Western Graduate&PostdoctoralStudies

Western University

Scholarship@Western

Electronic Thesis and Dissertation Repository

1-18-2023 4:30 PM

PT-Net: A Multi-Model Machine Learning Approach for Smarter Next-Generation Wearable Tremor Suppression Devices for Parkinson's Disease Tremor

Anas Ibrahim, *The University of Western Ontario*

Supervisor: Trejos, Ana Luisa, *The University of Western Ontario*
Co-Supervisor: Naish, Michael D., *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Electrical and Computer Engineering

© Anas Ibrahim 2023

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Part of the Electrical and Computer Engineering Commons

Recommended Citation

Ibrahim, Anas, "PT-Net: A Multi-Model Machine Learning Approach for Smarter Next-Generation Wearable Tremor Suppression Devices for Parkinson's Disease Tremor" (2023). *Electronic Thesis and Dissertation Repository*. 9164.
https://ir.lib.uwo.ca/etd/9164

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlswadmin@uwo.ca.

Abstract

According to the World Health Organization (WHO), Parkinson's Disease (PD) is the second most common neurodegenerative condition that can cause tremors and other motor and non motor related symptoms. Medication and deep brain stimulation (DBS) are often used to treat tremor; however, medication is not always effective and has adverse effects, and DBS is invasive and carries a significant risk of complications. Wearable tremor suppression devices (WTSDs) have been proposed as a possible alternative, but their effectiveness is limited by the tremor models they use, which introduce a phase delay that decreases the performance of the devices. Additionally, the availability of tremor datasets is limited, which prevents the rapid advancement of these devices.

To address the challenges facing the WTSDs, PD tremor data were collected at the Wearable Biomechatronics Laboratory (WearMe Lab) to develop methods and data-driven models to improve the performance of WTSDs in managing tremor, and potentially to be integrated with the wearable tremor suppression glove that is being developed at the WearMe Lab. A predictive model was introduced and showed improved motion estimation with an average estimation accuracy of 99.2%. The model was also able to predict motion with multiple steps ahead, negating the phase delay introduced by previous models and achieving prediction accuracies of 97%, 94%, 92%, and 90% for predicting voluntary motion 10, 20, 50, and 100 steps ahead, respectively. Tremor and task classification models were also developed, with mean classification accuracies of 91.2% and 91.1%, respectively. These models can be used to fine-tune the parameters of existing estimators based on the type of tremor and task, increasing their suppression capabilities. To address the absence of a mathematical model for generating tremor data and limited access to existing PD tremor datasets, an open-source generative model was developed to produce data with similar characteristics, distribution, and patterns to real data. The reliability of the generated data was evaluated using four different methods, showing that the generative model can produce data with similar distribution, patterns, and characteristics to real data. The development of data-driven models and methods to improve the performance of wearable tremor suppression devices for Parkinson's disease can potentially offer a noninvasive and effective alternative to medication and deep brain stimulation. The proposed predictive model, classification model, and the open-source generative model provide a promising framework for the advancement of wearable technology for tremor suppression, potentially leading to a significant improvement in the quality of life for individuals with Parkinson's disease.

Index terms— Parkinson's disease, pathological tremor, hand tremor, machine learning, deep learning, neural networks, tremor prediction, tremor classification, generative models.

Lay Summary

Parkinson's disease (PD) is a brain illness that can cause tremor. Tremor interferes with daily tasks that require controlled movement, such as eating and writing. Medication or deep brain stimulation (DBS) are used to treat tremor; however, medication is not always helpful, and DBS has substantial risks. Wearable devices have been recognized as a viable alternative, but their success in reducing tremor depends on the estimation models they use. Furthermore, developing algorithms for these devices to reduce shaking movements requires large datasets that are hard to find. In this study, tremor data were used to create prediction and classification models. These models help wearable devices to reduce tremor. A model that can produce tremor data was also developed to make tremor data widely available for research studies. The prediction model had an estimation accuracy of 97%, 94%, 92%, and 90% for predicting motion 10, 20, 50, and 100 steps ahead, respectively. The tremor and task classification models had average accuracies of 91.2% and 91.1%, respectively. The generative model was validated using four methods and is available for public use, where anyone can download it for free to generate tremor data.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Dr. Ana Luisa Trejos and Dr. Michael D. Naish, for their tireless dedication and commitment to my research, and for their constant guidance and support over the past five years. Their valuable insights, aspiration for perfection and excellence, and constant support have had a profound impact on my personal growth and development and molded me into the person and researcher that I am today. When I struggled with writing paper drafts, they patiently taught and corrected me until I learned from my mistakes. They also helped me to develop my ideas and expand on them to tell a bigger story. I am grateful to have had the opportunity to work with them and learn from them, not only about research but also about becoming a better person and reaching for excellence.

I would like to express my appreciation to my advisory committee members for their guidance and support throughout my PhD journey. Their insights and expertise have been priceless. Thank you for your invaluable contributions and for helping me to achieve my goals.

I am also grateful to have met my friends in the lab, including Taylor Stanbury, Jacob Tryon, Alex Lizotte, Emma Farago, Brandon Edmonds, Nicole Devos, Vaughn Murphy, Marco Gallone, Jason Gharibo, Parisa Daemi, Zahra Habibollahi, Mena Kamel, Mahshad Berjis, and Tyler Desplenter.

I am especially grateful to Yue Zhou and Memo Colli-Alfaro, who have been invaluable sources of knowledge and support. Our daily discussions, conversations, and the times we spent climbing have been incredibly valuable and fun. I am thankful to have had the opportunity to learn from you and to spend great times with you.

Lastly, I am deeply indebted to my family for their love and support throughout my journey. I am especially grateful to my mother and sister, who have always been my biggest inspiration and have taught me the value of hard work and determination. I am thankful for their unwavering belief in me and for the sacrifices they have made to support me in achieving this accomplishment.

Statement of Co-Authorship

The following thesis contains material from manuscripts that have been submitted, accepted, and published. The work presented herein has been written by Anas Ibrahim under the supervision of Dr. Ana Luisa Trejos and Dr. Michael D. Naish, and has been co-authored by Anas Ibrahim, Dr. Yue Zhou, Dr. Mary E. Jenkins, Dr. Ana Luisa Trejos, and Dr. Michael D. Naish. Three articles have been chosen to form the main body of the thesis, and the extent of the collaboration of the co-authors is listed below.

Chapter 3. Motion Estimation and Multi-Step Ahead Prediction

Paper: Real-Time Voluntary Motion Prediction and Parkinson's Tremor Reduction using Deep Neural Networks

Current Status: Published in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 29, pp.1413–1423, 2021.

Anas Ibrahim: First Author, analyzed the data, developed the neural network based estimator and predictor and evaluated its performance against different neural network architectures in a simulation study, compared the developed estimator and predictor with the WFLC using the data collected from the participants in an experimental setup, and wrote the manuscript.

Dr. Yue Zhou: Co-author, collected the data from 18 participants with PD, helped in the design of the experimental setup, and assisted in editing and correcting the manuscript.

Dr. Mary E. Jenkins: Co-author, recruited the 18 participants with PD, assessed each participant's tremor using the Unified Parkinson's Disease Rating Scale (UPDRS), helped in the data collection, and assisted in editing and correcting the manuscript.

Dr. Ana Luisa Trejos: Co-author, supervised the data collection, supervised the development of simulation study and the experimental setup, and edited and corrected the manuscript.

Dr. Michael D. Naish: Corresponding author, supervised the data collection, supervised the development of simulation study and the experimental setup, and edited and corrected the manuscript.

Chapter 4. Tremor and Task Classification

Paper: Parkinson's Tremor Type and Task Classification

Current Status: Submitted to Engineering Applications of Artificial Intelligence, January 2023.

Anas Ibrahim: First Author, analyzed and processed the data, performed feature engineering, trained the different machine learning models, evaluated and compared their performance, showed that feature engineering is not suited for this study, and proposed a classification model based on a convolutional neural network.

Dr. Yue Zhou: Co-author, collected the data from 18 participants with PD, assisted in editing and correcting the manuscript.

Dr. Mary E. Jenkins: Co-author, recruited the 18 participants with PD, assessed each participant's tremor using the Unified Parkinson's Disease Rating Scale (UPDRS), helped in the data collection, and assisted in editing and correcting the manuscript.

Dr. Ana Luisa Trejos: Co-author, supervised the data collection, supervised the data analysis process and the evaluation of the different machine learning models, and edited and corrected the manuscript.

Dr. Michael D. Naish: Corresponding author, supervised the data collection, supervised the data analysis process and the evaluation of the different machine learning models, and edited and corrected the manuscript.

Chapter 5. Tremor and Voluntary Motion Generative Model

Paper: PT-Net: Pathological Tremor and Voluntary Motion Generative Adversarial Network

Current Status: Will be submitted to Elsevier - Engineering Applications of Artificial Intelligence.

Anas Ibrahim: First Author, analyzed and processed the data, developed the architecture of the generative model, evaluated the validity of the data generated, made the model open access on GitHub, and wrote the manuscript.

Dr. Yue Zhou: Co-author, collected the data from 18 participants with PD, and assisted in editing and correcting the manuscript.

Dr. Mary E. Jenkins: Co-author, recruited the 18 participants with PD, assessed each participant's tremor using the Unified Parkinson's Disease Rating Scale (UPDRS), helped in the data collection, and assisted in editing and correcting the manuscript.

Dr. Ana Luisa Trejos: Co-author, supervised the data collection, supervised the development of the generative model, and edited and corrected the manuscript.

Dr. Michael D. Naish: Corresponding author, supervised the data collection, supervised the development of the generative model, and edited and corrected the manuscript.

Contents

Abstract **ii**

Lay Summary **iv**

Acknowledgements **v**

Statement of Co-Authorship **vi**

Table of Contents **ix**

List of Figures **xiii**

List of Tables **xv**

Nomenclature and Acronyms **xvi**

1 Introduction **1**

1.1 Motivation 1

1.2 General Problem Statement and Research Objectives 2

1.3 Thesis Outline and Contributions 3

2 Literature Review and Machine Learning Background **7**

2.1 Introduction 7

2.2 Types of Tremor 7

2.3 Tremor Suppression 8

2.4 Tremor Estimation 9

2.5 Machine Learning 10

2.6 Supervised Learning 12

2.6.1 Linear Regression 13
2.6.2 SVMs 14
2.6.3 KNN 14
2.6.4 Decision Trees 15
2.6.5 Neural Networks 16
2.6.5.1 Optimization 17
2.6.5.2 Backpropagation 18
2.6.5.3 Recurrent Neural Networks 20
2.6.5.4 Convolutional Neural Networks 24
2.6.5.5 Supervised Learning Pipeline 26
2.7 Summary 27

3 Motion Estimation and Multi-Step Ahead Prediction 29
3.1 Abstract 29
3.2 Introduction 30
3.3 One Dimensional Convolutional-Multilayer Perceptron and Recurrent Neural Networks 33
3.4 Methods 35
3.4.1 Data Collection 35
3.4.2 Data Preparation 37
3.4.3 Implementation and Accuracy Assessment 38
3.4.4 Statistical Analysis 39
3.4.5 Network Comparison 39
3.5 Simulation Results 41
3.6 Motion Reproduction: Experimental Setup 45
3.6.1 Experimental Setup and Evaluation 45
3.6.2 Statistical Analysis 47
3.6.3 Results and Discussion 48
3.7 Conclusion 53

4 Tremor and Task Classification 55
4.1 Abstract 55

4.2 Introduction 56
4.3 Data Recording and Processing 57
4.4 Task and Tremor Classification 60
    4.4.1 Data Segmentation and Feature Extraction 60
    4.4.2 Feature Normalization and Feature Reduction 60
    4.4.3 Cross Validation 61
    4.4.4 Classification Models 61
    4.4.5 Results 64
4.5 Autoencoders 64
4.6 Convolutional Neural Network Classification 67
    4.6.1 Continuous Wavelet Transform 67
    4.6.2 Hilbert-Huang Transform 68
    4.6.3 Network Architecture and Cross Validation 69
    4.6.4 Results 69
4.7 Discussion 70
4.8 Conclusion 71

5 Tremor and Voluntary Motion Generative Model 73
5.1 Abstract 73
5.2 Introduction 74
5.3 Generative Adversarial Networks 76
5.4 One-Dimensional Convolutional Neural Networks 78
5.5 Methods 78
    5.5.1 Data Collection 78
    5.5.2 Data Preparation 79
    5.5.3 Implementation 80
    5.5.4 Evaluation 82
        5.5.4.1 Train on Real Data, Test on Synthetic Data (TRTS) 82
        5.5.4.2 Train on Synthetic Data, Test on Real Data (TSTR) 83
        5.5.4.3 Power Spectrum Density (PSD) 83
        5.5.4.4 T-Distribution Stochastic Neighbouring Embedding (t-SNE) . . . 83
5.6 Results 84

5.6.1 Train on Real, Test on Synthetic Data (TRTS) 84
5.6.2 Train on Synthetic, Test on Real Data (TSTR) 85
5.6.3 Power Spectrum Density (PSD) 86
5.6.4 T-Distribution Stochastic Neighbouring Embedding (t-SNE) 87
5.7 Discussion 87
5.8 Conclusion 89

6 Conclusion **90**
6.1 Contributions 91
6.2 Limitations and Future Work 92

References **94**

Curriculum Vitae **103**

List of Figures

1.1 PT-Net: a multi-model appraoch for smarter next generation WTSDs. 3

2.1 A simple mathematical model of a neuron. 18
2.2 Visualization of the computation graph of the neuron model. 19
2.3 A simplified compressed structure of an RNN. 21
2.4 An unfolded structure of a single RNN cell. 21
2.5 An unfolded structure of a single LSTM cell. 22
2.6 An unfolded structure of a single GRU cell. 23
2.7 Illustration of convolving a filter over an input with stride 1 and no padding. . . . 25

3.1 A sample of network architecture configuration with 2 CNN and 1 MLP layers. . . 33
3.2 Visualization of the placement of the IMU sensors. 36
3.3 Visualization of randomly selected data from different PD participants to show the difference in the data between participants. 38
3.4 Comparison of five neural network architectures. 40
3.5 Visualization of a sample output of the 1D-CNN-MLP model using Task 6 data as an input. 42
3.6 Visualization of a sample output of the 1D-CNN-MLP model using Task 5 data as an input. 43
3.7 Visualization of a sample output of the 1D-CNN-MLP model using Task 4 data as an input. 44
3.8 Visualization of the experimental setup. 46
3.9 A block diagram of the experimental setup that specifies the input and the output signals. 46

3.10 Comparison between the performance of the 1D-CNN-MLP model and the WFLC in suppressing tremor. 48
3.11 Visualization of a sample output of the WFLC when data of Task 2 were used as an input. 49
3.12 Visualization of PSD of the original and suppressed tremor for the WFLC model. . . 49
3.13 Visualization of a sample output of the 1D-CNN-MLP model when data of Task 2 were used as an input. 50
3.14 Visualization of PSD of the original and suppressed tremor for the 1D-CNN-MLP model. 50
3.15 Comparison between the performance of the 1D-CNN-MLP model and the WFLC when tracking voluntary motion. 51
3.16 Visualization of a sample output of the WFLC when data of Task 4 were used as an input. 52
3.17 Visualization of a sample output of the 1D-CNN-MLP when data of Task 4 were used as an input. 52

4.1 The placement of the IMU sensors on the forearm and the different joints of the hand. 58
4.2 The tasks performed by the participants. 59
4.3 Data processing and training workflow of the different models. 63
4.4 Sample visualization of the autoencoder architecture. 66
4.5 Visualization of the CNN architecture. 70

5.1 Structure of a Generative Adversarial Network. 77
5.2 Visualization of an example of slicing the data using the sliding window approach. 80
5.3 Visualization of four sample outputs of the PT-Net model of 5 seconds — TRTS. . 84
5.4 Visualization of four sample outputs of the PT-Net model of 10 seconds — TRTS. 85
5.5 Visualization of four random real data samples selected from the real dataset — TSTR. 86
5.6 Visualization of the cluster(s) created by t-SNE of the real and synthetic data. . . 88

List of Tables

3.1 Voluntary motion estimation performance of the 1D-CNN-MLP model during Tasks 4, 5, and 6. 42

3.2 Voluntary motion prediction performance of the 1D-CNN-MLP model for different steps into the future. 43

3.3 Multistep prediction time for the 1D-CNN-MLP model. 44

4.1 The accuracies of the machine learning models trained with and without feature reduction. 65

4.2 The accuracies of the four models after using the autoencoder for feature reduction. 66

4.3 The accuracy of the CNN when classifying five tasks. 70

4.4 The accuracy of the CNN when classifying three tasks based on the three types of tremor. 71

5.1 The estimation accuracy of the model for estimating voluntary motion for TRTS for the three window sizes. 85

5.2 Comparison of the mean power proportion of 2$^{\text{nd}}$ and 3$^{\text{rd}}$ harmonics of tremor to the 1$^{\text{st}}$ harmonic of tremor in real and synthetic data. 87

Nomenclature and Acronyms

Latin Letters

X	Input features
Y	Target outputs
f	Function that maps inputs to outputs
F	Class of functions that can be learned based on an objective function
L	A scalar-valued objective function
y	Target output
$\hat{y}$	Predicted output
x_i	The ith input
w_i	The weight for the input x at the ith position
b	Bias
$\mathbb{R}$	Set of real numbers
W	Weight matrix
H	Number of neurons in a hidden layer
g	Average loss of input examples including the regularization penalty
$\breve{w}$	Sum of weighted inputs
h_t	Hidden state at time t
x_t	Input at time t
C_t	Cell state at time t
$\widetilde{C}_t$	Candidate cell state at time t

fg_t	Forget gate at time t
ig_t	Update gate at time t
og_t	Output gate at time t
rg_t	Reset gate at time t
zg_t	GRU update gate at time t
$\hat{h}_t$	GRU candidate hidden state at time t
K	Number of filters
Fs	Spatial filters
S	Stride
P	Number of zero padding
Hd	Height of images
Wd	Width of images
N	Length of the input sequence
$x^{'}$	Normalized input
n	Number of features
C_W	Continuous wavelet transform
a_j	The instantaneous amplitude of the jth IMF
v_j	The instantaneous frequency of the jth IMF
V	Value function of the generator and discriminator
G	Generator
D	Discriminator
$\mathbb{E}$	Expected value
z	Vector of randomly generated numbers (noise)

Greek Letters

λ	Strength of regularization parameter
θ	Vector of parameters to be learned

σ	Standard deviation
μ	Mean of a feature
ψ	A wavelet of the continuous wavelet transform
υ	Dilation parameter
τ	Position of the wavelet in the time domain
κ	Instantaneous phase function
Δ	Gradient carried back to the input

Acronyms

PD	Parkinson's Disease
ADL	Activities of Daily Living
WTSD	Wearable Tremor Suppression Device
WTSG	Wearable Tremor Suppression Glove
FLC	Fourier Linear Combiner
WFLC	Weighted-frequency Fourier Linear Combiner
BMFLC	Band-limited Multiple Fourier Linear Combiner
EBMFLC	Extended Band-limited Multiple Fourier Linear Combiner
KF	Kalman Filter
RL	Reinforcement Learning
IMU	Inertial Measurement Unit
SVM	Support Vector Machine
KNN	K-Nearest Neighbour
DT	Decision Tree
RF	Random Forest
XGBoost	eXtreme Gradient Boosting
L1	Lasso regression
L2	Ridge regression

GD	Gradient Descent
SGD	Stochastic Gradient Descent
RMSProp	Root Mean Squared Propagation
AdaGrad	Adaptive Gradient Algorithm
Adam	Adaptive Moment Estimation
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
BLSTM	Bidirectional Long Short-Term Memory
BGRU	Bidirectional Gated Recurrent Unit
CNN	Convolutional Neural Network
DBS	Deep Brain Stimulation
1D	One Dimensional
2D	Two Dimensional
MLP	Multilayer Perceptron
DOF	Degree of Freedom
DNN	Deep Neural Network
MCP	MetaCarpoPhalangeal
SPI	Serial Peripheral Interface
RMSE	Root Mean Squared Error
PPA	Prediction Percentage Accuracy
EAP	Estimation Percentage Accuracy
SPSS	Statistical Package for the Social Sciences
ReLU	Rectified Linear Unit
PDS	Power Density Spectrum
PT	Parkinsonian Tremor
ET	Essential Tremor

MAV	Mean Absolute Value
EMAV	Enhanced Mean Absolute Value
MMAV	Modified Mean Absolute Value
WL	WaveLength
EWL	Enhanced WaveLength
ZC	Zero Crossings
SSC	Slope Sign Change
RMS	Root Mean Square
AC	Average Amplitude Change
DASDV	Difference Absolute Standard Deviation Value
SSI	Simple Square Integral
VARS	Variance of Signal
WA	Willison Amplitude
MFL	Maximum Fractal Length
PCA	Principle Component Analysis
CWT	Continuous Wavelet Transform
STFT	Short-Time Fourier Transform
HHT	Hilbert-Huang Transform
EMD	Empirical Mode Decomposition
IMF	Intrinsic Model Function
HSA	Hilbert Spectral Analysis
PT-Net	Pathological Tremor Network
SMOTE	Synthetic Minority Oversampling Technique
GAN	Generative Adversarial Network
EMG	Electromyography
EEG	Electroencephalogram
ECG	Electrocardiogram

Conv1Dz	One dimensional convolution with zero padding
TRTS	Train on Real, Test on Synthetic
TSTR	Train on Synthetic, Test on Real
EA	Estimation Accuracy
T-SNE	T-distribution Stochastic Neighbouring Embedding

Chapter 1

Introduction

Parkinson's disease (PD) is a degenerative and progressive disorder that affects the nerves in the deep parts of the brain called the basal ganglia and the substantia nigra [1]. PD is also considered the second most common neurodegenerative disorder, where it is estimated to affect between seven and ten million people worldwide. According to the Public Health Agency of Canada, it is projected that the number of Canadians living with PD will double between 2011 and 2031 [2]. Tremor is one of the motor-related symptoms of PD. It may be intermittent (occurring at separate times) or constant, which causes involuntary, rhythmic muscle contractions leading to shaking movements of one or more parts of the body. Tremor can affect different parts of the body, such as the vocal cords, torso, and legs, however, it tends to occur in the hands, where it is often described as "pill-rolling".

