low-intensity electrical pulses over muscles. Out-of-phase technique consists of using this kind of stimulation over a pair of antagonists muscles. During a recording stage, tremor frequency and peak amplitude are computed and a prediction horizon for the next peaks is estimated, so antagonists muscles could be stimulated accordingly to reduce tremor amplitude [19].

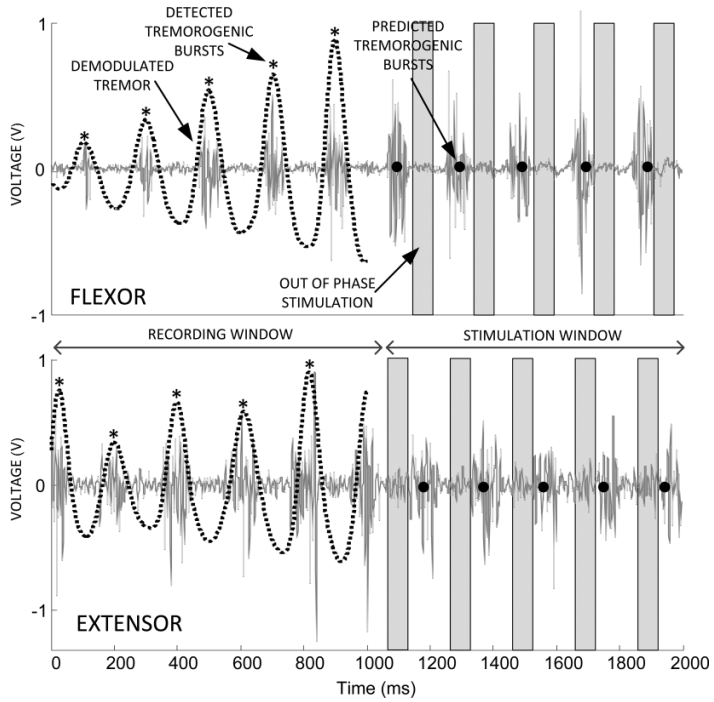


FIGURE 1.1: Out-Of-Phase strategy for FES [19]. During a fixed recording period, the EMG signal is analyzed and tremorgenic peaks are detected. The next time steps where tremor bursts might appear are estimated and the antagonists muscles are stimulated with a specified delay.

On the other hand, ESAP has shown a promising potential to reduce the amplitude of the electrical pulses applied and, still, achieving comparable results in tremor reduction. As out-of-phase FES, its objective is to stimulate coordinately a pair of antagonists muscles but, this time, via afferent pathways using spinal cord

reflexes and interneurons [20]. One of the most promising strategies in this area is SATS, which is based on alternating signal recording and electrical stimulation windows, sequentially. This closed-loop strategy brings the possibility to overcome the desynchronization issues from techniques like out-of-phase [21].

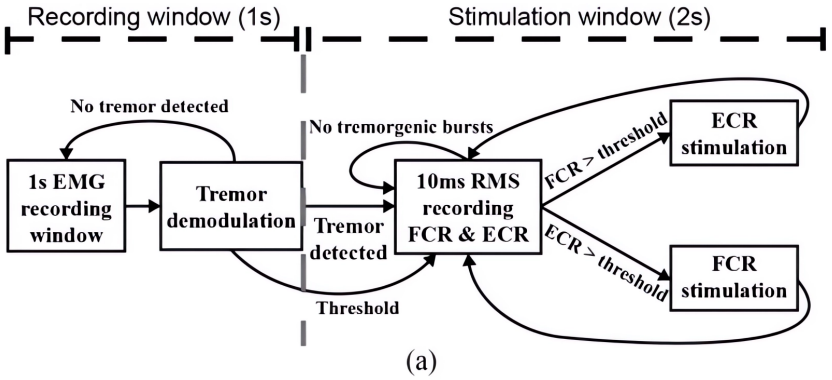


FIGURE 1.2: Control flow diagram for SATS strategy [21]. After tremor activity is detected within a 1 s recording window, every 10 ms, the Root Mean Square (RMS) values of the EMG signals from the antagonists muscles are computed and, if they are greater than a certain threshold, the stimulation begins. After 2 s of stimulation window, the recording stage is repeated.

Reference	Patients	Strategy	% Suppression
Javidan et al., 1992 [22]	3 ET, 4 PD	FES-Out-of-Phase	53±25 %
Gilard et al., 1999 [23]	3 PD	FES-Out-of-Phase	83±2 %
Popovic Maneski et al., 2011 [24]	3 ET, 4 PD	FES-Out-of-Phase	67±13 %
Dosen et al., 2015 [19]	2 ET, 4 PD	ESAP-Out-of-Phase	60±14 %
Dideriksen et al., 2017 [25]	5 ET, 4 PD	ESAP-Out-of-Phase	52 %
Pascual-Valdunciel et al., 2020 [20]	9 ET	ESAP-SATS	32 %

TABLE 1.1: Example studies using FES-Out-of-phase and ESAP-SATS strategies in ET and PD patients.

While the results obtained in recent studies by this modern techniques are promising, there exists still a large margin for improvement and various limitations due to physiological characteristics of pathological tremor and other factors. First of all, the systems need to deal with variations in tremor amplitude, frequency and patterns between different patients. Moreover, those features can vary for the same

patient during the day, due to medication and hormone levels, and tremor patterns can also change with aging, since most tremor disorders are neurodegenerative diseases. Last but not least, those techniques should be able to handle variations due to voluntary movements, which may affect tremor recognition and complicate correct stimulation.

Tremor suppression wearable devices

Another important area of investigation in the fields of ET and PD is the design of tremor suppression devices. This kind of technology can help patients who suffer from pathological tremor to be able to perform daily living activities, without the inconveniences of traditional surgical treatments or drugs. For that reason, the most relevant feature of those devices is to be wearable and comfortable. In this line, previously presented strategies for tremor management become interesting, as they aim to be minimum invasive, because the goal is to create an accurate sensing system to detect and quantify the tremor, and a control algorithm to perform the stimulation accordingly.

The integration of surface EMG (sEMG) in tremor suppression devices is limited due to the inconveniences that it presents in this case: the electrodes locations is patient dependent and there exists interference between recording and stimulation signals [26]. Instead, other studies relied on the usage of accelerometers and gyroscopes. However, in order to get accurate results, it is common to use more than 2 or 3 of this type of sensors, which can be very cumbersome for the patient [27]. Another possible approach is to combine both kinematic and sEMG signals to increase the prediction accuracy, while avoiding the aforementioned problems of employing just sEMG. In 2013, Basu et al. [28] developed an ON-OFF DBS system based on non-invasive measurements from sEMG and accelerometers which was able to achieve 85.7 % of tremor prediction accuracy for ET patients and 80.2 % for PD patients. They used spectral, entropy and recurrence quantification parameters to create an algorithm that was able to predict tremor onset, after a cycle of stimulation with DBS.

However, despite the results were promising, this algorithm was developed and

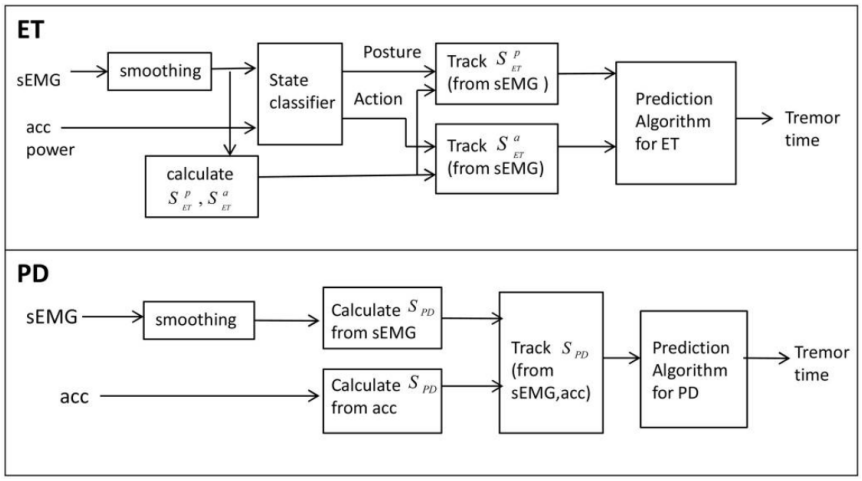


FIGURE 1.3: Signal flow diagram employed by Basu et al. [28] for tremor prediction. Authors designed different signal processing for ET and PD patients. After filtering the sEMG, spectral and non-linear time series parameters are used by the prediction algorithm to turn on the stimulation.

optimized assuming that their exists a DBS system that performs ON-OFF stimulation, so the goal was to predict future tremor episodes after a period of stimulation. Therefore, it is only applicable for those patients that already have a DBS device implanted, so the drawbacks and limitations of this method are still present. Finally, signal treatment was designed differently for ET and PD patients, which means that the system cannot be implemented directly to any patient and some variables have to be defined first.

1.2 Related studies

In recent years, taking advantage of the increasing progress in artificial intelligence (AI) applications, more specifically, Machine Learning (ML) and Deep Learning (DL), new studies on tremor classification and prediction using these novel methods have emerged as promising approaches to the management of involuntary movements. Most of the studies regarding this topic make use of tremor signals to various objectives: to identify tremor and non-tremor periods, classify tremor into unified types or to evaluate improvements after surgery. Those that are of interest for the research in this thesis are the ones related to tremor detection and tremor signal forecasting.

1.2.1 Detection of tremor episodes

In 2017, Jeon et al. [29] studied the application of ML algorithms to predict the severity of tremor in PD patients, following the Unified Parkinson's Disease Rating Scale (UPDRS), which is similar to how neurologists evaluate tremor in clinical practice. Authors designed a wearable device conformed by an accelerometer and a gyroscope, and recorded acceleration, angular velocity, displacement, and angle signals from 85 PD patients. Subsequently, they extracted nineteen features from each signal, applied dimensionality reduction and tested several ML algorithms for automatic scoring, namely: decision tree, random forest, support vector machine, discriminant analysis and k-nearest neighbors. All algorithms achieved over 80 % of accuracy in testing, being the decision tree the best performing one. Similar other studies have been conducted with alike results, extending the objectives to not only assessing tremor severity but, also, to identify tremor and non-tremor episodes or to classify signals from healthy and ET/PD patients [30]–[32]. All this research provides evidence to the helpfulness of ML algorithms in the problem of tremor detection.

Similarly, other investigations have tried to implement DL models in the same line. In 2021, Sriraam [33] developed a Recurrent Neural Network (RNN) capable of achieving a 99.81 % of accuracy on identifying presence of ET in EMG signals, by means of spectral features. The author experimented with a set of different estimation methods to compute Power Spectral Density (PSD) features from the EMG. Best results were obtained using Multiple Signal Classification (MUSIC) for feature extraction and a Recurrent Feedback Elman Neural Network (RFBEN) as a classifier. Nevertheless, with other methods such as Welch, Yule-Walker, covariance, modified covariance, accuracy results were above 90 %. More recently, in 2023, a research conducted by Pascual-Valdunciel et al. [34] provided promising results for tremor vs. no-tremor signal classification using both ML and DL algorithms. Authors employed minimally processed kinematic and EMG signals from ET patients, along with traditional ML models (K-Nearest Neighbors, Random Forest, Support Vector Machine) and a 2-layer Long Short-Term Memory (LSTM) architecture. Moreover, they assessed the capability of the algorithms to classify raw EMG recordings. All

models provided f1-scores over 0.8 in classification and, for the case of raw EMG sequences, the 2-layer LSTM achieved 0.98 of f1-score. Consequently, the feasibility of employing DL models for tremor detection has been demonstrated in some studies.

1.2.2 Prediction of tremor-associated signals

As well as detection and classification of tremor signals are crucial for the development of tremor management techniques, the capability of predicting next episodes of tremor, in order to apply appropriate stimulation, is at the same time a core feature for those systems. Some interesting studies have been conducted in this line of investigation.

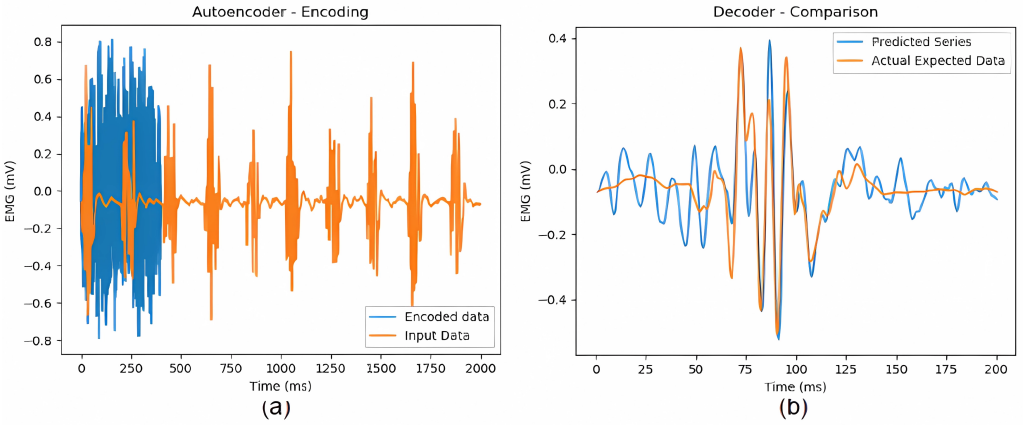


FIGURE 1.4: MLP Encoder for raw EMG encoding and prediction used in [35]. (a) shows the compression of a raw EMG sequence into a lower dimensional space by the encoder. (b) corresponds to a prediction example by the decoder, using as input the compressed signal.

Zanini et al. [35] performed a study using raw and envelope EMG signals from PD patients, whose objective was to experiment with a set of different DL architectures for prediction. Signals were acquired at a sample rate of 2000 Hz and split into 60 s sequences (120.000 samples per sequence). After removing high frequency noise and power line component, they created a dataset composed by raw EMG signals and another dataset with envelope EMG obtained using a moving-average filter. Then, they established a prediction window of 400 samples (200 ms), since that corresponds to the typical frequency at which tremor appears in PD. The key idea at this point is that they used an input window of 2 s (4000 samples) to predict the next 200 ms. They noted that it is possible to use smaller input sizes but the performance decreased considerably. Regarding the models they used, a Multilayer Perceptron

(MLP), multilayer LSTM, MLP Autoencoder and LSTM Autoencoder were included in the study. The reason to experiment with autoencoder architecture was that compressing the input data might ease the prediction task for the models, as well as provide useful insights about the signals. All models were trained and tested using both datasets and the results showed that MLP Autoencoder was the best performing one, in both cases, while the results were worse for raw signals than for the envelope. Consequently, authors concluded that data compression using autoencoder architecture can be beneficial for forecasting models, even more when they are fed with raw EMG sequences.

Other investigations have searched the same objective but using kinematic information instead of EMG signals. Ibrahim et al. [36] a hybrid DL architecture based on a Convolutional Neural Network (CNN) and a MLP. They acquired kinematic signals associated to specific parts of the hand and the wrist using IMUs, from a group of PD patients. Signals were recorded at a 100 Hz sampling rate and they were filtered between 3 and 17 Hz using a band-pass Butterworth filter to extract the tremor component. Also, a low-pass filter with cutoff frequency of 2 Hz was applied in order to remove voluntary movement component. The model architecture was composed by a combination of two consecutive 1D-CNN layers, followed by a MLP layer. Authors explained that the first part of the network is aimed to perform a feature extraction process over the signals using convolutions and pooling. Then, the extracted features are passed to the MLP which is in charge of performing the prediction stage.

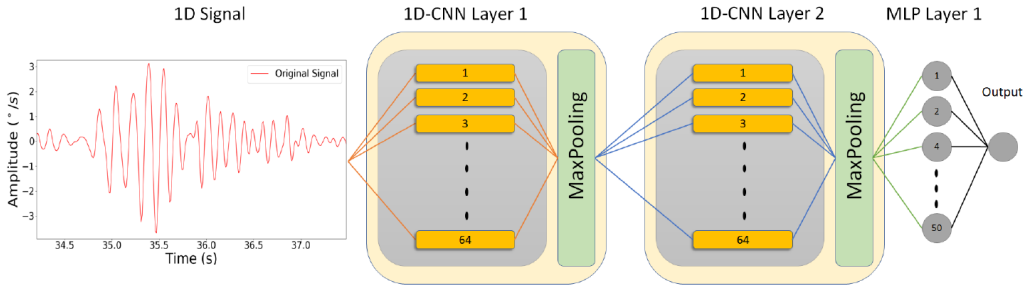


FIGURE 1.5: CNN-MLP model architecture diagram used in [36]. 1D-CNN layers were composed by 64 filters of size 2, 1D-MaxPooling were of size 2 and the MLP consisted of 50 neurons.

Although the model was also tested on estimating and predicting the voluntary

component of the signal, achieving about 99 % of estimation accuracy, the experiments related to tremor activity are the ones of interest in this case, whose results were not as good as for voluntary movement. Prediction accuracy for tremor signals was 97.3 %, 93.7 %, 91.4 % and 90.3 % for 10, 20, 50 and 100 ms prediction horizons, respectively. The lowest performance was given for signals associated with holding the hand against gravity in postural position, with an accuracy between 66 % and 73 %. Thus, results were highly dependent on the type of gesture and position of the patient.

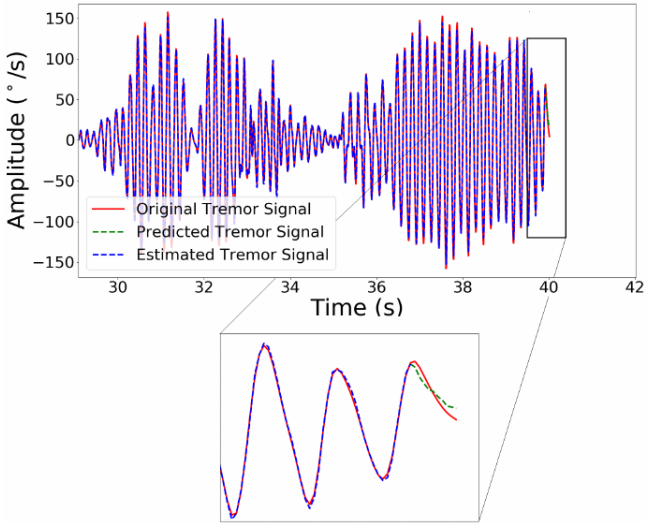


FIGURE 1.6: Example of estimation and prediction of tremor signal from [36]. Green scattered line corresponds to last 10 ms predicted segment.

Shahtalebi et al. [37] proposed a bidirectional RNN architecture called PHNet to extract and predict the voluntary component involved in pathological hand tremor (PHT) signals from PD and ET patients. Then, the tremor component can be also recognized and reduction techniques via stimulation could be applied. In this case, the data were acquired using an inline 3D accelerometer sensor and downsampled to 100 Hz. Patients were asked to perform different static tasks in order to obtain information from a variety of postures. Authors proposed an additive model for the voluntary component, assuming that the final measured signal is just the sum between the voluntary motion component and the tremor activity. Following this idea, they created the ground-truth for action tremor by mixing experimentally-collected

static tremor signals with a synthesized voluntary component created using a sinusoidal function. In this way, they could train the model to predict the artificial voluntary component in a large dataset and, after, evaluate the performance on real action tremor signals.

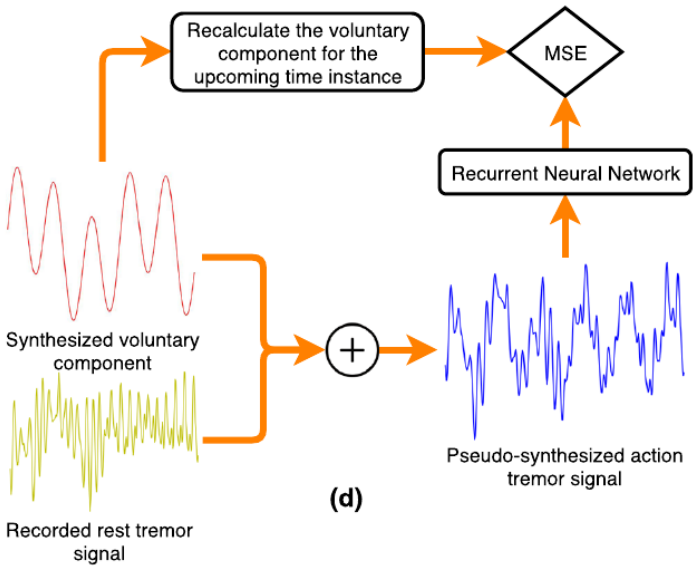


FIGURE 1.7: Workflow diagram for model training proposed by [37]. After adding synthesized and recorded components, the resulting signal is fed to the RNN model which is trained to predict the next voluntary component values.

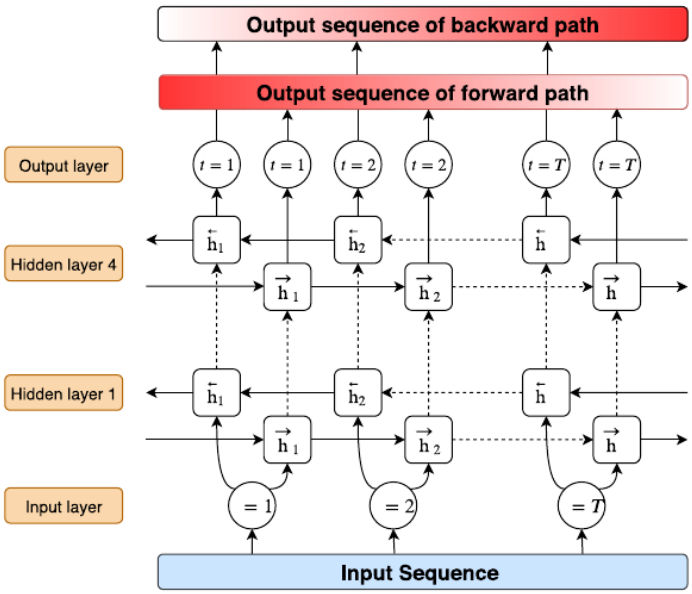


FIGURE 1.8: Bidirectional RNN architecture proposed by [37]. The information from the forward and backward paths are not mixed at the end to generate a unique output.

The bidirectional RNN architecture was composed by an specific type of recurrent cells: Gated Recurrent Unit (GRU) [38]. Authors decided to use a bidirectional approach because they wanted to build an online-offline prediction system. The forward path of the network corresponds to the online prediction, where past values are used to predict future unseen ones, whereas the backward path tries to reconstruct the previous part of the signal from current values. Then, they implemented the model so as to divide the output into two sequences (forward and backward paths) instead of mixing both information. The model was tested using different input length sequences: 1, 2, 3, 4 and 5 s. The best performance was achieved with input sequences of 4 s length and the results were extremely accurate (MSE<0.002). Even when the model was tested on real action tremor signals, the estimation of the voluntary component was precise (see Figure 1.9).

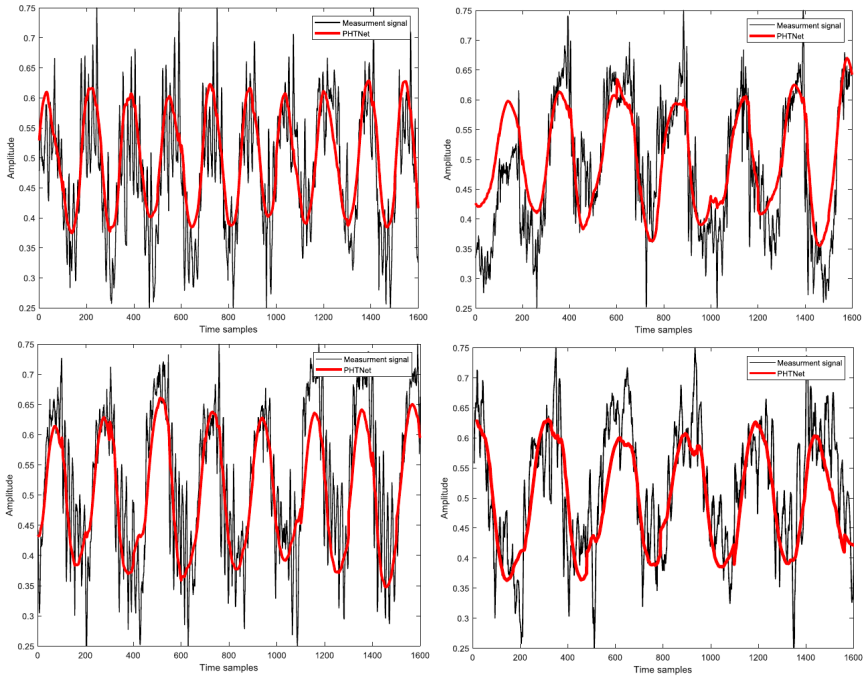


FIGURE 1.9: Voluntary component prediction examples from [37]. Black lines correspond to the real measured signal while red lines are the estimations provided by PHNet.

Finally, another recent and interesting study was the one performed by Pascual-Valdunciel et al. in 2021 [21]. The objective of this study was to develop a DL model

capable of predicting the full tremor waveform (amplitude and phase) with minimum phase delay from kinematic signals. They acquired data from ET patients using a configuration of 4 IMUs located over the dorsal side of both hands and at the forearms, with a sampling rate of 50 Hz. They empathized the need for several preprocessing steps before getting the data ready for training. First of all, raw quaternions had to be converted into Euler angles. Then, a band-pass Butterworth filter between 4 and 10 Hz was applied in order to extract the tremor component from the signals. The last step was to perform a min-max scaling over the signals with the purpose of easing the training for the network (see Figure 1.10). Two models consisting in one and two LSTM layers plus a linear layer were trained using different input and output lengths, namely: 600 and 1000 ms as input lengths, and 100, 200, 400, 600 and 1000 ms as prediction horizons. Performance was evaluated in terms of MSE, RMSE, Pearson correlation coefficient (PCC) and phase delay. Correlations were in the range of 0.70 in the worst case (input 600 ms, output 1000 ms) and 0.99 in the best case (input 1000 ms, output 100 ms). The prediction quality for 1000 ms horizon could be improved up to 0.76 correlation by using the same length at the input. In conclusion, authors demonstrated the potential usage of LSTM architectures for accurate prediction on appropriately processed kinematic tremor signals.