For those affected by PD, tremor interferes with their routine activities, such as eating, dressing up, writing, and many other tasks that require motor coordination. As a result, it negatively impacts their ability to perform activities of daily living (ADL) and can lead to social embarrassment and stigmatization, which affects their emotional well being.

1.1 Motivation

Tremor is often the first motor symptom of PD, where in its early stages it affects one side of the body. As the disease progresses over time, both sides may become affected.

Medication is usually considered as the first step for controlling the unwanted movements, where Levodopa is the medication that is most commonly given to people affected with PD. It controls their tremor by blocking acetylcholine in the brain [1]. However, medication is often ineffective and can cause significant mental and physical side effects. A second method that is considered for tremor management when medication is ineffective is deep brain stimulation (DBS); however, DBS is invasive in nature and carries a significant risk of complications.

Given the limitations and drawbacks of the above mentioned methods for treating tremor, researchers discovered that tremor can be effectively managed by mechanical loading [3]. Consequently, wearable tremor suppression devices (WTSD) were developed to help with tremor management [4–12]. The experiments done using these devices have shown promising results. However, the their performance is limited by the tremor models they use [13–15]. Therefore, there is a need to develop smarter algorithms in order to increase the performance of WTSDs for managing and suppressing tremor.

1.2 General Problem Statement and Research Objectives

Wearable tremor suppression devices are a viable alternative for tremor management, where with a fully developed WTSD, individuals with PD tremor could perform their activities of daily living with little to no effort. However, the WTSDs reported in the literature are still in the early stages of development and they have been only validated in simulation or on less than a handful of individuals with PD [12]. Moreover, the current tremor models used within the WTSDs are limited in their performance [13]. In order for WTSDs to achieve the desired level of performance and effectively reduce tremor, smarter algorithms have to be developed and integrated with these devices. To develop smarter algorithms, sufficient and comprehensive data that represent many of the activities of daily living have to be collected from individuals with PD tremor. However, this is not always possible.

As such, the overall goals of this thesis are to develop novel techniques and algorithms using the neural network modeling paradigm, which offers appealing and superior results, for smarter wearable tremor suppression devices. Consequently, these goals are addressed by introducing PT-Net. PT-Net consists of multiple machine learning models that are geared towards two main parts. The first part considers the development of data-driven models to be directly integrated with WTSDs in order to achieve higher levels of tremor suppression. The second part focuses on making tremor data widely accessible for anyone who is interested in studying and analyzing parkinsonian tremor in order to expedite the advancement of WTSDs.

1.3 Thesis Outline and Contributions

This dissertation develops models that directly learn from tremor data. In particular, a multi-model PT-Net is developed, as shown in Figure 1.1, that is geared towards solving three different tasks. The main contributions of each chapter are summarized as follows:

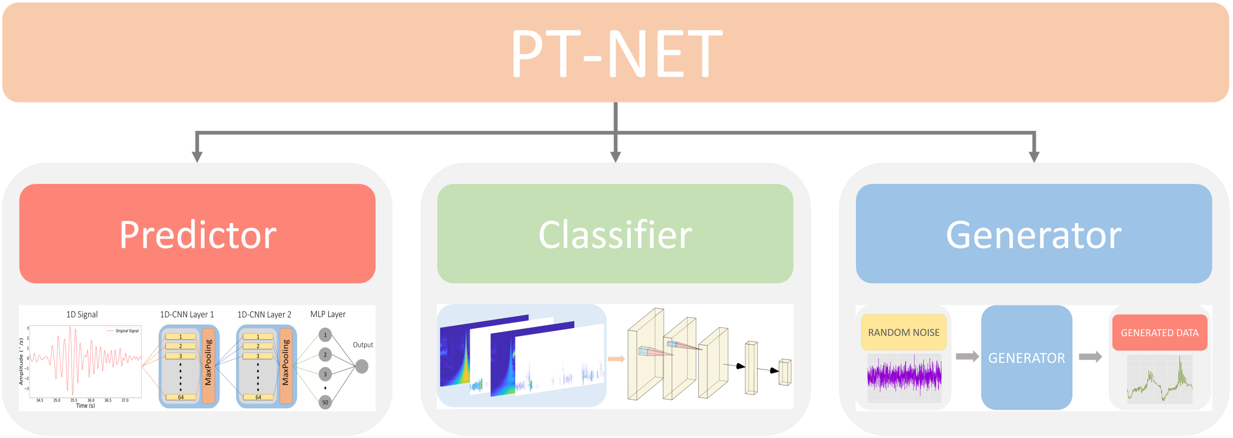


Figure 1.1: PT-Net: a multi-model machine learning approach for smarter next generation WTSDs. PT-Net consists of three models: a predictor, a classifier, and a generator that together form the contributions of the work that is presented in this thesis. Each of the models will be discussed in detail in Chapters 3, 4, and 5, respectively.

Chapter 2 Literature Review and Machine Learning Background: Before the main contributions of this work are introduced and discussed in detail, this chapter presents a review of the different types of tremor and tremor estimation methods. It also reviews the relevant mathematics used for neural networks, backpropagation, and optimization. Finally, the different neural networks architectures that have been used in this dissertation are described.

Chapter 3 Motion Estimation and Multi-Step Ahead Prediction: The PT-Net predictor is proposed in this chapter to overcome the drawbacks of the tremor estimators that are currently used with WTSDs. Current tremor estimators suffer from a major problem, namely the phase delay that they introduce, which degrades the real time tremor suppression performance of WTSDs. Another limitation of the existing tremor estimators is that they operate under the assumption that tremor is periodic and sinusoidal, however, recent research has shown that tremor is stochastic and nonlinear in nature. As a result, the PT-Net predictor utilizes the power of neural networks to learn the patterns and nonlinearities of tremor directly from the data without prior assumptions. Moreover, the PT-Net predictor is able to solve the problem of the phase delay by introducing the ability to predict multiple steps ahead. This chapter presents a comparison between the four neural network architectures in a simulation study. The network that performed the best was then compared to a commonly used tremor estimator in an experimental setup. As such, the contributions of this chapter are summarized as follows:

- Developing a predictive model that can learn directly from data, can adapt to the various tremor frequencies and amplitudes, and can learn the nonlinear relationships between voluntary motion and tremor.
- Developing a predictive model that can predict 10, 20, 50, and 100 steps ahead with high accuracy and low prediction time.
- Showing that automatic feature extraction using neural networks is preferable to tremor modeling when compared to the tremor estimators that are currently used with WTSDs.

Chapter 4 Tremor and Task Classification: This chapter presents the drawbacks of the tremor models used in WTSDs, where these estimators do not have the ability to know the type of tremor and the task being performed. Further, they do not have the ability to dynamically adapt their parameters based on the type of tremor that is active and the task being performed, which degrades the performance of the WTSDs. As a result, the PT-Net classifier is proposed to provide WTSDs with the necessary information to maximize tremor reduction. In order to develop accurate classification models, the work in this chapter shows that traditional feature engineering methods are not suited for this task, and proposes a novel approach that takes advantage of convolutional neural networks. The contributions of this work are as follows:

- Proposing the first method to deal with classifying PD tremor type and identifying the task performed to facilitate the development of smarter algorithms for WTSDs.
- Showing that traditional machine learning algorithms and feature engineering are not well suited to solving this problem.
- Introducing CNN-based classification models that outperform traditional machine learning methods to detect PD tremor type and identify the task being performed.

Chapter 5 Tremor and Voluntary Motion Generative Model: With the need for smarter algorithms for WTSDs, tremor data must be widely accessible to many research groups. Comprehensive tremor datasets in existence are private and usually cannot be shared due to privacy concerns. Moreover, these datasets are often limited in the number of data points collected due to the small number of participants with PD tremor that volunteered to have their tremor recorded. In addition, the ability to record tremor data is not always possible due to many reasons (e.g., spread of a disease or a lack of volunteers). As such, the work presented in this chapter removes the aforementioned restrictions by introducing an open-source tremor and voluntary motion generative model that was developed to generate data that have the same characteristics, distribution, and patterns as real tremor and voluntary motion data collected from individuals with PD tremor. As a result, the contributions of this work are as follows:

- A framework that is based on one-dimensional convolutional neural networks (1D-CNN) and generative adversarial networks (GANs) that leverages the advantages of the supervised and unsupervised learning paradigms of machine learning.
- A model that generates realistic motion that reflects a combination of voluntary motion and parkinsonian tremor.
- An open-access trained model that can be downloaded with its weights and used by anyone worldwide to generate and use realistic synthetic parkinsonian tremor data.
- A framework that can be trained and used with any type of biosignals, without being limited to tremors, e.g., electromyography (EMG) data, electroencephalogram (EEG) data, and electrocardiogram (ECG) data.

Chapter 6 Conclusion and Future Work: Summarizes the contributions presented in the previous chapters. The future work and research directions are also discussed in this chapter.

Chapter 2

Literature Review and Machine Learning Background

2.1 Introduction

In this chapter, a brief introduction to the different types of tremor (Section 2.2), an introduction to tremor suppression and wearable tremor suppression devices (Section 2.3), and a review of the literature in the areas of the tremor estimation methods used for wearable tremor suppression devices (WTSD) and their limitations (Section 2.4) is presented. The necessary technical background on machine learning (Section 2.5) and more specifically on supervised learning (Section 2.6) is also provided, with a focus on neural networks (Section 2.6.5), which will specifically cover convolutional and recurrent neural networks, the backbone of most of the work that will be presented in Chapters 3, 4, and 5.

2.2 Types of Tremor

Tremor can be labelled as either physiological or pathological tremor. Physiological tremor occurs in all healthy individuals. It is barely visible and it involves fine shaking of the hands, including the fingers. Physiological tremor is a normal human phenomenon that is the result of normal physical behaviour of the body, such as muscle activation. The magnitude of physiological tremor is negligible when compared to the magnitude of voluntary motion [16], as such, tremor suppression is only considered for tasks that require highly precise movements. Pathological tremor, on the other hand, is in an involuntary rhythmic oscillation and contraction that leads to shaking movements of different parts of the body. It affects the mobility of those affected and their ability to perform ADL [17].

Pathological tremor is associated with a number of neurological disorders that cause damage to the deep parts of the brain that control movement. Most of the neurological disorders where pathological tremors are observed have no known cause, however, there are some forms that appear to be inherited [17].

Parkinson's disease (PD) is the second most common neurodegenerative disorder, where it affects the nerve cells in the deep parts of the brain called the basal ganglia and the substantia nigra [17]. It is estimated that more than ten million people worldwide are living with PD [16, 18]. PD signs and symptoms include tremor that is time varying, nonlinear, stochastic, and nonstationary in nature [19].

2.3 Tremor Suppression

Tremor significantly affects and impacts people's mobility and their ability to perform ADLs [20–23]. The varying effectivness of traditional treaments for tremor management (medication and deep brain stimulation) was a motivation for research groups to develop unobtrusive methods and approaches to help reduce tremor. As a result, wearable tremor suppression devices have been developed as a viable alternative to the traditional treatments [24–27]. However, despite the good results that these devices showed, they were not tailored for the comfort of the people who will use them [10, 12]. Furthermore, these devices focused on suppressing tremor in the proximal joints, yet studies have shown that applying mechanical loading to the proximal joints increases the tremor magnitude in the distal joints, such as the fingers [12].

To address the limitations presented, a wearable tremor suppression glove (WTSG) was developed at the Wearable Biomechatronics Laboratory at Western University. The developed WTSG is lightweight (580 grams) and is based on a distributed cable-based transmission system to suppress tremor at the index finger, thumb, and wrist [12]. The preliminary assessment of the WTSG showed promising results in reducing tremor while allowing voluntary motion in the index finger, thumb and wrist [12]. Despite the promising results shown, the tremor suppresion performance of the developed WTSG is limited by the tremor estimator that it uses [12].

2.4 Tremor Estimation

Tremor estimators have been adopted for use with wearable tremor suppression devices (WTSD) to estimate the tremor component in the presence of voluntary motion. The main idea of using tremor estimators is to identify tremor oscillations that can be treated as noisy unwanted signals, such that they can be discarded. As such, in theory, a WTSD should allow the voluntary motion component of the signal, which results in tremor suppression [4–11, 26, 28, 29], in order to assist individuals with tremor perform ADLs.

In practice, the performance of assistive devices is directly impacted by the tremor estimators that they use for control [23]. In order to differentiate between tremor and voluntary motion data, simple filtering approaches, such as the low-pass filter and the Butterworth band-pass filter, can be applied to extract the desired signal(s). However, simply applying a filter has a major limitation, which is the time delay it introduces. The time delay is an inherent characteristic of these filters, in the sense that by the time a recorded signal from a certain joint is received, processed, filtered, then sent back to the actuator(s), some time would have elapsed and the position of the body part that is being monitored is at different position from where it initally was. As a result, the Fourier Linear Combiner (FLC) [30] was developed and used to overcome the drawbacks of these filters. The FLC operates under the assumption that tremor can be represented as a periodic signal that can be modelled by a Fourier series; however, Timmer *et al.* [19] showed that tremor is nonlinear, time varying, and stochastic in nature. Since the FLC operates using a predefined frequency, it is not able to dynamically track tremor frequencies.

The Weighted-Frequency Fourier Linear Combiner (WFLC) was proposed to overcome the drawbacks of the FLC. The WFLC is a commonly used method in tremor estimation [31] that adapts the frequency and amplitude of the model to the tremor signal. Similar to the FLC, the WFLC assumes that tremor is sinusoidal in nature, and it is unable to extract periodic signals containing more than one dominant frequency. As a result, it cannot handle the frequencies of the higher harmonics of tremor without degrading the performance of the WTSD.

To address the limitations of the WFLC, the Band-limited Multiple Fourier Linear Combiner (BMFLC) was proposed by Veluvolu *et al.* [32]. The limitation of the BMFLC is that it works with a predetermined set of frequencies and is unable to accurately track and adapt its frequency as tremor changes over time. Studies have shown that the calculated errors tend to be high [33], because the BMFLC is not robust to noise. Thus, if there is noise within the frequency band, the

BMFLC will add the noise to the estimation signal.

The Extended-BMFLC (EBMFLC) [34] was developed to overcome the limitations of the BMFLC. It showed promising results in both the accuracy and consistency of tremor signal extraction; however, the tremor is still considered as a series of sine and cosine signals, thus, the adaptive filtering methods face challenges in estimating and extracting tremor accurately.

Lastly, a common limitation of the mentioned estimators is the inherent time delay they introduce for tremor control in WTSDs [23, 35, 36]. Studies have shown that a delay as small as 20 ms substantially degrades the performance of wearable tremor suppression devices [35, 36].

Veluvolu *et al.* [36] demonstrated that integrating autoregressive models with the WFLC and the BMFLC with a Kalman Filter (KF) to perform one-step prediction of tremor, still resulted in poor performance. The accuracies achieved were as low as 40% for up to 20 ms of future prediction. Recently, the WAKE framework [37] was proposed, based on wavelet decomposition and adaptive Kalman filtering; however, it is limited to estimation and one-step ahead prediction. Another framework, PHTNet has also been proposed by Shahtalebi *et al.* [38]. It is limited to the prediction of one step only, and it introduces a time delay of 100–150 ms before it can properly estimate the signal. This delay is detrimental, as it will negatively affect the performance of any wearable assistive devices that aim to manage tremor.

2.5 Machine Learning

Machine learning is the science of getting computers to act without being explicitly programmed. Machine learning models can be classified into different types, including supervised learning, unsupervised learning, semi-supervised learning, self-supervised learning, reinforcement learning, and evolutionary computation.

Supervised learning can be described as a function approximation method. It is an approach that uses labelled datasets, which are designed to supervise algorithms in the training process, to solve a regression or a classification task(s). Supervised learning algorithms learn to map the input features to a target by training on the labelled data.

Unlike supervised learning algorithms that require large, carefully cleaned, and expensive to create datasets, in order to work well, unsupervised leaning algorithms are used to analyze and cluster unlabelled datasets based on similarities in some feature space, which removes the bottleneck of explicit human labelling. Unsupervised learning algorithms discover hidden patterns in the data without the need for human intervention [39].

Semi-supervised learning is used when a dataset consists of two portions, a small subset that is labelled and a larger portion that is unlabelled. A machine learning model is first trained on the labelled dataset and is then used to label the unlabelled portion of the dataset—a technique that is called pseudo-labelling. Once the unlabelled dataset is labelled by the trained model, the model is retrained again on the entire dataset, which is now comprised of the data that was truly labelled along with the data that was pseudo-labelled [40].

Self-supervised learning utilizes the intrinsic structure of the data as a supervisory signal for training. The general method of self-supervised learning is to predict a hidden part of the input data from any observed part of the input [41].

Reinforcement learning (RL) is a computational approach that allows a machine to learn from interaction with an environment [42]. RL can be described as the training of machine learning models to make a sequence of decisions, where an agent learns to achieve a goal in an uncertain and complex environment. In a reinforcement learning setting, an agent uses trial and error to find a solution to a problem based on a reward mechanism that encourages the agent to perform desired behaviour and a negative reward (punishment) to discourage unwanted behaviour (actions) [42]. The agent discovers which actions to take that result in the most reward, such that in many complex problems, actions to be taken affect not only the immediate reward but also the next state and, as a result, affect all subsequent rewards [42].

Evolutionary computation or evolutionary algorithms [43] employ mechanisms that are based on the theory of evolution to solve a search or an optimization problem, in which the fittest members of a population—those with traits that enable them to survive the longest—are the ones who produce the best offspring. The offspring inherit the positive characteristics of the parents and the population evolves over time to find a solution that is close to the unknown optimal solution [43].

In the following section, supervised learning is discussed in greater detail, providing a foundation for the work presented in this thesis.

2.6 Supervised Learning

Many problems can be formulated in a way to that allows computers to perform a mapping $f : X \rightarrow Y$, where X is the input and Y is the output or the target output that the computer should predict. However, in most of cases, it is hard to find the function f by conventional means, in the sense that it is unclear how to write a program to manually specify the function f that can differentiate between a cat and a dog, for example. As such, the supervised learning approach offers a powerful alternative that takes advantage of the data x and their labels y which are the desired mapping. Herein, the examples consist of IMU data collected from individuals with PD tremor.

In supervised learning, the assumption is that a training dataset of n independent examples $[(x_1, y_1), ...(x_n, y_n)]$ is available. As a result, the goal is to learn the mapping $f : X \rightarrow Y$ by searching over a set of candidate functions and finding the one that is most consistent with the training examples. As such, a class of functions F is considered that can be learned based on a scalar-valued objective function $L(\hat{y}, y)$, also called a cost or a loss function, to measure the disagreement between a predicted label $\hat{y_i} = f(x_i)$ for some $f \in F$ and a true label y_i. Consequently, the goal is to find a function f^* that minimizes the loss over the labels and the predictions as follows:

$$f^* = \underset{f \in F}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i), \tag{2.1}$$

where n is the number of training examples in a dataset, x_i is the feature vector of the i-th element, y_i is the corresponding true label, and L is the cost function value of each data point.

Once the function f^* is learned, it can then be used to map elements of X to Y. The function f^* is learned by optimizing the loss over the available training examples, and the hope is that it is a good proxy objective that can be used successfully on new unseen data.

In machine learning, f varies based on the machine learning algorithm used and its structure. Machine learning models that fall under the supervised learning category can be classified into regression and classification models, such as linear and logistic regression, support vector machines (SVM), k-nearest neighbours (KNN), decision trees and its derivatives, and neural networks. Regression is a type of a supervised learning task that uses an algorithm to understand the relationship between variables in order to predict a numerical value(s) based on some input.

Classification, on the other hand, is the task of assigning data into different categories, such as separating oranges from bananas, or cats from dogs.

2.6.1 Linear Regression

Linear regression models are relatively simple and provide an easy to interpret mathematical formula that can generate continuous predictions [44], based on the following equation:

$$\hat{y}(w, x) = w_1 x_1 + ... + w_n x_n + b, \tag{2.2}$$

where $\hat{y}$ is the predicted value, w is the vector of weights, b is the intercept, and x is the input features vector. A commonly used loss function in a regression setting is the squared difference between the target and the predicted value $L(\hat{y}, y) = (\hat{y} - y)^2$ as follows:

$$f = \underset{w,b}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} (w^T x_i + b - y_i)^2 \right]. \tag{2.3}$$

In order for Eq. 2.3 to generalize over all (x, y), regularization R is often used [44]. R is a scalar valued function that encodes preference for some functions over others, regardless of their fit to the training data, such that the regularization is a measure of complexity of a function. Ridge regression and Lasso regression are the two main methods used to increase the robustness of a model. Ridge regression imposes a penalty on the coefficients as follows:

$$R_2 = \lambda \sum_{i=1}^{n} w_i^2, \tag{2.4}$$

where λ is a parameter specifying the strength of regularization and w is the i-th coefficient. Putting Eq. 2.3 and 2.4 together, the problem to solve becomes:

$$f = \underset{w,b}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} (w^T x_i + b - y_i)^2 \right] + \lambda \sum_{i=1}^{n} w_i^2 \tag{2.5}$$

Lasso regression adds a penalty equal to the absolute value of the magnitude of coefficients. This type of regularization results in sparse coefficients, which is ideal for producing simple models.