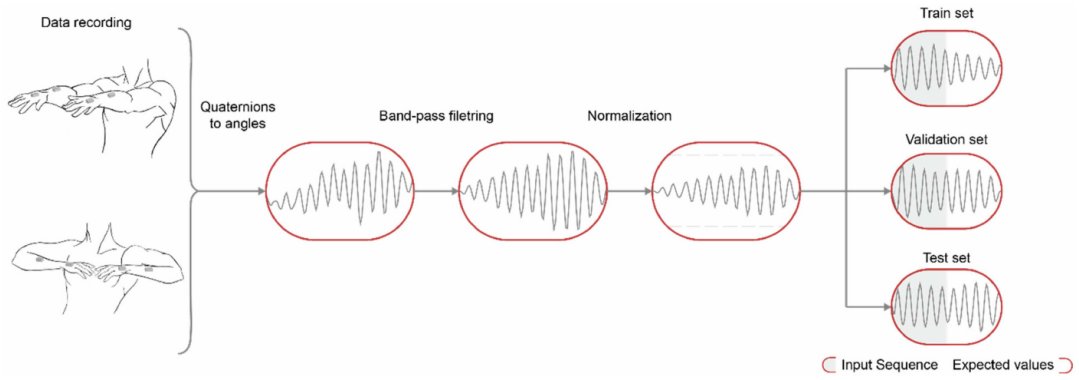


FIGURE 1.10: Workflow schema for the experiments conducted in [21].

1.3 Motivation

While the tremor detection problem is part of the development of tremor management systems and, thus, relevant for the background of this thesis, it has been shown that, in recent years, classification of tremorgenic signals using ML and DL algorithms is already explored and various studies have proved the potential usage of these methods in the field. What it is not entirely and generally demonstrated are the capabilities of AI models to predict tremor activity (amplitude and phase) with high accuracy for the typical tremor duration windows (>200 ms). In this context, this Master thesis is aimed to explore more extensively and generally the potential usage of DL models for tremor signals forecasting.

First of all, many investigations were carried out using kinematic information, recorded by means of accelerometers, gyroscopes and IMU devices, which usually require a heavy preprocessing stage. Instead, in this thesis, most experiments will be focused on using sEMG signals, which are a raw representation of electrophysiological data (muscle activity). Thus, models might be capable of learning intrinsic patterns from biological signals directly, which might be helpful for the generalization and characterization of tremor activity. In this regard, it should be remarked the importance of including information from both ET and PD disorders. This way, the models could be trained using separated or combined information and their generalization capabilities could be studied and tested. Therefore, this investigation will be carried out using data from ET and PD patients.

Regarding the design of DL architectures, other studies have shown the potential of RNN-based models, making use of different approaches in the filed (GRU, LSTM). Not so many investigation were conducted exploring other kind of designs, taking advantage of newer more powerful architectures which could bring an improved performance in prediction of tremor signals. Consequently, this Master thesis aims to build different non or less-explored approaches in this field, to test their efficacy and compare it with the state-of-the-art (SOTA).

Finally, another motivation for this investigation is to study the combined usage of kinematic and EMG signals for tremor activity forecasting, as there have not been found studies on this topic.

1.4 Objectives

Once analyzed the SOTA related to the investigation of this thesis and declared a convenient motivation as foundation, a series of objectives should be stated in order to specify the critical points to be assessed and to summarize the scope of this work:

- Design a sufficient but simple preprocessing pipeline and study the feasibility to use minimal preprocessed EMG signals to predict tremor oscillations.
- Design, implement and train predictive models based on a variety of contemporary DL architectures and approaches.
- Test the predictive capabilities of the models when trained separately with ET and PD data, and when trained using both information.
- Test models' performance for different combinations of input and output signal lengths.
- Study the combined usage of kinematic and EMG signals to train a predictive DL model and assess the performance.

Following these previous ideas, this thesis will be divided into 5 main parts:

1. Definition and description of the data to be used in the study.
2. Definition and implementation of preprocessing steps for the data and database creation.
3. Description of the DL models designed for the prediction of tremor signals and definition of the training algorithm and performance metrics.
4. Individual and comparative analysis of the results and a final overall discussion.
5. Declaration of the conclusions obtained from the investigation.

1.5 Hypothesis

To conclude with the introductory part for this Master thesis, it is convenient to add a set of hypothesis about some expected results of the investigation, that should be properly answered at the end of this work.

- Given the complex structure of DL models, it is expected that they will be able to extract patterns and learn predictive information from the data, even if the preprocessing effort is minimal.
- In terms of prediction accuracy, there will be a trade-off between input length and prediction horizon: performance will decrease as the input length becomes smaller or the output longer, while it will increase for longer input lengths or smaller prediction horizons.
- The combined usage of kinematic and EMG signals might bring an improved performance in terms of prediction accuracy for tremor activity, as models will be fed with two compatible sources of tremorgenic information.

Chapter 2

Materials and methods

This chapter involves the first three main parts of the thesis, which include all the information regarding the collected data, preprocessing steps, implemented DL models for prediction and definition of the performance metrics and training algorithm.

2.1 Materials

One of the objectives of this Master thesis is to evaluate predictive capabilities of the DL models in pathological tremor signals from the most prominent tremor disorders today: ET and PD. Therefore, a recruitment process for patients from both diseases was performed before acquiring the data.

Twelve ET patients were selected from the Movement Disorders Clinic of Gregorio Marañón Hospital (Madrid, Spain) to participate on the study performed by Pascual-Valdunciel et al. [21]. Fourteen PD patients were recruited and tested at Imperial College London (London, United Kingdom). Procedures were compliant with the Declaration of Helsinki and approved by Spanish Agency of Medicines and Medical Devices (record 714/18/EC) for the experiments with ET patients, and by Imperial College Ethical Committee (record 18IC4685) for the experiments with PD patients. The ET and PD patients included in the study satisfied the following criteria: diagnosis of ET or PD, present clinically postural or rest tremor; age between 18-85 years; tremor affecting at least one of the upper limbs; absence of other neurological or musculoskeletal pathology and ability to understand the procedure and sign the informed consent.

2.1.1 Data acquisition and description

For ET Patients, bipolar sEMG electrodes were placed over the muscle belly of FCR and ECR, after cleaning the skin with alcohol. sEMG signals were acquired at 2042 Hz. Additionally, kinematic signals were also recorded during the experiments using a motion capture system (Tech-MCS, Technaid S.L., Madrid, Spain) of four Inertial Measurement Units (IMUs). The sensors were located over the dorsal side of the hand and the forearm of both upper limbs (see Figure 1.10). Raw quaternions representing the spatial orientation of the IMUs were sampled at 50 Hz during 60 seconds. In two experimental sessions and, at least, three trials per patient, subjects were asked to hold the following postures against gravity to elicit tremor: both arms outstretched and pronated; or both arms with the elbows flexed and facing stretched fingers. On the other hand, only EMG data were acquired from PD patients and the recording stage of kinematic data using the IMUs system was excluded, due to the limited time and resources for the study. In this case, sEMG signals were acquired at 2000 Hz with a bio-signal amplifier (OT Bioelettronica, Italy) using bipolar electrodes placed on FCR and ECR muscles. Patients were asked to maintain a posture that facilitated the appearance of tremors for 30-60s: with both arms stretched out in front and pronated; with both arms raised, elbows flexed and fingers pointing to each other at the face level, or with both arms and hands resting on the table.

2.1.2 Data preprocessing

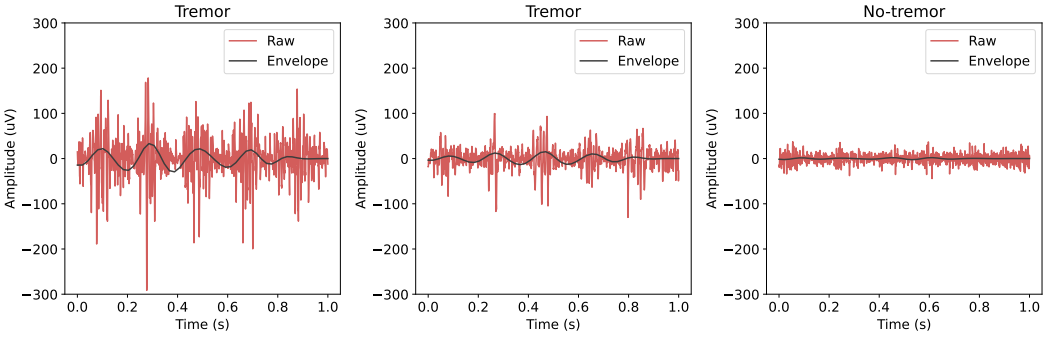
EMG data

The first part of data preprocessing was a quality selection procedure for sEMG signals. After experimental sessions were ended, each record was assessed and tagged according to its quality, based on visual inspection performed by an expert in EMG tremor signals. An heuristic 3-class quality scale was established as follows: 0 (Not-Acceptable), 1 (Acceptable) and 2 (Good). For the purpose of this investigation, only EMG records tagged as 1 or 2 quality classes were considered and included into the datasets (see Table 2.1). For the case of EMG recorded data from PD patients, all sequences satisfied the same signal quality requirements so it was not necessary to perform a selection process.

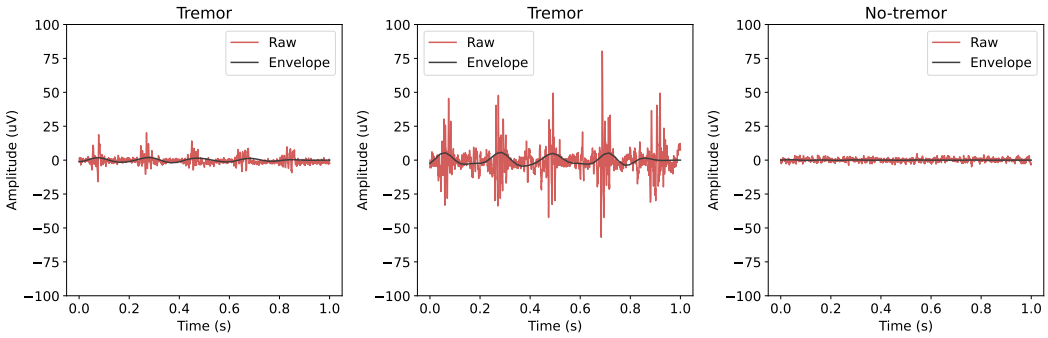
	EMG-ECR	EMG-FCR
Acceptable (1)	23	46
Good (2)	125	95
Total	148	141
Available records	289	

TABLE 2.1: Number of selected EMG records from ET and PD patients based on signal quality information.

sEMG recordings are known to generally contain different sources of artifacts and noise, such as: electrode inherent noise, movement artifacts, power line component, cross-talk between muscle fibers or electrocardiographic artifacts. Therefore, a preliminary filtering process is recommended [39].



(A) Source: ET data.



(B) Source: PD data.

FIGURE 2.1: Example EMG signals before and after preprocessing. Red lines correspond to the raw sEMG recordings and black lines to signals after being processed.

As commented in the introduction, pathological tremor in ET and PD patients can have frequencies between 3 and 12 Hz. However, at the time of the experimental sessions, it was noted that patients presented tremor with frequencies less than 10 Hz. Consequently, in order to extract the tremor component from raw signals, a

Butterworth band-pass filter of order 5 with cut-off frequencies 4 Hz and 9 Hz was applied. Butterworth filters have no ripple in their response and the transition is much more smooth than with other types of filters. Also, it was applied forward and backward to the signals, which provides a zero phase distortion [40]. Finally, the mean was removed from the signals and the recordings were downsampled from their original sampling rates, 2042 Hz and 2000 Hz for ET and PD patients, respectively, to a final sampling rate of 50 Hz. According to Shannon-Nyquist sampling theorem [41], to recover the information contained in a continuous signal with frequencies lower than f, it must be sampled at a rate f_s such that: $fs > 2f$. Therefore, given that the objective signal is the tremor component, whose frequency is less than 10 Hz in this case, a sampling rate of 50 Hz fulfills the Shannon-Nyquist theorem. Downsampling process helps to reduce as much as possible the computational and storage costs.

Kinematic data

Only the kinematic sequences associated to the selected sEMG recordings from ET patients were used in the dataset. Kinematic data were recorded in raw quaternions format with the IMUs system at a sampling rate of 50 Hz.

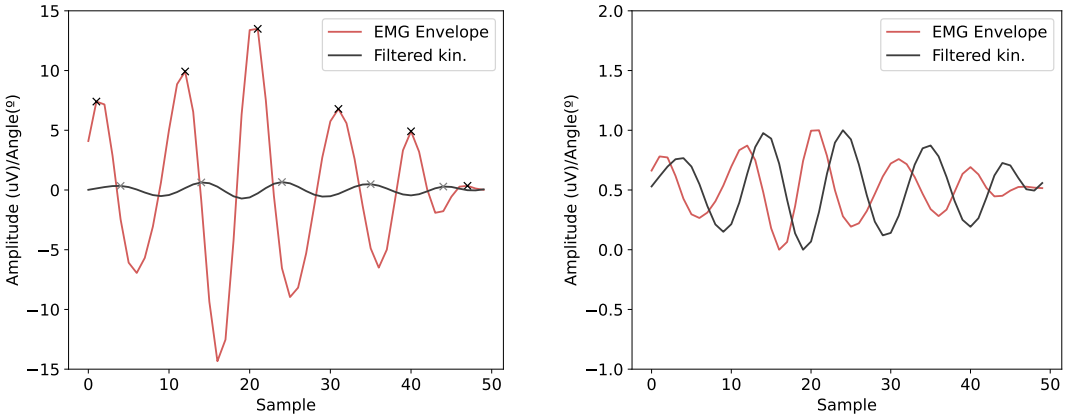
Quaternion number system is an extension to the complex numbers which was first described by William Rowan Hamilton, whose objective was to apply complex numbers to represent 3-dimensional rotations. A quaternion is generally defined as:

$$q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k} \tag{2.1}$$

where $a, b, c, d \in \mathbb{R}$ and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are the basis vectors which must fulfill the fundamental formula of quaternions [42]: $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$

Quaternions are used in problems that involve 3-dimensional rotations calculations because storing the captured data in this format demands less storage and it is more efficient, however, they are less intuitive to be interpreted. Consequently, they were transformed into Euler angles and represented based on the rotation axes of the IMUs (x, y, z). Then, only the variations in the angle corresponding to z-axis were taken, as that is the axis associated to the flexion-extension movement of the

wrist. Finally, sequences were filtered using the same configuration for EMG data (Butterworth band-pass filter of order 5 between 4 and 9 Hz), in order to apply the same signal processing strategy. In Figure 2.2, an example of EMG and its associated kinematic signal after processing is depicted. It can be noticed that their frequency is closely similar but the kinematic sequence is delayed with respect to the EMG, which is expected as the former represents directly electrophysiological activity and the kinematic signal represents the consecutive movement.



(A) Original. (B) Scaled view.

FIGURE 2.2: EMG+Kinematic signal from an ET patient example. (a) shows the data after processing and as stored in the dataset. (b) shows an scaled view in order to see clearer the delay between signals.

Tremor activity detection procedure

The principal objective of this Master thesis is to evaluate the predictive capabilities of DL models on tremor-associated signals. Given that tremor episodes are recurrent and not constant, it is expected that not every recorded sequence contains tremor component. Consequently, it is needed to have a tagging procedure for detecting and filtering the sequences in which there exists tremor activity. In this regard, frequency-based features are generally used in literature as a simple, computational-efficient approach to classify tremor signals, without resort to ML algorithms. For this study, the tremor detection process was based on the Power Spectral Density (PSD) of the sequences. PSD is defined as the Fourier transform of

the autocorrelation for a given signal X, formally:

$$S_X(e^{j\omega}) = \mathcal{F}\{R_X[m]\} = \sum_{m=-\infty}^{+\infty} R_X[m]e^{-j\omega m}, \qquad (-\pi \leq \omega < \pi) \tag{2.2}$$

where $\mathcal{F}$ denotes the Fourier transform and $R_X[m]$ the autocorrelation of signal X. PSD can be interpreted as a measure of the distribution of the mean power of a signal along the frequency domain [43]. Since the objective frequency band is between 4 Hz and 9 Hz, the values of the PSD associated to that interval can be used to determine a threshold for tremor activity. It should be noted that the definition of the PSD given by Equation 2.2 is true for a wide-sense stationary signal, that is, a signal whose statistical properties are approximately time invariant with respect to the origin. This property is assumed to be fulfilled by the EMG sequences, however, it is important to remark that this assumption is not totally true for biological signals in general, because they are not stationary.

The PSD of the EMG signals was estimated using Welch's method [44], which provides an estimate for the PSD based on splitting the signal into overlapping segments, computing the modified periodograms for each segment and averaging the results. Welch's method presents some important advantages with respect to other estimation methods, such as Blackman-Tuckey: it reduces spectral leakage by applying a window over the segments and variance by averaging various estimates [43]. For this study, Welch's method was applied using two segments with 50% overlap and a Hanning window to compute PSD estimates. The maximum value of the PSD between 4 Hz and 9 Hz was compared to an heuristic threshold to decide whether the an EMG sequence contains tremor or not. This threshold was defined as the median of the maximum PSD values between 4 Hz and 9 Hz from a sample of EMG recordings that represents more than 50% of the sequences. The mean value was discarded due to the presence of strong outliers in the distribution. Moreover, it was visually verified that the median value approximately served as threshold between tremor and no-tremor sequences. Nonetheless, it should be remarked that this method was developed based on this specific dataset and it is not guaranteed that it can be applied generally. The goal of this process was to easily filter tremor signals but, for general applications, classification methods as those mentioned in

the introduction might be used (see Section 1.2.1).

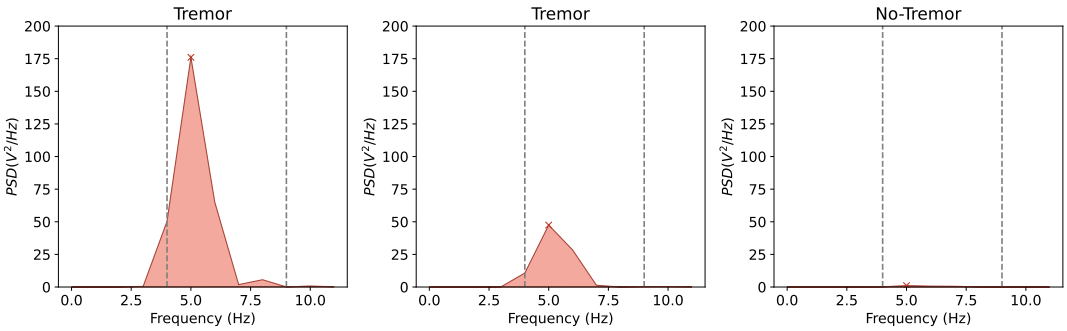


FIGURE 2.3: PSD for three processed EMG sequences of 1 s length. Dashed lines indicate the filtered frequency band in which tremor component is expected to appear. It is shown that the maximum of the PSD (x) is located inside this band.

2.1.3 Datasets

The datasets were created using sequences of 2 s duration. Only those signals in which tremor was detected during the first second were included. The first second will serve as the source for the input window in model training, whereas the resting part of the signal will be used as output window, i.e. prediction objective. Additionally, a 50% overlap was applied when loading the original recordings, with the purpose of increasing as much as possible the number of samples in the datasets. At the end, a total of 3800 EMG sequences of 2 s were selected from ET patients (along with their associated kinematic sequences), and 5200 from PD patients.

	Train		**Validation**		**Test**	
	ET	*PD*	*ET*	*PD*	*ET-P001*	*PD-P010*
Database ET*	3060	0	340	0	100*	0
Database PD	0	4590	0	510	0	100
Database ET+PD	3060	3060	340	340	100*	100

TABLE 2.2: Distribution of samples for the databases in train, test and validation subsets. *Database ET has one version including only EMG data and another including EMG and kinematic sequences. *Samples from patient ET-P001 were cropped to 100 in order to have the same number of testing samples from both diseases.

For the purposes of this thesis, four different datasets were created using the selected samples (see Table 2.2). A leave-one-patient-out method was used to create a test subset from the data, in order to evaluate the performance of the models. This methodology helps to assess the generalization capabilities of the models, as well

as to detect overfitting. Since the data contain information from two different diseases, one ET patient and one PD patient were selected and their recordings were removed from the training data to create the test subset. The remaining sequences were divided into train and validation subsets.

2.2 Methods

2.2.1 Development tools

This Master thesis was developed using the Python programming language and environment. Python is a general-purpose programming language that is increasing in popularity from past years, specially for tasks related to the field of data science, ML and DL. It has a large variety of libraries specifically designed for data manipulation, scientific analysis, algebra, ML modeling and automatic differentiation. For the purposes of this Master thesis, the principal libraries employed are: *SciPy* (scientific computing and signal processing), *NumPy* (matrix operations), *Pandas* (dataframes and data manipulation), *Matplotlib* (data visualization) and *PyTorch* (automatic differentiation). All DL models were implemented using the *PyTorch* library which, along with automatic differentiation features, it provides a large set of functions to load the data, train, test and deploy DL models. All code regarding the experiments of this work is available at a repository in GitHub[1].

2.2.2 LSTM

The Long Short-Term Memory network was proposed by Hochreiter et al. [45] in 1997 and it belongs to the field of recurrent neural networks (RNN). RNNs are designed to work with sequential data. Their architecture is characterized by the capability to save memory states, i.e. learn time dependencies in the input sequences and try to simulate memory notion. To this end, RNNs implement an update rule for their hidden states in which the information is shared across successive time steps. This strategy is known as sequence unrolling and it consists in connecting current

[1] https://github.com/Robolabo/. Access should be requested to the repository owner (a.gutierrez@upm.es).

inputs to previous outputs [46]. Formally, a RNN can be expressed in two equations:

$$\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}_h) \tag{2.3}$$

$$\mathbf{y}_t = g(\mathbf{V}\mathbf{h}_t + \mathbf{b}_y) \tag{2.4}$$

where $\mathbf{h}_t, \mathbf{h}_{t-1}$ are the hidden states from current and previous time steps, respectively; $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{V}$ are weights matrices, $\mathbf{x}_t$ is the current input vector, $\mathbf{b}_h$, $\mathbf{b}_y$ are bias vectors and f, g denote the applied activation function (e.g. $\tanh$, ReLU) which introduces a non-linear transformation.

This type of neural networks needs a different procedure to perform the learning process which, traditionally and generally for every neural network, is based on back-propagation and gradient descent techniques. The reason is that RNNs work with sequential data, meaning that each training iteration is over once the whole sequence is processed. Thus, there exists a time step dependency between the parameters of the neural network. This problem must be taken into account when computing the derivatives and updating the weights. For this purpose, a modified version of back-propagation was proposed for RNNs which is called back-propagation through time (BPTT). BPTT includes the time dimension on back-propagation process by assuming that the loss function ($\mathcal{L}$) at the current time step depends on previous ones, so the derivatives are taken from latest to first time steps (see Figure 2.4).

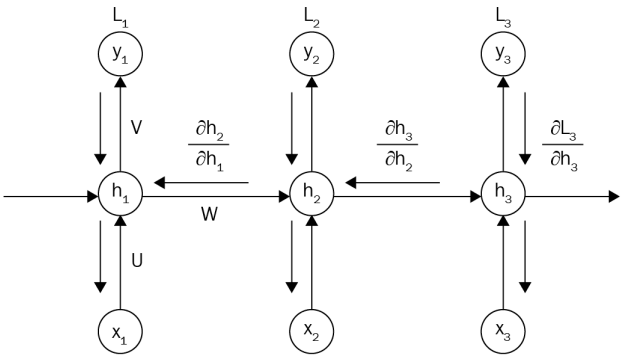


FIGURE 2.4: Computational graph of BPTT for an input sequence of three time steps [46]. At the last time step, the derivative of the loss function is computed with respect to that input. Then, the result is assumed to be dependent on the previous time steps and the derivatives are propagated until the beginning of the sequence.

BPTT solves the back-propagation problem in RNNs, however, one of the main

drawbacks of this type of architectures is the difficulty to learn long-term patterns as length of the input sequences increases. This is known as the vanishing gradients problem, which is explained in detail in Appendix A. LSTM was created as a solution to this issue. It is based on a memory cell composed by four non-linear gating units, whose purpose is to regulate the incoming and outgoing information flow. The gates are named as: input gate, forget gate, context gate and output gate. They are defined by Equations 2.5, 2.6, 2.7 and 2.8. Matrices first index corresponds to the vector they process, while second index refers to the gate [46].

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i) \tag{2.5}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f) \tag{2.6}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \tag{2.7}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_o) \tag{2.8}$$

for $\mathbf{x}_t \in \mathbb{R}^N$, where $\mathbf{x}_t$ is the input vector at time step t and N the feature length. $\odot$ represents the Hadamard product (element-wise multiplication). Then, the output gate generates the final hidden state which is propagated to the next step:

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{2.9}$$

These equations conform a model for human memory notion. LSTM cells build a representation for new ($\mathbf{i}_t$) and past ($\mathbf{f}_t$) information by applying linear combinations between the input ($\mathbf{x}_t$), previous hidden state ($\mathbf{h}_{t-1}$) and previous context ($\mathbf{c}_{t-1}$). Thus, they are adding additional information to the current time step (context) apart from the hidden state, as vanilla RNNs. Next, the current cell information ($\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c$) is filtered by applying a non-linear transformation ($\tanh$) and an element-wise multiplication with the input. The resulting information is added to the multiplication between the forget representation and the previous context. At this point, the output is computed from the new context vector and, finally, the next hidden state is generated.

Model and hyperparameters

LSTM architecture has been widely used in research when working with sequential/time series data. It has shown a great performance and, moreover, it was already applied to tremor-associated signals, as explained in the introduction. Therefore, the first implemented model in this work was based on the LSTM cell. A two-layer LSTM model called **LSTMp** is proposed as the simplest design that is going to be the reference to compare the next approaches. This idea is based on the previous work conducted by Pascual-Valdunciel et al. [47], where a similar model was proposed to perform signal prediction of kinematic signals.

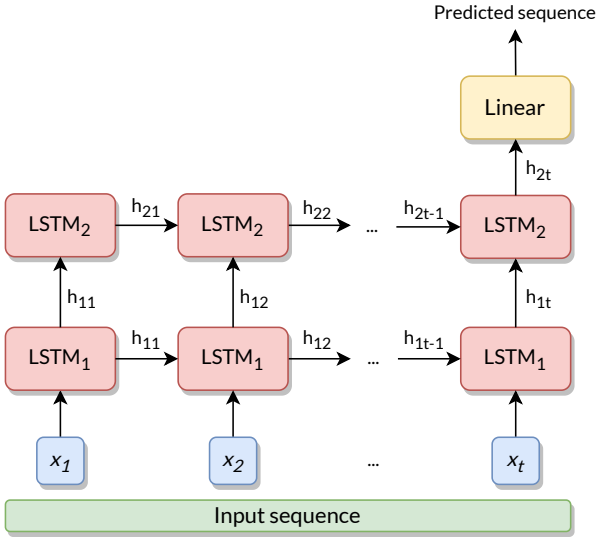


FIGURE 2.5: Diagram of the implemented LSTM model. Each layer corresponds to a different LSTM cell with its own parameters. The final hidden state of the first layer from each time step is passed as input to the second layer. Finally, the last hidden state from the second layer is fed to a linear layer to produce the entire predicted sequence.