Lasso regression can be calculated as follows:

$$R_1 = \lambda \sum_{i=1}^{n} |w_i|, \tag{2.6}$$

where λ is the regularization strength and w is the i-th coefficient. As such, putting Eq. 2.3 and Eq. 2.6 together, the problem to solve becomes:

$$f = \underset{w,b}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} (w^T x_i + b - y_i)^2 \right] + \lambda \sum_{i=1}^{n} |w_i|. \tag{2.7}$$

2.6.2 SVMs

Support vector machines [45] is a powerful supervised learning algorithm. It can be used for both classification and regressions tasks, however, SVMs is mainly used for classification problems. SVMs works by mapping data to a high-dimensional feature space from a low-dimensional feature space so that data points can be linearly separated. It uses separating hyperplanes, also called decision boundaries, with support vectors to distinctly classify between data points corresponding to different classes. In order to classify data points into different classes, SVMs aims to maximize the distance between the hyperplane and the supporting vectors using a kernel function. As such, the objective function of SVM is as follows:

$$\frac{1}{2} ||w||^2 + C \sum_{i=1}^{n} \max\left(0, 1 - y_i(w^T x_i + b)\right), \tag{2.8}$$

where $||w||$ is the norm of the weight vector, C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification error, x is the ith data point of the feature vector, y is the ith label of the ith training example, b is a bias term, and n is the number of data points.

A linear kernel function is recommended when linear separation of the data is straightforward. For complex cases, a nonlinear function should be used, such as the radial basis function (RBF), polynomial kernel, or sigmoid kernel.

2.6.3 KNN

K-nearest neighbours (KNN) is among the simplest supervised learning algorithms. It is considered a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of data points. KNN works on the assumption that similar points can be found near one another [46]. In a classification problem, a class label is assigned on the basis of a majority vote, based on finding the majority class label of the k nearest neighbours using a distance metric (e.g., Euclidean distance, Manhattan distance, Hamming distance, etc.). As a result, the better the distance metric reflects label similarity, the better the classification. Minkowski distance is the most common metric used in KNN. k, which represents the number of nearest neighbours, is another important metric to consider. A KNN model might underfit the data when k is too large, or overfit the data when k is too small.

2.6.4 Decision Trees

Decision tree, (DT) [47] is a nonparametric supervised learning algorithm that can be used for both regression and classification. DT has a hierarchical tree structure, which consists of a root node, branches, internal nodes, and leaf nodes. DT learning employs a divide and conquer approach by conducting a greedy search to identify the optimal split points within a tree. The process of splitting is repeated in a top-down, recursive manner until all, or the majority of examples in a dataset are classified under specific classes. Many powerful algorithms have been proposed based on DT. The two main algorithms proposed are Random Forests (RF) and extreme gradient boosting (XGBoost) [48].

RF consists of a large number of individual decision trees that are the building blocks of a RF. Decision trees operate as an ensemble, where each individual tree performs a prediction, and the class with the most votes becomes the prediction of the RF model [49]. RF has shown promising results in different areas, given its ease of use and flexibility in handling both classification and regression problems. RF operates by selecting a random subset of the features of the data in a training set, such that after several random subsets of features are selected by each tree, the decision trees are trained independently, and the average or majority of the predictions yield a more accurate prediction.

XGBoost [48] is another popular tree-based algorithm that outperforms DTs and other tree-based algorithms in terms of performance and training time. XGBoost works by improving a single weak tree (model) by combining it with a number of other weak trees in order to generate a collectively strong model. It utilizes gradient descent and boosting methods to combine shallow DT, where the next input sample of a new DT is related to the results of previous DTs, and it has been highly successful for structured and tabular data.

2.6.5 Neural Networks

Neural networks were originally developed to be a mathematical abstraction of the brain. They are represented by a fairly simple mathematical expression—a sequence of matrix multiplications with some nonlinearities (nonlinear functions) added to them. A neural network consists of many nodes, also known as neurons, that are loosely related to the synapsis of the brain. They are trainable and modifiable, with the goal of finding the proper settings (parameters) of the neurons to make the neural network perform something desirable, in other words, to perform a certain task(s) very well.

A vanilla neural network neuron (without the nonlinear functions) is basically a linear model as seen in Section 2.6.1. Expanding on this, consider the following simple example: a dataset of ten 2-dimensional points such that $X \to \mathbb{R}^2$, where each X is annotated with a scaler value $Y \to \mathbb{R}$. A linear function f^* can be used to map X to Y, where $f^* = w^T x + b \mid w \in \mathbb{R}^2, b \in \mathbb{R}$. In this example, the parameter space is measured and defined by (w_1, w_2, b), where w_1 and w_2 are vectors of weights and b is the bias. The squared difference between the target and the predicted values(s) is a common loss function that is used in a regression problem, as seen in Eq. 2.5. It can be extended by adding a nonlinear function, such as the tangent hyperbolic or the sigmoid function as follows:

$$f = \arg\min_{W_1, b_1, w_2, b_2} \left[\frac{1}{n} \sum_{i=1}^{n} (w_2 \tanh(W_1^T x_i + b_1) + b_2 - y_i)^2 \right] + \lambda(W_1^2 + w_2^2). \tag{2.9}$$

Note that f becomes more complex due to the addition of the nonlinear function. As a result, instead of searching over a linear function $f(x) = w^T x + b$, one searches over $f(x) = w_2 \tanh(W_1^T x + b_1) + b_2$, where W_1, w_2, b_1 and b_2 are the parameters to be trained in order to find the proper settings to minimize the loss function $L(\hat{y}, y) = (\hat{y} - y_i)^2$. Assuming there are 10 neurons in the hidden layer, such that the number of neurons is denoted as H, W_1 is a matrix of size $H \times 2$, b_1 is a bias vector of size H, w_2 is a vector of size H, b_2 is a single scaler, and tanh is the hyperbolic tangent function that is applied element-wise to “squash” the values between –1 and 1. Consequently, in neural networks, the space of functions F to be explored/learned will be a neural network with certain settings (parameters, or weights and biases), a loss function L, which is the

Euclidean loss in the regression example above, and the regularization R, where the L2 norm or the ridge regression regularization is the most commonly used (R_2 is the sum of squares of all the weights). As such, the problem of learning a model for a supervised learning problem is reduced to an optimization problem of the form $\theta = \text{argmin}_\theta \ g(\theta)$, where θ is a parameter vector and $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(f_\theta(x_i), y_i) + R(f_\theta)$.

2.6.5.1 Optimization

In the previous section the supervised learning task was reduced to solving an optimization problem of the form $\theta = \text{argmin}_\theta g(\theta)$, where θ is a vector of parameters and g combines the average loss of all examples and the penalty using regularization. As such, in order to perform efficient optimization, the problem is restricted to using differentiable functions in order to compute the gradient with respect to the weights and biases $\nabla_{\theta g}$ using backpropagation, which will be discussed in the following section. The gradient is a vector of partial derivatives that gives the slope of g for all of the parameters of θ. The gradient can be used as a search direction to improve θ to lower g, which is done by adding to it the negative gradient, since the goal is to minimize it. This process can be utilized in the form of a gradient descent (GD) algorithm that then can be used to calculate the gradient and backpropagate it to update the parameters of the neurons in the neural network by taking a small step in the direction of the negative gradient. In case of large datasets, GD becomes very slow. As a result, the stochastic gradient descent (SGD) [44] algorithm was proposed. SGD is used to estimate the gradient using a small batch of samples from the dataset. This process allows many approximate updates to be performed instead of fewer precise updates.

SGD uses a parameter ϵ, which is also called the learning rate, that is used as a step size at each iteration while moving toward a minimum of a loss function. The learning rate can cause the gradient descent to diverge if it is too large, or it could cause the gradient descent to take too long to converge if it is too small. In practice, the learning rate is modified dynamically in order to obtain faster convergence. For example, one method that can be used is the Momentum update, which is used to encourage the progress in a small and consistent directions of the gradient. Different methods have been developed that use the notion of momentum such as the RMSProp (Root Mean Squared Propagation) [50], AdaGrad (Adaptive Gradient algorithm) [51], and Adam (Adaptive moment estimation) [52].

2.6.5.2 Backpropagation

In order to evaluate the gradient of the loss function, an optimizer is used to minimize it, such as stochastic gradient descent or Adam. As a result, during this process a mapping $f \in F$ is found, such that f achieves a low regularization cost and maps X to Y.

Backpropagation is the efficient process used by the optimizer to calculate the gradients with respect to the weights and biases. Backpropagation is a recursive application of the chain rule from calculus. As an example, suppose there is an expression $y = (3x + 5)^2$, where the goal is to compute $\frac{\partial y}{\partial x}$. In a neural network setting, x is the vector that contains the input data and the network parameters, and y is the total loss. As such, assume that $a = 3x$, $b = a + 5$, and $y = b^2$. The gradient of every transformation with respect to its input can be written as follows: $\frac{\partial a}{\partial x} = 3$, $\frac{\partial b}{\partial a} = 1$, and $\frac{\partial y}{\partial b} = 2b$. As a result, the local derivatives can be used to find the desired global derivative $\frac{\partial y}{\partial x}$ by iteratively applying the chain rule, which can be written as follows: $\frac{\partial y}{\partial x} = \frac{\partial y}{\partial b}\frac{\partial b}{\partial a}\frac{\partial a}{\partial x}$.

Given the example described above, the same logic can be applied to a neuron in a concrete example as shown in Figure 2.1.

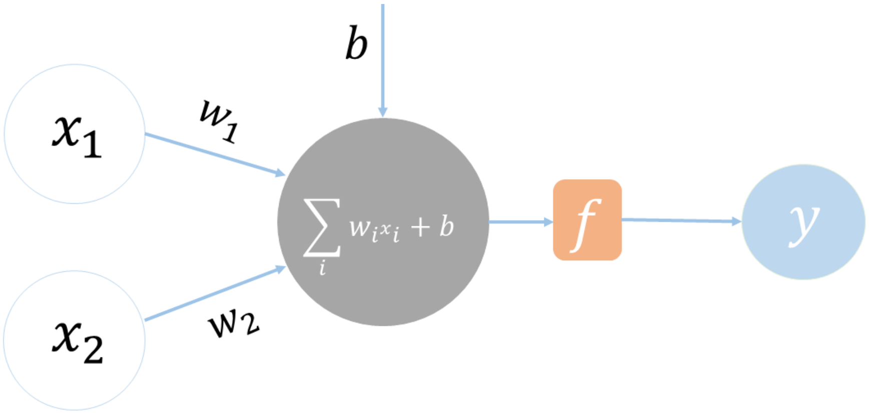


Figure 2.1: A simple mathematical model of a neuron. Some input x_i are passed through synapsis (connections)—that are associated with weights, to the neuron cell body. The synapse interacts with the input to the neuron multiplicatively. The neuron has its own bias b, that acts as a threshold of firing of the neuron regardless of the input. The sum of $w_i x_i$ and the added bias is passed through an activation function f to get the output y.

Assume the inputs x_1 and x_2, where $x_1 = 2$ and $x_2 = -1$. Each input has its own weight w, where $w_1 = -3$ and $w_2 = 1$. The weights are initialized randomly and represent the synaptic strength for each input. The neuron has a bias $b = 7.7$, and f is the "tanh" activation function. According to the model presented in Figure 2.1, during the forward pass, each input needs to be multiplied by its own weight and then a bias is added to the sum of the weighted input. The result is then passed through the activation function f to produce the output y, as follows:

$$\begin{aligned}
x_1w_1 &= x_1 \times w_1,\\
x_2w_2 &= x_2 \times w_2,\\
\breve{w} &= x_1w_1 + x_2w_2,\\
o &= \breve{w} + b,\\
y &= \tanh(o),
\end{aligned} \tag{2.10}$$

where $\breve{w}$ is the sum of the weighted inputs, o is the raw output of the neuron, and y is the prediction after o was transformed using the activation function f. As a result, the computational graph of Eq. 2.10 is shown in Figure 2.2

In order to get a desired output based on some loss function, we need to propagate the gradient backwards. As such, the derivative of y with respect to y, $\frac{\partial y}{\partial y} = 1$. Next, we need to backpropagate through the "tanh", so $\frac{\partial y}{\partial o} = 1 - \tanh^2(o)$, which is $1 - y^2 = 0.63474$, and as a result, the gradient of o is 0.63474. The additive symbol is a simply a distributor of gradient, so the 0.63474 will simply flow to b and to $\breve{w}$ equally, and that is because the local derivative of the addition operation is 1 for every one of its nodes. Subsequently, the gradient of each $\breve{w}$ and b is 0.63474, respectively. Again, the gradient from $\breve{w}$ will flow to both expressions x_1w_1 and x_2w_2. At every point in time, the gradient is saying that, if we want the output of this neuron to increase, then the influence of these expressions on the output is positive. Lastly, backpropagating to x_1w_1 first through a multiplication node, the local derivative of x_1 is going to be the gradient of x_1w_1 multiplied by the value of w_1, and the gradient of w_1 is going to be the gradient of x_1w_1 multiplied by the value of x_1, and the same logic applies to x_2 and w_2.

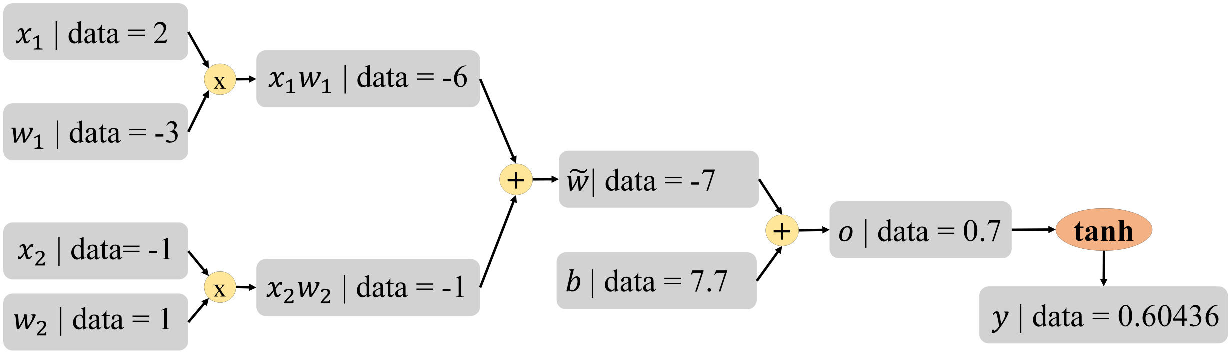


Figure 2.2: Visualization of the computation graph of the neuron model.

In a typical neural network setting, the derivative of the neurons on the weights w and biases b is what matters the most, because they are the parameters that will be changed as part of the optimization process. In this example, a single neuron is shown that is usually a piece of a bigger puzzle, whereas in a neural network, there are many neurons that are connected, resulting in a more complex computational graph.

In the following sections, more complex architectures are discussed that are well suited for time series PD tremor data.

2.6.5.3 Recurrent Neural Networks

A recurrent neural network (RNN) is a class of neural network that is used to process sequential data [44]. RNNs take advantage of parameter sharing, where the output produced by each RNN cell at each time step is shared with other RNN cells through recurrent connections as illustrated in Figure 2.3. A single RNN cell (purple box) expanded is shown in Figure 2.4. A vanilla RNN utilizes the connectivity pattern to process a sequence of vectors x_t using the recurrence formula $h_t = f_\theta(h_{t-1}, x_t)$, where h is the hidden vector that is the running summary of all previous vectors x to that time step, x is the input at time t, f is the function to be learned, and θ represents the parameters to be learned. The previous hidden vector h_{t-1} and the input x at time t are concatenated and multiplied by the weights vector W and then transformed using a nonlinear function, such as the *tanh* function. As such, the RNN formula becomes:

$$h_t = \tanh(W(h_{t-1}, x_t)), \tag{2.11}$$

where W consists of two weights vectors, one for the input x and the second for h.

Despite the ability of a vanilla RNN to process sequential data, it suffers from two main problems when trained over long sequences of data: the exploding gradient and the vanishing gradient [44, 53]. The exploding gradient can make the learning unstable, while the vanishing gradient makes it difficult to know in which direction the parameters of the hidden units should be updated to improve the objective function.

The Long Short-Term Memory (LSTM) architecture was introduced by Hochreiter *et al.* [54] to solve the problems of the vanilla RNN. The LSTM consists of a memory cell and gated units that regulate the flow of information by changing the recurrence formula to be more computationally complex. The new recurrence formula allows the LSTM network to effectively propagate the

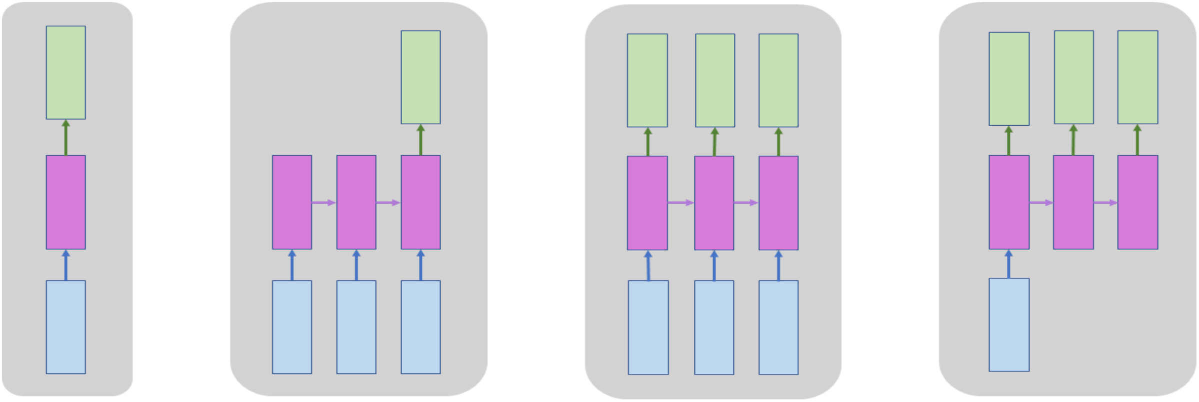


Figure 2.3: A simplified compressed structure of an RNN. Blue boxes are the input vector(s), purple boxes represent some hidden layer(s), and the green boxes represent the output vector(s). An RNN can take an input vector(s), transform it through some hidden layer, and produce an output. The arrows indicate the functional dependencies, where an RNN is able to process sequences of vectors at the input, at the output, or both either serially or in parallel. The hidden layer manipulates the set of internal variables h_t from both the current input and the previous hidden state h_{t-1} using the recurrent connections.

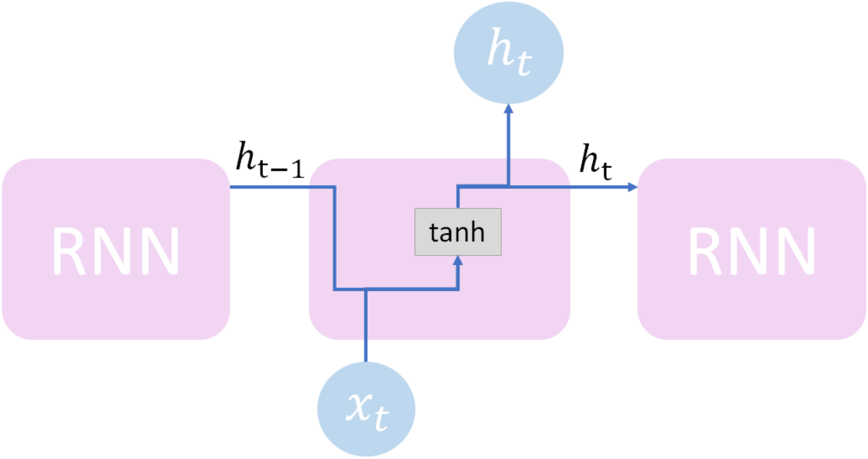


Figure 2.4: A single RNN cell expanded, where it takes the input x at time t and the hidden state from previous time step, and it outputs h the hidden state at a hidden layer. The sequential data are processed using the recurrent connections and the recurrent formula $h_t = f_\theta(h_{t-1}, x_t)$, where θ are some parameters to be learned.

gradients backwards in time, thereby enabling them to learn the important data from long input sequences. The LSTM cell consists of a forget gate, an update gate, an output gate, and a memory cell [54], as shown in Figure 2.5. The memory cell carries relevant information throughout the processing of the input sequence, and information is added to and removed from the memory cell through the different gates. The forget gate decides which information should be kept and which information should be removed from the previous states. The update gate decides which information should be updated and which information is essential to keep from the current step. The output gate decides what the prediction is and what the hidden state should be. The the more computationally complex structure for the LSTM cell is presented in Figure 2.5 and is described as follows:

$$
\begin{aligned}
fg_t &= \text{sigmoid}(W_f(h_{t-1}, x_t) + b_f), \\
ig_t &= \text{sigmoid}(W_i(h_{t-1}, x_t) + b_i), \\
\widetilde{C}_t &= \tanh(W_C(h_{t-1}, x_t) + b_C), \\
C_t &= fg_t \times C_{t-1} + ig_t \times \widetilde{C}_t, \\
og_t &= \text{sigmoid}(W_o(h_{t-1}, x_t) + b_o), \\
h_t &= og_t \times \tanh(C_t),
\end{aligned}
\tag{2.12}
$$

where W and b are the weights biases vectors. The LSTM receives the cell state C_{t-1} from the LSTM node at step $t-1$, and it acts like a conveyor belt, where information flow along it is either unchanged or changed with some minor interactions. The LSTM node has the ability to remove or add information from the cell state, which is carefully regulated by the different gates. The first gate (fg_t) decides which information to let through and which information to throw away from the cell state. The decision is made by the sigmoid function. The forget gate looks at h_{t-1} and x_t and outputs a number between 0 and 1 for each value in the cell state C_{t-1}, with 0 meaning get

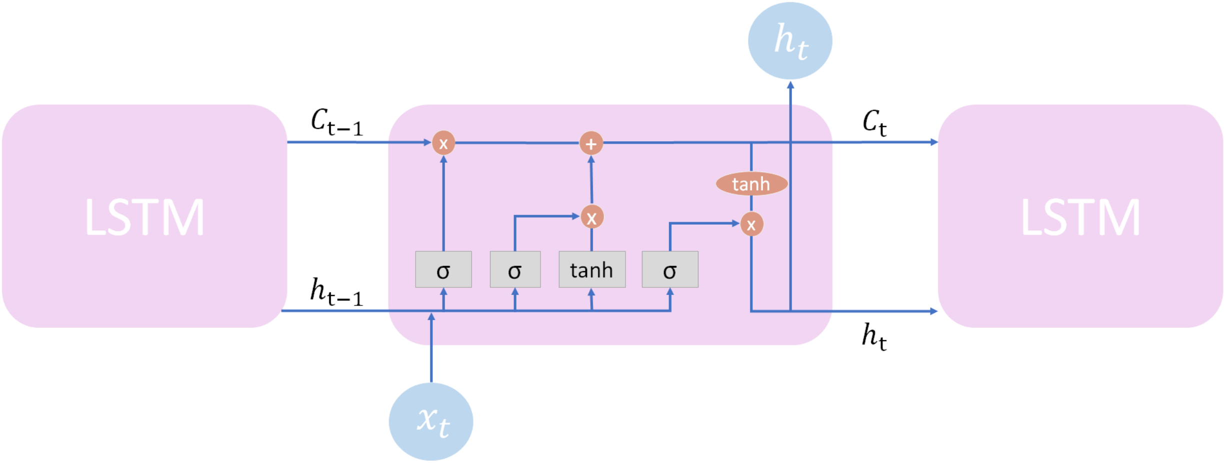


Figure 2.5: A single LSTM cell expanded, receiving the cell state (C_{t-1}) and hidden state (h_{t-1}) of previous LSTM node and the input (x_t), and outputting the current cell state (C_t) and the hidden state (h_t). A single LSTM node consists of the forget gate (fg_t), the update gate that consists of (ig_t and $\widetilde{C}_t$), and the output gate (og_t).