A grid-search strategy was employed to find the best performing model, using the following predefined hyperparameters sets based on similar studies:

- **Hidden size (num. of neurons):** {35, 50}.
- **Learning rate:** {0.001, 0.0005, 0.0001}.

2.2.3 CNN

CNNs are a specific type of neural networks specialized in processing arrays of data, e.g.: images. They have far superior performance on extracting spatial patterns from

two dimensional data, with respect to other types of neural networks [48]. The reason is that they are inspired by the visual cortex functioning when it comes to data processing. CNNs implement a special type of neural layer called convolutional layer. It performs a convolution between the input and a series of matrices called filters or kernels, which are also known as feature detectors. The convolution consists in computing a dot product between the image and each of the filters, for different regions of the input (perceptive field). The output of each operation conforms a feature map, which represents the patterns found by each filter in the input. Then, a non-linear transformation is applied to generate the final set of feature maps. Therefore, the training process of a CNN has the objective of learning the appropriate set of weights for the filters in order to perform the given task [49].

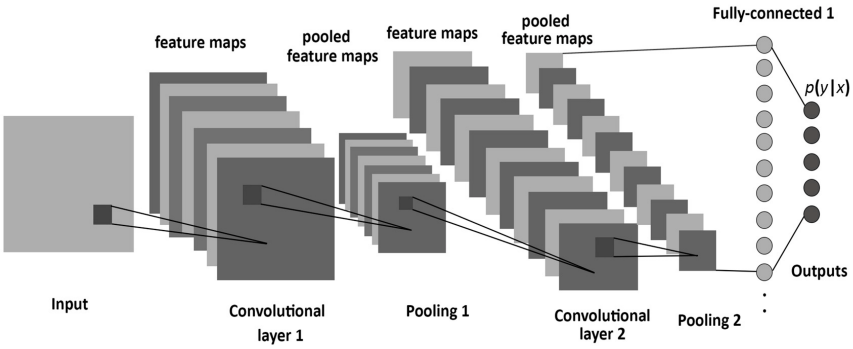


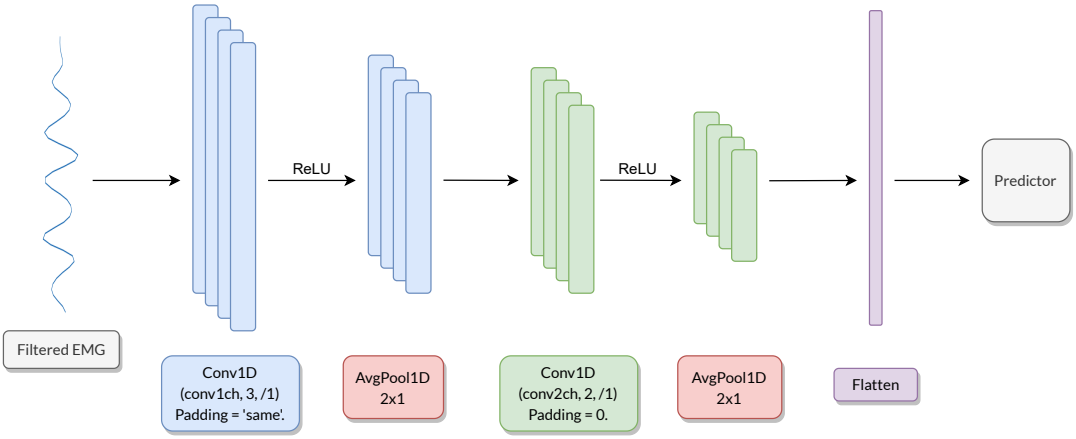
FIGURE 2.6: Diagram of an example CNN architecture [49]. The model is conformed by two convolutional layers, two pooling layers and a fully-connected network. The dimensionality reduction in the process is visually represented by the decreasing size of the feature maps.

In addition to convolutional layers, CNNs also are characterized by the application of downsampling operations to reduce input dimensionality. This procedure is driven by the pooling layers which, similarly to convolutional layers, are composed by a set of filters but, this time, their parameters are not trainable. This happens because the objective of pooling layers is to apply an aggregation function to fixed size regions in the input. After, the resulting matrices are compressed by selecting the maximum value of each of them (max pooling) or averaging all values (average pooling). This way, the dimensionality of the data is sequentially reduced.

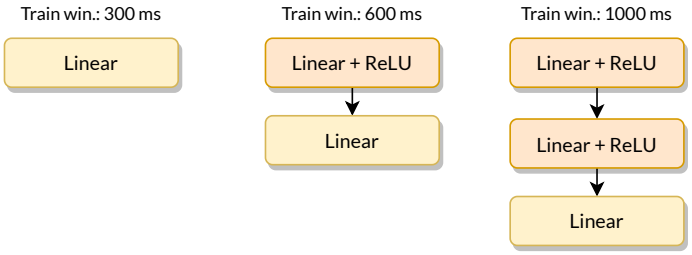
CNNs have been extensively applied in computer vision tasks but, also, various studies have proved the feasibility of applying these architectures to sequential/time series data [50], [51]. However, some modifications have to be implemented since

the input have no longer more than one dimension. Consequently, the kernels of the neural network are one-dimensional and the operations (convolutions and pooling) are perform along a unique axis. In this case, the feature maps can be interpreted as the recognized temporal patterns through the signal. Thus, the objective of this type of CNNs is similar to that of RNNs but, contrary to them, CNNs process the whole signal at once in parallel, which supposes an advantage in terms of computing efficiency for long sequences.

Model and hyperparameters



(A) Complete model. Conv1D notation: (Num. of filters, Kernel size, Stride)



(B) *Predictor* module for each training window.

FIGURE 2.7: Diagram of **LeNetp** model.

The first CNN-based architecture in this work is called **LeNetp** and it based on the LeNet network presented by LeCun et al. [52] in 1998. The original purpose of this network is to work with gray-scale (1 color channel) images of 28x28 pixels and perform a classification task. It is a basic CNN architecture based on subsequent convolutions and downsampling stages. Thus, due to its simplicity and design, it was selected as a reference to build a CNN network for the signal prediction task.

The architecture of the model is presented in Figure 2.7. It should be noted that, regarding the convolution stages, two different combinations of number of filters per convolution were employed during training, therefore, the number of filters in the diagram is referred as *conv1ch*, *conv2ch* for the first and the second convolution, respectively. Padding *'same'* consists on applying the necessary padding length in order to have the same output length as at the input. There exist other ways to perform padding, if necessary. In this case, having the same output length as at the input helps to add more complexity to the network. The input length (training window) determines directly the size of the final flattened vector, since the output length of the signal after each convolution or pooling is given by:

$$L_o = \frac{L_i + 2 \cdot p - d \cdot (k_s - 1) - 1}{s} + 1 \tag{2.10}$$

where L_i is the input length, p is the padding, d is the dilation, k_s is the kernel size and s is the stride. Consequently, a different sequence of linear layers and activation functions were applied for each training window. The number of layers increases with the input length, in order to take advantage of the more incoming information from the signal. In the simplest case, there is only one single linear layer that performs directly the prediction.

A grid-search strategy was employed to find the best performing model, using the following predefined hyperparameters sets, :

- **conv1ch**: {3, 6}.
- **conv2ch**: {12, 16}.
- **Learning rate**: {0.001, 0.0005, 0.0001}.

The hyperparameters were selected based on the original LeNet structure which used 6 and 16 kernels for the convolutions and filters of size 2x2 for the average pooling layers. The strides, paddings and kernel sizes for the convolutional layers were adapted to the input size of the signals, which will be between 15 (300 ms) and 50 (1000 ms) samples, in order to maintain a minimum sequence length through the processing for the convolutions and pools. Finally, the sigmoid function was substituted by the ReLU. The main reason is that ReLU helps to accelerate the training

process compared to other functions, because its derivative is 1 for positive inputs. It should be noted that this activation function is not applied on the final linear layer since the output signal can have positive and negative. In other words, if ReLU were applied at the end of the process, predicted values would be always positive.

Another CNN-based architecture called **incepCNNp** is proposed as a more sophisticated approach to the LeNetp model presented before. It is inspired by *ChronoNet* architecture developed by Subhrajit et al. [53] in the context of abnormal EEG detection.

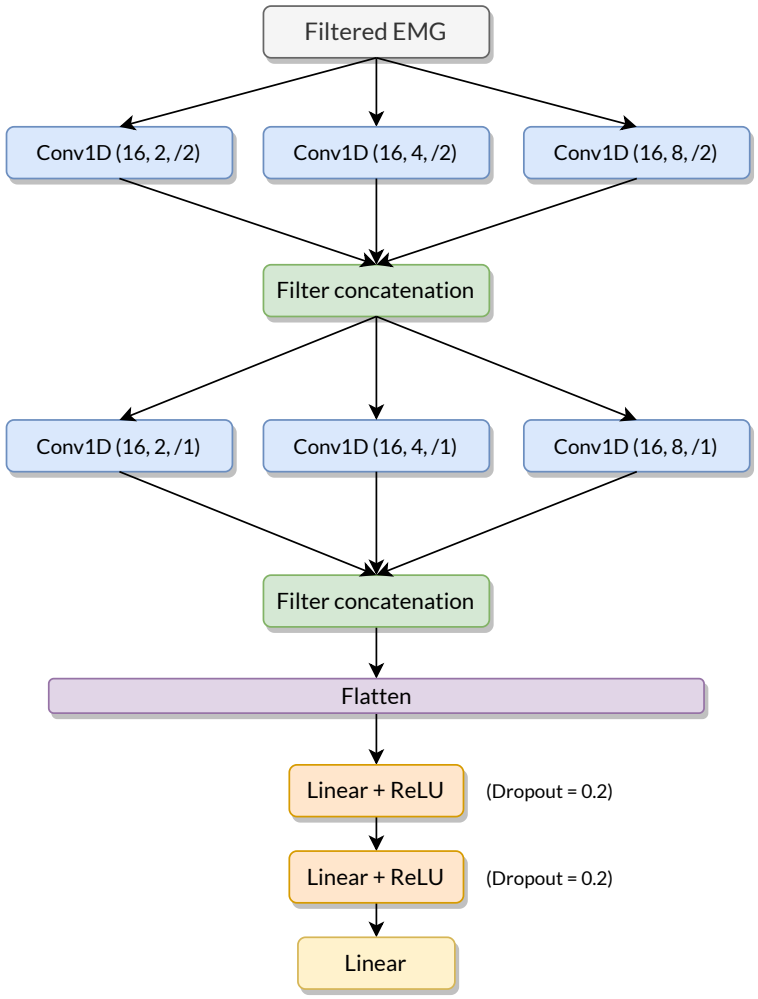


FIGURE 2.8: Diagram of **incepCNNp** architecture. Conv1D notation: (Num. of filters, Kernel size, Stride).

The authors experimented with a set of modules inspired by the inception architecture designed by Google [54]. This architecture was proposed as a solution to overfitting and to improve computing efficiency in deeper CNNs, in which the complexity

of the models started to suffer from problems related to vanishing and exploding gradients, as well as training expenses. Authors of the *ChronoNet* architecture inspired their model on the simplest version of inception. These modules consist in same-level convolutions with exponentially increasing kernel size, followed by a filter concatenation operation. In addition, *ChronoNet* includes recurrent networks and residual connections to improve the performance with time series data and prevent overfitting. For the purposes of this Master thesis, the proposed model follows the same idea but replacing the recurrent parts of the network by a sequence of linear layers and activation functions, in order to assess the performance of the model without sequences-oriented layers. The resulting architecture is shown in Figure 2.8. Regarding the hyperparamter selection, only different learning rates ({0.001, 0.0005, 0.0001}) were used in the grid-search process. The resting parameters were chosen according to the *ChronoNet* architecture, but reducing the number of kernels and their sizes since, in this case, the task is simpler (one channel of EMG instead of several channels from EEG).

2.2.4 CNN-LSTM

The next proposed model for tremor signal prediction is a combination between CNN and recurrent structures. While in literature it is common to find studies that make use of RNNs and, less frequently, CNNs for time series forecasting, the combined usage of both architectures has not been widely explored. The most interesting study that was found in this regard is the one performed by Subhrajit et al. [53], that was mentioned before. The authors experimented with a combination of convolutional and recurrent modules to process biological signals, adding skip-connections to avoid overfitting and vanishing gradients. In the previous section, their model was used as reference to create a CNN-based architecture (incepCNNp) but, now, another similar model called **CNN-LSTMp** is proposed including recurrent parts (see Figure 2.9). The objective of this model is to assess the possible improvements of combining convolutional with recurrent structures when processing one-dimensional signals.

In this case, the grid-search process was performed using the following hyperparameters sets:

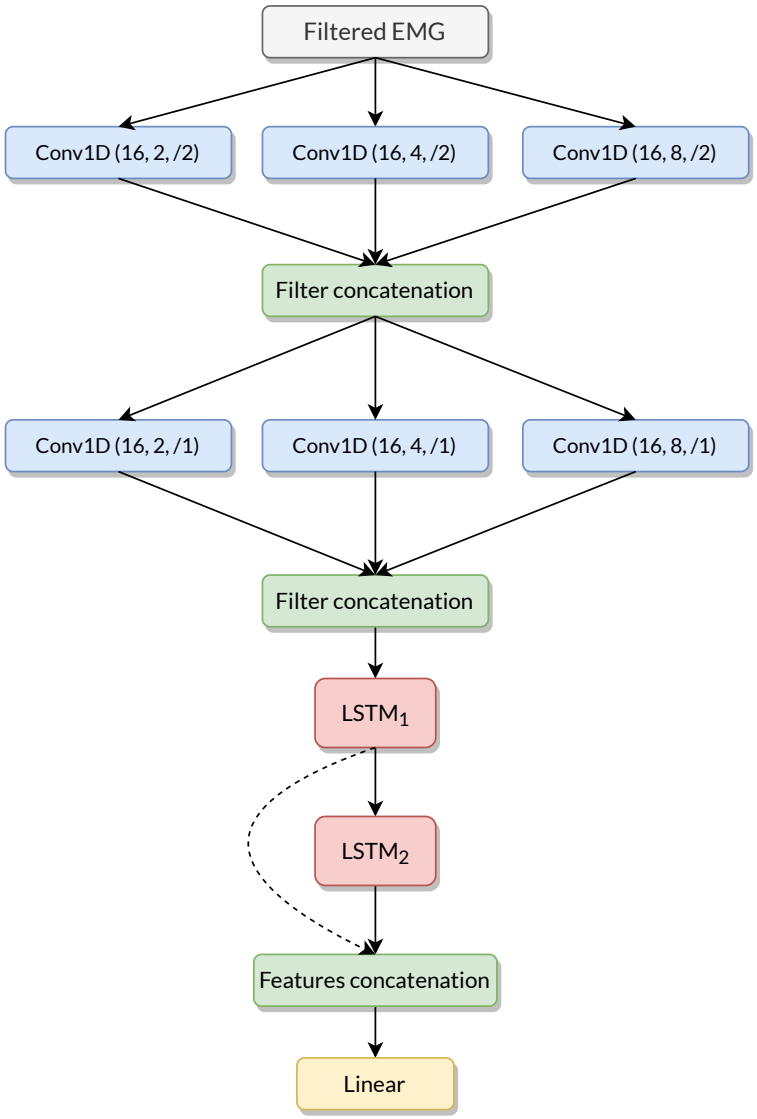


FIGURE 2.9: Diagram of **CNN-LSTMp** architecture. Conv1D notation: (Num. of filters, Kernel size, Stride).

- **Hidden size (num. of neurons):** {35, 50}. Same for both LSTM cells.
- **Learning rate:** {0.001, 0.0005, 0.0001}.

2.2.5 Transformer

In 2017, Vaswani et al. [55] published a paper that would change the field of DL for the upcoming years. In that paper, *Attention Is All You Need*, they presented the Transformer architecture: an encoder-decoder approach in the context of machine translation which, contrary to the SOTA models, substituted any recurrent structure by only self-attention modules. Authors showed that the Transformer was capable

of outperforming any other sequence-to-sequence model in most tasks. The great innovation of this new approach is mainly the total independence of recurrent structures in the processing. Authors demonstrated that attention modules were enough for a model to learn patterns and create fair, strong representations for text data.

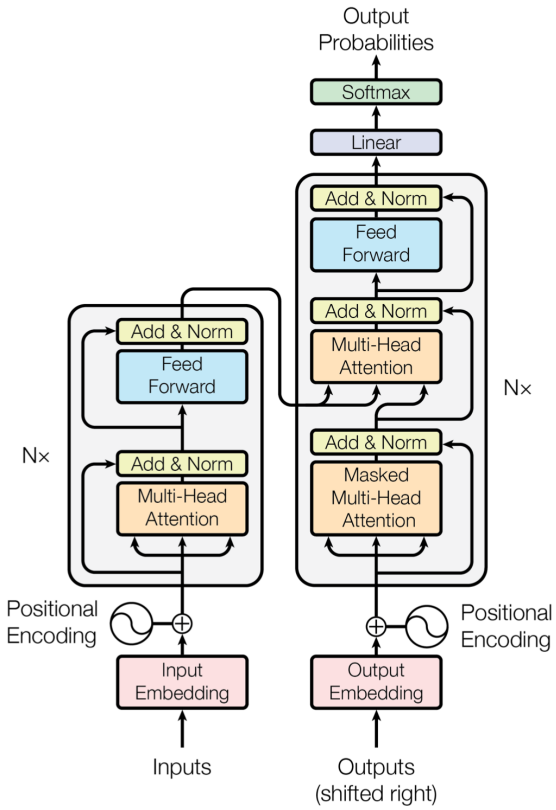


FIGURE 2.10: Original diagram of the Transformer architecture [55].

In neural networks, attention is a mechanism that allows models to create more complex representations for sequences, applying weights to the tokens that are computed based on the relative importance of each token in the sequence. These attention mechanisms can be interpreted as an imitation of the visual attention system in human beings [56], since its objective is to provide information about the relative importance of the elements in some situation. There exists several variants of the attention mechanism, regarding the operations involved in the computation of the attention weights. However, the core idea is the same for almost every method and it consists on a function that maps a query with a set of key-value pairs to an output, where all elements are vectors. The nomenclature comes from the field of databases and the meaning of each element is similar in this case. The query represents the

searched element, that is, the token of the sequence for which the attention weights from other tokens is going to be computed. These other tokens and their respective importance regarding the query are represented by the key-value pairs. Therefore, the attention system will apply some compatibility function between the query and the keys, providing the corresponding values. The final output is a weighted sum of those values. Vaswani et al. proposed a self-attention mechanism named as scaled dot-product attention:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.11}$$

where Q, K, V are the query, keys and values matrices consisting on the stacked individual query, keys and values vectors, and d_k is the embedding size. Authors claimed that this implementation is much faster and efficient than other similar approaches, since it is based on simple matrix multiplications.

In addition, authors decided to perform this attention process several times in parallel, by using different representations or heads, thus, implement multi-head attention. This method allows the model to create deeper and richer representations that will be jointly used at inference time. Formally, multi-head attention can be expressed as follows:

$$\begin{aligned} Multihead(Q, K, V) &= Concat(head_1, \cdots, head_h)W^O \\ \text{with } head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned}$$

where W_i^Q, W_i^K, W_i^V are parameters matrices that are learned during the training process.

Apart from the attention mechanism in the Transformer architecture, authors declared the necessity of introducing information related to the absolute or relative position of the tokens in the sequence. This is extremely important for the model to learn the temporal (positional) dependencies in the data, otherwise, it would not be able to establish these connections between the tokens, which is also crucial for the training of the attention mechanism. Consequently, authors added a positional encoding vector to the input embeddings. This positional encoding can be computed

in several ways, both fixed and learned terms. However, in the original architecture, authors noted that both approaches had similar performances so they decided to create a fixed function based on cosine and sine functions:

$$PE_{(pos,2i)} = \sin(pos/n^{2i/d_k})$$
$$PE_{(pos,2i+1)} = \cos(pos/n^{2i/d_k})$$

where pos is the token position in the sequence, i is the embedding dimension index, n is an scalar and d_k is the embedding dimension size. Therefore, the positional encoding is defined as a couple of sinusoidal functions of the same frequency, which reduces as the embedding dimension increases. For a fixed dimension index, an specific sine or cosine is added to the input embedding, so every token in the sequence carries information about its position and the model would be able to attend to relative positions. Moreover, the calculation of these functions is extremely fast and efficient and only depends on the selected embedding dimension.

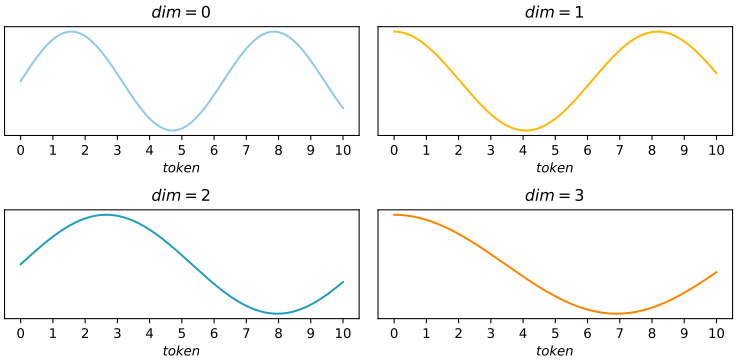


FIGURE 2.11: Example visualization of sine and cosine functions used for positional encoding. For even dimensions, the sine function is employed, whereas the cosine is added for odd dimensions. In both cases, as the dimension depth increases, the frequency reduces.

The Transformer architecture has an encoder-decoder structure, in which those previously presented concepts of self-attention and positional encoding are implemented. The encoder takes the input sequence, adds positional information, performs the self-attention and, then, the resulting vectors are passed to feed-forward networks that process each token individually. These last neural networks have the purpose of preventing the model to stop learning by applying linear transformations

along with a ReLU non-linear activation function. Additionally, there exist residual connections between the input and the output of the self-attention module. This encoder structure can be repeated N times, which supposes more flexibility when designing models of this kind. Finally, the outputs of the encoders are fed to the decoder, which has a similar structure. However, the decoder is meant to work in an autoregressive process, that is, it is sequentially receiving the previous outputs and, applying attention between them and the encoders outputs, it generates the probabilities for the next token.

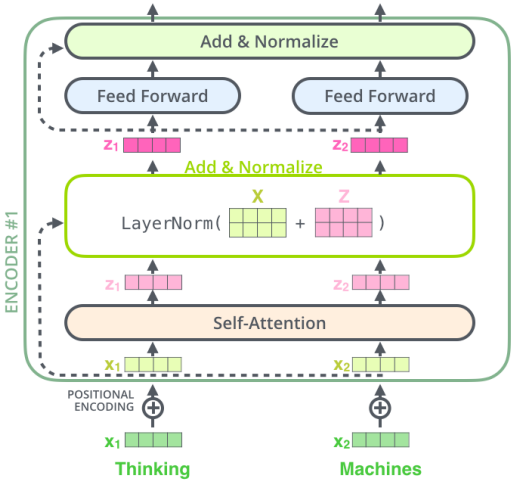


FIGURE 2.12: Tranformer encoder illustration by [57].

Besides major implementations and results using the Transformer architecture are associated to the field of natural language processing, since its release researchers have successfully tried to create variants of the Transformer to work with images [58] and time series data. At this moment, there exist several Transformer-based networks for time series forecasting, such as: LogTrans [59], Pyraformer [60], Autoformer [61], Informer [62] and FEDFormer [63]. The main difference between those studies comes from the implementation of the self-attention module for the time series. It should be noted that, with this kind of data, one of the main problems is the creation of a good methodology to perform both embedding and attention. In the field of natural language processing, there exists the concept of word embeddings, which is a unique map between a word an a vector in certain subspace. The unique representations for words allows a proper self-attention learning, because the model is capable of learning and identifying unique tokens. However, in time series data,

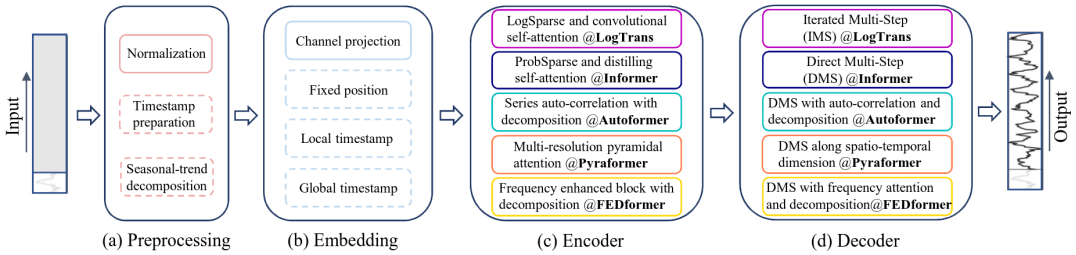


FIGURE 2.13: Summary of the pipeline from existing Transformer-based architectures for time series forecasting [64]. Solid boxes are essential operations while dotted boxes are optional.

the problem is different because both the input and the embeddings are in continuous space. That is the reason for the amount of different approaches in time series forecasting (see Figure 2.13).

Nevertheless, the efficacy of those models was called into question with the recent publication of the paper: *Are Transformers Effective for Time Series Forecasting?*, where Zeng et al. [64] designed a much more simpler feed-forward neural network that overcome the aforementioned Transformer-based architectures in most popular test datasets. In this context, by the end of 2022, Yuqi Nie et al. [65] proposed a novel architecture for long-term time series forecasting called *PatchTST*. This new model obtained better results than the existing approaches and, also, overcome the feed-forward network proposed by Zeng et al. *PatchTST* is based on feeding a Transformer Encoder with patches from the original signal. For the case of multi-channel signals, it implements channel-independence, i.e. each channel is fed to the model individually. The authors showed that the combination of instance normalization, patching and channel-independence is enough to obtain better results than those from previous architectures. Thus, for the case of EMG signal prediction, the proposed model is based on the ideas of *PatchTST*.

Model and hyperparameters

The last proposed model in this Master thesis for tremor signal prediction is called **TSTp**. It is mainly conformed by a group of Transformer encoders, along with other operations and layers. Firstly, before signals are projected to the model space, they are normalized and divided into patches. The number of patches depends on their length and the stride applied: $N_p = \frac{S_L - P_L}{s} + 1$, where N_p is the number of patches,

S_L is the sequence length, P_L is the patch length and s is the stride. Then, the patches are passed through the projection layer and the positional information is added. The resulting vectors are fed to the Transformer encoders, which have the same structure as in the original Transformer architecture. Finally, the flattened outputs pass through the prediction head, which is composed by a linear projection and the *de-normalization* step.