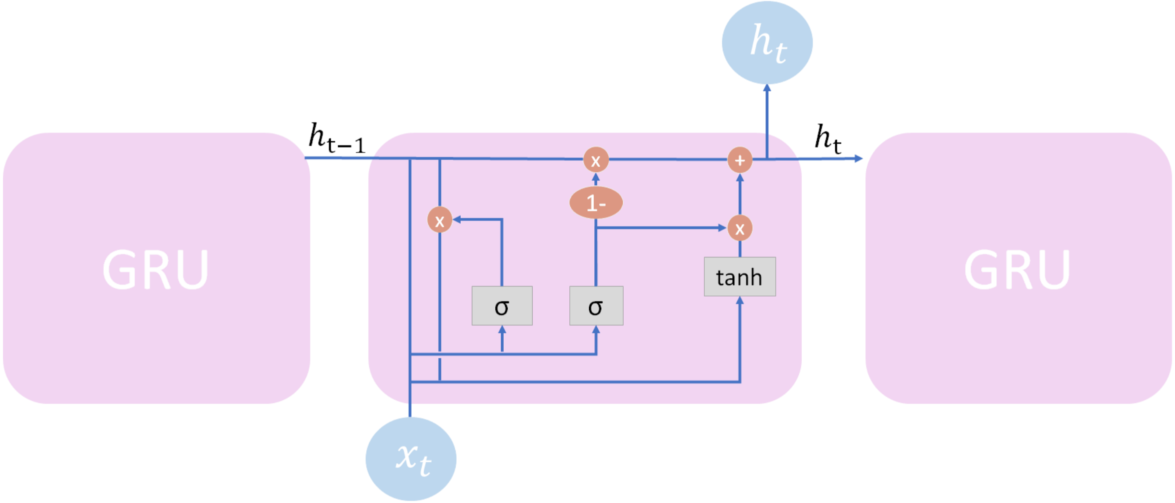


Figure 2.6: A single GRU cell expanded, receiving the hidden state (h_{t-1}) from a GRU node at time $t-1$ and the input (x_t), and outputting the hidden state (h_t). A single GRU node consists of the reset (rg_t) and the update gate (zg_t).

rid of the information completely, and 1 meaning keep the information. The next step is to decide what new information to store in the cell state, which is done in two steps. The first step includes a sigmoid function that decides which values to update, and in the second step a tanh function is used to create a vector of new candidate values ($\widetilde{C}_t$), where $\widetilde{C}_t$ represents the candidate values that could be added to the state. The combination of these two steps form the update to the cell state (C_{t-1}) to create C_t using $C_t = fg_t \times C_{t-1} + ig_t \times \widetilde{C}_t$. Finally, the output og_t is based on the filtered information from x_t and h_{t-1} using the sigmoid function, then the cell state is passed through a tanh function and its output is multiplied by og_t to form h_t.

The Gated Recurrent Unit (GRU) architecture was introduced by Cho *et al.* [55]. Similar to LSTMs, GRUs consist of gate units but do not contain memory cells. The GRU structure is simpler than the LSTM structure, which only consists of reset and update gates as shown in Figure 2.6. The reset gate allows the GRU cell to drop any information considered irrelevant in the future, thus allowing a more compact representation of the data. On the other hand, the update gate controls how the information from previous hidden states carries over to the current hidden state, which allows the network to remember long-term information and decide whether the hidden state is to be updated with a new hidden state. The structure of a GRU cell is shown in Figure 2.6 and is described as follows:

$$\begin{aligned} rg_t &= \text{sigmoid}(U_r h_{t-1} + W_r x_t + b_r), \\ zg_t &= \text{sigmoid}(U_{zg} h_{t-1} + W_{zg} x_t + b_z), \\ \hat{h_t} &= \tanh(W_h x_t + U_h(rg_t \times h_{t-1}) + b_h), \\ h_t &= zg_t \times \hat{h_t} + (1 - zg_t) \times h_{t-1}, \end{aligned} \tag{2.13}$$

where W and U are the weight vectors and b is the biases vector(s). The GRU node receives the hidden state (h_{t-1}) from the GRU node at step $t-1$ and the input (x_t) that will pass through the reset gate. The reset gate (rg_t) is responsible for the short-term memory of the network and decides what information to let through and what information to throw away from h_{t-1} using the sigmoid function. The reset gate is then used to find the candidate hidden state ($\hat{h_t} = \tanh(W_h x_t + U_h(rg_t \times h_{t-1}) + b_h)$), where rg_t control how much influence the previous hidden state can have on the candidate state. The update gate (zg_t), on the other hand, is responsible for the long-term memory and it regulates both, the flow of historical information from h_{t-1}, as well as the new information from the candidate state $\hat{h_t}$.

The Bidirectional LSTM (BLSTM) and the Bidirectional GRU (BGRU) are enhanced architectures of the LSTMs and GRUs, respectively. A BLSTM or a BGRU consists of one network that processes the sequential data from left to right (from the past to the future) and another separate network that processes the sequential data from right to left (from the future to the past) [56]. The hidden vectors from both sides are concatenated to produce a representation at each position of the input, where the hidden states of the LSTMs or GRUs can preserve information from both the past and the future at any point in time and enrich the input at any position by its own context.

2.6.5.4 Convolutional Neural Networks

A convolutional neural network (CNN) [44] is a neural network architecture that was originally developed and designed to handle data with a spatial topology, such as images. A CNN takes in a multi-dimensional array, e.g., a 225 × 225 image is a 225 × 225 × 3 array if it is a coloured image (3 is for the colour channels red, green, and blue), or a 225 × 225 × 1 array if it is not, and produces an output by convolving the input with a set of filters. For example, assume that the input is a coloured image X with width and height 225 × 225 × 3, a CNN with a filter w of 5 × 5 × 3, where w consists of 75 parameters (5*5*3 = 75) that will be updated and modified for the model to learn. In CNNs, the filter is convolved by sliding it across the spatial positions of

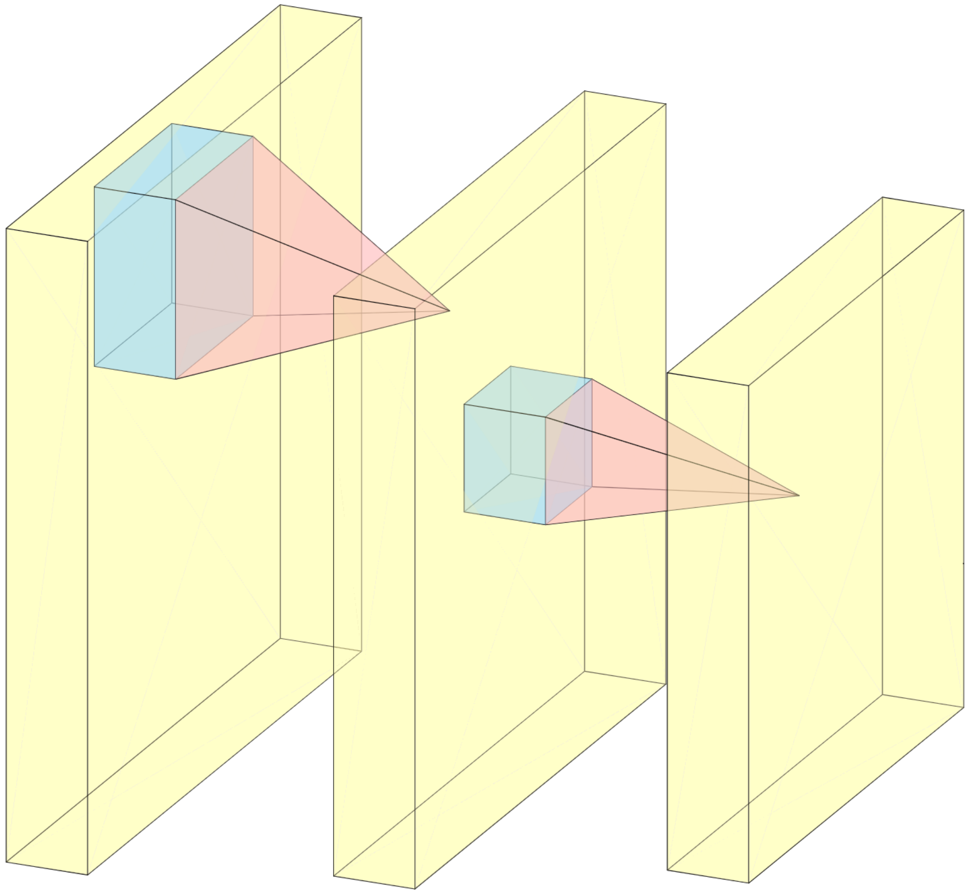


Figure 2.7: Illustration of convolving a filter over an input with stride 1 and no padding.

the input and the dot product between the small window of X that w passed on is calculated at each position, as seen in Figure 2.7. The output of this dot product is an activation map that will have a dimension of 221×221, where 221 is the number of unique positions that a filter of 5 elements can be placed over an input of size 225. Some methods can be used to control the output dimension(s) of the activation map(s), such as padding. For example, padding with a border of zeros of thickness of 3 would give an activation map of dimension 227×227 (the formula used to perform the calculations will be presented shortly). Another method that can be applied when convolving the filter is to use stride, which shifts the convolution filter by a larger amount. For instance, applying a $5 \times 5 \times 3$ filter to a $225 \times 225 \times 3$ image with no padding and a stride of 2 would result in an output activation map of size 111×111. A CNN layer consists of multiple filters and not just 1. As such, if 64 filters are used, then each would calculate its own 111×111 activation map, which are stacked to produce the final output of the layer. In this case, it would be $111 \times 111 \times 64$. Each filter would look for specific features in the input and their parameters are trained using backpropagation.

In order to control or calculate the dimensions of the outputs of a CNN layer, four parameters need to be specified. The number of filters K, the spatial extent of said filters Fs, the stride S by which they are applied, and the number of zero padding P that is added to the borders of the input. As a result, to compute the output dimensions of the CNN layer, the output dimension of one filter, is calculated as follows:

$$Wd_2 = (Wd_1 - Fs + 2P)/S + 1, Hd_2 = (Hd_1 - Fs + 2P)/S + 1, \tag{2.14}$$

where the Wd_2 and Hd_2 are the width and height of the activation map (the output), respectively, and based on the number of filters used, the output volume would be of size $Wd_2 \times H_2 \times K$.

The result of the dot product of one of the filters at a certain location represents the output of one neuron that only has connections to the specific position of the input. Each filter is slid over the input and it uses the same weights at every position or location, which results in effectively introducing a parameter sharing scheme, as such that all of the neighbouring neurons in an activation map use the same weights. Consequently, this results in a smaller number of parameters in each CNN layer. For instance, an input image of dimensions $225 \times 225 \times 3$ to a CNN layer with 64 filters of size $5 \times 5 \times 3$ with padding of 1 and stride of 1, the output would be $223 \times 223 \times 64$. The number of parameters from the example given is 3,182,656 outputs that only use $5 * 5 * 3 * 64 + 64 = 4864$ parameters that represent the weights and biases, which is a very small number of parameters to learn compared to number of parameters needed for a vanilla neural network layer (also known as a multi-layer perceptron, or a fully connected layer).

Lastly, pooling layers are another element that is usually used alongside convolutional layers, more specifically after a CNN layer. Pooling layers are used to control the overfitting problem, in the sense that the pooling layer(s) decrease the number of representations using a downsampling transformation. The pooling layer is used independently on each activation map to downsample them spatially.

2.6.5.5 Supervised Learning Pipeline

In order to use supervised learning, a dataset has to be available. In this dissertation, a dataset that has been collected at the Wearable Biomechatronics Laboratory at Western University is used. The dataset is comprised of data from 18 participants with PD tremor. The PD group included eleven males and seven females with varying tremor severity. The data were collected usingd inertial measurement units (IMU) sensors that were affixed to the participants' wrist and the hand (the sensors placements will be discussed in more detail in the following chapter). The dataset was split into a training set, a validation set, and a testing set and will be discussed in more detail in Chapters 3 and 4.

Preprocessing the data can help improve the convergence of supervised learning models. For motion data, common preprocessing techniques involve filtering the data using a low-pass filter to extract the voluntary motion component of the signal and Butterworth band-pass filter to extract tremor component of the signal. The data are then normalized, which is a crucial step to help supervised learning algorithms learn effectively.

Following the data preprocessing step, it is necessary to decide on a family of architectures to explore based on the task at hand. In this dissertation, different supervised learning algorithm families were selected and explored in each of Chapters 3, 4, and 5, where each chapter will provide more details about the problem and the selected algorithms.

Once the architecture is chosen, hyperparameter optimization is performed. Hyperparameter optimization is the process of selecting the best values for the hyperparameters of a machine learning model. Hyperparameters are the parameters of a model that are set before training, and they control the behavior and performance of the model. For example, the learning rate and the regularization coefficient are both hyperparameters. In order to find the best hyperparameters, the model is trained multiple times with different values for the hyperparameters, and the performance of each set of hyperparameters is evaluated. The best performing set of hyperparameters is then selected. Hyperparameter optimization is important because it can significantly improve the performance of a machine learning model.

After the model(s) parameters are chosen and it is trained, the best trained model is identified; typically it is the model with lowest validation loss. The model is then evaluated on the test set to assess its performance.

2.7 Summary

This chapter presents a review of the different types of tremor and a review of the literature in the areas of tremor estimation methods.

In the review of the tremor estimation methods that are used for WTSDs, a few limitations have been identified. First, most of these methods are not suited to be used in a real-life setting because they introduce a delay in tremor estimation and cannot predict motion ahead of time, which would degrade the performance of the WTSDs. As such, a new approach capable of multi-step ahead prediction is required.

Second, the review of current WTSDs showed that the tremor estimators used do not adapt their parameters dynamically based on the type of tremor that is active and the task that is being performed. As a result, the lack of knowledge about the type of active tremor and the task being performed decreases the ability of the WTSDs to suppress tremor optimally.

In addition, machine learning models have demonstrated their ability to produce excellent results. However, to fully capitalize on their potential, large, clean, and expensive-to-create datasets must be available to train these models. Unfortunately, PD tremor datasets are not openly accessible, and when they are available, they cannot be shared, and sometimes data cannot be collected. Therefore, a novel approach to generating data that is similar to actual PD tremor data in characteristics and distribution should be investigated.

Lastly, the relevant technical background for supervised learning, neural networks, optimization, and backpropagation was provided, and the neural network architectures used in Chapters 3, 4, and 5 were presented.

The following chapter will discuss the limitations of tremor estimators and will introduce a deep learning-based solution that overcomes these limitations and is capable of multi-step ahead prediction.

Chapter 3

Motion Estimation and Multi-Step Ahead Prediction

This chapter is adapted from “Real-Time Voluntary Motion Prediction and Parkinson’s Tremor Reduction using Deep Neural Networks,” with minor modifications to the version published in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 29, pp. 1413–1423, 2021.

3.1 Abstract

Wearable tremor suppression devices (WTSD) have been considered as a viable solution to manage parkinsonian tremor. WTSDs showed their ability to improve the quality of life of individuals suffering from parkinsonian tremor, by helping them to perform activities of daily living (ADL). Since parkinsonian tremor has been shown to be nonstationary, nonlinear, and stochastic in nature, the performance of the tremor models used by WTSDs is affected by their inability to adapt to the nonlinear behaviour of tremor. Another drawback that the models have is their limitation to estimate or predict one step ahead, which introduces delay when used in real time with WTSDs, which compromises performance. To address these issues, this work proposes a deep neural network model that learns the correlations and nonlinearities of tremor and voluntary motion, and is capable of multi-step prediction with minimal delay. A generalized model that is task and user-independent is presented. The model achieved an average estimation percentage accuracy of 99.2%. The average future voluntary motion prediction percentage accuracy with 10, 20, 50, and 100 steps ahead was

97.0%, 94.0%, 91.6%, and 89.9%, respectively, with prediction time as low as 1.5 ms for 100 steps ahead. The proposed model also achieved an average of 93.8% ± 1.5% in tremor reduction when it was tested in an experimental setup in real time. The tremor reduction showed an improvement of 25% over the Weighted Fourier Linear Combiner (WFLC), an estimator commonly used with WTSDs.

3.2 Introduction

Parkinson's Disease (PD) is a disorder that affects the nerve cells in the deep parts of the brain called the basal ganglia and the substantia nigra [17]. More than ten million people worldwide are living with PD [16, 18]. PD signs and symptoms include tremor that is time varying, nonlinear, stochastic, and nonstationary in nature [19]. Tremor significantly affects and impacts people's mobility and their ability to perform activities of daily living (ADLs) [20–23].

The current approach for the treatment of PD tremor is medication and deep brain stimulation (DBS) [1, 57]; however, medication is connected with a series of adverse effects and DBS carries significant risks [1] and could cost up to $100,000 per patient [16].

Recently, assistive technologies, such as wearable assistive devices, have been considered and are being developed as an alternative approach to manage tremor [4–11, 26, 28, 29]. The mechanical suppression of tremor by these devices serves to assist individuals with tremor perform ADLs. However, the performance of the assistive devices is directly impacted by the tremor models that they use for control [23]. The models use adaptive filtering techniques to estimate tremor. Initially, the Fourier Linear Combiner (FLC) [30] was developed and used. The FLC operates under the assumption that tremor can be represented as a periodic signal that can be modelled by a Fourier series; however, Timmer *et al.* showed that tremor is not periodic but rather time varying, nonlinear, and stochastic. The FLC works using a predefined frequency, and it is not able to adaptively track tremor frequencies. The Weighted-Frequency Fourier Linear Combiner (WFLC) was proposed to overcome the drawbacks of the FLC. The WFLC is a commonly used method in tremor estimation [31], which adapts the frequency and amplitude of the model to the tremor signal. Similar to the FLC, the WFLC assumes that tremor is sinusoidal in nature, and it is unable to extract periodic signals containing more than one dominant frequency. As a result, it cannot adapt to the frequency of the higher harmonics of tremor, since the model uses Fourier series, and the frequency parameters of the higher harmonics are set proportional to the estimated frequency of the dominant harmonic. Therefore, it cannot be used to accurately estimate tremor.

The Band-limited Multiple Fourier Linear Combiner (BMFLC) is an algorithm that is based on the FLC. The BMFLC was developed to address the shortcomings of the WFLC. The limitation of the BMFLC is that it works with a predetermined set of frequencies and is unable to accurately track and adapt its frequency to tremor frequencies. Studies have shown that the calculated errors tend to be high [33], because the BMFLC is not robust to noise. Thus, if there is noise within the frequency band, the BMFLC will add the noise to the estimation as well. As a result, the Extended-BMFLC (EBMFLC) [34] was developed to overcome these limitations. It showed promising results in both the accuracy and consistency of extracting tremor signals; however, the tremor is still considered as a series of sine and cosine signals, thus, the adaptive filtering methods face challenges in estimating and extracting tremor accurately.

One common limitation of these estimators is the inherent time delay that they introduce for tremor control in wearable assistive devices [23, 35, 36]. Studies have shown that a delay as small as 20 ms degrades the performance of wearable devices [35, 36].

In an attempt to overcome the drawbacks of the FLC and its derivatives and to increase tremor suppression, Taheri *et al.* [26] proposed an algorithm that uses torque to suppress tremor, showing promising results. Taheri *et al.*, extended their work in [29] to estimate the muscle torque that produced the tremor instead of estimating tremor, and then applied an equal and opposite torque to suppress tremor. Other research has focused on predicting future steps of tremor in an attempt to improve tremor suppression; however, Veluvolu *et al.* [36] demonstrated that integrating autoregressive models, the WFLC, and the BMFLC with a Kalman Filter (KF) to perform one-step prediction of tremor, still resulted in poor performance. The accuracies achieved were as low as 40% for up to 20 ms of future prediction. Recently, the WAKE framework [37] was proposed, based on wavelet decomposition and adaptive Kalman filtering; however, it is limited to estimation and one-step prediction. Another framework, PHTNet has been recently proposed by Shahtalebi *et al.* [38]. It is limited to the prediction of only one step ahead, and it introduces a time delay of 100 and 150 ms before it can properly estimate the signal. This delay is detrimental when applied to real-time control, as it will negatively affect the performance of the wearable assistive devices that manage tremor.