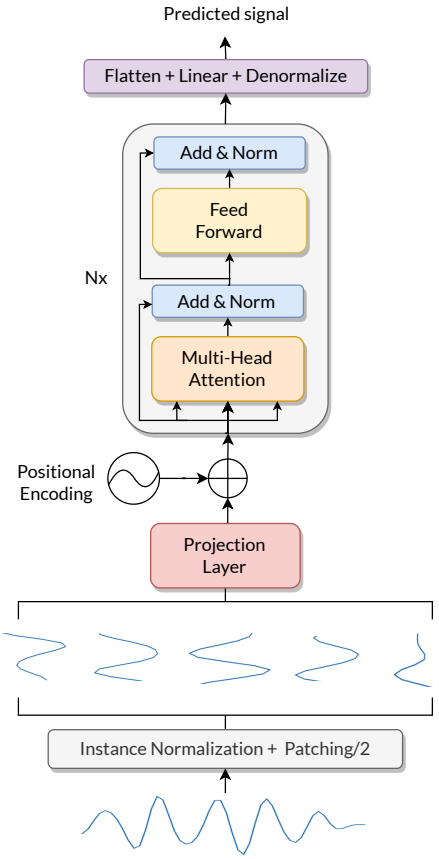


FIGURE 2.14: Transformer-based model architecture (**TSTp**).

As well as for the other models, a grid-search strategy was used to find the best set of hyperparameters for each situation, given the following predefined sets:

- **Patch length**: {160 ms, 200 ms}. (Stride fixed to 2.)
- **Model dimension:** {128, 256}. (Feed-forward dimension fixed to 128.)
- **Num. Attention Heads:** {4, 8}.
- **Num. Encoder layers**: {4, 6}.
- **Learning rate**: {0.001, 0.0005, 0.0001}.

2.2.6 Training algorithm

A common training function was defined for all the presented models. The algorithmic view of this function is available below (see Algorithm 1). Regarding the

Algorithm 1: Train function pseudo-code.

```
Define num. epochs, validation ratio and steps until early-stopping;
Data: X_t ← Load train dataset
Data: X_v ← Load validation dataset
criterion ← MSELoss;
optimizer ← AdamOptimizer;
for epoch in epochs do
    forall sample in X_t do
        Separate the input and target signals;
        optimizer.zero_grad();
        output ← model(input_signal);
        loss ← criterion(output, target_signal);
        loss.backward();
        optimizer.step();
        if validation then
            Disable gradient update;
            forall sample in X_v do
                Separate the input and target signals;
                output ← model(input_signal);
                loss ← criterion(output, target_signal);
            end
            valid_loss ← Average validation loss;
            if valid_loss < best_valid_loss then
                best_valid_loss ← valid_loss;
                stop_counter ← 0;
            else
                stop_counter ← stop_counter + 1 ;
            end
            Unable gradient update;
        end
    end
    if stop counter > early stopping steps then
        Save model;
        break;
    end
end
```

training parameters, a number of 1000 epochs was selected along with an early stopping criterion of 200 steps, that is, the training will be aborted if no improvement is achieved during 200 subsequent validation steps. These validation steps occur with a frequency that depends on the selected batch size and the size of the train dataset.

As optimizer, Adam was selected. There exist several optimizers that can be used depending on the specific dataset or task. For the purposes of this work, Adam was considered as a proper solution due to its computational efficiency and low memory requirements [66]. Finally, the MSE was chosen as loss function, since this is a regression/prediction task.

It should noted that the separation of the samples in train and target signals is performed inside the training function. Thus, the complete 2 s sequences are loaded at the beginning and, before feeding the model, the original sequence is divided according to the selected train and prediction windows:

- **Train window**: {300 ms, 600 ms, 1000 ms}.
- **Target/Prediction window**: {100 ms, 200 ms, 300 ms, 600 ms, 1000 ms}.

In this way, train, validation and test sets can be created once and used as many times as needed. Moreover, models could be assessed for different combinations of input and output lengths, thus, in terms of predictive capabilities, which is one of the main objectives in this investigation

2.2.7 Performance metrics

In order to evaluate the prediction models, a set of three different metrics was selected: the Mean Squared Error (MSE), the Root-Mean-Square Error (RMSE) and Pearson's Correlation Coefficient (PCC):

$$MSE = \left(\frac{1}{N}\right)\sum_{i=1}^{N}(y_i - \hat{y}_i)^2; \qquad RMSE = \sqrt{\left(\frac{1}{N}\right)\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{2.12}$$

$$PCC \equiv \rho = r_{y,\hat{y}} = \frac{\sum_{i=1}^{N}(y_i - \overline{y})(\hat{y}_i - \overline{\hat{y}})}{\sqrt{\sum_{i=1}^{N}(y_i - \overline{y})^2}\sqrt{\sum_{i=1}^{N}(\hat{y}_i - \overline{\hat{y}})^2}} \tag{2.13}$$

where N is the number of predictions made, y_i the true value of the signal, $\hat{y}_i$ the predicted value and $\overline{y}, \overline{\hat{y}}$ are the sample means.

The MSE value provides information about the prediction accuracy of the model, since it consists on computing the differences between real and predicted values. On the other hand, PCC represents the similarity between the real and the predicted

signal. Values of this coefficient closer to 1 or -1 mean higher linear relationship between them, so the objective is to maximize it. In this work, special attention will be given to PCC over the other metrics when evaluating the models and extracting conclusions, because the main interest is that the models learn to reproduce the overall behavior of tremor activity in the signals.

2.2.8 Statistical tools

All models were trained following a grid-search strategy to choose the best combination of hyperparemeters. After this process was conducted, the best performing models for each combination of train/target windows were selected and compared, using an appropriate statistical procedure to determine whether there exists statistical significance between the results or not. To this end, there exist different statistical tests that can be applied, depending on the specific situation and objectives. For the case of this investigation, the goal is to compare the populations of correlations obtained by the different models on the test sets. One alternative is to apply a repeated measurements ANOVA test with five factors (one per model), whose objective is to analyse the difference between the means of various populations. Therefore, it tests the null hypothesis of populations having the same mean. Nonetheless, in order to apply this test, it has to be assumed that the observations are independent, follow a normal distribution and have similar variance [67]. Unfortunately, it was obtained via Shapiro-Wilk normality tests that the populations of correlations in the experiments do not follow a normal distribution. Consequently, the repeated measurements ANOVA was substituted by a Friedman test [68], which is its equivalent non-parametric version. Non-parametric tests are statistical tests which are not based on parametrized probability distributions, thus, they can be applied to many situations while still providing proper robustness. However, for cases in which a parametric test is appropriate, the non-parametric versions are less powerful [67].

Friedman tests do not provide information about the specific pairs of populations (models) between which there exist differences. Therefore, another complementary statistical test is required to compare the performance of the models rigorously. To this end, the most popular test that is employed to compare the means from two populations is the Student's t-test, however, it has to be substituted by its

non-parametric equivalent as occurred with the repeated measurements ANOVA. In this case, the Wilcoxon signed-rank test [69] was employed.

Finally, as a principal representation tool, box plots will be used to show the results. In descriptive statistics, box plots are graphical representations to visually inspect the distribution, spread and skewness of a population of samples, by means of their quartiles (data points that divide the population in four equal parts). Box plots are composed by a box, whose limits represent the first ($Q_1 \rightarrow 25\%$) and third ($Q_3 \rightarrow 75\%$) quartiles, a line inside the box which represents the median or second quartile ($Q_2 \rightarrow 50\%$), and two additional lines named as whiskers, which can be computed in different ways [67]. For the case of this investigation, the whiskers are drawn at 1.5 times the interquartile range (IQR) below Q_1 and above Q_3. The remaining points that fall outside the resulting limits are depicted individually and they are called outliers.

Chapter 3

Results

This chapter is dedicated to the presentation of the results obtained throughout this Master thesis. As explained in the introduction, the main objective of this work is to assess the predictive capabilities of various DL models, based on different architectures and approaches, for EMG signals associated with tremor activity. Therefore, the first part of the chapter is conformed by the comparisons between the performance of the models for each database (ET, PD and ET+PD), separately (Sections 3.1.1, 3.1.2, 3.1.3).

Another core objective is to test the predictive capabilities of models that have been trained with data from one disease in data from the other disease, e.g.: a model trained using only ET data is tested using PD data and vice versa. These experiments are denoted as cross-testing and Section 3.1.4 is dedicated to show and comment the results obtained in this matter.

Finally, the last part of the chapter is focussed on the results obtained from tremor signal prediction using combined kinematic and EMG information (Section 3.2) and a final discussion of all results (Section 3.3).

3.1 EMG signal prediction

Table 3.1 shows the results of the Friedman tests for each database and, also, for the experiments conducted on cross-testing. In order to simplify the presentation of the tables, the cross-testing experiments are denoted as: **ETPD** for models trained on ET data and tested on PD, and **PDET** for models trained using PD data and tested on ET samples.

Train win. (ms)/Pred. win. (ms)	ET	PD	ETPD	PDET	ET+PD
300/100	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4
300/200	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4
300/300	0.074	<1e-4	<1e-4	<1e-4	<1e-4
300/600	0.229	<1e-4	<1e-4	0.0192	<1e-4
300/1000	0.1496	<1e-4	<1e-4	0.0482	0.0003
600/100	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4
600/200	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4
600/300	0.0139	0.1666	<1e-4	0.0002	<1e-4
600/600	0.3199	0.0019	<1e-4	0.0212	0.0002
600/1000	0.7299	0.2415	<1e-4	0.0152	<1e-4
1000/100	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4
1000/200	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4
1000/300	0.0137	0.0002	0.0002	0.0069	<1e-4
1000/600	0.2211	0.0552	<1e-4	0.5314	<1e-4
1000/1000	0.79	<1e-4	<1e-4	0.6155	0.0097

TABLE 3.1: P-values of Friedman tests for the different databases and cross-prediction.

In all tests, the populations of correlations obtained by every model were compared, in order to check whether there exist differences between their means or not. Nevertheless, these tests serve as an overview of the situations where there exist differences between the models, thus, there might be ones better than others; and those scenarios where no statistical significance was obtained so the performances of all models were similar.

It can be noticed that there less statistical differences exist for the cases where the models are trained and tested on ET data (**ET** column), where the number of cases in which there exists statistical significance ($p < 0.05$) is 8 out of 15 total tests. On the other hand, more statistical significances are found for the resting cases: 12/15 in **PD** column, 15/15 in **ETPD** column, 13/15 in **PDET** column and 15/15 in **ET+PD** column. These observations suggest that the performance of the models might depend on the disease in certain grade. Figure 3.1 shows the correlation boxplots where all models are compared regarding the training windows and databases. It can be seen that the distributions of the correlations for database **PD**, cross-testing **ETPD** and database **ET+PD** are more concentrated in high values, while for the cases of **ET** and **PDET** the distributions are more spread.

Additionally, Table 3.1 shows that the differences between the models decrease as the prediction window enlarges. This can be a consequence of the increasing complexity when the target signal becomes longer. While there might be models that perform better for short and intermediate prediction horizons, there could be

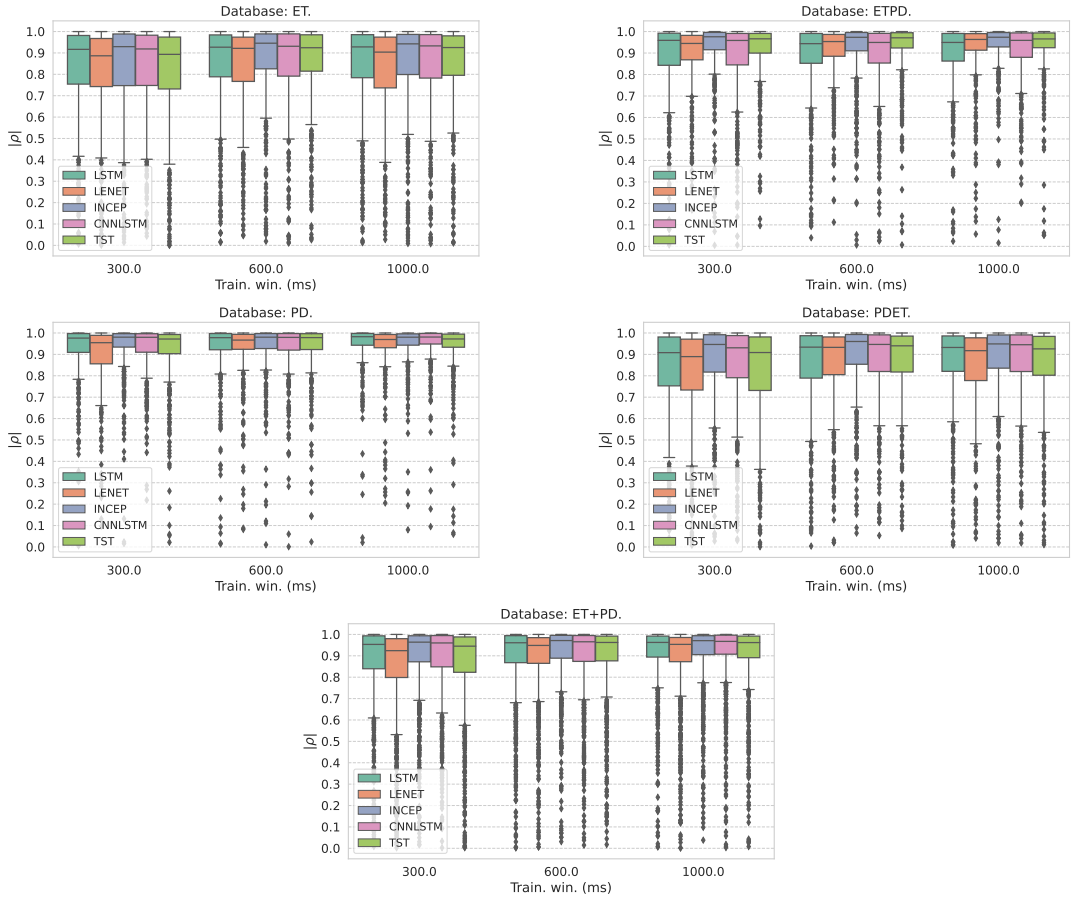


FIGURE 3.1: Boxplots of correlations obtained by the models, separated by training windows and for each database.

a threshold from which models are less capable of accurately predicting the next samples and their results become more similar in terms of performance.

Given that there are five different models, the total possible pairs without repetition ascends to ten, so ten different Wilcoxon tests were conducted for each database (see Appendix B). The results were summarized in the tables that are presented in the following sections, along with correlation boxplots for visualization.

3.1.1 ET dataset

The first part of the results corresponds to the models that were trained and tested on the ET database. In Table 3.2, a comparison between the best models for each scenario is presented. The average correlation coefficient (ρ) was computed from all test samples and, then, the set of Wilcoxon tests was performed in order to check the presence of statistical significance in the results. The table is divided in three main rows, corresponding to the three different training windows (input lengths)

Train Database: ET - Test Database: patient ET-P001

Train win. (ms)	Pred. win. (ms)	LSTMp	LeNetp	incepCNNp	CNN-LSTMp	TSTp
300	100	**0.9865**$^{\bullet\star}$	0.8922$^{*\dagger\diamond\star}$	0.9816$^{\bullet\star}$	0.9854$^{\bullet\star}$	0.9145$^{*\bullet\dagger\diamond}$
	200	0.9173$^{\bullet\star}$	0.8775$^{*\dagger\diamond}$	**0.9373**$^{\bullet\diamond\star}$	0.9233$^{\bullet\dagger}$	0.8887*†
	300	0.8418	0.8150	**0.8521***	0.8462	0.8218†
	600	0.7113	0.6816	**0.7312***	0.7125	0.6643†
	1000	0.5760	0.5628	**0.5946**	0.5794	0.5635
600	100	0.9889$^{\bullet\star}$	0.9721$^{*\dagger\diamond\star}$	0.9887$^{\bullet\star}$	**0.9906**$^{\bullet\star}$	0.984$^{*\bullet\dagger\diamond}$
	200	0.9487$^{\bullet\dagger}$	0.9248$^{*\dagger\diamond}$	**0.9645**$^{*\bullet\star}$	0.9569$^{\bullet\star}$	0.9427$^{\dagger\diamond}$
	300	0.8668†	0.8757†	**0.9074**$^{*\bullet\diamond\star}$	0.8951†	0.8912†
	600	0.7544†	0.7509†	**0.7846**$^{*\bullet}$	0.7608	0.7797
	1000	0.5829	0.6117	0.6121	0.6294	**0.6425**
1000	100	0.9907$^{\bullet\star}$	0.8952$^{*\dagger\diamond\star}$	0.9872$^{\bullet\star}$	**0.9909**$^{\bullet\star}$	0.9759$^{*\bullet\dagger\diamond}$
	200	0.9185$^{\bullet}$	0.8746$^{*\dagger\diamond\star}$	**0.9385**$^{\bullet}$	0.9384$^{\bullet}$	0.9284$^{\bullet}$
	300	0.8676$^{\bullet}$	0.8074*†*	0.8621$^{\bullet}$	0.8486	**0.8693**$^{\bullet}$
	600	0.6997*	0.7023	0.7135	0.6838	**0.7274***
	1000	0.6800	0.6569	0.6558	0.6439	**0.6927**
# Times as best		1	0	**8**	2	4

TABLE 3.2: Mean correlation coefficient (ρ) comparison between models for database ET.. Symbols to denote presence of statistical significance ($p < 0.05$) with respect to the models: LSTMp ($*$), LeNetp ($\bullet$), incepCNNp ($\dagger$), CNN-LSTMp ($\diamond$), TSTp ($\star$).

and each of these rows contains another five sections that correspond to the prediction horizons (output lengths). The highest correlations in each scenario is presented in bold and, at the bottom of the table, an additional row is included to count the number of scenarios where a certain model obtained the best result. Then, the presence of statistical significance between the results of two models, according to Wilcoxon signed-rank test, is denoted by a set of different symbols.

Regarding the first input window (300 ms/5 samples), the best correlation results are obtained for the shortest prediction horizon, as expected. Almost every model had a mean correlation on the test set over 0.9, excepting the LeNetp which had an average correlation of 0.89. Also, results from LeNetp were statistically different from the ones obtained by the rest of the models, which are better. This suggests that it is truly performing worse in this case. Meanwhile, the best model for this combination of train/target windows was the LSTMp, nonetheless, incepCNNp and CNN-LSTMp provided closely similar results with no statistical significance between them and the LSTMp, so these three models have equivalent performance. Continuing with the second prediction horizon (200 ms/10 samples), still three models were capable of provide a mean correlation over 0.9 as for the previous scenario. In this case, incepCNNp obtained a 0.94 average correlation and presence statistical significance with respect to three out of the four resting models. It should be noted that

this prediction window is now inside the typical duration of a tremor period and most of the models are showing an acceptable performance. In the next prediction window (600 ms/15 samples), a drop in terms of performance occurred for all models, whose mean correlations are now around 0.83. Again, the model that shows the best performance is the incepCNNp with 0.85 but, in this case, the results are only significantly different from those of TSTp. Finally, for the two remaining prediction windows (600 ms/30 samples and 1000 ms/50 samples) the results descend down to 0.7 for 600 ms and less than 0.6 for 1000 ms. The best performing model appeared to be the incepCNNp but in these cases there are almost no evidences of statistical significance with respect to the other models.

For the case of the second training window (600 ms/30 samples), the results appear to be similar for the first two prediction horizons. The main improvement comes for the intermediate output windows (300 ms and 600 ms) where the mean correlations are all over 0.75, so doubling the size of the input window conducts to a notable improvement, which is expected as the models would have more information during training. Still, the last prediction window contains the worst results and all models appeared to be equivalent.

Finally, the last input window (1000 ms/50 samples) had similar results for the first two output lengths while, contrary to what might be expected, for the intermediate prediction horizons the results were worse than those obtained using a 600 ms training window. Probably, longer input windows contain more variability in the signals and, thus, the uncertainty in the predictions could increase for intermediate horizons. However, in the last scenario, an improvement can be seen in the average correlations with respect to the previous input window. These results suggest that the models perform better in intermediate prediction lengths with intermediate input windows but, for the longest output where the complexity of the task should be the highest, models benefit from having maximum information as possible.

In Figure 3.2, the boxplots show the previously commented observations about the behaviour of the models, with respect to train and prediction windows. The distributions of the correlations are narrower for short prediction lengths. For the first two training windows, every model obtains correlations over 0.9 for more than 75% of the test samples, regardless of the input length. while as it increases the variability

in the results is more prominent. Lastly, the model that for most scenarios has the highest median and narrowest distribution of correlation values is the incepCNNp, which fits with the times that it was recognized as best model in Table 3.2. Nonetheless, it is clear that, for the longest output length, the variability in the predictions is high, besides in the best case (Train. win.: 1000 ms) the medians are always above 0.7, but this can be expected assuming that as longer the signal is, the greater the chance to find variations in the sequence, moreover, it cannot be guaranteed that the whole target signal contains tremor activity. It should be remarked that presence of tremor is only ensured by a detection threshold during the first second of the input but, as in a real scenario, the target is completely unknown by the models and it might or might not contain tremor patterns.

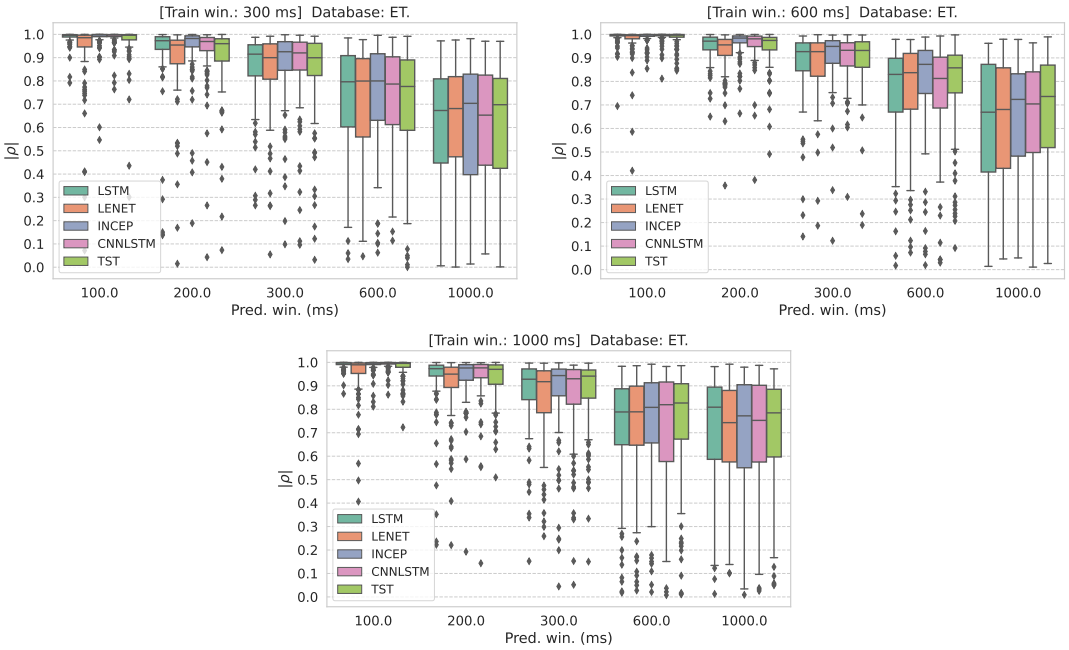


FIGURE 3.2: Boxplots of correlations obtained by the models for ET database, separated by training and prediction windows.

3.1.2 PD dataset

The results corresponding to database PD are presented in Table 3.3, with the same format as commented above. There are more cases in which statistical significance exists between pairs of models, also, the average correlations are noticeably higher in general.

Train Database: PD - Test Database: patient PD-P010

Train win. (ms)	Pred. win. (ms)	LSTMp	LeNetp	incepCNNp	CNN-LSTMp	TSTp
300	100	**0.9991** $^{\bullet\dagger\diamond\star}$	0.9738$^{*\dagger\diamond}$	0.9970$^{*\bullet\diamond\star}$	0.9985$^{*\bullet\dagger\star}$	0.9914$^{*\dagger\diamond}$
	200	0.9907$^{\bullet\star}$	0.9771$^{*\dagger\diamond\star}$	**0.9928**$^{\bullet\diamond\star}$	0.9904$^{\bullet\dagger\star}$	0.9804$^{*\bullet\dagger\diamond}$
	300	0.9652$^{\bullet\dagger\star}$	0.9439$^{*\dagger\diamond\star}$	**0.9743**$^{*\bullet\diamond\star}$	0.9645$^{\bullet\dagger}$	0.9469$^{*\bullet\dagger}$
	600	0.9000$^{\bullet\dagger}$	0.8611$^{*\dagger\diamond\star}$	**0.9198**$^{*\bullet\diamond\star}$	0.8992$^{\bullet\dagger}$	0.8787$^{\bullet\dagger}$
	1000	0.7931$^{\bullet\dagger}$	0.7308$^{*\dagger\diamond}$	**0.8080** $^{*\bullet\star}$	0.8008$^{\bullet}$	0.7461†
600	100	**0.9994** $^{\bullet\dagger\star}$	0.9940 $^{*\dagger\diamond\star}$	0.9986$^{*\bullet\diamond}$	0.9992$^{\bullet\dagger\star}$	0.9984$^{*\bullet\diamond}$
	200	0.9899$^{\bullet\star}$	0.9826$^{*\dagger\diamond\star}$	**0.9906**$^{\bullet\star}$	0.9904$^{\bullet}$	0.9879$^{*\bullet\dagger}$
	300	0.9644$^{\bullet}$	0.9598*†	**0.9694**$^{\bullet}$	0.9642	0.9633
	600	0.8943†	0.8961$^{\diamond}$	**0.9032**$^{*\diamond}$	0.8820$^{\bullet\dagger\star}$	0.8975$^{\diamond}$
	1000	0.7931	0.8071	**0.8111**	0.8076	0.8039
1000	100	**0.9986** $^{\bullet\dagger\star}$	0.9844$^{*\dagger\diamond\star}$	0.997$^{*\bullet\diamond\star}$	0.9984$^{\bullet\dagger\star}$	0.9944$^{*\bullet\dagger\diamond}$
	200	0.9907$^{\bullet\star}$	0.9771$^{*\dagger\diamond}$	0.9875$^{\bullet\diamond\star}$	**0.9915**$^{\bullet\dagger\star}$	0.9794$^{*\dagger\diamond}$
	300	**0.9675** $^{\bullet\star}$	0.9520 $^{*\diamond}$	0.9632	0.9660$^{\bullet\star}$	0.9564$^{*\diamond}$
	600	0.9092	0.8959$^{\dagger\diamond}$	0.9141$^{\bullet}$	**0.9206**$^{\bullet}$	0.8937
	1000	0.8834*	0.8716$^{\dagger\diamond}$	0.8812$^{\bullet\diamond\star}$	**0.8946**$^{\bullet\dagger\star}$	0.8578$^{*\dagger\diamond}$
# Times as best		4	0	**8**	3	0

TABLE 3.3: Mean correlation coefficient (ρ) comparison between models for database PD. Symbols to denote presence of statistical significance ($p < 0.05$) with respect to the models: LSTMp ($*$), LeNetp ($\bullet$), incepCNNp ($\dagger$), CNN-LSTMp ($\diamond$), TSTp ($\star$).