Due to the challenges and limitations presented in the previous work, deep neural networks

(DNN) [58] were considered. DNNs have shown their robustness in learning from data, and high prediction accuracies in different areas including robotics and assistive devices that use motion data as their input. As such, in order to effectively suppress tremor with wearable assistive devices and overcome the time delay problem, this work implements a novel approach based on a one-dimensional Convolutional-Multilayer Perceptron model (1D-CNN-MLP) for multistep prediction. The robustness of the 1D-CNN-MLP model for voluntary and tremor time series prediction and estimation was shown in a previous study by the authors [59], where it was demonstrated that the 1D-CNN-MLP model was able to learn correlations between past and present events, and future events. Given a time series, such as tremor signals and voluntary motion, the 1D-CNN-MLP model reads a string of numbers that represent tremor or voluntary motion (e.g., acceleration, velocity, or position) and predicts the number that is most likely to occur next.

In this paper, the performance of the proposed 1D-CNNMLP model was evaluated and compared to four known DNN architectures that are based on Recurrent Neural Networks (RNN) to find the best architecture that could be used for WTSDs. Even though the networks have never been used for WTSDs, nor for multi-step prediction, the RNN-based architectures have been recognized and shown to be robust when used with time-series data [60–64]. The RNN-based architectures are the Long–Short Term Memory (LSTM) [54], the Gated Recurrent Unit (GRU) [55], the Bidirectional LSTM (BLSTM), and the Bidirectional GRU (BGRU) [56]. While the 1D-CNN-MLP has been shown to be robust to time-series data, it also requires less computational power and performs automatic feature extraction, as will be discussed in Section 3.3. As such, the 1D-CNN-MLP was compared to the RNN-based architectures to find the best model to use with WTSDs. The performance of the 1D-CNN-MLP was then evaluated and compared to the Weighted Fourier Linear Combiner (WFLC), which is commonly used with WTSDs.

Therefore, the contribution of this work is a model that can learn directly from data, can adapt to the various tremor frequencies and amplitudes, and can learn the nonlinear relationships between voluntary motion and tremor in order to increase tremor suppression when used with WTSDs. Another contribution of this work is a DNN model that can predict 10, 20, 50, and 100 steps ahead with high accuracy, and low prediction time.

3.3 One Dimensional Convolutional-Multilayer Perceptron and Recurrent Neural Networks

A Convolutional Neural Network (CNN) is one type of Neural Network (NN) architecture that takes advantage of the concept of convolutions to learn higher-order features [44, 59, 65]. CNNs are mainly used for image classification and object recognition. They are also used for complex tasks such as sound and text recognition [66–68]. CNNs are known for their automatic feature extraction process, where they learn an internal representation of n-dimensional data. The same process can be exploited for one-dimensional sequences of data, such as time-series data of hand motion that are collected using Inertial Measurement Units (IMU). In this work, the angular velocity data from the three axes (x, y, and z) were used in the training and testing sets; however, the prediction was performed on a single axis. Angular velocity represents the rate at which an object rotates or revolves around an axis with respect to time. CNNs learn to map the internal features extracted from sequences of observations (angular velocity data) to follow a motion sequence. A sequence of convolutions are performed in the first convolutional layer. The sum of convolutions is passed through the activation function f, followed by a sub-sampling operation before passing the one-dimensional sequences of data to the following layers to learn and extract the features. Herein, the output of the last CNN layer was used in the prediction task that is performed by the Multi-Layer Perceptron (MLP) layer [59] that estimates and predicts voluntary motion. Due to the nonlinear, nonstationary, and stochastic nature of PD tremor, a hybrid of a one-dimensional CNN (1D-CNN) and a MLP network were considered based on the previous work proposed by the authors [59], as shown in Fig. 3.1.

The LSTM architecture was introduced by Hochreiter *et al.* [54] to solve the vanishing and exploding gradients problem that RNNs face. The concept of LSTMs relies on memory cells

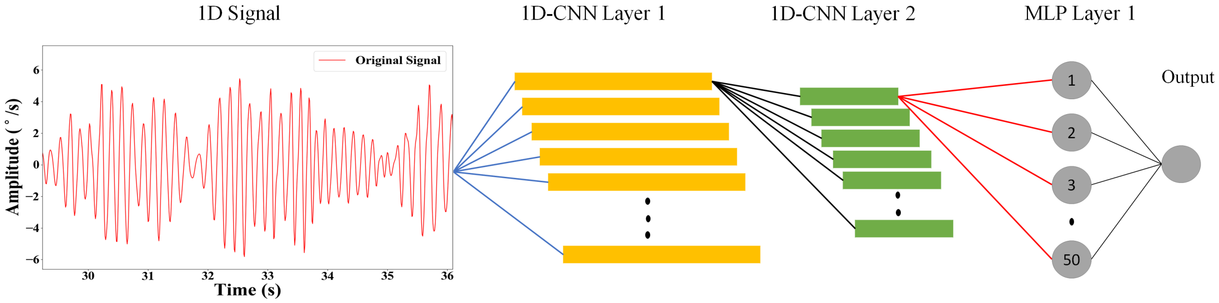


Figure 3.1: A sample of network architecture configuration with 2 CNN and 1 MLP layers.

and gated units that regulate the flow of information, and enables them to learn the important data from the input sequence to make a prediction. The LSTM cell consists of a forget gate, an update gate, an output gate, and a memory cell [54]. The memory cell carries relevant information throughout the processing of the input sequence, and information is added to and removed from the memory cell through the different gates. The forget gate decides which information should be kept and which information should be removed from the previous states. The update gate decides which information should be updated, and which information is important to keep from the current step. The output gate decides what the prediction is and what the hidden state should be.

The GRU architecture was recently introduced by Cho *et al.* [55]. Similar to LSTMs, GRUs consist of gate units but do not contain memory cells. The GRU architecture is simpler than that of the LSTM, as it only consists of a reset gate and an update gate. The reset gate allows the GRU cell to drop any information that is considered to be irrelevant in the future, thus, allowing for a more compact representation of data. The update gate, on the other hand, controls how the information from previous hidden states carries over to the current hidden state. The update gate acts similarly to the memory cell in the LSTM network. It allows the network to remember long-term information, and decides whether the hidden state is to be updated with a new hidden state.

The BLSTM and BGRU work by training the network simultaneously in the positive and negative time direction. In other words, the network will run from the past to the future, and from the future to the past [56].

LSTMs and GRUs are able to preserve information from both the past and the future at any point in time due to their hidden states, which makes them robust for time-series data analysis. The RNN-based networks have complex architectures that require more computation and memory to hold the information learned, while the 1D-CNN-MLP requires less computational power and memory storage than the LSTMs and GRUs, which leads to a faster training process.

Regardless of the limitations that the RNN-based architectures have, which are their complexity and their slower training process, they have been shown to be robust with time-series data. Moreover, the 1D-CNN-MLP model is a new approach that has been shown to be robust with time-series data [59]. As a result, the five architectures were considered and compared to find the best architecture to be used with WTSDs.

3.4 Methods

To develop a voluntary motion estimator and a multi-step predictor, a two-step method was followed. The first step presents a comparison between the five different neural network (NN) architectures. The best architecture was then used in a simulation study that shows the performance of the model when predicting voluntary motion. In the second step, the model with the best performance in simulation is used in a real-life experimental setup, where one degree-of-freedom (DOF) motion data were reproduced using a DC brushless motor. A second motor was used to track and predict the voluntary motion using the best model from the first step. The goal of this study is to find a method that is based on neural networks, that can learn directly from data to estimate and predict voluntary motion when used with WTSDs to increase tremor suppression. The method was evaluated to show the performance of the predictor on voluntary motion up to 100 steps ahead.

The following subsections present the data collection and preparation, the implementation and accuracy measures, and the comparison of the five DNNs.

3.4.1 Data Collection

A total of 18 subjects with PD participated in this study. The data were recorded at the Wearable Biomechatronics Laboratory at Western University [20]. The participants were recruited by a movement disorders neurologist, and the study was approved by the Health Sciences Research Ethics Board (#106172). The PD group included eleven males and seven females with varying tremor severity. Their ages ranged from 60 to 84 (mean ± standard deviation is 69 ± 7). 16 out of the 18 participants were on medication (Levodopa, Pramipexole, and Amantadine), but did not stop taking their medicine before they arrived at the lab for data recording. Furthermore, the tremor scores from the MDS-UPDRS Motor Exam Part 3 were recorded, and they are as follows: for Item 15, the mean score was 1.56 ± 0.92. For Item 16, the mean score was 0.83 ± 0.71. For Item 17, the mean score was 2.72 ± 0.67. For Item 18, the mean score was 3.61 ± 0.92. Data were collected from the hand with the dominant tremor for each of the PD participants.

IMUs were used to record the data, and they were placed as shown in Fig. 3.2. IMU 1 was affixed to the wrist support (above the ulna and radius bones) at the distal end of the forearm, IMUs 2, 3, 4 and 5 were affixed to the thumb metacarpal bone, the index and middle fingers

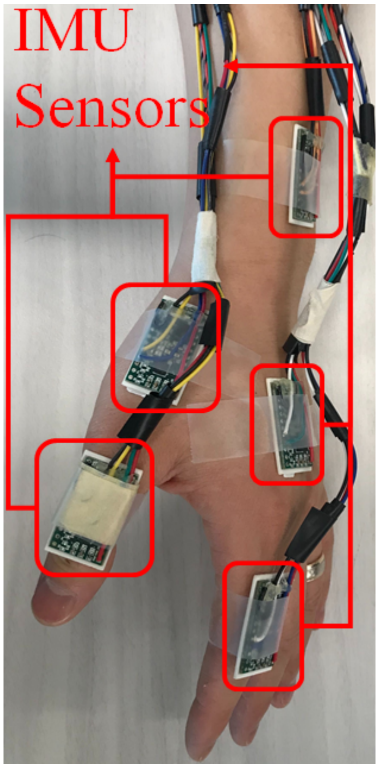


Figure 3.2: Visualization of the placement of the IMU sensors on the wrist support, the thumb support, the hand support, and the (MCP) joints of the thumb and the index finger.

metacarpal bones (dorsal side), and the metacarpophalangeal (MCP) joints of the thumb and the index finger, respectively. The motion of each target joint was measured differentially with the two nearby IMUs. The sensors communicated with an NXP LP1768 microcontroller through a serial peripheral interface (SPI) at a sampling rate of 100 Hz.

PD tremor can be classified into three types of tremor, resting, postural, and action tremor. Resting tremor occurs when a limb is in a resting position, postural tremor can be observed when an individual with PD tremor maintains a hand position against gravity, and action tremor is observed when an individual with PD tremor performs a voluntary movement. To assess the three types of PD tremor, the participants performed six different tasks, as follows:

1. The participant's hand was in a resting position with the palm facing down.
2. The participant's hand was in a resting position with the palm facing up.
3. The participant's hand was held in a postural position.
4. The participant was asked to extend and flex their wrist, and pinch a pencil when extending the wrist joint.

5. The participant was asked to move a pencil with the thumb and index finger when extending and flexing the wrist joint.

6. The participant was asked to draw a spiral.

The data for the first three tasks were recorded for sixty seconds. Both tasks four and five were repeated five times, and the sixth task was performed once.

3.4.2 Data Preparation

Recent studies have shown that the frequencies of parkinsonian tremor range from 3 to 17 Hz [20, 23], and voluntary motion frequencies range from 0 to 2 Hz [20, 23]. For the network to be able to learn and distinguish the difference between tremor and voluntary motion, it was trained on both tremor and voluntary motion data. To extract the tremor signal, the IMUs were aligned to the user's joint in the direction of flexion–extension, and the angular velocity data of the three axes were used to train the neural network. A 4^{th} order Butterworth band-pass filter with cutoff frequency between 3 and 17 Hz was applied to the original motion from each axis. To extract and differentiate the ground truth voluntary motion from the action tremor, a low-pass filter with cutoff frequency of 2 Hz was applied to the original motion data of tasks four, five and six. After filtering the data, zero-phase shifting with forward–backward filtering was applied offline to remove the resulting phase shift before training the neural network models.

In order to develop a generalized model and to assess its performance on new unseen data, the data were divided into a training set, a validation set, and a testing set. The training set consisted of data from twelve PD participants, the validation dataset consisted of data from a different set of three PD participants, and the testing dataset consisted of a final set with the remaining three PD participants. The test set was only used to evaluate the performance of the neural networks after they were trained using the training and validation sets. The training process consisted of five-fold cross validation that was repeated 50 times using the data from the training and validation datasets combined to make sure that the model was not overfitted. It was then tested on new unseen data with the data from the testing set. The training, validation, and testing sets included all the repetitions of all the tasks of the participants' data that were in each set. Furthermore, after analyzing the recorded data from the PD participants, it was found that there is large variability in the data on which the model can be evaluated, as can be seen in Fig. 3.3.

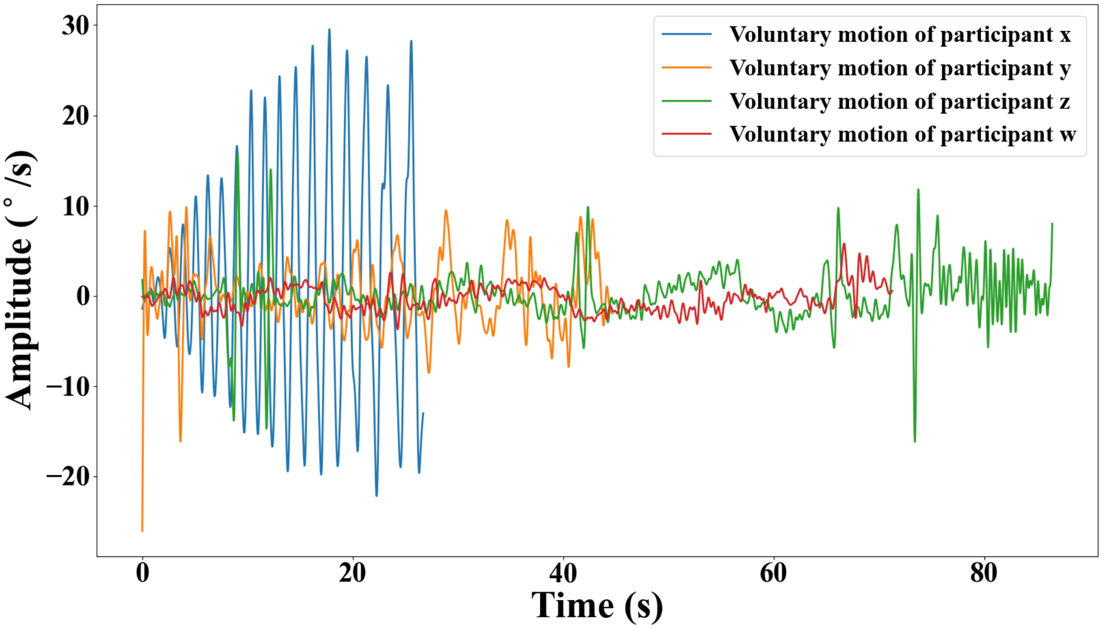


Figure 3.3: Visualization of randomly selected data from different PD participants to show the difference in the data between participants.

These data represent the unseen data from the testing dataset. The voluntary motion frequencies differ from one participant to another, and the motion is affected by the tremor that alters the patterns in each participants' data. Given this variability, the data provides a realistic basis from which to assess the applicability of the models to real-life scenarios.

3.4.3 Implementation and Accuracy Assessment

Data processing and the implementation of the five DNN architectures were performed using Python 3.7 and TensorFlow 2.0. The five networks were trained on an Intel® Core™ i7-9700 CPU @ 4.7 GHz PC, using an NVIDIA GeForce RTX 2060 to speed up the training process.

The grid search approach was used to tune the hyperparameters of the models. Grid search is a common method that involves creating a grid of possible values for each hyperparameter of the model(s) and evaluating the model's performance for each unique combination of the hyperparameters.

The root mean squared error (RMSE) metric was used to assess the performance of the models, and it is calculated as follows:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}}, \tag{3.1}$$

where $\hat{y}_i$ is the predicted value, y_i is the ground truth value, and N is the length of the input sequence.

To show the performance of the models when tracking voluntary motion, the prediction percentage accuracy (PPA) for multi-step future prediction, and the estimation percentage accuracy (EPA) were computed. Estimation is the process of approximating seen observations, while prediction is the process of forecasting future unseen observations. Multi-step ahead prediction is the sequential prediction of multiple successive future outputs. The PPA is calculated as follows:

$$\left(1 - \frac{\mid \text{True Value} - \text{Predicted Value} \mid}{\mid \text{True Value} \mid}\right) \times 100. \tag{3.2}$$

The EPA was calculated using the PPA equation and replacing the “Predicted Value” by “Estimated Value”. The network with the lowest RMSE was then used in the simulation analysis and in the experimental analysis.

3.4.4 Statistical Analysis

A statistical analysis was conducted to compare the performance of the five networks with five different numbers of layers. The data were tested for normality, and it was found that they are nonparametric. As a result, a Kruskal-Wallis test was performed to identify differences between each predictive model. The analysis was performed using the IBM Statistical Package for the Social Sciences (SPSS version 25) statistics software.

3.4.5 Network Comparison

To find the network architecture that performed the best, the five networks were trained with tremor and voluntary motion data separately, in order for the networks to learn the difference between them. They were compared using the 5-fold cross validation repeated 50 times and with different numbers of hidden layers, as presented in Fig. 3.4. The RMSE accuracy measure was used to compare the models, as the problem is a regression problem, and it gives a clear indication of which one performed better. In the four RNN based architectures, 100 nodes were used in each layer. In the CNN-MLP network, each of the CNN layers consists of 64 filters, which are the learnable weights of the network, a kernel of size two, and a 1D-MaxPooling layer of size two; the MLP network consisted of one layer that consisted of 50 nodes, and the output layer.

The activation function that was used in all of the hidden layers is the Rectified Linear Unit (ReLU) [69]. The networks were trained using the *Adam* optimizer [52] and a learning rate of

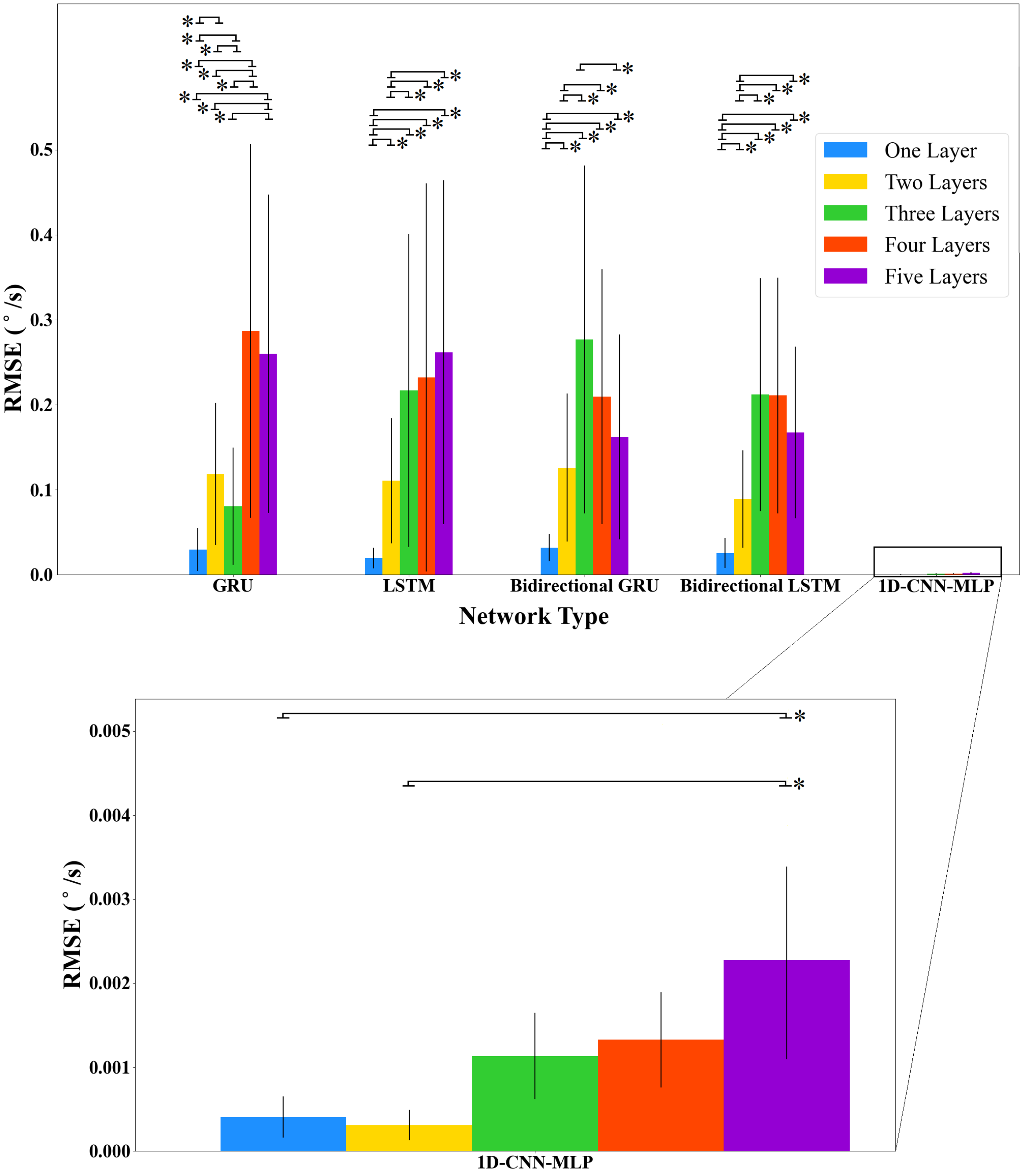


Figure 3.4: Comparison of the five neural network architectures with five different layers. The black bars represent the standard deviation (±). A significant difference ($p < 0.05$) between the number of layers in each network is indicated by a "*".

0.0001 with a decay of 1e-6.

The results of the five DNNs showed that they performed very well using hand motion data, where the average RMSE of each of the five networks with different layers was below 0.5°/s. The 1D-CNN-MLP network had the lowest RMSE (RMSE $< 0.003°/s$). One of the reasons why the 1D CNNs performed well is because they were designed with a similar concept to 2D CNNs. Vanilla

CNN or 2D CNN are designed to operate exclusively on 2D data such as images and videos, and learn higher-order features from the data. To work with one-dimensional data, a modified version of the 2D CNNs has been developed. Recent studies [59] have shown that 1D CNNs are advantageous when dealing with 1D signals, such as time-series data. The forward propagation and backward propagation of the 1D CNNs require simple array operations, which result in a lower computational complexity compared to 2D CNNs and to RNN-based architectures. The first 1D CNN layer defines a set of filters to learn multiple features, which will be the input to the following 1D CNN layer. The second 1D CNN layer will also define a set of layers that will learn more features, and so on. As a result, the 1D CNN is able to automatically learn features about the data, which makes it more robust compared to the RNN-based architectures, which retain information using the memory cells, but do not perform automatic feature extraction.

The statistical analysis showed that there is a significant difference between the 1D-CNN-MLP with different layers and with the other networks with different layers. The statistical analysis showed that within each of the first four networks (GRU, LSTM, BGRU, BLSTM), there are significant differences between the different layers.

With regard to the 1D-CNN-MLP networks, the results showed that there is a significant difference between 1D-CNN-MLP with one layer and five layers, and between 1D-CNN-MLP with two layers and five layers. There were no significant differences between networks with any other combination of layer numbers. Given what has been presented, the 1D-CNN-MLP network with two 1D CNN layers was chosen arbitrarily since the 1D-CNN-MLP with one 1D CNN layer and the 1D-CNN-MLP with two 1D CNN layers had the same performance and there was no advantage of choosing one over the other. In addition to performance, and based on how the data were structured and prepared for training, there was no noticeable difference computation and training wise when choosing one over the other, where both networks took the same amount of time to train. As such, the 1D-CNN-MLP network with two 1D CNN layers and one MLP layer was used in simulation as presented in the following section.

3.5 Simulation Results

In this section, the performance of the 1D-CNN-MLP model for estimation and for multi-step future prediction of voluntary motion is evaluated in different scenarios. Figs. 3.5, 3.6, and 3.7

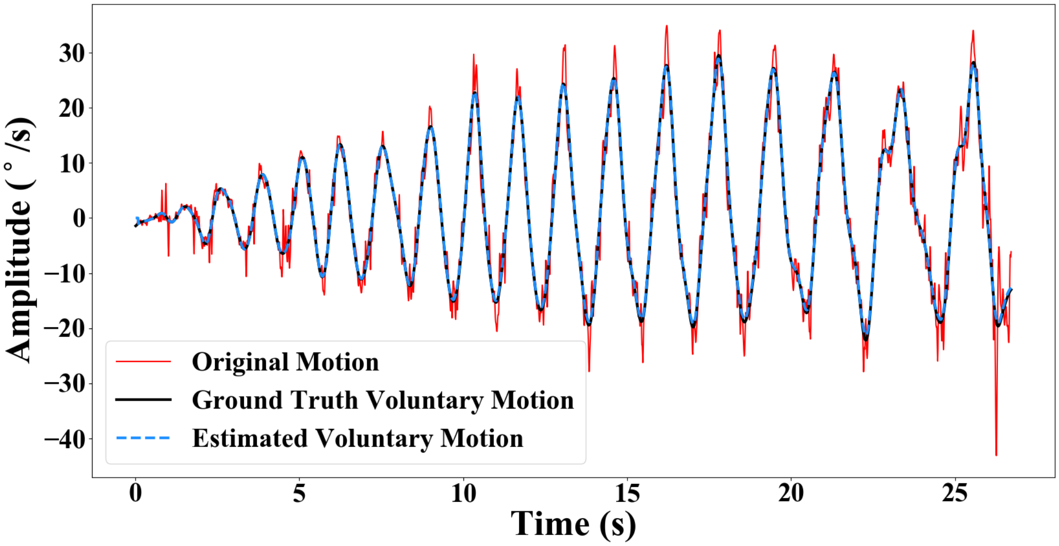


Figure 3.5: Visualization of a sample output of the 1D-CNN-MLP model when the data of Task 6 for Participant 16 were used as an input. Task 6 represents the data when the PD participants drew a spiral.

show the performance of the model for different tasks that were chosen randomly from the testing dataset. Fig. 3.5 shows data from the original motion performed by Participant 16, the ground truth voluntary motion (black) that was extracted as discussed in Section 3.4.2, and the estimated voluntary motion (blue). Figs. 3.6 and 3.7 show the performance of the model on new unseen participant data. They show the original voluntary motion (black), the estimated voluntary motion (blue), and the motion predicted (green) 20 steps ahead, where the prediction is highlighted in the second inset in both figures. Table 3.1 shows the performance of the model when estimating voluntary motion, and Table 3.2 shows the performance of the model for different multistep predictions of voluntary motion. The time it took the model for multistep prediction was also measured, and the results are shown in Table 3.3.

Table 3.1: Voluntary motion estimation performance of the 1D-CNN-MLP model during Tasks 4, 5, and 6.

Subject	**Voluntary Motion - EPA**	**SD ±**
16	99.5%	0.2%
17	99.0%	0.23%
18	99.1%	0.15%
Average	99.2%	0.19%

The 1D-CNN-MLP model achieved an average EPA of 99.2% for voluntary motion when tested

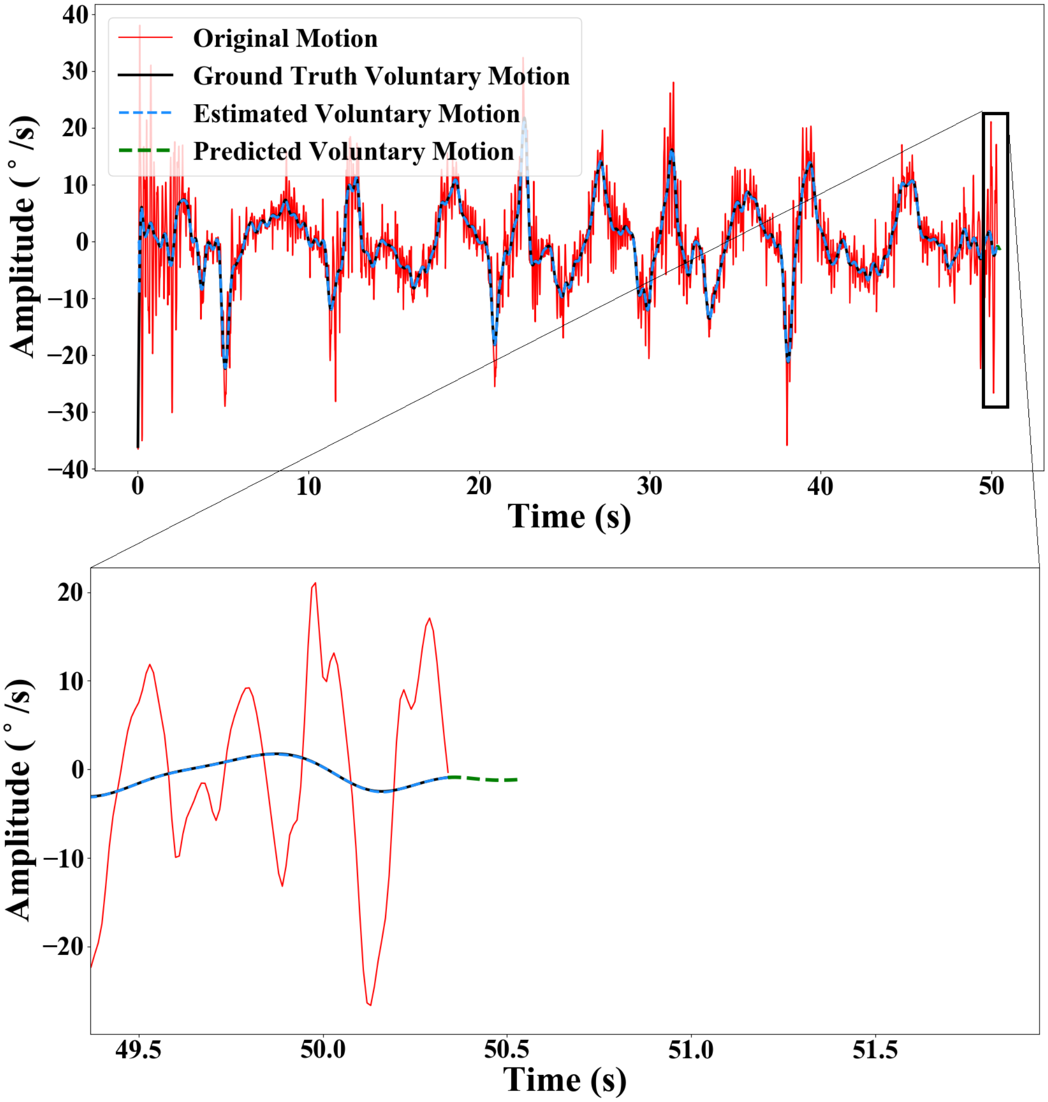


Figure 3.6: Visualization of a sample output of the 1D-CNN-MLP model when the data of Task 5 for Participant 17 were used as an input. Task 5 represents the data when the PD participants were asked to move a pencil with their thumb and index finger while extending and flexing their wrist joint.

Table 3.2: Voluntary motion prediction performance of the 1D-CNN-MLP model for different steps into the future.

Subject	**10 steps**	**20 steps**	**50 steps**	**100 steps**
16	97.0%	94.8%	90.2%	89.3%
17	96.2%	93.6%	92.6%	91.4%
18	98.4%	95.8%	91.7%	90.2%
Average - PPA	97.2%	94.7%	91.5%	90.3%
SD ±	0.8%	1.2%	0.8%	1%

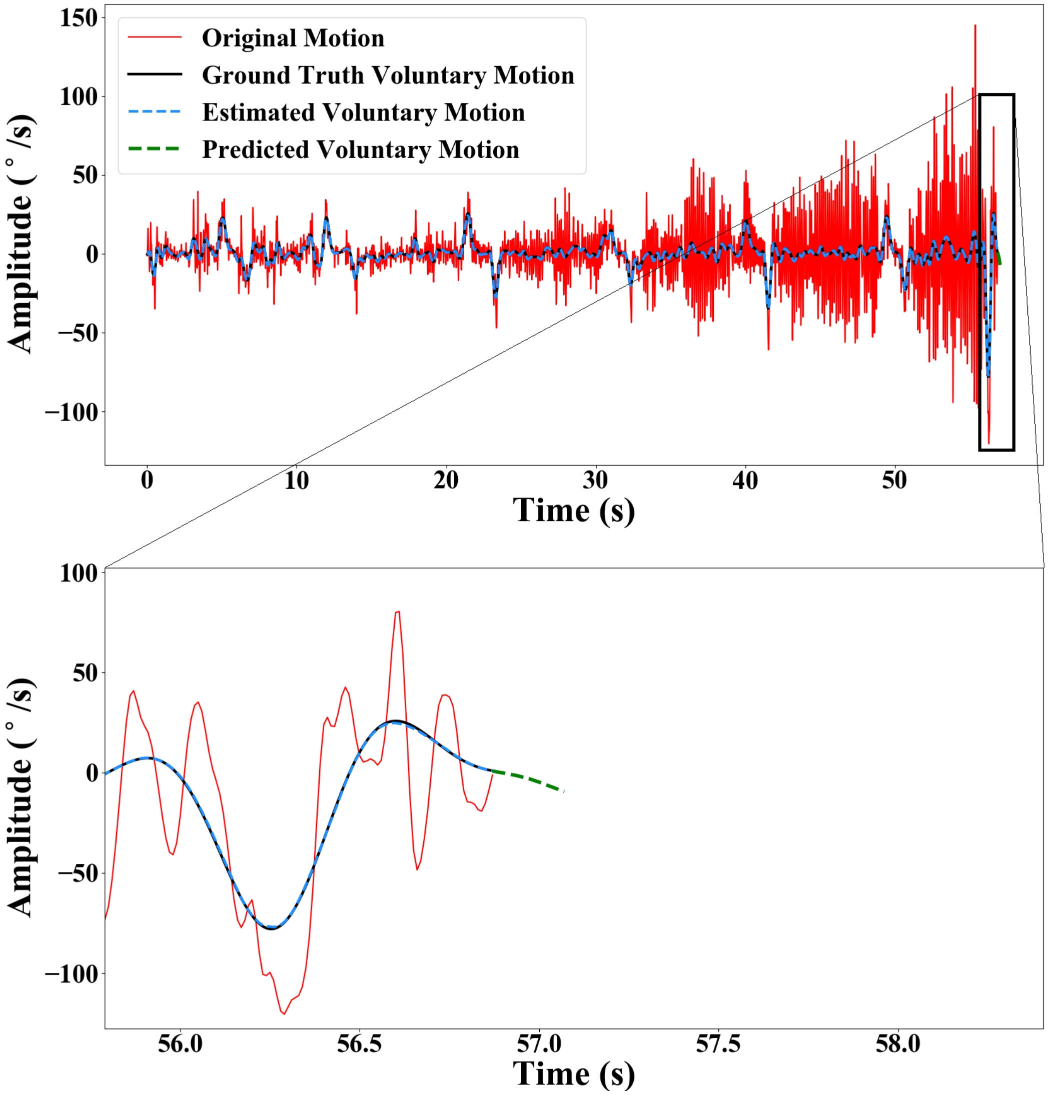


Figure 3.7: Visualization of a sample output of the 1D-CNN-MLP model when the data of Task 4 for Participant 18 were used as an input. Task 4 represents the data when the PD participants were asked to extend and flex their wrist and tap their thumb and index finger together when their wrist joint is extended.

Table 3.3: Multistep prediction time for the 1D-CNN-MLP model.

Number of steps ahead	**Time (ms)**
10	0.9
20	0.96
50	1
100	1.5

using the nine tasks of new data with an RMSE of 0.000313°/s. On the other hand, the 1D-CNN-MLP model achieved an average PPA of 97.0%, 94.0%, 91.7%, and 90.6% for 10, 20, 50, and 100 steps ahead of voluntary motion, respectively. The 1D-CNN-MLP model achieved an RMSE of 0.001, 0.0018, 0.007, and 0.04°/s for 10, 20, 50, and 100 steps ahead, respectively.

The results presented in this section show the robustness of the proposed model to track and predict voluntary motions. The results also show that the proposed model outperforms the ones proposed in the literature [23, 30, 31, 33, 36–38] for both estimation and multistep future prediction. This is important and integral to the development and enhancement of wearable tremor suppression gloves (WTSGs) and wearable assistive devices, in order to improve the effectiveness of tremor management delivered by these devices.

The following sections present the design of an experimental setup for motion simulation. The proposed 1D-CNN-MLP model will be compared to one of the most common estimators used in robotic devices that manage tremor—the Weighted Fourier Linear Combiner (WFLC) [31]. Both models will be used and evaluated based on their ability to help a tremor suppressor track voluntary motion while suppressing tremor.

3.6 Motion Reproduction: Experimental Setup

This section presents the experimental setup used to compare the WFLC and the proposed 1D-CNN-MLP model. The objective of this experiment was to evaluate the performance of the two models in tracking voluntary motion while suppressing tremor. This is important since the end goal is to effectively predict voluntary motion and help in tremor suppression when integrated with WTSDs for individuals living with parkinsonian tremor.

3.6.1 Experimental Setup and Evaluation

The design of the bench-top setup is shown in Figs. 3.8 and 3.9. It consists of two brushless DC motors, each with a planetary gearhead (EC-max 16, reduction ratio 29:1, Maxon Motors, with EPOS2 24/2 controllers).

The first motor (motion generator) was used as a simulator to reproduce the recorded motions of the index finger of the PD participants. The second motor (voluntary motion predictor and tremor suppressor) was controlled by the 1D-CNN-MLP model and the WFLC to compare their performances. The data from the 18 PD participants were used in the bench top experimental setup. The recorded participants' motion data were sent directly to the motion generator motor

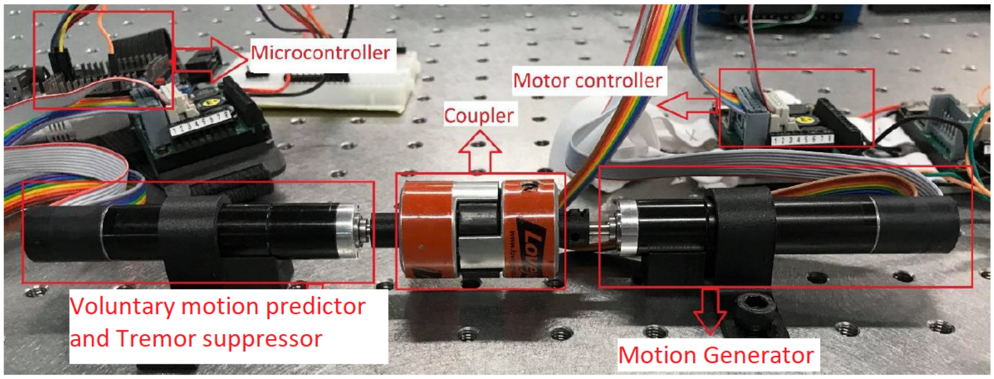


Figure 3.8: Visualization of the experimental setup—showing the two motors connected by a coupler, one motor was used to simulate hand motion, and the second was controlled by either of the models to track voluntary motion and suppress tremor.

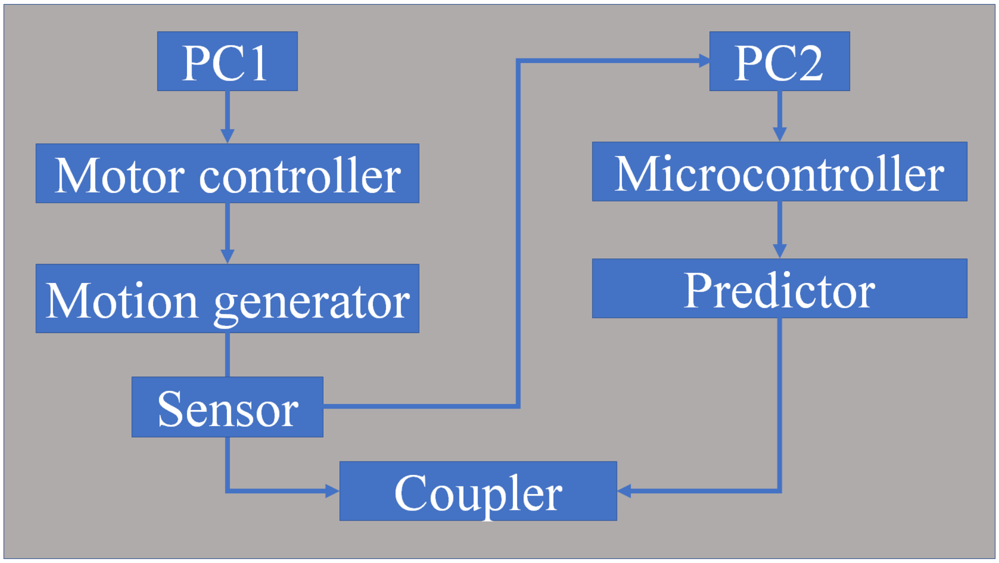


Figure 3.9: A block diagram of the experimental setup that specifies the input and the output signals.

controller using LabView Software (Version 2014, NI). The voluntary motion predictor was connected to the same PC described in Section 3.4.3. The two motors were connected by a coupler, and an IMU sensor was placed on the right side of the coupler to read the motion reproduced by the simulator. The data were then sent to the PC to be processed. The processed data were used as the input to the WFLC and the 1D-CNN-MLP models. The output of the models—which is the estimated voluntary motion—was then used as the command motion for the motor. The performances of the WFLC and the 1D-CNN-MLP models were evaluated by calculating the percentage reduction of tremor using the power spectral density (PSD) of the outputs of the WFLC and the 1D-CNN-MLP model. The PSD estimates the power distribution of the input signal over a specific frequency range. This frequency domain feature captures the overall frequency content of the specific signal and is expressed as the Fourier transform of the autocorrelation of the obtained tremor signal. The PSD of both the original signal and the output signal of the model were then calculated as discussed in [27] as follows:

$$\phi(f) = \frac{\mathrm{FFT}(x)}{N}, \tag{3.3}$$

$$\mathrm{PSD} = \sum_{i=f_1}^{f_2} \frac{\phi(i)}{T} = \frac{1}{Nt_\mathrm{s}} \sum_{i=f_1}^{f_2} \phi(i), \tag{3.4}$$

where x is the signal in the time domain, N is the length of the signal, t_s is the sampling period, and f_1 and f_2 are the lower and upper limits of the range of interest. The tremor PSD reduction was then calculated by subtracting the PSD of the measured tremor motion after the tremor suppressor was activated, from the PSD of the original tremor motion, then dividing by the PSD of the original tremor motion, and then multiplying by 100 as follows:

$$\mathrm{PSD}_{\mathrm{reduction}} = \frac{\mathrm{PSD}_{\mathrm{original}} - \mathrm{PSD}_{\mathrm{suppressed}}}{\mathrm{PSD}_{\mathrm{original}}} \times 100. \tag{3.5}$$

The RMSE was also used to calculate how well both models performed in tracking voluntary motion.

3.6.2 Statistical Analysis

A statistical analysis was conducted to compare the performance of the 1D-CNN-MLP model to the WFLC during all six tasks. The data have been tested for normality, and the results showed that they are not normally distributed for the 1D-CNN-MLP with Tasks 1 and 3, and for the WLFC with Task 4. As such, the Kruskal-Wallis test was performed to identify differences between the two models across these tasks, and a two-by-six repeated measures ANOVA with a Bonferroni correction post hoc test was performed to identify differences between the two models and the six tasks for the normally distributed data. The α was set to 0.05. The analysis was performed using SPSS.

3.6.3 Results and Discussion

In this section, the performance of the WFLC and the 1D-CNN-MLP models for tracking voluntary motion and suppressing tremor is presented. The statistical analysis showed that there is a significant difference ($p < 0.05$) between the 1D-CNN-MLP model and the WFLC across the six tasks, and that there is interaction between the two main factors. Fig. 3.10 shows a comparison between the performance of the WFLC and 1D-CNN-MLP model in reducing tremor across the six tasks. It is important to note that the analysis showed that there was no significant difference between the six tasks.