For the first input window of 300 ms, the situation is very similar as for ET database in the two first output horizons: most models obtain mean correlations in the test set over 0.9. Nonetheless, in this case, it happens that the performances of the models for these scenarios are optimum, since most of them obtain correlations around 0.99. Additionally, significant differences exist in the results when making the statistical comparisons. This suggests that, even if all models are providing similar mean correlations, their distributions are different between them. Consequently, for the case of PD data, the models might be treating the sequences in distinct ways. Continuing with the next prediction windows, it is remarkable that for 300 ms of output length the mean correlations are still over 0.9, more specifically, around 0.95. The best model is the incepCNNp in this case, with a 0.97 average correlation and presenting statistical significance with respect to all other models. LeNetp model is the one whose performance decreased the most, down to 0.94. For the next output window (600 ms), only LSTMp and incepCNNp achieved results over 0.9, being the former the best performing model. The worst is again the LeNetp with 0.86 mean correlation. Lastly, for the longest prediction horizon of 1000 ms, it occurs the biggest drop in terms of performance, as expected. Only three models (LSTMp, incepCNNp, CNN-LSTMp) provided mean correlations around 0.8 (about 0.1 less than for the previous window), while the rest of them achieved results closer to 0.7. Also, the

presence of statistical significances is less prominent. This behaviour was also noticed for the case of ET database, where as prediction window increased, the models were starting to become more similar regarding their results. About the drop in the mean correlations, it might be a similar situation as for ET sequences: as signals become longer, there is more room for variability and, moreover, more chance that there is absence of tremor activity in the prediction window. In any case, taking into account that the models were trained only with 300 ms (15 samples), a performance of 0.8 mean correlation for the most difficult scenario is promising.

The results for the next input length (600 ms) are very similar to those of the previous window, specially for the first four prediction horizons, where results of the best models are also above 0.9. A tiny decrease in the mean correlations can be noticed with respect to the previous input window. However, these differences correspond to hundredths and thousandths of correlation. The most remarkable case is the 1000 ms output window, where an improvement of 0.01 (4%) for the best model can be seen. There are no statistical significances between the results of the models, thus, they should perform approximately the same but it seems that having more information at the input supposes better performance for the longest output.

For the last training window (1000 ms), average correlations for the first three output lengths are similar to the previous ones. The improvements correspond to the last two output lengths, where the effect of more input information is an increase of performance, specially for the longest prediction horizon. In this case, all models experienced a significant improvement, going from mean correlations around 0.8 up to 0.89, with a 12% better performance for the CNN-LSTMp with respect to previous input windows. Regarding intermediate windows, it is interesting that, contrary to the ET database, the results are better using the longest input length.

In Figure 3.3, the boxplots show the distributions of the correlations for each scenario. Visually, it is obvious that, for the first three output windows, the distribution of the correlations are much more concentrated around the median than for ET database. Also, the presence of outliers is minor. On the other hand, for the last two prediction horizons, the improvement given by the increasing length at the input is represented by tighter distributions of the results. As commented before, in this case, models always perform better for longer prediction horizons when they

are trained with more information.

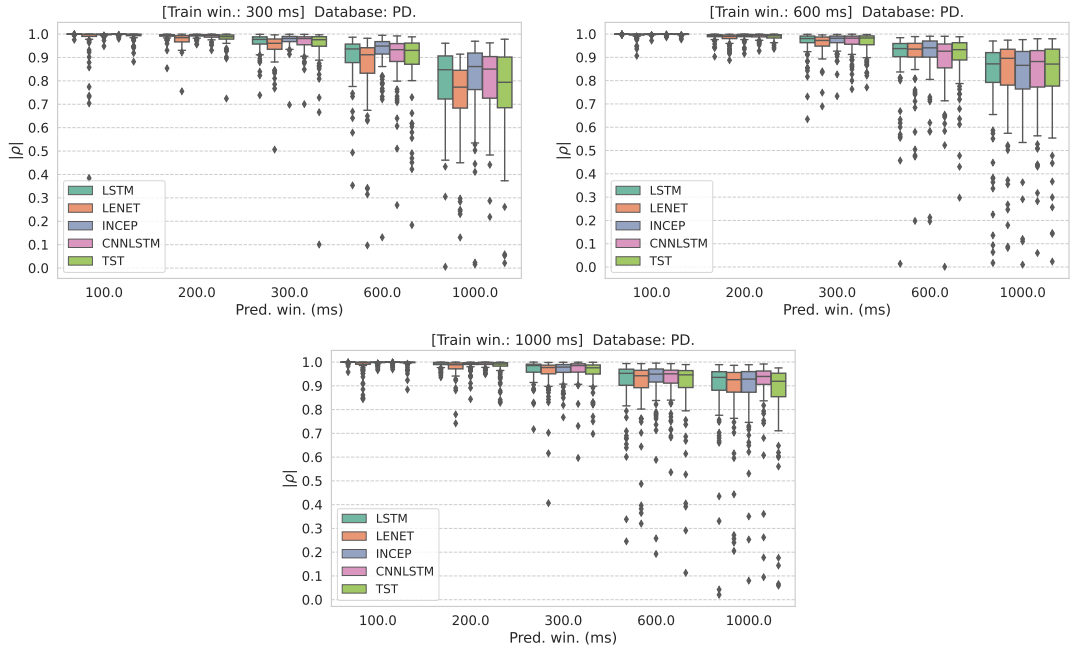


FIGURE 3.3: Boxplots of correlations obtained by the models for PD database, separated by training and prediction windows.

3.1.3 ET+PD dataset

Train Database: ET + PD - Test Database: patients ET-P001 and PD-P010

Train win. (ms)	Pred. win. (ms)	LSTMp	LeNetp	incepCNNp	CNN-LSTMp	TSTp
300	100	0.9939$^{\bullet\dagger\diamond\star}$	0.9286$^{*\dagger\diamond\star}$	0.9906$^{*\bullet\diamond\star}$	**0.9963**$^{*\bullet\dagger\star}$	0.9594$^{*\bullet\dagger\diamond}$
	200	0.9643$^{\bullet\dagger\star}$	0.9292$^{*\dagger\diamond\star}$	**0.9710** $^{*\bullet\diamond\star}$	0.9628$^{\bullet\dagger\star}$	0.9371$^{*\bullet\dagger\diamond}$
	300	0.9131$^{\bullet\dagger\star}$	0.8837$^{*\dagger\diamond}$	**0.9319**$^{*\bullet\diamond\star}$	0.9223$^{\bullet\dagger\star}$	0.8920 $^{*\dagger\diamond}$
	600	0.8316$^{\bullet\dagger}$	0.7853$^{*\dagger\diamond\star}$	**0.8466**$^{*\bullet\diamond\star}$	0.8300$^{\bullet\dagger}$	0.7997$^{\bullet\dagger}$
	1000	0.6676$^{\dagger\diamond}$	0.6436$^{\dagger\diamond}$	**0.7191**$^{*\bullet\diamond\star}$	0.6996$^{*\bullet\dagger\star}$	0.6504$^{\dagger\diamond}$
600	100	0.9957$^{\bullet\diamond}$	0.9789$^{*\dagger\diamond\star}$	0.9939$^{\bullet\diamond\star}$	**0.9959**$^{*\bullet\dagger\star}$	0.9932$^{\bullet\dagger\diamond}$
	200	0.9765$^{\bullet\star}$	0.9565$^{*\dagger\diamond\star}$	0.9803$^{\bullet\star}$	**0.9807**$^{\bullet\star}$	0.9704$^{*\bullet\dagger\diamond}$
	300	0.9396$^{\bullet\dagger}$	0.9252$^{*\dagger\diamond}$	**0.9496**$^{*\bullet\diamond\star}$	0.9449$^{\bullet\dagger\star}$	0.9322$^{\dagger\diamond}$
	600	0.8346†*	0.8260†*	**0.8548**$^{*\bullet\diamond}$	0.8241†*	0.8486$^{*\bullet\diamond}$
	1000	0.6836$^{\bullet\dagger\diamond\star}$	0.7097*	**0.7383**$^{*\diamond}$	0.7255*†	0.7222*
1000	100	0.9933$^{\bullet\diamond}$	0.9425$^{*\dagger\diamond\star}$	0.9943$^{\bullet\diamond}$	**0.9966**$^{*\bullet\dagger\star}$	0.9896$^{\bullet\diamond}$
	200	0.9649$^{\bullet\dagger\diamond}$	0.9316$^{*\dagger\diamond\star}$	0.9705$^{*\bullet\diamond\star}$	**0.9798**$^{*\bullet\dagger\star}$	0.9574$^{\bullet\dagger\diamond}$
	300	0.9245$^{\bullet\diamond}$	0.8865$^{*\dagger\diamond\star}$	0.9302$^{\bullet\star}$	**0.9317**$^{*\bullet\star}$	0.9149$^{\bullet\dagger\diamond}$
	600	0.8327$^{\bullet\dagger}$	0.8049$^{*\dagger\diamond\star}$	**0.8513**$^{*\bullet\star}$	0.8486$^{\bullet}$	0.8236$^{\bullet\dagger}$
	1000	0.8061	0.7841$^{\dagger\diamond\star}$	0.8103$^{\bullet}$	**0.8126**$^{\bullet}$	0.8004$^{\bullet}$
# Times as best		0	0	**8**	7	0

TABLE 3.4: Mean correlation coefficient (ρ) comparison between models for database ET+PD. Symbols to denote presence of statistical significance ($p < 0.05$) with respect to the models: LSTMp ($*$), LeNetp ($\bullet$), incepCNNp ($\dagger$), CNN-LSTMp ($\diamond$), TSTp ($\star$).

Some differences have been noted between the results using PD data and those for the ET database. This could be a consequence of the different nature of the signals which, while always associated with tremor activity, they correspond to distinct

diseases. In order to analyse the effect of having tremor signals from both diseases together, it is interesting to assess the performance of models that have been trained using information from both sources. Table 3.4 shows the mean correlation results for the case where all models were trained using a combined database of ET and PD data. The structure and notation is still the same, as explained for the other cases. This time, the analysis of the results is more focused on the main differences with respect to the previous databases, highlighting the situations where clear improvements or decrease in performance exist.

For all training windows and the three first output lengths (100-300 ms), all models obtain average correlations over 0.9, being the incepCNNp and CNN-LSTMp the best ones, depending on the specific scenario. This supposes an improvement with respect to the ET database results, where mean correlations for the same situations were oscillating around 0.85. However, the first noticeable improvement appears with 600 ms prediction window. The results are in between the ones obtained for the ET database (≈ 0.75) and the ones from PD database (> 0.90), since in all cases the average correlation of the best model is around 0.85. The second most prominent improvement corresponds to the longest prediction horizon (1000 ms). For input lengths of 300 ms and 600 ms, the mean correlations are now above 0.7, contrary to the results corresponding to the ET database. However, the correlations for this database were already close to 0.7, so for these training windows the increase in performance is small. Now, for the last input window of 1000 ms, the mean correlation for the best model reaches 0.8 which supposes a worse performance, in comparison with PD database, but, regarding ET database, there is an improvement of about 17% with respect to the mean correlation value.

Figure 3.4 shows the boxplots of the correlation values obtained by each model for every combination of train/target windows. The behaviour of the models is now similar to the one observed for the PD database: the longer the training length, the better the performance. Also, it can be noticed how the values are getting greater and more concentrated around the median, as the training window increases. It is important to note that the test set in this case contains samples from ET and PD in the same proportion (50%/50%). Therefore, higher values of mean correlations are not only consequence of having PD sequences in which models have great performance.

Clearly, having information from both diseases is helping the models to generally perform more accurate predictions.

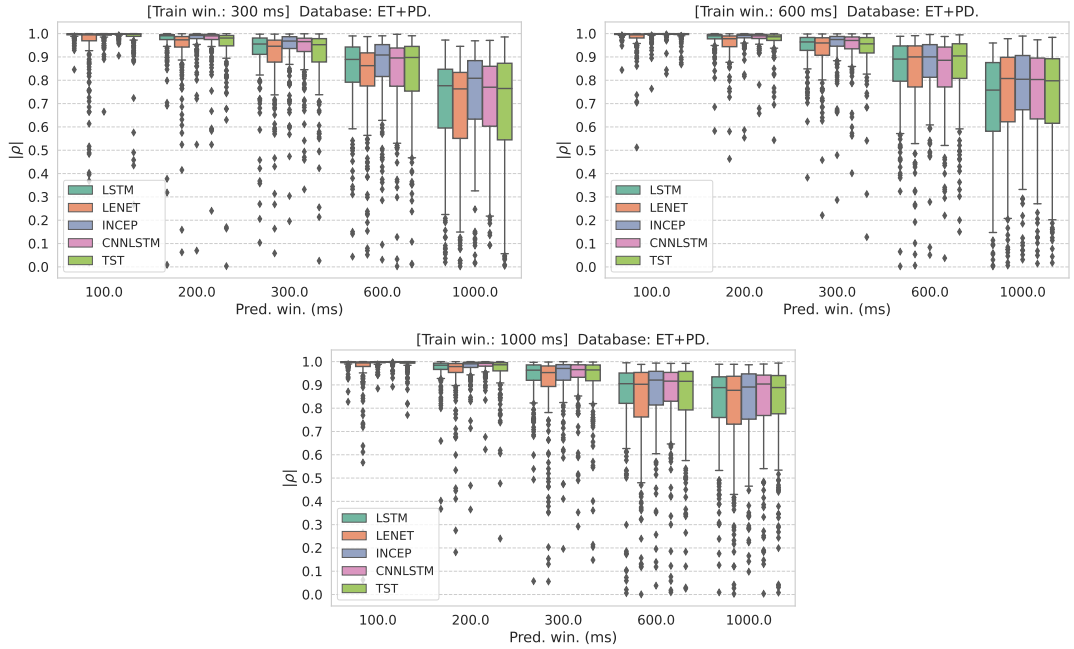


FIGURE 3.4: Boxplots of correlations obtained by the models for ET+PD database, separated by training and prediction windows.

3.1.4 Cross-testing

Train Database: ET - Test Database: patient PD-P010

Train win. (ms)	Pred. win. (ms)	LSTMp	LeNetp	incepCNNp	CNN-LSTMp	TSTp
300	100	0.9902$^{\bullet\dagger}$	0.9677$^{*\dagger\diamond\star}$	**0.9959**$^{*\bullet\diamond\star}$	0.9892$^{\bullet\dagger}$	0.9874$^{\bullet\dagger}$
	200	0.9834$^{\bullet\dagger}$	0.9629$^{*\dagger\diamond\star}$	**0.9901**$^{*\bullet\diamond\star}$	0.9822$^{\bullet\dagger}$	0.9782$^{\bullet\dagger}$
	300	0.9372†*	0.9324†*	**0.9662**$^{*\bullet\diamond\star}$	0.9398†	0.9524$^{*\bullet\dagger}$
	600	0.8413$^{\bullet\dagger\star}$	0.8637*†*	**0.9073**$^{*\bullet\diamond\star}$	0.8594†*	0.8762$^{*\bullet\dagger\diamond}$
	1000	0.5862$^{\bullet\dagger\star}$	0.7349$^{*\diamond}$	0.7417$^{*\diamond}$	0.6202$^{\bullet\dagger\star}$	**0.7545**$^{*\diamond}$
600	100	**0.9978** $^{\bullet}$	0.9823$^{*\dagger\diamond\star}$	0.9976$^{\bullet}$	0.9970$^{\bullet}$	0.9972$^{\bullet}$
	200	0.9701$^{\dagger\diamond\star}$	0.9702†*	**0.9873**$^{*\bullet\diamond\star}$	0.9754*†	0.9831$^{*\bullet\dagger}$
	300	0.9288$^{\bullet\dagger\star}$	0.9445$^{*\dagger\diamond\star}$	0.9583$^{*\bullet\diamond}$	0.9250$^{\bullet\dagger\star}$	**0.9595**$^{*\bullet\diamond}$
	600	0.8220$^{\bullet\dagger\star}$	0.8637$^{*\diamond\star}$	0.8741$^{*\diamond}$	0.8165$^{\bullet\dagger\star}$	**0.8844**$^{*\bullet\diamond}$
	1000	0.6416$^{\bullet\dagger\star}$	0.7656$^{*\dagger\diamond}$	0.7275$^{*\bullet\diamond\star}$	0.6792$^{\bullet\dagger\star}$	**0.7816**$^{*\dagger\diamond}$
1000	100	0.9916$^{\bullet\diamond}$	0.9798$^{*\dagger\diamond\star}$	0.9929$^{\bullet\star}$	**0.9948**$^{*\bullet\star}$	0.9865$^{\bullet\dagger\diamond}$
	200	0.9669$^{\dagger\diamond\star}$	0.9657$^{\dagger\diamond\star}$	**0.9840** $^{*\bullet\star}$	0.9789$^{*\bullet}$	0.9708$^{*\bullet\dagger}$
	300	0.9079$^{\bullet\dagger\star}$	0.9374*†	**0.9568**$^{*\bullet\diamond}$	0.9298†*	0.9428$^{*\diamond}$
	600	0.7911$^{\bullet\dagger\diamond\star}$	0.8671$^{*\diamond}$	**0.8877**$^{*\diamond}$	0.8448$^{*\bullet\dagger\star}$	0.8748$^{*\diamond}$
	1000	0.7777$^{\bullet\dagger\star}$	0.8270 $^{*\dagger\diamond}$	**0.8571**$^{*\bullet\diamond}$	0.7847$^{\bullet\dagger\star}$	0.8452$^{*\diamond}$
# Times as best		1	0	**9**	1	4

TABLE 3.5: Mean correlation coefficient (ρ) comparison between models for ETPD cross-testing. Symbols to denote presence of statistical significance ($p < 0.05$) with respect to the models: LSTMp ($*$), LeNetp ($\bullet$), incepCNNp ($\dagger$), CNN-LSTMp ($\diamond$), TSTp ($\star$).

The last part of this section is dedicated to analyse the cross-prediction experiments. As shown in the previous results, the combined usage of ET and PD sequences provides better performance in terms of correlations obtained by the models. This suggests that the models benefit from being trained with data from both diseases. However, in order to properly analyse and discuss the individual effects of one database with respect to the other, it is interesting to test the models trained on ET (or PD) on the test set of the other disease. Table 3.5 and Table 3.6 show the results in this matter. The first one corresponds to the mean correlation values obtained for models trained using ET data and tested in PD. The second correspond to the complementary situation: models trained using PD data and testes in ET.

Train Database: PD - Test Database: patient ET-P001

Train win. (ms)	Pred. win. (ms)	LSTMp	LeNetp	incepCNNp	CNN-LSTMp	TSTp
300	100	$0.9896^{\bullet\diamond\star}$	$0.9065^{*\dagger\diamond\star}$	$\mathbf{0.9929}^{\bullet\star}$	$0.9916^{*\bullet\star}$	$0.9415^{*\bullet\dagger\diamond}$
	200	$0.9340^{\dagger\diamond}$	$0.8968^{\dagger\diamond}$	$\mathbf{0.9601}^{*\bullet\diamond\star}$	$0.9437^{*\bullet\dagger\star}$	$0.8914^{\dagger\diamond}$
	300	$0.8706^{\dagger}$	$0.8439^{\dagger}$	$\mathbf{0.9226}^{*\bullet\diamond\star}$	$0.8847^{\dagger}$	$0.8483^{\dagger}$
	600	$0.7340^{\dagger}$	$0.7057^{\dagger}$	$\mathbf{0.7866}^{*\bullet\star}$	0.7605	$0.7125^{\dagger}$
	1000	$0.5682^{\dagger}$	$0.5704^{\dagger}$	$\mathbf{0.6340}^{*\bullet\star}$	$0.5972^{\star}$	$0.5528^{\dagger\diamond}$
600	100	$0.9795^{\bullet}$	$0.9824^{*\dagger\diamond\star}$	$0.9897^{\bullet\star}$	$\mathbf{0.9909}^{\bullet}$	$0.9898^{\bullet\dagger}$
	200	$0.9474^{\dagger\diamond}$	$0.9493^{\dagger\diamond}$	$\mathbf{0.9737}^{*\bullet\diamond\star}$	$0.9684^{*\bullet\dagger\star}$	$0.9560^{\dagger\diamond}$
	300	$0.9051^{\dagger}$	$0.9041^{\dagger\diamond}$	$\mathbf{0.9282}^{*\bullet\diamond\star}$	$0.9157^{\bullet\dagger}$	$0.9052^{\dagger}$
	600	$0.7613^{\dagger}$	$0.7734^{\dagger}$	$\mathbf{0.8104}^{*\bullet\star}$	0.8000	$0.7881^{\dagger}$
	1000	$0.6288^{\dagger}$	0.6380	$\mathbf{0.6814}^{*\diamond}$	$0.6433^{\dagger}$	0.6710
1000	100	$0.9835^{\bullet\diamond\star}$	$0.9211^{*\dagger\diamond\star}$	$0.9917^{\bullet\star}$	$\mathbf{0.9934}^{*\bullet\star}$	$0.9808^{*\bullet\dagger\diamond}$
	200	$0.9425^{\bullet\diamond}$	$0.9066^{*\dagger\diamond}$	$0.9553^{\bullet\star}$	$\mathbf{0.9601}^{*\bullet\star}$	$0.9414^{\dagger\diamond}$
	300	$0.8850^{\bullet}$	$0.8535^{*\dagger\diamond}$	$0.8920^{\bullet}$	$\mathbf{0.8977}^{\bullet}$	0.8755
	600	0.7654	0.7431	0.757	**0.7666**	0.7448
	1000	0.6882	0.7179	0.7062	**0.7215**	0.6980
# Times as best		0	0	**9**	6	0

TABLE 3.6: Mean correlation coefficient (ρ) comparison between models for PDET cross-testing. Symbols to denote presence of statistical significance ($p < 0.05$) with respect to the models: LSTMp ($*$), LeNetp ($\bullet$), incepCNNp ($\dagger$), CNN-LSTMp ($\diamond$), TSTp ($\star$).

The results show that the models are predicting with better correlations ET signals when they were trained with PD data. If the results of Table 3.2 are compared to the ones of the PDET cross-prediction (Table 3.6), they happen to be better, suggesting that models benefit from being trained in PD data. It can be hypothesized that the reason to this behaviour is related to the quality of the training data, regarding the number of differentiable tremor events that can be found in the sequences. Since an heuristic method was applied to tag and filter the recordings, it is possible that the quality of one of the datasets in that sense is poorer. In this case, results suggest that the ET dataset might contain less useful information regarding tremor events

than the PD dataset. On the other hand, the ETPD cross-prediction shows mean correlation values always above 0.75 for the best performing models. Thus, ET data can be used to predict PD sequences but less accurately than using samples from the same disease.

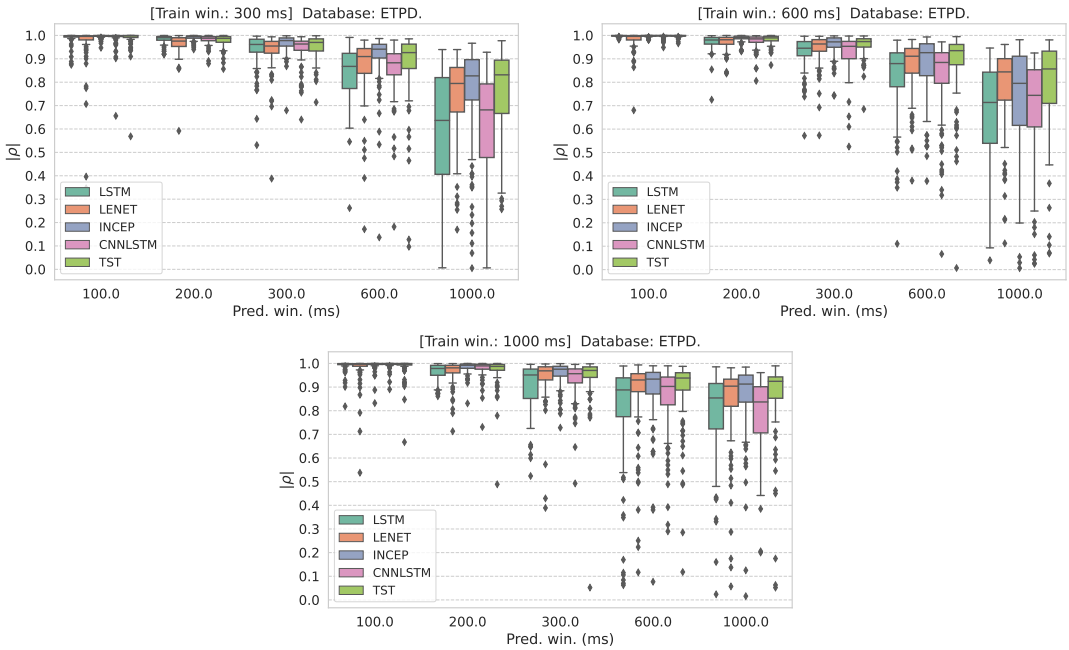


FIGURE 3.5: Boxplots of correlations obtained by the models for ETPD cross-testing, separated by training and prediction windows.

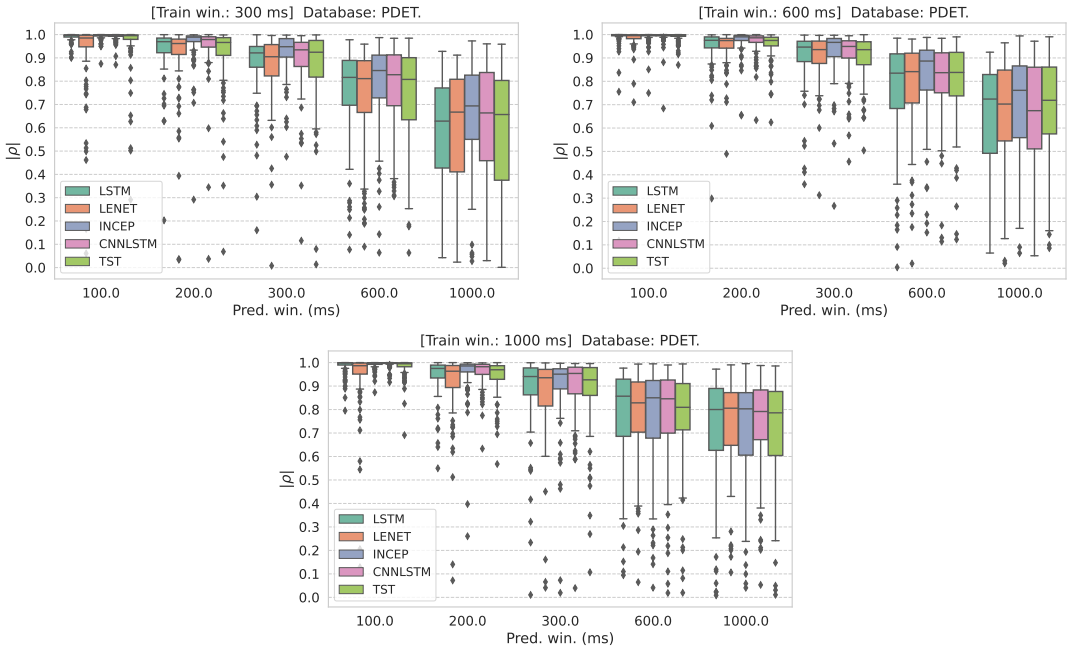


FIGURE 3.6: Boxplots of correlations obtained by the models for PDET cross-testing, separated by training and prediction windows.

Figures 3.5 and 3.6 show the boxplots corresponding to the correlations obtained by each model for each situation in the cross-prediction experiments. The poor performance of the LSTMp model is seen in the distributions, specially for the largest output windows.

3.2 EMG+KIN signal prediction with LSTM

As commented in Section 1.2, no studies were found at the time of this investigation that made prediction experiments of tremor-associated signals using both information. Some DL architectures make this kind of approaches possible, since they are capable of working simultaneously with various features for every time steps. One of these architectures is, for instance, the LSTM, which can take for each time step not only the values of the time series itself but, also, additional values that could represent complementary or associated information to the signal. Moreover, these additional values could correspond to another time series that is simultaneous to the other. The objective of this part of the investigation is to train a LSTM model that takes both kinematic and EMG signals to predict the next samples of the EMG envelope. Therefore, the target sequence is the same as for previous experiments but, this time, the model will have information of the associated kinematic signal to the EMG sequence during training. The main advantage of kinematic signals is that they are significantly less noisy than EMG, because the IMUs are recording just movement.

The same LSTM model as the one for EMG signal prediction (LSTMp) was employed in this experiment. Table 3.7 shows the results obtained regarding the mean correlation coefficient, MSE and RMSE. It should be noted that the values of MSE and RMSE are high because of the lack of signal normalization. The most important metric for this investigation, as explained in previous sections, is the mean correlation coefficient.

The metrics show that the LSTM model is providing acceptable performance ($\rho \approx 0.8$) for the first three prediction windows (100, 200, 300 ms) in all cases. Then, for the longest output windows (600, 1000 ms) the mean correlations are always below 0.7. Moreover, even when the LSTM is trained using the longest input window, it is only able to achieve a mean correlation of 0.62 for 1000 ms prediction horizon.

Train Database: ET - Test Database: patient ET-P001

Train win. (ms)	Pred. win. (ms)	Hidden size	Learning rate	ρ	MSE	RMSE
300	100	35	0.001	0.9817	31.2603	2.5811
	200	50	0.001	0.8993	100.1529	5.7557
	300	50	0.0001	0.8164	133.0573	7.3333
	600	50	0.0001	0.6474	183.7293	9.2523
	1000	50	0.0001	0.5063	267.7385	11.3056
600	100	50	0.0005	0.9795	33.5611	2.5782
	200	35	0.001	0.9454	55.6044	4.6832
	300	35	0.0005	0.8321	84.2085	6.9043
	600	35	0.0005	0.6729	244.0873	10.3264
	1000	50	0.001	0.5164	351.7870	12.4779
1000	100	50	0.001	0.9819	15.9296	1.8218
	200	35	0.001	0.8953	102.4482	5.8664
	300	50	0.001	0.7941	117.1362	7.1764
	600	35	0.0005	0.6650	192.8847	9.5610
	1000	35	0.0001	0.6260	171.3992	8.8281

TABLE 3.7: Prediction results from LSTMp model trained in extended database ET (kinematic and EMG signals).

Now, if these results are compared to those obtained when training the LSTM using only EMG data (see Table 3.2) a noticeable decrease regarding the mean correlations can be seen, specially for the longest output windows. For intermediate prediction horizons, a similar behaviour is noticed: the model works better when it is trained using also intermediate input windows. Nonetheless, the metrics are worse in general and very far from the results that were obtained using the combined ET+PD database or even cross-prediction. This suggests that the addition of kinematic information is not only unhelpful but, also, it hinders the training process of the model and reduces its predictive capabilities, regarding the EMG target sequences. Nevertheless, it should be remarked that these results are conditioned to the data used in this investigation.

3.3 Discussion

The investigation conducted in this Master thesis has compared the performances of five DL models based on different architectures and designs. Based on the literature, a 2-layer LSTM model (LSTMp) was employed as the baseline reference to compare the possible improvements of other approaches. The results obtained showed that this simple architecture can be outperformed in the prediction task by other more complex models, which add convolutional and attention structures. Moreover, it

was shown that the usage of convolutional modules is beneficial for the time series forecasting of tremor-associated EMG signals. The most complex CNN-based model (incepCNNp), inspired by the architecture of Inception modules, achieved the best performance in most scenarios throughout all databases. Thus, it is fair to consider incepCNNp as the overall best performing model but it should be remarked that all models are generally close in terms of performance, even if statistical significance was found.

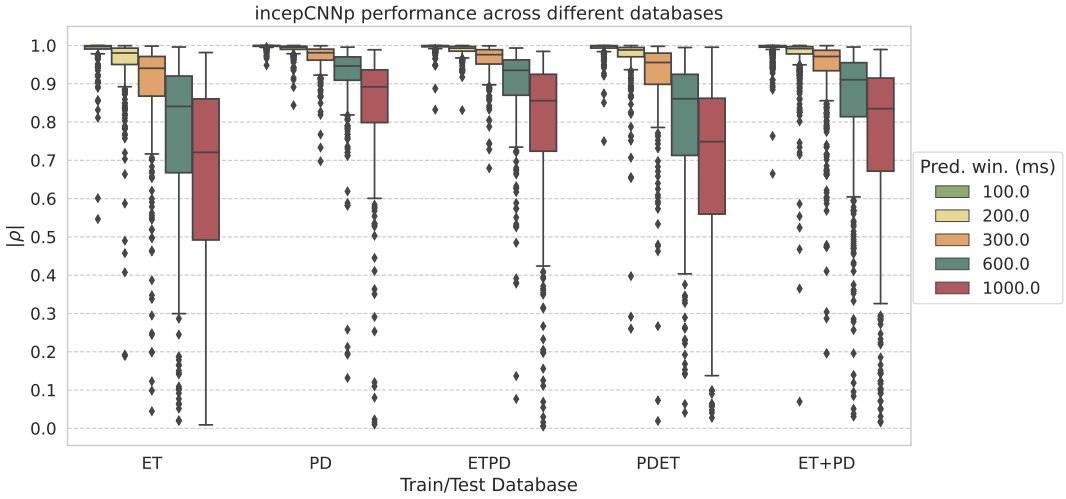


FIGURE 3.7: Boxplots of correlations obtained by incepCNNp model, separated by prediction windows and databases. For each situation, correlations from the three training windows were included together.

The worst results were obtained using the simplest CNN-based model (LeNetp), which suggests that, when the model is simple, it is better to have sequence-oriented structures, otherwise, parallel feature extraction techniques are insufficient. On the other hand, the combined usage of recurrent and convolutional modules did not suppose a noticeable improvement in the performance. Even, results showed that the CNN-LSTMp model had worse generalization capabilities, as in the cross-prediction experiments a general decrease in the mean correlations was noticed. Nonetheless, this might be a consequence inherited from the LSTM architecture, because this decrease in performance was also evident for the LSTMp model. This suggests that, in this case, a recurrent architecture is more dependent on the specific disease in which it is trained, while feature extraction modules have better generalization capabilities. It can be hypothesized that, in this case, CNNs can extract several and complex statistical properties of the signals, by means different feature maps and a deeper

architecture. These patterns are computed using all the information at once in parallel and might be more generalizable than those obtained using recurrent structures, which are limited by sequential processing.

Regarding the Transformer-based model (TSTp), its performance is usually better than that of the LSTMp, but not as good as those of incepCNNp and CNN-LSTMp. Thus, attention mechanisms can outperform recurrent structures in some cases for tremor signal prediction task, but convolutional architectures appear to be a better alternative. Nevertheless, both CNNs and Transformers overcome one of the biggest drawbacks of employing recurrent approaches, which is the limitations of sequential processing. Convolutions and attention are applied in parallel to the whole sequences, providing faster and more computationally efficient training and inference, which leads to better scalability. Consequently, CNN-based architectures could be the best option for tremor signal prediction, based on the results of this investigation.

The influence of training with data from different diseases on the performance of the models was studied in two perspectives: using a combined database with both diseases and evaluating the models in a cross-prediction scenario. The results regarding the mixed database ET+PD showed that models benefit from having information of the two diseases. Mean correlations were improved in comparison with the case of ET database, which is the dataset where the performances are the worst. It is worth noting that the differences in terms of prediction accuracy between ET and PD database are significant. This suggests that the data source is influencing noticeably the training process. It is true that the PD database is larger that ET one but it is difficult to assume that the difference regarding the number of samples is the main cause to such an improvement. Probably, the disease and the acquisition methodology have also some influence in this regard. It should be remarked that the data from ET and PD patients was not provided by the same source and they were acquired in different circumstances, without the possibility to apply a proper inter-patient normalization, such that the characteristics of the signals in amplitude and noise were equivalent. Having said that, it is important to note that the models could overcome this normalization issue and it seems that they were even capable

of working with signals from different sources and, moreover, they can improve the results if they are trained with the information mixed.

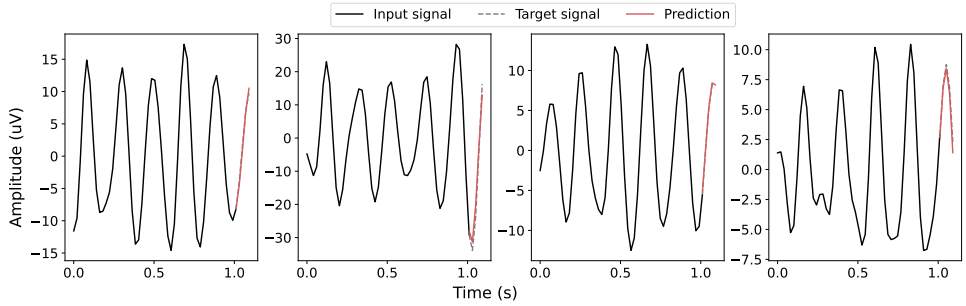
On the other hand, the cross-prediction experiments have shown that, to perform accurate predictions in signals from one disease, it is not necessary to train the models with the same information. They are capable of learning patterns that, more than overfitting the signals from one of the diseases, they generalize at some grade tremor activity, even if the characteristics regarding frequency and amplitude are different. It was also noticed that the performance of the models in ET signals can be improved by using only PD information in the training process, while models trained in ET data provided worse but similar results on PD test set, than models trained in PD database. Consequently, with the data and methodology employed in this Master thesis, it is demonstrated that, for the case of ET and PD, the two most common tremor-associated disorders, DL models can work indistinctly with signals from both sources, providing promising results.

Other relevant factors regarding time series forecasting are the training and prediction windows. In this investigation, three input lengths and five target windows of increasing size were employed. Regarding the influence of the training signal length, it was noticed that its effects for the first three output windows (100, 200, 300 ms) is minimum. The accuracies in the predictions were highly similar using the three input lengths for all models. Apparently, it begins to be more important when the prediction window is longer than 600 ms, where a noticeable improvement is seen when the models are trained with the longest input window (1000 ms). This result is expected as for longer predictions, where the variability of the signal increases, models benefit from having seen the behaviour of the signal during more time.

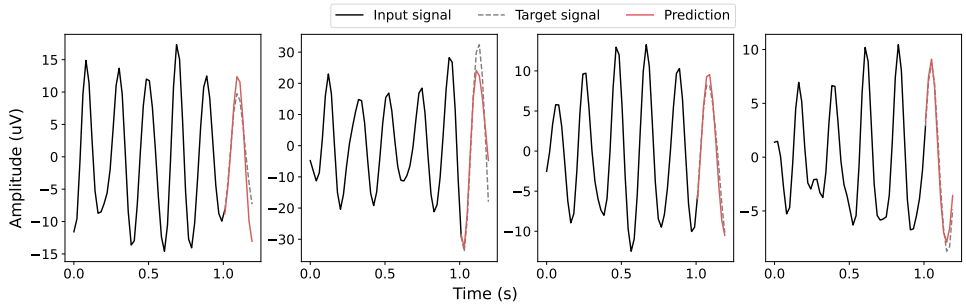
On the other hand, the influence of the prediction window in the results is clear: as far as the models must predict more future samples, their performances are lower and the variability in the results increases, as it is shown in the boxplots of correlations presented before. Nonetheless, for the first four prediction horizons (100, 200, 300, 600 ms), the mean correlations obtained by the best model in the ET+PD database were always above 0.8, which is a remarkable prediction performance. For

longer target windows, a noticeable decrease in accuracy is observed and the distributions of the correlations are wider. Also, there is less evidence of statistical significances between the results from all models, which suggests that they are reaching a bottleneck. This can be expected regarding the nature of the data. Besides it has some general features, tremor activity is the consequence of a neurological disorder, in which different factors have to be taken into account at the time of characterizing its associated biological signals. These factors might include non-stationary and random components in time which, at the end, cause bottlenecks and limitations regarding the predictive capabilities of any system. Nonetheless, correlation results show that the models are capable of fitting both amplitude and phase of tremor activity with high accuracy ($\rho > 0.8$) for most combinations of training and target windows (see Figure 3.8).

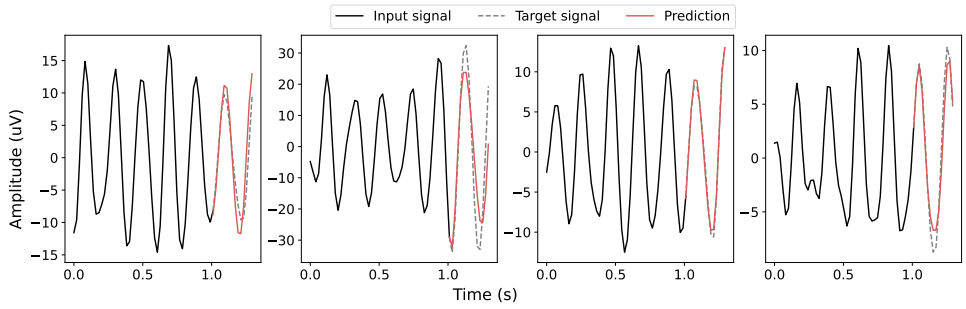
Finally, the combined usage of kinematic and EMG information was assessed in terms of prediction performance for tremor-associated EMG signals. It was hypothesized that the addition of another source of information from tremor activity might help the models to learn better patterns to perform the predictions more accurately. However, it was observed that the results of an LSTM trained with both kinematic and EMG signals were worse than those of the models trained only in EMG data. Then, regarding the data employed in this study, kinematic information seems to be unhelpful to the models, moreover, it hinders the prediction task. As it was already commented, kinematic signals are the mechanic translation of EMG activity and, thus, it should provide the same information but delayed. It is reasonable to assume that, contrary to what was expected, kinematic signals are adding a layer of complexity, instead of helping the models to learn deeper patterns. Nevertheless, these results are only applicable to the ET database, since no kinematic information could be acquired from PD patients. In order to generalize these conclusions to other diseases, extended experiments should be conducted.



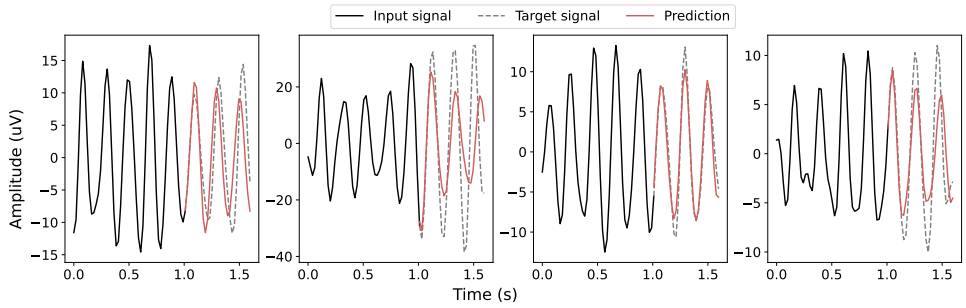
(A) Pred. win: 100 ms.



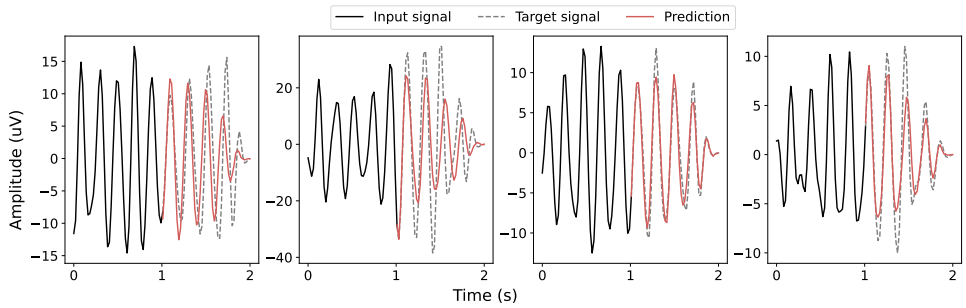
(B) Pred. win: 200 ms.



(C) Pred. win: 300 ms.



(D) Pred. win: 600 ms.



(E) Pred. win: 1000 ms.

FIGURE 3.8: Prediction examples for ET+PD database and training window of 1000 ms using incepCNNp model.

Chapter 4

Conclusions

In this Master thesis, an in-depth research on the field of tremor-associated signal prediction was conducted. First of all, a simple preprocessing pipeline for the EMG signals was proposed, based on basic filtering and a downsampling procedure to reduce computational costs of training and inference. Compared to the SOTA, the designed pipeline is significantly simpler, while the results show that models are not suffering from an insufficient and incorrect signal processing. Afterwards, a wide variety of DL models has been proposed, including architectures and modules from different branches of DL, thus, extensively exploring the viability of using artificial intelligence to predict tremor signals. No other studies were found at the time of this Master thesis that involved this amount and variety of architectures, making an overall comparison between them and looking for the best possible approach. The results showed that the proposed incepCNNp model, entirely based on convolutional and linear layers, is the one that achieves the best performance, demonstrating that CNN-based approaches can be more powerful in time series forecasting of tremor-associated signals than traditional recurrent architectures. This performance was also assessed using information from various tremor disorders: ET and PD, which is a novelty with respect to the SOTA studies that were focused only in a single disease. It was shown that the combined usage of different sources of information can help to improve the prediction performance of the models. Furthermore, it was demonstrated that DL models are capable of performing accurate signal prediction on sequences from other diseases, different from those in which they were trained. This is a promising useful result, as it opens the possibility to create general models for tremor management, no matter the type of tremor it aims to handle. Regarding

the training and prediction windows, the results of this investigation showed that an input length of 300 ms is enough to obtain a correlation coefficient over 0.8 in the prediction of the next 600 ms window, when using the combined ET+PD database. These results are also very encouraging because, as typical tremor frequencies are between 4 and 12 Hz, it is crucial to obtain high accuracies for prediction windows $\leq$ 250 ms. Then, the results showed that this is affordable by using only a short input window of 300 ms, which corresponds to just 15 samples with a sampling rate of 50 Hz. Finally, this Master thesis included an additional experiment regarding the combined usage of kinematic and EMG signals for tremor prediction. This is also a novelty in the field, since no other studies were found in which authors employed both signals to perform the predictions. However, contrary to what it was expected, the addition of kinematic information to the EMG signals caused a decrease in performance, rather than improving the results. This suggests that it is better to use those signals separately to perform the predictions. Pascual-Valdunciel et al. [47] already showed that it is possible to obtain similar correlation metrics using kinematic signals. However, the usage of this type of information for tremor prediction has an important disadvantage: the signals do not correspond directly to the activity of the muscles, instead, they are the mechanic manifestation of biological activity. Therefore, there exists a loss in terms of information due to the transduction from EMG to kinematic activity. This step supposes a filter for the biological signals which may contain useful and deeper information regarding the characterization of tremor. Consequently, the usage of EMG signals, which are a direct translation of the activity from the nervous system, can be enriching for the learning stage and generalization capabilities of the models. However, the designing of an EMG-based approach has some drawbacks regarding the setup of the experiments and the necessity of having a minimum grade of signal to noise ratio in the acquisition.

The field of time series forecasting for pathological tremor signals using DL techniques has still a wide margin for improvement, even more regarding practical and real applications of these novel methods. It is increasing in popularity with the arrival of more powerful and efficient approaches based on artificial intelligence. This Master thesis contributes in several aspects to the investigation of automatic tremor management systems that, in the near future, could improve the life of people who

suffer from tremor disorders. Also, it can be another reference for studies regarding other medical topics, in which time series forecasting of biological signals is involved.

4.1 Future work

Further investigations in the same line of this Master thesis should be focused on:

- Study the effects of including an inter-patient normalization process to unify the signals from different diseases and sources.
- Extend the database with more patients, other types of tremor apart from Essential and Parkinsonian variants, and additional postures or activities during recording time.
- Given that CNN-based approaches are more powerful in this case, explore the possibility of improving the performance and find the best balance between complexity and rapid training and inference, by optimizing the proposed architectures.
- In-depth analysis of the CNN-based models to explore the feature maps and gain insights about what are the models learning from the data, which might help clinicians to better understand tremor characteristics.
- Implement the models in real-time tremor suppression devices to assess the feasibility of employing these approaches in a tremor management system.
- Combine the tremor prediction and detection models to create a complete tremor management pipeline.

References

[1] R Bhidayasiri, "Differential diagnosis of common tremor syndromes," *Postgrad. Med. J.*, vol. 81, no. 962, pp. 756–762, 2005.

[2] *Involuntary movements: Types, causes, and examples, stanford 25*, https://stanfordmedicine25.stanford.edu/the25/involuntary-movements-and-tremors.html, Accessed: 2023-5-19.

[3] G. Deuschl, J. Raethjen, M. Lindemann, and P. Krack, "The pathophysiology of tremor," *Muscle & Nerve*, vol. 24, no. 6, pp. 716–735, DOI: https://doi.org/10.1002/mus.1063. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mus.1063. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mus.1063.

[4] N. Kamble and P. K. Pal, "Tremor syndromes: A review," *Neurol. India*, vol. 66, no. Supplement, S36–S47, 2018.

[5] R. C. Helmich, I. Toni, G. Deuschl, and B. R. Bloem, "The pathophysiology of essential tremor and parkinson's tremor," *Curr. Neurol. Neurosci. Rep.*, vol. 13, no. 9, p. 378, 2013.

[6] E. D. Louis, "Clinical practice. essential tremor," *N. Engl. J. Med.*, vol. 345, no. 12, pp. 887–891, 2001.

[7] L. N. Clark and E. D. Louis, "Essential tremor," *Handb. Clin. Neurol.*, vol. 147, pp. 229–239, 2018.

[8] S. Sveinbjornsdottir, "The clinical symptoms of parkinson's disease," en, *J. Neurochem.*, vol. 139 Suppl 1, pp. 318–324, 2016.

[9] D. E. Haines and G. A. Mihailoff, *Fundamental Neuroscience for Basic and Clinical Applications*, 5th ed. Philadelphia, PA: Elsevier - Health Sciences Division, 2017.

[10] J Benito-León and M León-Ruiz, "Epidemiología del temblor esencial," es, *Rev. Neurol.*, vol. 70, no. 4, pp. 139–148, 2020.

[11] C Marras, J. C. Beck, J. H. Bower, *et al.*, "Prevalence of parkinson's disease across north america," en, *NPJ Parkinsons Dis.*, vol. 4, p. 21, 2018.

[12] M. Plumb and P. Bain, *Essential tremor: The facts*. London, England: Oxford University Press, 2006.

[13] P. Hedera, F. Cibulčík, and T. L. Davis, "Pharmacotherapy of essential tremor," *J. Cent. Nerv. Syst. Dis.*, vol. 5, pp. 43–55, 2013.

[14] A. Puschmann and Z. K. Wszolek, "Diagnosis and treatment of common forms of tremor," en, *Semin. Neurol.*, vol. 31, no. 1, pp. 65–77, 2011.

[15] J Kulisevsky, M. R. Luquin, J. M. Arbelo, *et al.*, "Enfermedad de parkinson avanzada. características clínicas y tratamiento (parte i)," *Neurologia*, vol. 28, no. 8, pp. 503–521, 2013.

[16] R. F. Dallapiazza, D. J. Lee, P. De Vloo, *et al.*, "Outcomes from stereotactic surgery for essential tremor," *J. Neurol. Neurosurg. Psychiatry*, vol. 90, no. 4, pp. 474–482, 2019.

[17] J. S. Perlmutter and J. W. Mink, "Deep brain stimulation," *Annu. Rev. Neurosci.*, vol. 29, no. 1, pp. 229–257, 2006.

[18] M Rueda, "Efectividad de la estimulación cerebral profunda de núcleo subtalámico en pacientes con enfermedad de parkinson: Experiencia de antioquía," *Acta neurol. colomb.*, 2016.

[19] S. Dosen, S. Muceli, J. L. Dideriksen, *et al.*, "Online tremor suppression using electromyography and low-level electrical stimulation," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 23, no. 3, pp. 385–395, 2015.

[20] F. B. A. Pascual-Valdunciel and J. Pons, "Motor inhibition elicited by electrical stimulation of afferent pathways and its application in tremor suppression."

[21] A. Pascual-Valdunciel, M. Gonzalez-Sanchez, S. Muceli, *et al.*, "Intramuscular stimulation of muscle afferents attains prolonged tremor reduction in essential tremor patients," *IEEE Trans. Biomed. Eng.*, vol. 68, no. 6, pp. 1768–1776, 2021.

[22] M Javidan, J Elek, and A Prochazka, "Attenuation of pathological tremors by functional electrical stimulation. II: Clinical evaluation," *Ann. Biomed. Eng.*, vol. 20, no. 2, pp. 225–236, 1992.

[23] D. M. Gillard, T Cameron, A Prochazka, and M. J. Gauthier, "Tremor suppression using functional electrical stimulation: A comparison between digital and analog controllers," *IEEE Trans. Rehabil. Eng.*, vol. 7, no. 3, pp. 385–388, 1999.

[24] L. Popović Maneski, N. Jorgovanović, V. Ilić, *et al.*, "Electrical stimulation for the suppression of pathological tremor," *Med. Biol. Eng. Comput.*, vol. 49, no. 10, pp. 1187–1193, 2011.