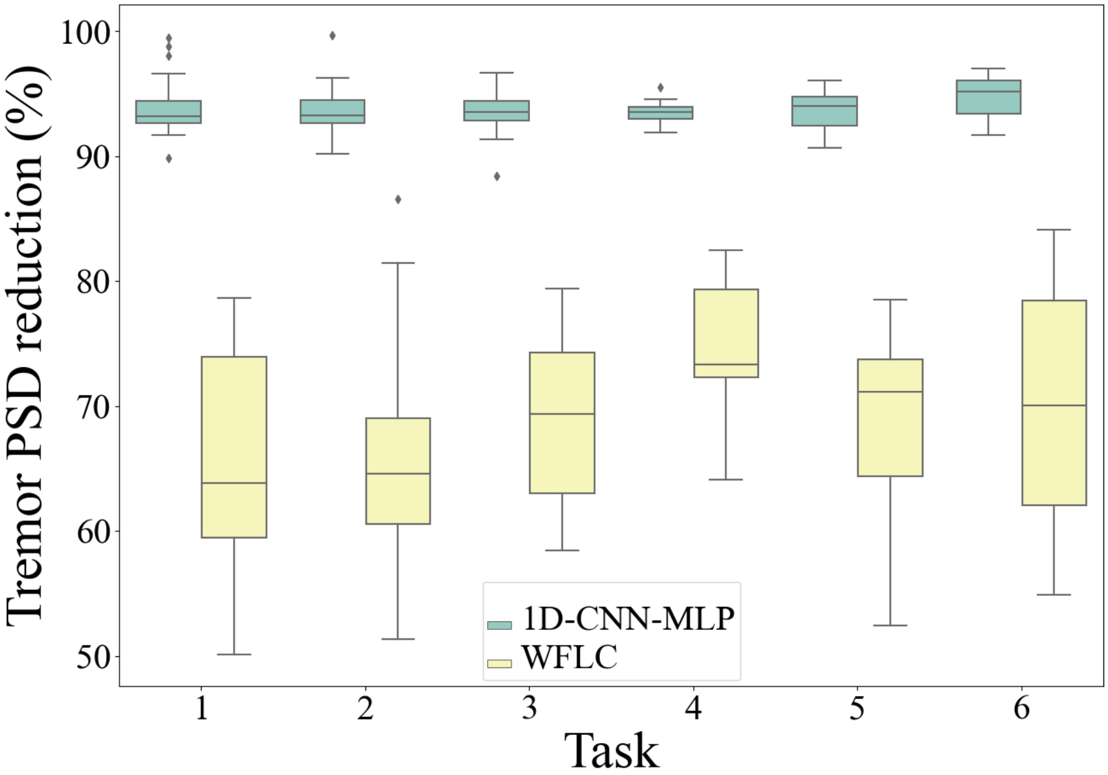


Figure 3.10: Comparison between the performance of the 1D-CNN-MLP model and the WFLC in suppressing tremor. The figure shows the tremor PSD percentage reduction by the WFLC, and the tremor PSD percentage reduction by the 1D-CNN-MLP model. The black line indicates the median, the box indicates the range between the 25% and the 75% quartiles. The x axis indicates the task number, and the y axis indicates the tremor PSD reduction.

The results show that the WFLC achieved an average of 68.8% ± 7.5% in tremor reduction, while the 1D-CNN-MLP model achieved an average of 93.8% ± 1.5% in tremor reduction, which is an improvement of 25%.

Figs. 3.11 and 3.13 show sample data with resting tremor and the output of the WFLC and

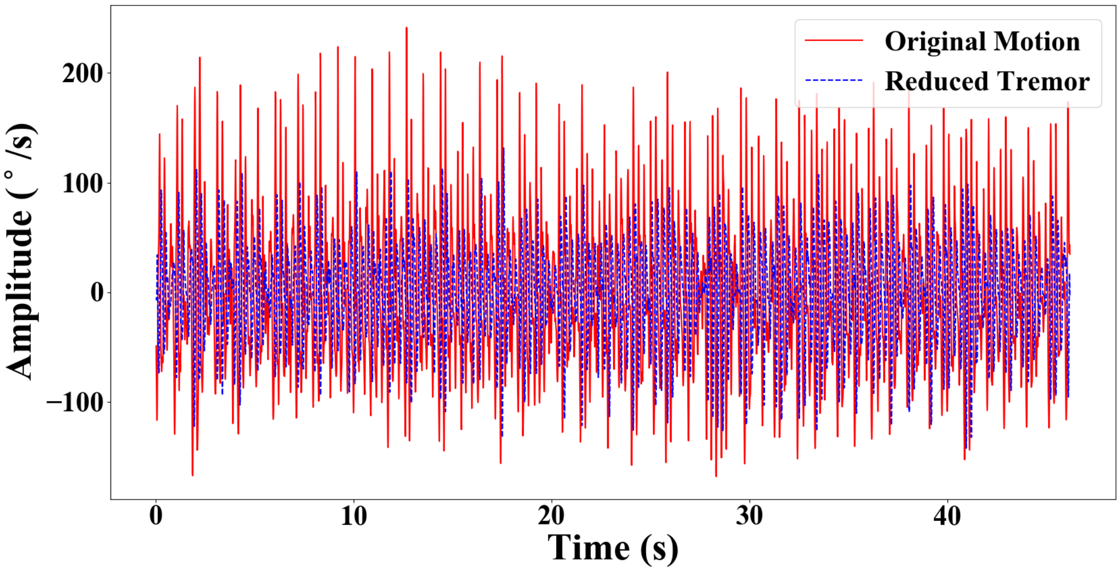


Figure 3.11: Visualization of a sample output of the WFLC when data of Task 2 were used as an input. Task 2 represents the data when the hand was in a resting position with the palm facing up.

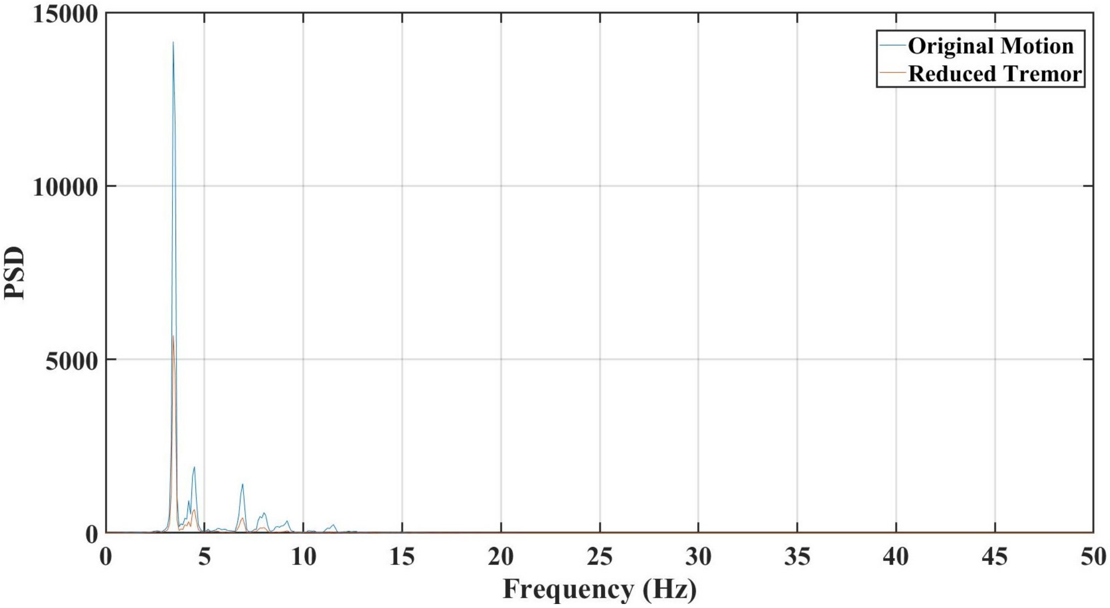


Figure 3.12: Visualization of PSD of the original and suppressed tremor for the WFLC model.

1D-CNN-MLP in suppressing tremor. Figs. 3.12 and 3.14 show the PSD of the original tremor and the suppressed tremor by the WFLC and the 1D-CNN-MLP respectively.

Fig. 3.15 shows a comparison of the performance of the WFLC and the 1D-CNN-MLP model when tracking voluntary motion of Tasks 4, 5 and 6, where the analysis has also shown that there was no significant difference between these tasks.

Figs. 3.16 and 3.17 show sample data of Task 4 that contains action tremor and voluntary

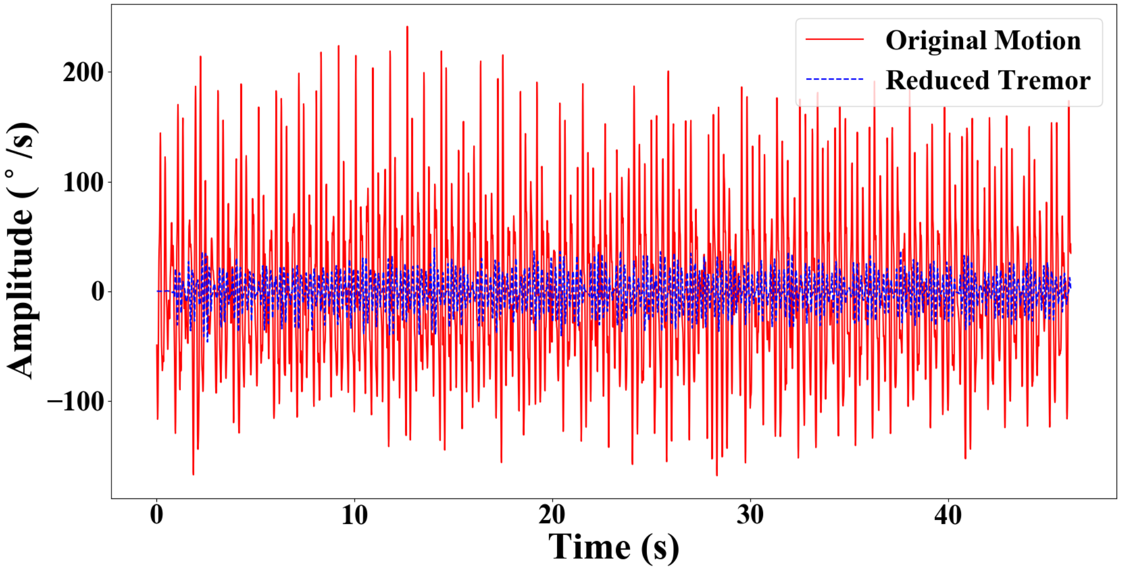


Figure 3.13: Visualization of a sample output of the 1D-CNN-MLP model when data of Task 2 were used as an input. Task 2 represents the data when the hand was in a resting position with the palm facing up.

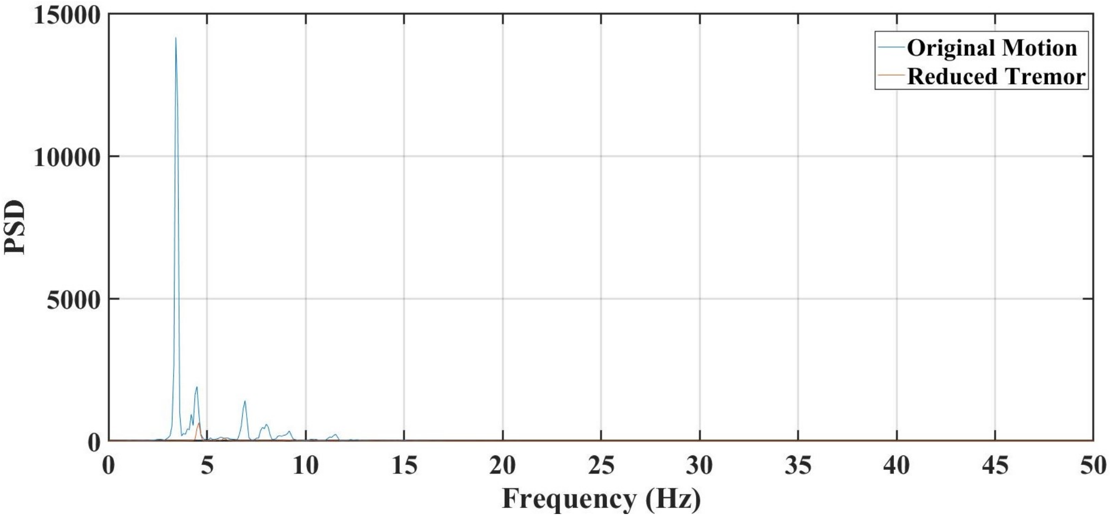


Figure 3.14: Visualization of PSD of the original and suppressed tremor for the 1D-CNN-MLP model.

motion. Fig. 3.16 shows the performance of the WFLC in estimating voluntary motion, while Fig. 3.17 shows how the 1D-CNN-MLP performed in real time when predicting voluntary motion. The input to both models was fed in real time using the data from the IMU, which measures the motion reproduced by the simulator. The 1D-CNN-MLP model was configured to predict 50 ms ahead to compensate for the time delay introduced by the system (data processing and motor response times). This is important to move the motor to the predicted position to follow voluntary motion and suppress tremor in real time. When estimating voluntary motion, the RMSE achieved

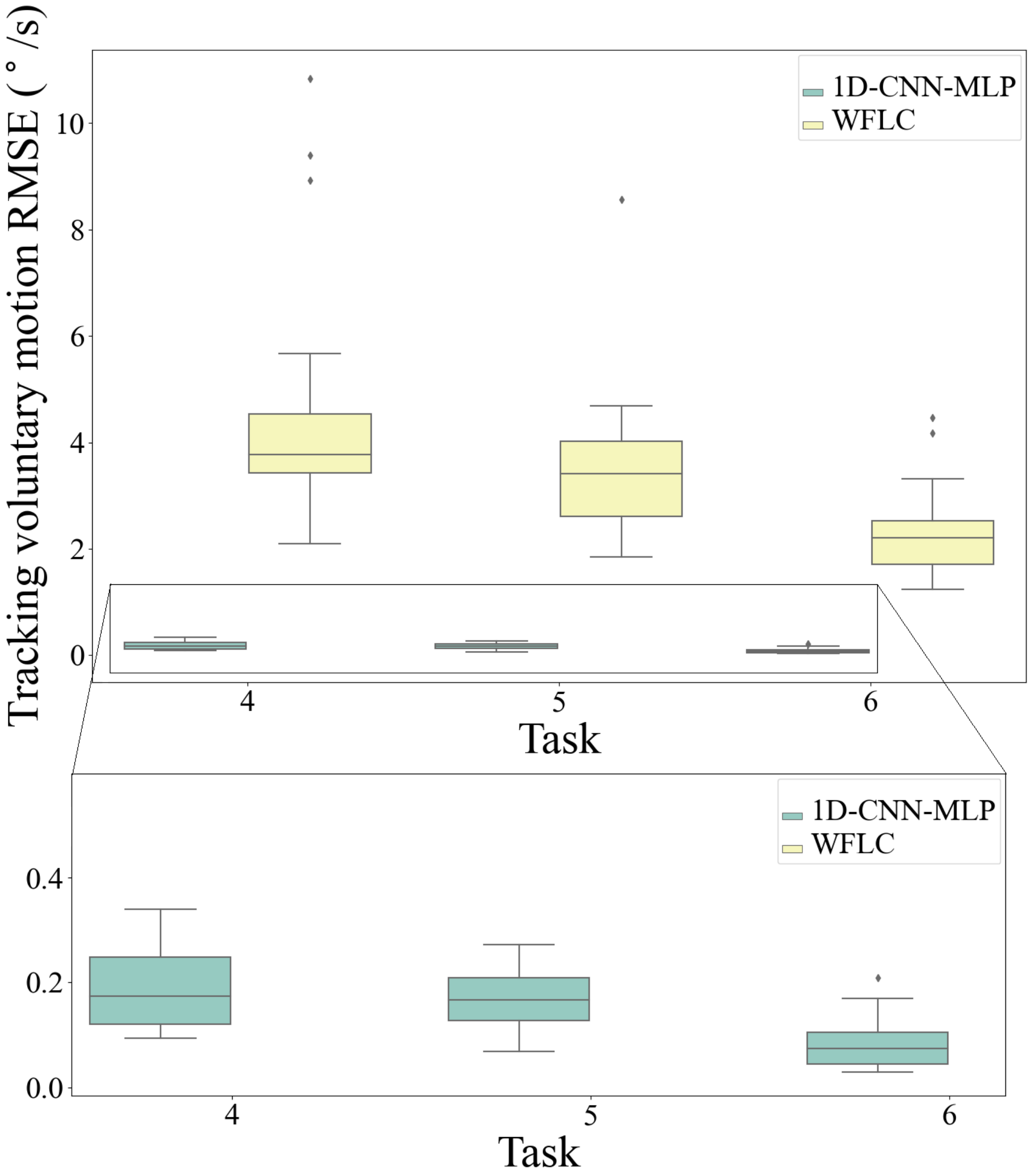


Figure 3.15: Comparison between the performance of the 1D-CNN-MLP model and the WFLC when tracking voluntary motion. The figure shows the RMSE of the WFLC, and the RMSE of the 1D-CNN-MLP model. The black line indicates the median and the box indicates the range between the 25% and the 75% quartiles. The x axis indicates the task number, and the y axis indicates the tremor PSD reduction. Note the different scales in the y axis.

by the WFLC in this scenario was 3.7°/s, while the 1D-CNN-MLP model achieved an RMSE of 0.07°/s.

The results of the WFLC output can be seen in Fig. 3.16, where it can not accurately track voluntary motion. This is likely due to the fact that the WFLC cannot track higher tremor harmonics adaptively and separately. Furthermore, the WFLC estimates a signal, which results

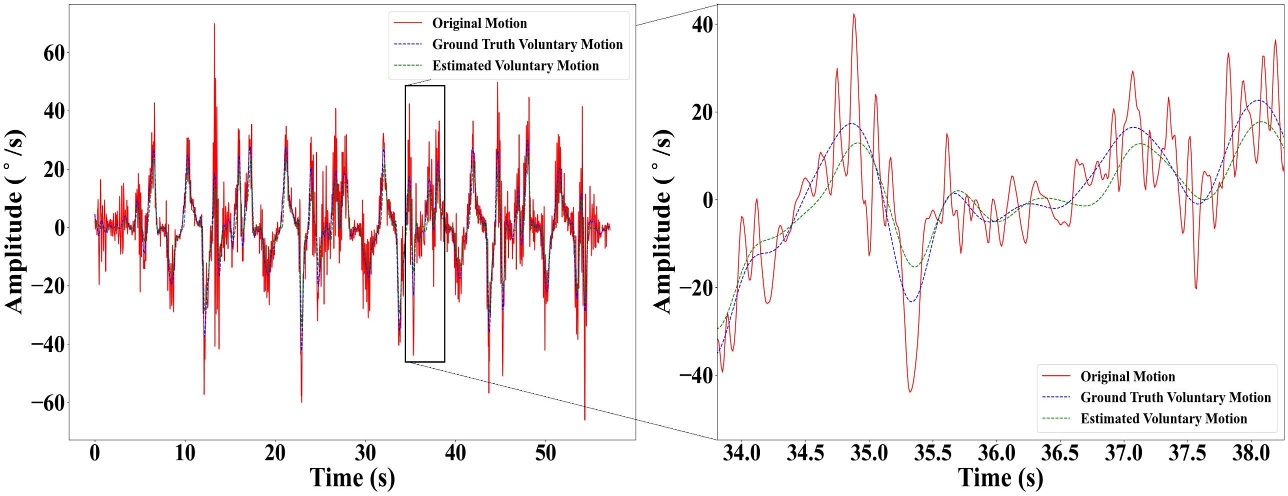


Figure 3.16: Visualization of a sample output of the WFLC when data of Task 4 were used as an input. Task 4 represents the data when the participants were asked to extend and flex their wrist, and tap their thumb and index finger together when their wrist joint is extended.

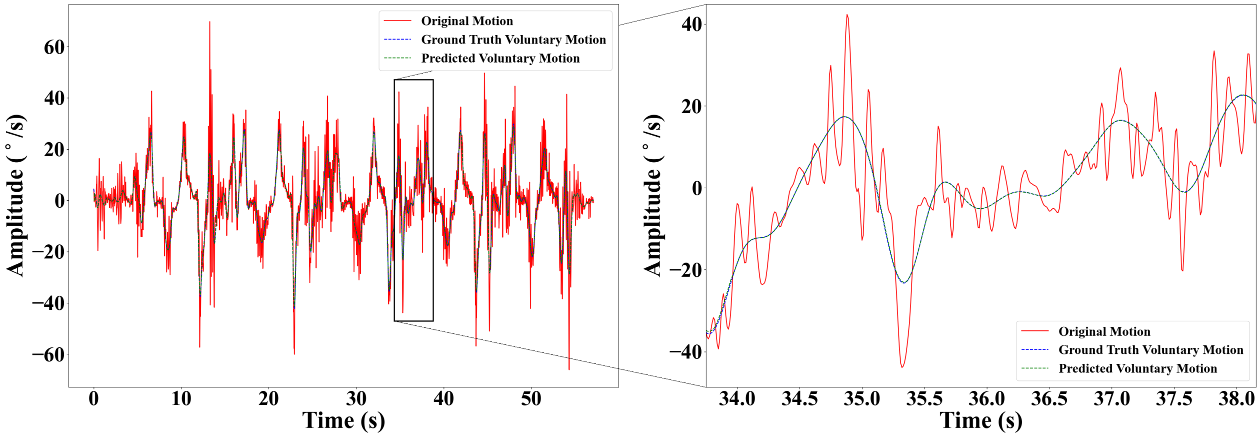


Figure 3.17: Visualization of a sample output of the 1D-CNN-MLP when data of Task 4 were used as an input. Task 4 represents the data when the participants were asked to extend and flex their wrist, and tap their thumb and index finger together when their wrist joint is extended.

in a delayed response from the motors, while the proposed model can learn directly from the data to predict the future steps.