[25] J. L. Dideriksen, C. M. Laine, S. Dosen, *et al.*, "Electrical stimulation of afferent pathways for the suppression of pathological tremor," *Front. Neurosci.*, vol. 11, p. 178, 2017.

[26] L. P. Maneski, N. Jorgovanovic, V. Ilić, *et al.*, "Electrical stimulation for the suppression of pathological tremor," *Medical and Biological Engineering and Computing*, vol. 49, no. 10, pp. 1187–1193, Jul. 2011. DOI: 10.1007/s11517-011-0803-6. [Online]. Available: https://doi.org/10.1007/s11517-011-0803-6.

[27] A. D. Karamesinis, R. V. Sillitoe, and A. Z. Kouzani, "Wearable Peripheral Electrical Stimulation Devices for the Reduction of Essential Tremor: A Review," *IEEE Access*, vol. 9, pp. 80 066–80 076, Jan. 2021. DOI: 10.1109/access.2021.3084819. [Online]. Available: https://doi.org/10.1109/access.2021.3084819.

[28] I. Basu, D. Graupe, D. Tuninetti, *et al.*, "Pathological tremor prediction using surface electromyogram and acceleration: potential use in 'ON–OFF' demand driven deep brain stimulator design," *Journal of Neural Engineering*, vol. 10, no. 3, p. 036 019, May 2013. DOI: 10.1088/1741-2560/10/3/036019. [Online]. Available: https://doi.org/10.1088/1741-2560/10/3/036019.

[29] H. Jeon, W. Lee, H. Park, *et al.*, "Automatic classification of tremor severity in parkinson's disease using a wearable device," *Sensors (Basel)*, vol. 17, no. 9, 2017.

[30] M. N. Alam, B. Johnson, J. Gendreau, K. Tavakolian, C. Combs, and R. Fazel-Rezai, "Tremor quantification of parkinson's disease - a pilot study," in *2016 IEEE International Conference on Electro Information Technology (EIT)*, IEEE, 2016.

[31] R. LeMoyne, N. Tomycz, T. Mastroianni, C. McCandless, M. Cozza, and D. Peduto, "Implementation of a smartphone wireless accelerometer platform for establishing deep brain stimulation treatment efficacy of essential tremor with machine learning," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2015.

[32] A. C. A. de Araújo, E. G. d. R. Santos, K. S. G. de Sá, *et al.*, "Hand resting tremor assessment of healthy and patients with parkinson's disease: An exploratory machine learning study," *Front. Bioeng. Biotechnol.*, vol. 8, p. 778, 2020.

[33] N. Sriraam, "EMG-Based Essential Tremor Detection Using PSD Features With Recurrent Feedforward Back Propogation Neural Network," *International Journal of E-Health and Medical Communications*, vol. 12, no. 6, pp. 1–16, Nov. 2021. DOI: 10.4018/ijehmc.20211101.oa10. [Online]. Available: https://doi.org/10.4018/ijehmc.20211101.oa10.

[34] A. Pascual-Valdunciel, V. Lopo-Martínez, A. J. Beltrán-Carrero, *et al.*, "Classification of Kinematic and Electromyographic Signals Associated with Pathological Tremor Using Machine and Deep Learning," *Entropy*, vol. 25, no. 1, p. 114, Jan. 2023. DOI: 10.3390/e25010114. [Online]. Available: https://doi.org/10.3390/e25010114.

[35] R. A. Zanini, E. L. Colombini, and M. C. F. de Castro, "Parkinson's disease EMG signal prediction using neural networks," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, IEEE, 2019.

[36] A. Ibrahim, Y. Zhou, M. E. Jenkins, A. Luisa Trejos, and M. D. Naish, "The design of a parkinson's tremor predictor and estimator using a hybrid convolutional-multilayer perceptron neural network," *Annu Int Conf IEEE Eng Med Biol Soc*, vol. 2020, pp. 5996–6000, 2020.

[37] S. Shahtalebi, S. F. Atashzar, O. Samotus, R. V. Patel, M. S. Jog, and A. Mohammadi, "PHTNet: Characterization and deep mining of involuntary pathological hand tremor using recurrent neural network models," *Sci. Rep.*, vol. 10, no. 1, p. 2195, 2020.

[38] K. Cho, B. van Merrienboer, C. Gulcehre, *et al.*, *Learning phrase representations using rnn encoder-decoder for statistical machine translation*, 2014. arXiv: 1406.1078 [cs.CL].

[39] R. H. Chowdhury, M. B. I. Reaz, M. T. Ali, A. A. A. Bakar, K. Chellappan, and T.-G. Chang, "Surface Electromyography Signal Processing and Classification Techniques," *Sensors*, vol. 13, no. 9, pp. 12 431–12 466, Sep. 2013. DOI: 10.3390/s130912431. [Online]. Available: https://doi.org/10.3390/s130912431.

[40] J. Karl, *Introduction to Digital Signal Processing*, en. San Diego, CA: Academic Press, 1989.

[41] A. V. Oppenheim, *Discrete-Time Signal Processing*. Pearson Education, Jan. 2014.

[42] D. A. Lewis, "Quaternion Algebras and the Algebraic Legacy of Hamilton's Quaternions," *Iris Mathematical Society*, vol. 0057, pp. 41–64, Jan. 2006. DOI: 10.33232/bims.0057.41.64. [Online]. Available: https://doi.org/10.33232/bims.0057.41.64.

[43] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. John Wiley and Sons, Apr. 1996.

[44] P. Welch, "The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms," *IEEE Transactions on Audio and Electroacoustics*, vol. 15, no. 2, pp. 70–73, 1967. DOI: 10.1109/TAU.1967.1161901.

[45] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[46] N. Adaloglou and S. Karagiannakos, "Recurrent neural networks: Building a custom lstm cell," *https://theaisummer.com/*, 2020. [Online]. Available: https://theaisummer.com/understanding-lstm.

[47] A. Pascual-Valdunciel, V. Lopo-Martínez, R. Sendra-Arranz, *et al.*, "Prediction of pathological tremor signals using long short-term memory neural networks," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 12, pp. 5930–5941, 2022. DOI: 10.1109/JBHI.2022.3209316.

[48] K. O'Shea and R. Nash, *An introduction to convolutional neural networks*, 2015. arXiv: 1511.08458 [cs.NE].

[49] S. Albelwi and A. Mahmood, "A framework for designing the architectures of deep convolutional neural networks," *Entropy*, vol. 19, no. 6, 2017, ISSN: 1099-4300. DOI: 10.3390/e19060242. [Online]. Available: https://www.mdpi.com/1099-4300/19/6/242.

[50] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017. DOI: 10.21629/JSEE.2017.01.18.

[51] S. Bai, J. Z. Kolter, and V. Koltun, *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling*, 2018. arXiv: 1803.01271 [cs.LG].

[52] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[53] S. Roy, I. Kiral-Kornek, and S. Harrer, *Chrononet: A deep recurrent neural network for abnormal eeg identification*, 2018. DOI: 10.48550/ARXIV.1802.00308. [Online]. Available: https://arxiv.org/abs/1802.00308.

[54] C. Szegedy, W. Liu, Y. Jia, *et al.*, *Going deeper with convolutions*, 2014. arXiv: 1409.4842 [cs.CV].

[55] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2017. arXiv: 1706.03762 [cs.CL].

[56] D. Soydaner, "Attention mechanism in neural networks: Where it comes and where it goes," *Neural Computing and Applications*, vol. 34, no. 16, pp. 13 371–13 385, 2022. DOI: 10.1007/s00521-022-07366-3. [Online]. Available: https://doi.org/10.1007%2Fs00521-022-07366-3.

[57] J. Alammar, *The Illustrated Transformer*, Accessed Jun. 15, 2023. [Online]. Available: https://jalammar.github.io/illustrated-transformer/.

[58] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021. arXiv: 2010.11929 [cs.CV].

[59] S. Li, X. Jin, Y. Xuan, *et al.*, *Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting*, 2020. arXiv: 1907.00235 [cs.LG].

[60] S. Liu, H. Yu, C. Liao, *et al.*, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=0EXmFzUn5I.

[61] H. Wu, J. Xu, J. Wang, and M. Long, *Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting*, 2022. arXiv: 2106.13008 [cs.LG].

[62] H. Zhou, S. Zhang, J. Peng, *et al.*, *Informer: Beyond efficient transformer for long sequence time-series forecasting*, 2021. arXiv: 2012.07436 [cs.LG].

[63] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, *Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting*, 2022. arXiv: 2201.12740 [cs.LG].

[64] A. Zeng, M. Chen, L. Zhang, and Q. Xu, *Are transformers effective for time series forecasting?* 2022. DOI: 10.48550/ARXIV.2205.13504. [Online]. Available: https://arxiv.org/abs/2205.13504.

[65] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, *A time series is worth 64 words: Long-term forecasting with transformers*, 2022. DOI: 10.48550/ARXIV.2211.14730. [Online]. Available: https://arxiv.org/abs/2211.14730.

[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[67] D. C. Montgomery and G. C. Runger, *Applied statistics and probability for engineers*, en. 2018.

[68] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Am. Stat. Assoc.*, vol. 32, no. 200, pp. 675–701, 1937.

[69] W. J. Conover, *Practical Nonparametric Statistics*, en, 3rd ed. Nashville, TN: John Wiley & Sons, 1998.

Appendix A

Vanishing Gradients problem

The vanishing gradients problem is a well known issue in the field of RNNs, which is related to the training process of neural networks that use gradient-based learning strategies. The problem consists on the gradients tending to zero as a consequence of training with long input sequences. In these cases, it might occur that the partial derivatives of the loss function, with respect to the weights, become very small, causing difficulties in the updating stage and, thus, in learning. An example of this problem, regarding the weights matrix **W**, is shown below, being the cases of matrix **U** and vector **b** similar to it.

Let the partial derivative of loss function $\mathcal{L}$ with respect to **W** for time step t be:

$$\begin{aligned} \frac{\partial \mathcal{L}_t}{\partial \mathbf{W}_t} &= \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t}\frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t}\cdots\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial \mathbf{W}_t} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t}\frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t}\left(\prod_{t=2}^{T}\frac{\partial \mathcal{L}_t}{\partial \mathbf{W}_t}\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}}\right)\frac{\partial \mathbf{y}_t}{\partial \mathbf{W}} \\ \mathbf{h}_t &= f(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}) \end{aligned} \tag{A.1}$$

Then, the evaluation of the partial derivatives in the sequential product leads to:

$$\frac{\partial \mathcal{L}_t}{\partial \mathbf{W}_t} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t}\frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t}\cdots\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial \mathbf{W}_t} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t}\frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t}\left(\prod_{t=2}^{T} f'(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b})\mathbf{U}\right)\frac{\partial \mathbf{y}_t}{\partial \mathbf{W}} \tag{A.2}$$

where f is the activation function.

The most popular activation functions are $\tanh$ and *sigmoid*. $\tanh$ maps their entries between -1 and 1, while the *sigmoid* does it between 0 and 1. The derivatives of these functions are upper-bounded to 1, which leads to the derivative of $\mathcal{L}$ tending to 0 for any t time step. Therefore, for very long sequences, it may happen that

gradients stop being updated or the update becomes significantly small.

$$\begin{aligned} &\frac{\partial \mathcal{L}_t}{\partial \mathbf{W}_t} \to 0 \\ &\mathbf{W}_{t+1} = \mathbf{W}_t - \alpha \frac{\partial \mathcal{L}_t}{\partial \mathbf{W}_t} \approx \mathbf{W}_t \end{aligned} \tag{A.3}$$

Appendix B

Results of Wilcoxon tests between prediction models

Train win. (ms)/Pred. win. (ms)	LSTM-LENET	LSTM-INCEP	LSTM-CNNLSTM	LSTM-TST	LENET-INCEP	LENET-CNNLSTM	LENET-TST	INCEP-CNNLSTM	INCEP-TST	CNNLSTM-TST
300/100	<1e-4	0.4873	0.3322	0.0354	<1e-4	<1e-4	0.0007	0.2594	<1e-4	0.001
300/200	0.0002	0.1648	0.1312	0.0053	0.0002	0.023	0.7544	0.0019	<1e-4	0.133
300/300	0.5822	0.1586	0.2638	0.518	0.1393	0.0743	0.7622	0.5247	0.021	0.0996
300/600	0.2438	0.1811	0.9534	0.1867	0.1777	0.2912	0.5158	0.6084	0.0348	0.2209
300/1000	0.9534	0.4724	0.9698	0.5682	0.1607	0.8339	0.7912	0.6824	0.1744	0.5202
600/100	<1e-4	0.2727	0.9178	0.036	0.0003	<1e-4	0.0092	0.3622	0.0077	0.0316
600/200	0.0052	0.0032	0.0789	0.5247	<1e-4	<1e-4	0.0504	0.5202	0.0006	0.0092
600/300	0.6278	0.0058	0.1973	0.0825	0.0316	0.536	0.1024	0.0322	0.0124	0.9097
600/600	0.4852	0.0142	0.7129	0.1312	0.0137	0.9671	0.2912	0.0711	0.7387	0.1777
600/1000	0.3862	0.536	0.1202	0.06	0.7701	0.1867	0.2249	0.5705	0.3104	0.8446
1000/100	<1e-4	0.2424	0.7939	<1e-4	<1e-4	<1e-4	0.0001	0.2184	0.0008	<1e-4
1000/200	0.0011	0.3357	0.8258	0.645	<1e-4	0.0001	0.0023	0.6699	0.0705	0.1961
1000/300	0.0058	0.7727	0.5941	0.518	0.0222	0.0954	0.0033	0.1766	0.9753	0.1243
1000/600	0.4873	0.3391	0.5988	0.0428	0.39	0.5136	0.334	0.552	0.3237	0.0643
1000/1000	0.6181	0.317	0.2682	1.0	0.7859	0.8689	0.3514	0.6849	0.3862	0.168

TABLE B.1: P-values of Wilcoxon tests between models for database ET.

Train win. (ms)/Pred. win. (ms)	LSTM-LENET	LSTM-INCEP	LSTM-CNNLSTM	LSTM-TST	LENET-INCEP	LENET-CNNLSTM	LENET-TST	INCEP-CNNLSTM	INCEP-TST	CNNLSTM-TST
300/100	0.0018	0.0008	0.1286	0.4895	<1e-4	<1e-4	<1e-4	0.0009	0.0028	0.8151
300/200	<1e-4	0.0002	0.7284	0.2315	<1e-4	<1e-4	0.0001	<1e-4	<1e-4	0.4073
300/300	0.3695	<1e-4	0.8018	0.0268	<1e-4	0.375	0.0054	<1e-4	0.0166	0.0619
300/600	0.0087	<1e-4	0.1856	0.0002	0.0003	0.1347	0.06	<1e-4	0.0075	0.0185
300/1000	<1e-4	<1e-4	0.2697	<1e-4	0.5566	<1e-4	0.2058	<1e-4	0.3604	<1e-4
600/100	<1e-4	0.896	0.3409	0.5941	<1e-4	<1e-4	<1e-4	0.6599	0.1193	0.606
600/200	0.9479	<1e-4	0.0233	<1e-4	<1e-4	0.1294	0.0002	<1e-4	0.0162	0.1024
600/300	0.0055	<1e-4	0.9069	<1e-4	0.0029	0.0049	0.001	<1e-4	0.9808	<1e-4
600/600	0.0002	<1e-4	0.7439	<1e-4	0.2551	<1e-4	0.0366	<1e-4	0.1722	<1e-4
600/1000	<1e-4	0.0015	0.0968	<1e-4	0.0155	<1e-4	0.4392	0.0366	0.0009	<1e-4
1000/100	0.0005	0.1269	0.0021	0.4917	<1e-4	<1e-4	0.0032	0.0658	0.0113	0.0013
1000/200	0.3938	<1e-4	0.0004	0.001	<1e-4	0.0025	0.0033	0.1032	0.0001	0.3104
1000/300	0.0079	<1e-4	0.0275	0.0001	0.0046	0.2355	0.1106	0.0014	0.1286	0.0126
1000/600	<1e-4	<1e-4	0.0727	<1e-4	0.2396	0.0209	0.1099	0.0014	0.9863	0.001
1000/1000	0.001	<1e-4	0.7596	<1e-4	0.0354	<1e-4	0.0653	<1e-4	0.937	<1e-4

TABLE B.2: P-values of Wilcoxon tests between models for ETPD cross-prediction. Models were trained using train set of ET database and tested on PD test samples.

Train win. (ms)/Pred. win. (ms)	LSTM-LENET	LSTM-INCEP	LSTM-CNNLSTM	LSTM-TST	LENET-INCEP	LENET-CNNLSTM	LENET-TST	INCEP-CNNLSTM	INCEP-TST	CNNLSTM-TST
300/100	<1e-4	<1e-4	0.0014	<1e-4	<1e-4	<1e-4	0.0624	0.0003	<1e-4	<1e-4
300/200	<1e-4	0.317	0.2082	<1e-4	<1e-4	<1e-4	0.0283	0.0161	<1e-4	0.0015
300/300	<1e-4	0.0055	0.6976	0.0074	<1e-4	<1e-4	0.0145	0.0039	<1e-4	0.06
300/600	0.0014	0.0003	0.8205	0.375	<1e-4	0.0004	0.0021	0.0006	0.0003	0.1449
300/1000	0.0003	0.0216	0.3391	0.0658	<1e-4	<1e-4	0.0679	0.0901	<1e-4	0.0546
600/100	<1e-4	0.0007	0.4809	0.0056	<1e-4	<1e-4	<1e-4	0.0004	0.8906	0.0224
600/200	<1e-4	0.4291	0.9342	0.0443	<1e-4	0.0001	0.0104	0.6648	0.0046	0.0638
600/300	0.0322	0.4171	0.9507	0.6475	0.0112	0.0529	0.2667	0.241	0.1985	0.7258
600/600	0.9397	0.0053	0.05	0.2438	0.1269	0.0357	0.926	0.0004	0.0901	0.0187
600/1000	0.0577	0.3187	0.3137	0.1243	0.3221	0.8285	0.8716	0.9863	0.8933	0.9452
1000/100	<1e-4	0.0002	0.3996	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4	0.0084	<1e-4
1000/200	<1e-4	0.0669	0.5775	0.0003	<1e-4	<1e-4	0.1507	0.0446	0.0002	0.0002
1000/300	0.0072	0.1566	0.9643	0.0039	0.1949	0.0076	0.4034	0.2383	0.1003	0.0239
1000/600	0.2301	0.6133	0.7129	0.2943	0.0248	0.0259	0.7387	0.6699	0.0927	0.1365
1000/1000	0.4371	0.9288	0.0755	0.022	0.0496	0.002	0.0605	0.0266	0.0006	<1e-4

TABLE B.3: P-values of Wilcoxon tests between models for database PD.

Train win. (ms)/Pred. win. (ms)	LSTM-LENET	LSTM-INCEP	LSTM-CNNLSTM	LSTM-TST	LENET-INCEP	LENET-CNNLSTM	LENET-TST	INCEP-CNNLSTM	INCEP-TST	CNNLSTM-TST
300/100	<1e-4	0.0831	0.048	0.0115	<1e-4	<1e-4	<1e-4	0.4034	<1e-4	<1e-4
300/200	0.0516	<1e-4	0.0469	0.1402	<1e-4	0.0034	0.7859	0.0004	<1e-4	0.001
300/300	0.587	<1e-4	0.2537	0.3039	<1e-4	0.0669	0.8312	<1e-4	<1e-4	0.1153
300/600	0.9835	0.0121	0.0813	0.4494	0.0298	0.2609	0.6133	0.4639	0.0016	0.0766
300/1000	0.4598	0.0095	0.2565	0.9151	0.028	0.2009	0.9288	0.1973	0.0112	0.0454
600/100	<1e-4	0.2396	0.1669	0.1811	<1e-4	<1e-4	0.0067	0.5337	0.0212	0.0521
600/200	0.2865	<1e-4	0.0013	0.9534	<1e-4	<1e-4	0.1384	0.029	<1e-4	0.0005
600/300	0.2742	0.0375	0.5728	0.4351	0.0026	0.0256	0.8988	0.0218	<1e-4	0.1973
600/600	0.5682	0.0144	0.1153	0.4151	0.0112	0.3732	0.4291	0.2082	0.0345	0.6133
600/1000	0.6157	0.0404	0.5846	0.1269	0.0648	0.7336	0.1468	0.0107	0.2803	0.0813
1000/100	0.0002	0.9835	0.0013	0.0428	<1e-4	<1e-4	0.0003	0.39	<1e-4	<1e-4
1000/200	0.0153	0.0888	0.0369	0.3514	<1e-4	<1e-4	0.0881	0.496	0.0002	0.0008
1000/300	0.0348	0.8071	0.3187	0.6724	0.0016	0.019	0.0989	0.3769	0.1177	0.2712
1000/600	0.3919	0.0577	0.645	0.3154	0.6549	0.8446	0.5543	0.9015	0.1596	0.5497
1000/1000	0.4171	0.9342	0.8608	0.8473	0.3479	0.7181	0.1755	0.989	0.6875	0.3938

TABLE B.4: P-values of Wilcoxon tests between models for PDET cross-prediction. Models were trained using train set of PD database and tested on ET test samples.

Train win. (ms)/Pred. win. (ms)	LSTM-LENET	LSTM-INCEP	LSTM-CNNLSTM	LSTM-TST	LENET-INCEP	LENET-CNNLSTM	LENET-TST	INCEP-CNNLSTM	INCEP-TST	CNNLSTM-TST
300/100	<1e-4	0.0407	<1e-4	0.0003	<1e-4	<1e-4	<1e-4	<1e-4	0.0016	<1e-4
300/200	<1e-4	0.0422	0.8932	<1e-4	<1e-4	<1e-4	0.0159	0.0493	<1e-4	<1e-4
300/300	0.0007	0.0005	0.345	0.008	<1e-4	0.0007	0.5548	0.002	<1e-4	0.0003
300/600	<1e-4	0.0278	0.7557	0.1389	<1e-4	0.0014	0.0125	0.0108	0.0073	0.6272
300/1000	0.2075	0.0005	0.0385	0.3083	<1e-4	0.011	0.171	0.0421	0.0042	0.0315
600/100	<1e-4	0.9029	<1e-4	0.2075	<1e-4	<1e-4	<1e-4	0.0014	0.0421	<1e-4
600/200	<1e-4	0.1784	0.1639	0.0063	<1e-4	<1e-4	<1e-4	0.0995	<1e-4	<1e-4
600/300	0.0171	0.0011	0.1363	0.3031	<1e-4	0.001	0.1389	0.0229	<1e-4	0.0204
600/600	0.6908	0.0305	0.1917	0.0058	0.0309	0.7884	0.0002	0.001	0.4185	0.0015
600/1000	0.0032	<1e-4	0.0007	0.0008	0.089	0.7308	0.3218	0.0376	0.3764	0.8749
1000/100	<1e-4	0.6438	<1e-4	0.0777	<1e-4	<1e-4	<1e-4	<1e-4	0.0631	<1e-4
1000/200	0.0008	0.0007	<1e-4	0.277	<1e-4	<1e-4	0.0009	0.0074	<1e-4	<1e-4
1000/300	<1e-4	0.2484	0.0354	0.3971	<1e-4	<1e-4	0.0261	0.9727	0.0079	0.023
1000/600	0.0107	0.0411	0.123	0.6702	<1e-4	0.0008	0.0067	0.5266	0.0077	0.1721
1000/1000	0.0604	0.2929	0.0512	0.583	0.0032	0.0463	0.0293	0.7034	0.2347	0.7474

TABLE B.5: P-values of Wilcoxon tests between models for database ET+PD.

Appendix C

Tables of prediction results

Training database: ET - Test data: patient ET-P001

Train win.(ms)	Pred. win. (ms)	Hidden size	Learning rate	ρ	MSE	RMSE
300	100	50	0.001	0.9865	77.0607	3.5920
	200	35	0.001	0.9173	186.4097	6.7244
	300	35	0.0005	0.8418	153.3340	7.9956
	600	50	0.0005	0.7113	250.9704	10.4011
	1000	50	0.0001	0.5760	502.2108	14.2719
600	100	50	0.001	0.9889	118.2013	4.0750
	200	35	0.001	0.9487	187.0405	6.8334
	300	50	0.0005	0.8668	232.7564	8.6526
	600	50	0.001	0.7544	382.0917	12.0545
	1000	50	0.0005	0.5829	497.1880	14.1765
1000	100	35	0.001	0.9907	84.9150	3.4720
	200	50	0.001	0.9185	211.7232	7.2486
	300	35	0.001	0.8676	284.2798	9.0826
	600	35	0.0001	0.6997	544.4229	14.2175
	1000	35	0.0005	0.6800	296.4531	10.6443

TABLE C.1: Prediction results from LSTMp model trained in database ET.

Training database: ET - Test data: patient PD-P010

Train win.(ms)	Pred. win. (ms)	Hidden size	Learning rate	ρ	MSE	RMSE
300	100	35	0.001	0.9902	0.3978	0.5702
	200	50	0.001	0.9834	2.3060	1.3361
	300	50	0.001	0.9372	6.8298	2.3146
	600	35	0.001	0.8413	15.8189	3.6336
	1000	50	0.0001	0.5862	35.5500	5.4789
600	100	50	0.001	0.9978	0.3305	0.4920
	200	35	0.0005	0.9701	3.9557	1.7701
	300	35	0.0005	0.9288	7.5630	2.4871
	600	35	0.0005	0.8220	19.7817	3.9267
	1000	50	0.0005	0.6416	36.8870	5.4647
1000	100	35	0.001	0.9916	0.3783	0.5275
	200	50	0.001	0.9669	4.6577	1.8467
	300	50	0.001	0.9079	7.4006	2.4761
	600	50	0.001	0.7911	17.7511	3.7565
	1000	50	0.001	0.7777	12.9237	3.2751

TABLE C.2: Prediction results from LSTMp model trained in database ET and tested on database PD.

Training database: PD - patient PD-P010

Train win.(ms)	Pred. win. (ms)	Hidden size	Learning rate	ρ	MSE	RMSE
300	100	50	0.001	0.9991	0.0787	0.2347
	200	50	0.001	0.9907	1.1972	0.9408
	300	35	0.0005	0.9652	3.9010	1.7328
	600	35	0.001	0.9000	11.8236	3.0638
	1000	35	0.0005	0.7931	23.4499	4.3520
600	100	50	0.001	0.9994	0.0712	0.2309
	200	35	0.001	0.9899	1.4273	1.0235
	300	35	0.001	0.9644	3.5383	1.6126
	600	50	0.0005	0.8943	11.6317	2.9583
	1000	35	0.001	0.7931	21.0517	4.0346
1000	100	50	0.001	0.9986	0.0612	0.2171
	200	50	0.001	0.9907	1.4252	0.9281
	300	35	0.001	0.9675	3.3517	1.5725
	600	35	0.0005	0.9092	8.3406	2.5561
	1000	50	0.0005	0.8834	8.0309	2.4312

TABLE C.3: Prediction results from LSTMp model trained in database PD.

Training database: PD - Test data: patient ET-P001

Train win.(ms)	Pred. win. (ms)	Hidden size	Learning rate	ρ	MSE	RMSE
300	100	50	0.0005	0.9896	202.1284	5.5787
	200	50	0.0005	0.9340	296.1274	8.5179
	300	35	0.0005	0.8706	331.1071	9.9468
	600	50	0.0005	0.7340	448.7246	12.7218
	1000	50	0.0001	0.5682	592.7478	15.4725
600	100	50	0.001	0.9795	301.6852	6.3293
	200	35	0.0005	0.9474	409.3758	9.3187
	300	50	0.0005	0.9051	365.8860	9.7689
	600	35	0.0005	0.7613	550.5888	13.4246
	1000	35	0.0005	0.6288	580.5461	15.0582
1000	100	35	0.001	0.9835	347.7738	7.3216
	200	35	0.0005	0.9425	466.8848	10.4072
	300	35	0.001	0.8850	524.8913	11.5759
	600	50	0.0005	0.7654	481.7008	13.0623
	1000	35	0.001	0.6882	384.3894	12.1726

TABLE C.4: Prediction results from LSTMp model trained in database PD and tested on database ET.

Training database: ET+PD - Test data: patients ET-P001 and PD-P010

Train win.(ms)	Pred. win. (ms)	Hidden size	Learning rate	ρ	MSE	RMSE
300	100	50	0.001	0.9939	25.6099	1.7637
	200	50	0.001	0.9643	68.4585	3.5146
	300	50	0.001	0.9131	86.3624	4.7935
	600	50	0.001	0.8316	133.1646	6.6002
	1000	35	0.0001	0.6676	281.5488	9.6443
600	100	50	0.001	0.9957	65.8185	2.2893
	200	50	0.001	0.9765	74.9813	3.4454
	300	35	0.001	0.9396	109.1421	4.7549
	600	35	0.001	0.8346	167.2260	7.1195
	1000	35	0.0005	0.6836	260.6097	9.1814
1000	100	35	0.001	0.9933	55.9864	2.0147
	200	35	0.0005	0.9649	90.4897	3.9348
	300	35	0.001	0.9245	137.1865	5.2343
	600	50	0.001	0.8327	166.6103	6.9498
	1000	50	0.0005	0.8061	152.8588	6.4760

TABLE C.5: Prediction results from LSTMp model trained in database ET+PD.

Train Database: ET - Test Database: patient ET-P001

Train win. (ms)	Pred. win. (ms)	Filters	Learning rate	ρ	MSE	RMSE
300	100	6/16	0.001	0.8922	23.5993	3.58
	200	6/16	0.001	0.8775	70.0111	5.8073
	300	6/16	0.001	0.815	93.7299	7.1209
	600	6/16	0.001	0.6816	176.9437	9.7037
	1000	6/16	0.001	0.5628	364.5609	12.998
600	100	6/16	0.001	0.9721	24.7854	3.2279
	200	6/16	0.001	0.9248	70.4169	5.8549
	300	6/16	0.001	0.8757	69.3508	6.1624
	600	6/16	0.001	0.7509	234.9962	10.1248
	1000	6/16	0.001	0.6117	434.0286	13.3847
1000	100	6/16	0.001	0.8952	25.5472	3.5398
	200	6/16	0.001	0.8746	129.5295	7.0686
	300	6/16	0.001	0.8074	171.67	8.2647
	600	6/16	0.001	0.7023	308.6369	11.2198
	1000	6/16	0.001	0.6569	303.5724	10.6547

TABLE C.6: Prediction results from LeNetp model trained in database ET.

Train Database: ET - Test Database: patient PD-P010

Train win. (ms)	Pred. win. (ms)	Filters	Learning rate	ρ	MSE	RMSE
300	100	6/16	0.0005	0.9677	2.1186	1.2304
	200	6/16	0.001	0.9629	4.4155	1.8295
	300	6/16	0.0005	0.9324	6.6672	2.3486
	600	6/16	0.0001	0.8637	14.5883	3.4508
	1000	6/16	0.0001	0.7349	28.2143	4.8812
600	100	6/16	0.001	0.9823	1.3917	1.0219
	200	6/16	0.001	0.9702	4.1634	1.7627
	300	6/16	0.0001	0.9445	6.1798	2.1683
	600	3/12	0.0001	0.8637	14.9715	3.532
	1000	3/12	0.0001	0.7656	22.6129	4.2996
1000	100	6/16	0.0005	0.9798	1.5103	1.0379
	200	6/16	0.0005	0.9657	4.7772	1.8269
	300	6/16	0.001	0.9374	6.746	2.2052
	600	6/16	0.001	0.8671	11.8401	3.0075
	1000	6/16	0.0001	0.827	10.2048	2.8705

TABLE C.7: Prediction results from LeNetp model trained in database ET and tested on database PD.

Train Database: PD - Test Database: patient PD-P010

Train win. (ms)	Pred. win. (ms)	Filters	Learning rate	ρ	MSE	RMSE
300	100	6/16	0.001	0.9065	21.6306	3.248
	200	6/16	0.0005	0.8968	54.6053	5.3498
	300	6/16	0.0001	0.8439	74.5626	6.5082
	600	3/12	0.001	0.7057	183.1599	9.7423
	1000	6/16	0.0001	0.5704	352.201	12.9507
600	100	6/16	0.001	0.9824	23.0797	2.9804
	200	3/12	0.001	0.9493	46.6701	4.6948
	300	6/16	0.001	0.9041	79.6405	6.0991
	600	3/12	0.0005	0.7734	215.6342	9.6731
	1000	6/16	0.0005	0.638	385.8215	12.6397
1000	100	6/16	0.001	0.9211	27.9527	3.5257
	200	6/16	0.0005	0.9066	86.4764	5.9961
	300	3/12	0.001	0.8535	186.5759	7.8767
	600	6/16	0.0005	0.7431	289.4431	10.6833
	1000	6/16	0.0005	0.7179	261.2029	10.0542

TABLE C.8: Prediction results from LeNetp model trained in database PD.

Train Database: PD - Test Database: patient ET-P001

Train win. (ms)	Pred. win. (ms)	Filters	Learning rate	ρ	MSE	RMSE
300	100	6/16	0.0005	0.9738	0.8573	0.772
	200	6/16	0.001	0.9771	3.0371	1.5261
	300	6/16	0.0005	0.9439	5.8601	2.1481
	600	3/12	0.0005	0.8611	14.0693	3.4279
	1000	3/12	0.001	0.7308	27.4713	4.8684
600	100	3/12	0.001	0.994	0.6957	0.6735
	200	3/12	0.001	0.9826	2.2991	1.2848
	300	6/16	0.001	0.9598	4.6408	1.8523
	600	6/16	0.0005	0.8961	11.3174	2.9749
	1000	6/16	0.001	0.8071	19.0528	3.8565
1000	100	6/16	0.001	0.9844	1.4504	0.9694
	200	3/12	0.001	0.9771	3.5319	1.4974
	300	3/12	0.0001	0.952	5.4634	1.9366
	600	6/16	0.0001	0.8959	9.3853	2.7156
	1000	6/16	0.0001	0.8716	8.5831	2.5751

TABLE C.9: Prediction results from LeNetp model trained in database PD and tested on database ET.

Train Database: ET+PD - Test Database: patients ET-P001 and PD-P010

Train win. (ms)	Pred. win. (ms)	Filters	Learning rate	ρ	MSE	RMSE
300	100	6/16	0.001	0.9286	11.5123	2.1699
	200	6/16	0.0005	0.9292	30.4733	3.6196
	300	6/16	0.001	0.8837	37.4137	4.2664
	600	6/16	0.001	0.7853	93.4875	6.4664
	1000	3/12	0.001	0.6436	187.5261	8.7636
600	100	3/12	0.001	0.9789	12.6129	2.1322
	200	6/16	0.001	0.9565	29.2761	3.3812
	300	6/16	0.001	0.9252	38.645	4.0188
	600	6/16	0.001	0.826	119.7771	6.5377
	1000	6/16	0.001	0.7097	212.7684	8.3872
1000	100	6/16	0.001	0.9425	14.2636	2.2109
	200	6/16	0.001	0.9316	47.9066	3.8965
	300	3/12	0.001	0.8865	76.9854	4.8875
	600	3/12	0.0005	0.8049	141.6623	6.7643
	1000	3/12	0.0005	0.7841	145.1352	6.3191

TABLE C.10: Prediction results from LeNetp model trained in database ET+PD.

Train Database: ET - Test Database: patient ET-P001

Train win. (ms)	Pred. win. (ms)	Learning rate	ρ	MSE	RMSE
300	100	0.001	0.9816	6.6104	1.7621
	200	0.001	0.9373	32.5608	3.9863
	300	0.0001	0.8521	80.0774	6.4955
	600	0.0005	0.7312	212.8938	10.0396
	1000	0.0001	0.5946	317.8019	12.316
600	100	0.0005	0.9887	4.202	1.4637
	200	0.001	0.9645	24.7289	3.52
	300	0.0001	0.9074	48.4397	5.344
	600	0.0001	0.7846	218.8105	9.7157
	1000	0.0001	0.6121	408.4256	13.186
1000	100	0.001	0.9872	10.3889	2.1232
	200	0.0005	0.9385	83.925	5.4762
	300	0.0001	0.8621	133.0693	7.2714
	600	0.0005	0.7135	325.2867	11.2714
	1000	0.0001	0.6558	270.0531	10.252

TABLE C.11: Prediction results from incepCNNp model trained in database ET.

Train Database: ET - Test Database: patient PD-P010

Train win. (ms)	Pred. win. (ms)	Learning rate	ρ	MSE	RMSE
300	100	0.0001	0.9959	0.3235	0.4962
	200	0.001	0.9901	1.9933	1.1318
	300	0.001	0.9662	4.6008	1.7899
	600	0.0001	0.9073	10.3008	2.8166
	1000	0.0001	0.7417	24.4693	4.4722
600	100	0.0001	0.9976	0.2244	0.4182
	200	0.0001	0.9873	1.8025	1.1676
	300	0.0001	0.9583	4.4853	1.8528
	600	0.0001	0.8741	12.4002	3.1031
	1000	0.0005	0.7275	24.4394	4.4611
1000	100	0.001	0.9929	0.2797	0.4641
	200	0.0001	0.984	2.4281	1.2642
	300	0.0001	0.9568	5.3643	1.9604
	600	0.0001	0.8877	10.3363	2.8517
	1000	0.0001	0.8571	8.9011	2.652

TABLE C.12: Prediction results from incepCNNp model trained in database ET and tested on database PD.

Train Database: PD - Test Database: patient PD-P010

Train win. (ms)	Pred. win. (ms)	Learning rate	ρ	MSE	RMSE
300	100	0.0001	0.997	0.1852	0.3774
	200	0.001	0.9928	1.0514	0.8827
	300	0.001	0.9743	3.3036	1.555
	600	0.001	0.9198	9.4642	2.6638
	1000	0.0001	0.808	20.4606	4.0942
600	100	0.0005	0.9986	0.1422	0.3243
	200	0.0001	0.9906	1.3286	0.9403
	300	0.0005	0.9694	3.6164	1.6295
	600	0.0001	0.9032	10.0575	2.777
	1000	0.0001	0.8111	19.0348	3.8824
1000	100	0.0001	0.997	0.2015	0.3748
	200	0.0001	0.9875	1.873	1.107
	300	0.0005	0.9632	4.3965	1.7787
	600	0.0001	0.9141	8.1596	2.4993
	1000	0.0001	0.8812	7.7118	2.4189

TABLE C.13: Prediction results from incepCNNp model trained in database PD.

Train Database: PD - Test Database: patient ET-P001

Train win. (ms)	Pred. win. (ms)	Learning rate	ρ	MSE	RMSE
300	100	0.001	0.9929	8.477	2.0516
	200	0.001	0.9601	21.798	3.2756
	300	0.001	0.9226	60.7803	5.375
	600	0.001	0.7866	127.8493	8.3494
	1000	0.0001	0.634	275.3451	11.841
600	100	0.0001	0.9897	3.2654	1.2405
	200	0.0001	0.9737	13.4008	2.6914
	300	0.0001	0.9282	47.8706	4.8726
	600	0.0001	0.8104	207.7331	9.0577
	1000	0.0001	0.6814	394.114	11.9809
1000	100	0.001	0.9917	6.4954	1.56
	200	0.0001	0.9553	49.1982	4.4842
	300	0.0001	0.892	117.7127	6.7787
	600	0.0001	0.757	259.7352	10.4858
	1000	0.001	0.7062	258.6327	10.0415

TABLE C.14: Prediction results from incepCNNp model trained in database PD and tested on database ET.

Train Database: ET+PD - Test Database: patients ET-P001 and PD-P010

Train win. (ms)	Pred. win. (ms)	Learning rate	ρ	MSE	RMSE
300	100	0.001	0.9906	6.1483	1.3491
	200	0.001	0.971	13.1104	2.2185
	300	0.001	0.9319	34.9078	3.7166
	600	0.001	0.8466	85.7923	5.8833
	1000	0.001	0.7191	199.2199	8.4191
600	100	0.0005	0.9939	1.7417	0.8091
	200	0.0005	0.9803	10.8128	2.0681
	300	0.001	0.9496	26.1572	3.2686
	600	0.0005	0.8548	117.1698	6.2608
	1000	0.0005	0.7383	230.7899	8.5256
1000	100	0.001	0.9943	3.966	1.1436
	200	0.0001	0.9705	28.5649	2.8298
	300	0.001	0.9302	64.3899	4.2082
	600	0.0005	0.8513	166.5103	6.5522
	1000	0.0005	0.8103	152.3684	6.1979

TABLE C.15: Prediction results from incepCNNp model trained in database ET+PD.

Train Database: ET - Test Database: patient ET-P001

Train win. (ms)	Pred. win. (ms)	Hidden size	Learning rate	ρ	MSE	RMSE
300	100	32	0.001	0.9854	47.2400	2.9450
	200	50	0.001	0.9233	121.4252	6.3903
	300	50	0.0001	0.8462	177.7581	8.2933
	600	50	0.0001	0.7125	287.3850	10.9318
	1000	50	0.0001	0.5794	419.0272	13.5853
600	100	50	0.0005	0.9906	77.9495	3.0613
	200	32	0.0005	0.9569	132.8318	5.7128
	300	32	0.0005	0.8951	176.3752	7.7931
	600	50	0.0001	0.7608	383.7545	11.9120
	1000	32	0.0001	0.6294	484.2425	13.9923
1000	100	32	0.001	0.9909	54.4558	2.6155
	200	50	0.001	0.9384	144.1817	6.3595
	300	32	0.0005	0.8486	280.1692	9.6185
	600	32	0.0005	0.6838	302.8548	11.5328
	1000	50	0.0005	0.6439	280.3851	10.7280

TABLE C.16: Prediction results from CNN-LSTMp model trained in database ET.

Train Database: ET - Test Database: patient PD-P010

Train win. (ms)	Pred. win. (ms)	Hidden size	Learning rate	ρ	MSE	RMSE
300	100	50	0.001	0.9892	0.4067	0.5468
	200	32	0.001	0.9822	3.3466	1.5514
	300	50	0.0001	0.9398	6.1832	2.2942
	600	50	0.0001	0.8594	16.7873	3.6984
	1000	50	0.0001	0.6202	37.2273	5.5802
600	100	50	0.0005	0.9970	0.2032	0.4100
	200	32	0.001	0.9754	3.3464	1.6159
	300	32	0.001	0.9250	10.2065	2.8124
	600	50	0.0001	0.8165	18.7922	3.9544
	1000	32	0.0001	0.6792	30.0829	5.0177
1000	100	50	0.001	0.9948	0.2580	0.4453
	200	50	0.001	0.9789	2.9982	1.4407
	300	32	0.0005	0.9298	7.5205	2.4895
	600	50	0.001	0.8448	15.7763	3.6375
	1000	50	0.0001	0.7847	17.0362	3.7229

TABLE C.17: Prediction results from CNN-LSTMp model trained in database ET and tested on database PD.

Train Database: PD - Test Database: patient PD-P010

Train win. (ms)	Pred. win. (ms)	Hidden size	Learning rate	ρ	MSE	RMSE
300	100	50	0.0005	0.9985	0.0946	0.2701
	200	32	0.0005	0.9904	1.5579	1.0316
	300	32	0.0001	0.9645	3.9449	1.7267
	600	50	0.0001	0.8992	11.0061	2.9409
	1000	32	0.0001	0.8008	22.7369	4.2589
600	100	50	0.001	0.9992	0.0907	0.2503
	200	50	0.001	0.9904	1.448	1.0072
	300	50	0.0005	0.9642	3.6891	1.6944
	600	50	0.0001	0.882	12.5682	3.1061
	1000	50	0.0001	0.8076	19.0738	3.8225
1000	100	32	0.001	0.9984	0.0591	0.2076
	200	32	0.001	0.9915	1.4373	0.9615
	300	32	0.001	0.966	4.0444	1.6079
	600	32	0.0001	0.9206	7.6707	2.4894
	1000	50	0.0001	0.8946	7.1026	2.2749

TABLE C.18: Prediction results from CNN-LSTMp model trained in database PD.

Train Database: PD - Test Database: patient ET-P001

Train win. (ms)	Pred. win. (ms)	Hidden size	Learning rate	ρ	MSE	RMSE
300	100	32	0.001	0.9916	208.4822	5.6266
	200	32	0.001	0.9437	273.1626	8.2079
	300	50	0.0001	0.8847	314.5978	9.8320
	600	32	0.0005	0.7605	450.3237	12.6343
	1000	32	0.0005	0.5972	532.1020	14.6606
600	100	32	0.001	0.9909	315.0364	6.4910
	200	32	0.001	0.9684	395.4348	9.0367
	300	32	0.0001	0.9157	445.2228	10.7028
	600	32	0.0001	0.8000	526.3105	13.2133
	1000	32	0.0001	0.6433	587.0131	15.1916
1000	100	32	0.001	0.9934	325.1829	7.1028
	200	50	0.0001	0.9601	449.2589	10.1872
	300	32	0.0005	0.8977	513.5740	11.5959
	600	32	0.0005	0.7666	523.7467	13.5994
	1000	32	0.001	0.7215	379.6608	11.9586

TABLE C.19: Prediction results from CNN-LSTMp model trained in database PD and tested on database ET.

Train Database: ET+PD - Test Database: patients ET-P001 and PD-P010

Train win. (ms)	Pred. win. (ms)	Hidden size	Learning rate	ρ	MSE	RMSE
300	100	32	0.001	0.9963	18.1015	1.2813
	200	32	0.001	0.9628	46.4576	3.1741
	300	32	0.0005	0.9223	74.9674	4.6153
	600	32	0.0005	0.8300	136.6239	6.6976
	1000	32	0.001	0.6996	219.2020	8.8879
600	100	50	0.001	0.9959	39.2736	1.5205
	200	32	0.001	0.9807	58.6371	3.1022
	300	32	0.0005	0.9449	79.3752	4.2510
	600	50	0.0001	0.8241	175.5901	7.1884
	1000	32	0.001	0.7255	243.8218	8.9126
1000	100	50	0.001	0.9966	26.4256	1.2845
	200	32	0.001	0.9798	73.5775	3.2744
	300	50	0.001	0.9317	113.3134	4.9217
	600	32	0.0005	0.8486	137.7013	6.5219
	1000	32	0.0005	0.8126	147.2053	6.2838

TABLE C.20: Prediction results from CNN-LSTMp model trained in database ET+PD.

Train Database: ET - Test Database: patient ET-P001

Train win. (ms)	Pred. win. (ms)	Patch len. (ms)	Dim. Model	Num. Heads	Enc. Layers	Learning rate	PCC	MSE	RMSE
300	100	200	256	4	6	0.001	0.9145	13.2657	2.5917
	200	160	256	4	6	0.001	0.8887	67.2772	5.7423
	300	160	256	4	6	0.001	0.8218	144.7922	7.9181
	600	200	256	4	4	0.001	0.6643	248.611	10.7638
	1000	160	256	8	4	0.001	0.5635	365.4423	13.2544
600	100	200	128	4	4	0.0005	0.984	9.1343	2.0773
	200	160	128	4	4	0.0005	0.9427	50.0117	4.7906
	300	160	128	8	4	0.001	0.8912	83.3474	6.3481
	600	160	256	8	4	0.0001	0.7797	245.0274	10.4311
	1000	200	128	8	6	0.0001	0.6425	507.5738	13.9868
1000	100	160	256	4	4	0.0005	0.9759	21.3603	2.9228
	200	160	256	8	4	0.001	0.9284	122.8206	6.3247
	300	160	128	4	6	0.001	0.8693	183.7428	7.7981
	600	160	256	4	4	0.0001	0.7274	312.6755	10.6949
	1000	200	128	8	4	0.0005	0.6927	300.2678	10.4898

TABLE C.21: Prediction results from TSTp model trained in database ET.

Train Database: ET - Test Database: patient PD-P010

Train win. (ms)	Pred. win. (ms)	Patch len. (ms)	Dim. Model	Num. Heads	Enc. Layers	Learning rate	PCC	MSE	RMSE
	100	200	256	4	6	0.001	0.9874	0.663	0.6852
	200	200	256	4	6	0.001	0.9782	3.5197	1.5695
300	300	160	256	4	6	0.001	0.9524	6.0992	2.1415
	600	200	256	4	4	0.001	0.8762	13.4571	3.1731
	1000	200	256	4	4	0.0001	0.7545	26.1008	4.5126
	100	200	128	4	4	0.001	0.9972	0.3836	0.5115
	200	160	128	4	4	0.0005	0.9831	2.678	1.3628
600	300	200	128	8	4	0.0001	0.9595	5.2521	1.9785
	600	200	128	4	4	0.0001	0.8844	12.387	3.0518
	1000	160	256	4	4	0.001	0.7816	22.2197	4.1488
	100	200	128	4	4	0.001	0.9865	0.7111	0.6912
	200	160	128	4	4	0.0001	0.9708	3.7742	1.5468
1000	300	160	256	4	4	0.001	0.9428	6.5618	2.1617
	600	200	128	4	4	0.0005	0.8748	11.1348	2.9273
	1000	200	128	4	4	0.0001	0.8452	10.4269	2.7914

TABLE C.22: Prediction results from TSTp model trained in database ET and tested on database PD.

Train Database: PD - Test Database: patient PD-P010

Train win. (ms)	Pred. win. (ms)	Patch len. (ms)	Dim. Model	Num. Heads	Enc. Layers	Learning rate	PCC	MSE	RMSE
	100	200	256	4	4	0.001	0.9914	0.5768	0.6161
	200	200	256	4	6	0.001	0.9804	2.7908	1.3992
300	300	200	256	4	6	0.001	0.9469	5.4674	1.983
	600	160	256	4	6	0.001	0.8787	11.6797	3.0585
	1000	160	256	4	4	0.001	0.7461	24.36	4.4917
	100	200	128	4	4	0.001	0.9984	0.2187	0.37
	200	200	128	8	4	0.001	0.9879	2.1497	1.2358
600	300	160	128	4	6	0.001	0.9633	4.4262	1.774
	600	200	128	4	4	0.0001	0.8975	11.1344	2.9255
	1000	200	256	8	6	0.001	0.8039	19.7898	3.8827
	100	200	128	4	6	0.001	0.9944	0.3271	0.4803
	200	200	128	4	4	0.001	0.9794	2.7018	1.326
1000	300	160	128	4	4	0.001	0.9564	5.0959	1.9287
	600	160	128	4	4	0.0001	0.8937	10.1239	2.7489
	1000	200	128	8	4	0.0001	0.8578	9.4176	2.6722

TABLE C.23: Prediction results from TSTp model trained in database PD.

Train Database: PD - Test Database: patient ET-P001

Train win. (ms)	Pred. win. (ms)	Patch len. (ms)	Dim. Model	Num. Heads	Enc. Layers	Learning rate	PCC	MSE	RMSE
	100	160	256	4	6	0.001	0.9415	16.3645	2.5515
	200	200	256	4	6	0.001	0.8914	47.6536	4.8868
300	300	160	256	8	6	0.001	0.8483	89.0267	6.7242
	600	160	256	4	4	0.001	0.7125	202.7454	9.9821
	1000	200	256	4	6	0.001	0.5528	378.2583	13.2565
	100	160	256	8	4	0.001	0.9898	8.0796	1.7302
	200	160	128	4	4	0.0001	0.956	38.8262	4.2454
600	300	160	128	8	4	0.0001	0.9052	70.9044	6.0405
	600	160	256	4	4	0.0005	0.7881	261.6052	10.249
	1000	160	256	8	6	0.001	0.671	408.1844	12.5314
	100	200	128	8	4	0.001	0.9808	10.0232	2.2022
	200	200	256	4	4	0.001	0.9414	92.2452	5.7448
1000	300	160	256	8	4	0.001	0.8755	170.3688	7.6952
	600	160	128	4	4	0.0001	0.7448	301.2495	10.7962
	1000	200	128	4	4	0.0001	0.698	265.1431	10.2659

TABLE C.24: Prediction results from TSTp model trained in database PD and tested on database ET.

Train Database: ET+PD- Test Database: patients ET-P001 and PD-P010

Train win. (ms)	Pred. win. (ms)	Patch len. (ms)	Dim. Model	Num. Heads	Enc. Layers	Learning rate	PCC	MSE	RMSE
	100	200	256	4	6	0.001	0.9594	7.6578	1.6436
	200	160	256	4	4	0.001	0.9371	27.4754	3.3811
300	300	160	256	4	6	0.001	0.892	52.228	4.5613
	600	160	256	4	6	0.001	0.7997	121.1521	6.6683
	1000	200	256	4	4	0.001	0.6504	222.2162	8.9996
	100	160	256	4	4	0.001	0.9932	3.1735	1.0753
	200	200	128	4	4	0.0001	0.9704	23.5897	2.8608
600	300	160	128	4	6	0.0001	0.9322	40.9268	4.0166
	600	200	128	4	4	0.0001	0.8486	117.898	6.369
	1000	160	256	8	4	0.0001	0.7222	237.8405	8.7242
	100	200	128	4	4	0.001	0.9896	4.6981	1.2125
	200	200	128	8	4	0.0005	0.9574	51.1513	3.5228
1000	300	200	128	8	4	0.0005	0.9149	85.2684	4.6872
	600	200	128	8	6	0.0001	0.8236	159.8057	6.6893
	1000	200	128	4	4	0.0001	0.8004	142.1019	6.1948

TABLE C.25: Prediction results from TSTp model trained in database ET+PD.