The prediction accuracies of the 1D-CNN-MLP model decreased slightly in the experimental setup compared to the simulation results presented in Section 3.5, where the average RMSE increased from 0.012°/s to 0.12°/s, however, it is still considered relatively small. The increase in the RMSE could be the result of different factors. One of these factors is that the use of the model in the experimental setup is closer to the real-life behaviour of motion data under study, and the simulation results represent the behaviour of the model in an ideal situation. Other factors to be considered are the data transfer or missing data during data transfer, and motor(s) response time. One main limitation of using the DNN model proposed is the time it takes to find the optimal parameters for the network, and the training time of the network.

Despite the slight increase in the RMSE when predicting voluntary motion, the results of the 1D-CNN-MLP model still showed its robustness to learn and differentiate between voluntary motion and tremor from the data directly, and to predict future voluntary motion with high accuracies. It was able to increase the tremor reduction to 93.8% on average compared to the reduction achieved by the WFLC.

Lastly, although finding the optimal parameters and training the networks consumes time, once the network is trained it can be used without the need to retrain and find the parameters again. The computational time of the proposed model to predict future voluntary motion was as low as 1.5 ms for 100 steps ahead.

3.7 Conclusion

This work proposes the design and implementation of a novel user and task-independent voluntary motion predictor that is based on DNN. The results of the proposed work show its potential when tracking voluntary motion, and predicting up to 100 steps ahead with high accuracy. In both simulation and an experimental assessment, the proposed 1D-CNN-MLP model showed that it overcomes the drawbacks of the estimators and predictors proposed in the literature, especially the time delay for real-time tremor management required by wearable assistive devices.

Future work will focus on investigating the factors that contributed to the increase of RMSE when predicting voluntary motion for the bench-top experimental setup, and to minimize their effect as much as possible so that the model has better performance in a real-life situation. Future work will also investigate the integration of the developed model with the WTSD being developed in the Wearable Biomechatronics Laboratory at Western University. It will also be important to investigate whether there are any advantages to using the cloud to train the network online and on the go; however, this will require further investigation into encrypting patients' data to ensure that their privacy is not compromised. An alternative is to investigate the use of a more powerful computing unit (e.g., Nvidia Jetson Nano, Raspberry Pi4, or Google Coral) to perform overnight training to increase the model's performance. However, the drawbacks of using a more powerful computing unit are the increase in the price, size, and power consumption of the device, along with the likely need for a cooling mechanism.

Chapter 4

Tremor and Task Classification

This chapter is adapted from "Parkinson's Tremor Types and Tasks Classification," with minor modifications to the version submitted to Engineering Applications of Artificial Intelligence, January 2023.

4.1 Abstract

Parkinson's Disease (PD) is the second most common neurodegenerative disorder. It is chronic and progressive in nature and affects millions of people worldwide. As PD tremor progresses, affected individuals become unable to control their upper limb movements normally. Wearable tremor suppression devices (WTSD) have been proposed as a potential tool to help people with PD tremors perform their activities of daily living (ADL); however, improvements to wearable devices are needed to achieve higher suppression ratios. Changes in the system dynamics caused by the tasks that the individuals with PD perform, such as ADL, and the type of active tremor, immensely affect the stability of the system. As a result, the performance of WTSDs can be improved by developing smarter algorithms to control tremors by identifying the tremor type and the task(s) being performed. This work proposes a solution that uses angular velocity data and user-independent deep learning models to detect the type of PD tremor and the tasks that individuals with PD perform. The models achieved a mean classification accuracy of 91.2% ± 0.02% and 91.1% ± 1.9%, respectively. The results demonstrate the ability to use nonlinear and stochastic signals to classify different tremor types and tasks, which can improve the robustness of wearable tremor suppression devices.

4.2 Introduction

Pathological tremor is described as involuntary rhythmic oscillations and muscle contractions that affect different parts of the body, such as the hands, head, trunk, and legs [70]. Parkinsonian tremor (PT) and essential tremor (ET) are the most often observed types of pathological tremor. The similarities in the physiological and psychological symptoms of PT and ET led different research groups to work on developing two main approaches to deal with essential and Parkinson's disease (PD) tremors. The first approach was concerned with the development of classification algorithms to differentiate between the two kinds of tremors [71–79] to help physicians with their diagnosis. The second approach focused on the development of wearable tremor suppression devices (WTSD) to help with tremor suppression [6, 10, 12, 23, 26, 80–82].

Although tremor is not life-threatening, it is responsible for the social embarrassment and functional disability of those affected [70]. PD is the second most common neurodegenerative disorder and tremor is one of its motor-related symptoms [1, 83]. To help with tremor management, WTSDs have been used as an alternative approach to medication and deep brain stimulation, allowing affected individuals to perform activities of daily living (ADL). However, despite their performance, these devices are still limited in their use. This limitation is caused by the adaptive filtering techniques employed to estimate tremors by the algorithms that are used within the WTSDs. The Fourier Linear Combiner (FLC) [30] was developed first, and it operates under the assumption that tremor is periodic and that a Fourier series can model it. Subsequently, studies showed that tremor is nonlinear, time-varying, and stochastic in nature [19]. The Weighted-Frequency Fourier Linear Combiner (WFLC) [31], the Band-limited Multiple Fourier Linear Combiner (BMFLC) [33], and the Extended-BMFLC (EBMFLC) [34] were proposed to overcome the drawbacks of the FLC. However, they still have serious shortcomings [13], where each of these algorithms has tunable parameters that need to be adjusted for each individual with tremor using the WTSD. As the tremor progresses, the parameters need to be tuned again for each individual. This is caused by the concept drift phenomenon that time-series data experience [84], for which the FLC-based estimators do not account for such occurrences and abnormalities. Concept drift describes a situation where the underlying relationship between the inputs, which are motion data that consist of voluntary motion and tremor, and outputs (tremor estimation) has changed. This type of drift is challenging to diagnose, and it can manifest itself gradually over time, where unpredictable abnormal events can occur occasionally [84]. As a result, this can cause changes in the statistical distributions of the data, which would lead to performance degradation of a WTSD. Another limitation that these algorithms have is that their parameters are tuned once, when instead, they should be dynamically adapted based on the active tremor type, and the task individuals with PD perform. These factors are not considered by the aforementioned algorithms. This drawback hampers the performance of the tremor management devices in real-life settings. The limitations and challenges that WTSD algorithms face show the need for a solution that makes tremor management devices smarter and helps them avoid performance degradation over time.

As a result, due to the challenges presented, this work focuses on introducing a novel approach to make WTSDs able to avoid any degradation in their performance better. This approach is based on deep learning [58] to classify the type of PD tremor and to detect the task that the individuals with PT are trying to perform. To our knowledge, no work has been done in this area before. The algorithms currently used in WTSDs can be enhanced by making them adaptable to the changes that they would encounter in real-life settings. Deep learning has shown its effectiveness and reliability in making neural networks learn from high-dimensional data in different areas, including healthcare, robotics, and assistive devices that use motion data as their input. To effectively develop smart algorithms for tremor suppression, this work first demonstrates that traditional machine learning algorithms and feature engineering are not well suited to tackle the problem. A novel solution based on convolutional neural networks (CNN)s is then introduced to classify the type of active PD tremor and to detect the task performed. As such, the main contributions of this work are summarized as follows:

1. Proposing the first method to deal with classifying PD tremor type and identifying the task performed to facilitate the development of smarter algorithms for WTSDs.
2. Showing that traditional machine learning algorithms and feature engineering are not well suited to solving this problem.
3. Introducing classification models that are based on CNNs that outperform the traditional machine learning methods to detect PD tremor type and identify the task being performed.

4.3 Data Recording and Processing

To develop a tremor classification and task classification models, 18 individuals with PT were recruited to have their tremors recorded while performing different tasks. Angular velocity data were recorded using Inertial Measurement Units (IMU)s at the Wearable Biomechatronics Laboratory [20], where Angular velocity represents the rate at which an object rotates or revolves around an axis with respect to time. The study was approved by the Health Sciences Research Ethics Board at Western University (#106172).

The IMUs were placed as shown in Figure 4.1, where data of the wrist joint, the thumb metacarpal bone, the index finger metacarpal bone (dorsal side), and the metacarpophalangeal (MCP) joints of the thumb and the index finger, were recorded at a sampling rate of 100 Hz.

The participants performed six tasks as described in [15] and as shown in Figure 4.2. The first two tasks were performed in order to measure the resting tremor with the palm facing up and down, respectively. In the third task, the participants had their hands held in a postural position. The last three tasks were performed in order to measure the action tremor, while the participants were moving their hands (flexion and extension of the wrist with and without pinching a pen, and

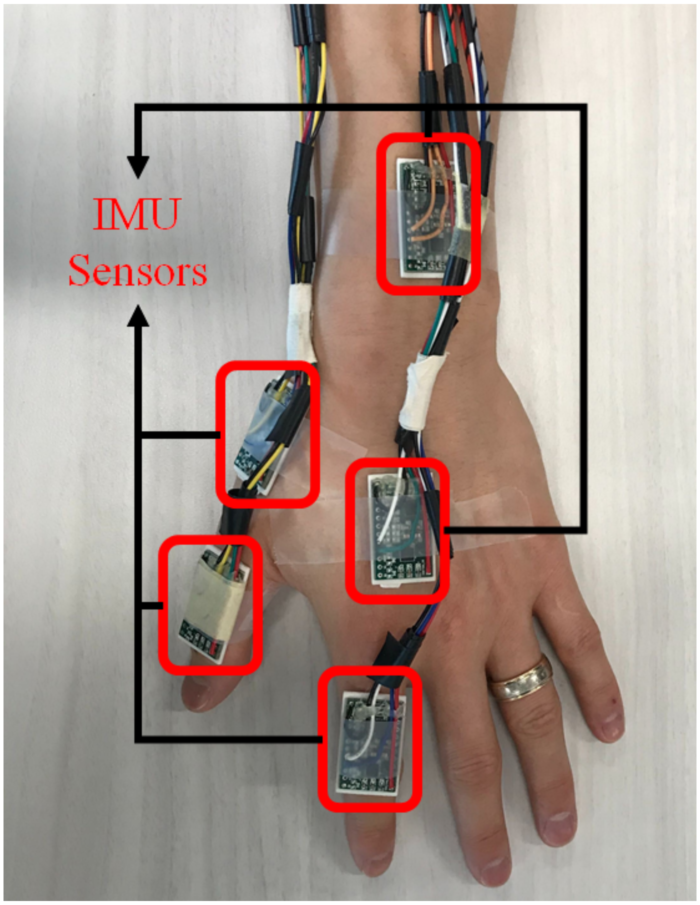


Figure 4.1: The placement of the IMU sensors on the forearm, thumb, and hand—above and below the (MCP) joints of the thumb and the index finger. The motion of each target joint was measured differentially with the two nearby IMUs.

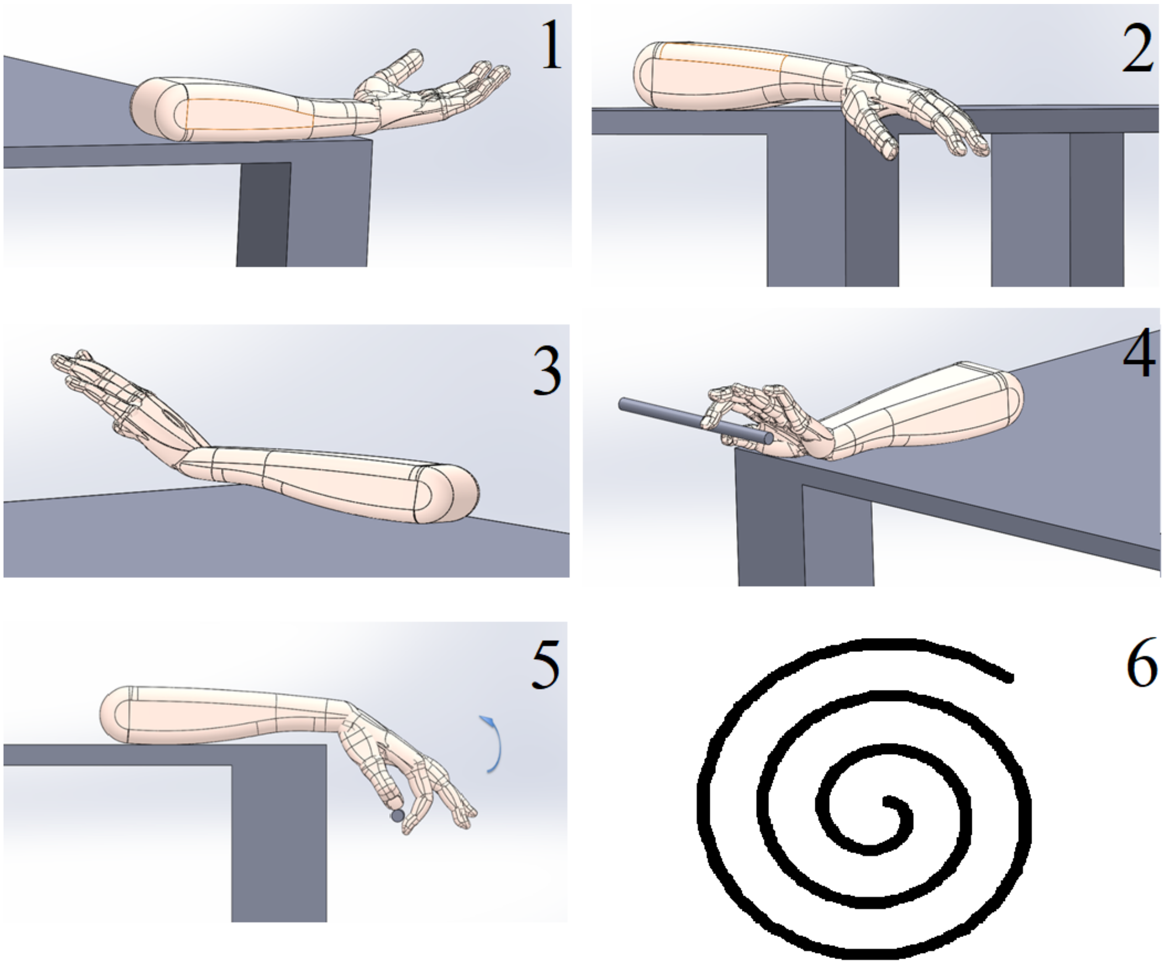


Figure 4.2: The tasks performed by the participants. In Tasks 1 and 2, resting tremor was measured when the palm was facing up and down, respectively. In Task 3, data were measured while the participant's hand was held at approximately 45 degrees against gravity. In Task 4, data were measured while the participants extended and flexed their wrist, and pinched a pencil when extending the wrist joint. In Task 5, data were measured while the participants pinched a pencil with their thumb and index finger, then extended and flexed the wrist joint. In Task 6, the participants attempted to draw a spiral by following a sketch of a spiral on a piece of paper.

drawing a spiral). The tasks performed by the participants simulate the three types of tremors: resting, postural, and action tremors. The data processing and analysis were performed offline using Python to develop the tremor classification and task classification models. The data were passed through a fourth-order Butterworth band-pass filter with a cut-off frequency between 3 and 17 Hz [15, 85] to extract tremor data. A zero-phase shifting with forward–backward filtering to remove the phase shift that took place after passing the signal through the band-pass was then applied to the filtered signals.

4.4 Task and Tremor Classification

The tasks were categorized into five classes to develop a user-independent task classification model. Class 0 consisted of the data of the first two tasks, Class 1 consisted of the data of the third task, and the last three tasks were categorized into Classes: 2, 3, and 4, respectively.

In addition, to develop a tremor classification model that is also user independent, the data were categorized into three classes, Class 0 for resting tremor, Class 1 for postural tremor, and Class 2 for action tremor.

4.4.1 Data Segmentation and Feature Extraction

Once the data were prepared for each of the classification models, each signal recording was segmented using a 250 ms sliding window with 50% overlap as recommended by Englehart *et al.* [86]. The segmentation process increased the number of observations from each signal recording to extract features from each window.

The features that were extracted from each segment were the mean absolute value (MAV), enhanced mean absolute value (EMAV), modified mean absolute value (MMAV), modified mean absolute value 2 (MMAV2), wavelength (WL), enhanced wavelength (EWL), zero crossings (ZC), slope sign change (SSC), root mean square (RMS), average amplitude change (AC), difference absolute standard deviation value (DASDV), simple square integral (SSI), the variance of signal (VARS), Willison amplitude (WA), and the maximum fractal length (MFL) [87, 88].

The MAV is the average of summation of the absolute value of a window of a signal. EMAV is an enhanced version of the MAV where the wavelet coefficient is raised to power of parameter p that is used to identify the influence of a sample window within the signal. The MMAV and MMAV2 are also modified versions of the MAV, where the feature is weighted based on some conditions [87, 88]. EWL is also a modified version of WL, where the wavelet feature is raised to power of parameter p that changes in value based on the sliding window used to calculate the wavelet coefficient [87, 88].

4.4.2 Feature Normalization and Feature Reduction

Once the features were calculated, uninformative features were removed using the variance threshold method. The variance threshold method is a simple baseline approach for feature selection.

It removes all features whose variance does not meet a certain threshold and therefore does not meaningfully contribute to the predictive capability of the model [89].

Following the step above, the remaining features were normalized using the z-score normalization method, calculated for each feature as follows:

$$x^{'} = \frac{x - \mu}{\sigma}, \tag{4.1}$$

where x is the data point in each feature column, μ is the mean of each feature, and σ is the standard deviation of each feature.

An essential tool for data analysis is the Principal Component Analysis (PCA). PCA has been widely used and applied in high dimensionality data analysis and features extraction [90]. PCA is a linear dimensionality reduction technique that seeks to map or embed data points from a high-dimensional space to a low-dimensional space, called principal components, while keeping all of the relevant linear structures intact. PCA is a statistical technique for determining critical variables in the high-dimensional dataset that explains the differences in the observations and can simplify the analysis of data without the loss of important information. It has been shown to increase the accuracy of machine learning models.

4.4.3 Cross Validation

In order to develop generalized models, a 5-fold cross-validation method was performed and repeated five times. The Stratified-k-fold method [91] was used to ensure that each fold contains a balanced class distribution. Stratified-k-fold is used to maintain the same class ratio throughout the k folds as the ratio in the original dataset, which ensures that each fold is representative of all strata of the data. As such, the classification algorithm does not get biased towards a particular class over the others.

4.4.4 Classification Models

The traditional machine learning algorithms that were used in the first part of this work are the Random Forest (RF) [49], K-Nearest Neighbour (KNN) [46], Support Vector Machine (SVM) [45], and the Multilayer Perceptron (MLP) [44] classifiers. The grid search approach was used to tune the hyperparameters of the four classifiers and they were trained using the segmented data with and without the feature reduction. It is important to note that the dataset that did not include the extracted features was also normalized using the z-score method before training the models. It is also important to note that the two datasets (with and without feature reduction) were used to investigate the effect of feature reduction on classification accuracy.

Random forest (RF): consists of a large number of individual decision trees that are the building blocks of a RF. Decision trees operate as an ensemble, where each individual tree performs a prediction, and the class with the most votes becomes the prediction of the RF model [49]. RF has shown promising results in different areas, given its ease of use and flexibility in handling both classification and regression problems. RF operates by selecting a random subset of the features of the data in a training set, such that after several random subsets of features are selected by each tree, the decision trees are trained independently, and the average or majority of the predictions yield a more accurate prediction.

K-nearest neighbours (KNN): is considered among the simplest supervised learning algorithms. It uses proximity to make classifications or predictions about the grouping of a data point. It works on the assumption that similar points can be found near one another [46]. In a classification problem, a class label is assigned on the basis of a majority vote, based on finding the majority class label of the k nearest neighbours using a distance metric (e.g., Euclidean distance, Manhattan distance, Hamming distance, etc.).

Support vector machine (SVM): is a robust classification and regression algorithm that maximizes the predictive accuracy of a model without overfitting the training data [45]. It works by mapping data to a high-dimensional feature space to categorize data points, even when the data are not otherwise linearly separable. A separator between the categories is found, and the data are then transformed such that the separator could be drawn as a hyperplane. As a result, the characteristics of new data can be used to predict the group to which a new data point should belong.

Multilayer perceptron (MLP): also known as a feed-forward neural network, an MLP is inspired by the human brain, mimicking the way that biological neurons signal to one another. An MLP consists of an input layer, one or more hidden layers, and the output layer. Each neuron connects to the neurons of the following layer and is associated with weight(s) and a bias. MLPs are known for their ability to learn nonlinear relationships and patterns in the data to perform tasks that were not achievable before [44].

As such, eight models were trained, as shown in Figure 4.3. Each of the models was tuned to find the optimal parameters. The optimal parameters for the RF were 100 trees, and the Gini index [92] was used to calculate the probability that a specific feature is classified incorrectly. The optimal maximum depth was three, the number of features to consider when looking for the best split was the square root of the maximum number of features, and the maximum leaf nodes were 100.

The KNN model was trained using the *KD-Tree* algorithm. The number of neighbours was five and the leaf size was 30.

The SVM model was trained using the radial basis function with a kernel coefficient of 0.016. The degree of the polynomial kernel was three and the gamma value was scaled based on Eq. 4.2:

$$\text{gamma} = \frac{1}{n \times \text{variance}(f_i)}, \tag{4.2}$$

where n is number of features, and f is the i^{th} feature.

The MLP network consisted of three hidden layers. The first layer contained 2048 neurons, the second layer contained 1024 neurons, and the third layer contained 512 neurons. Each of the hidden layers was followed by a batch normalization layer and a dropout layer with a dropout value of 50%. The batch normalization and the dropout layers are used to prevent the network from overfitting. The network was trained using the *Adam* optimizer [52] and the ridge regression regularization. The activation function that was used in all of the hidden layers is the Rectified Linear Unit (ReLU) [69]. The activation function of the output layer was the softmax function [93].

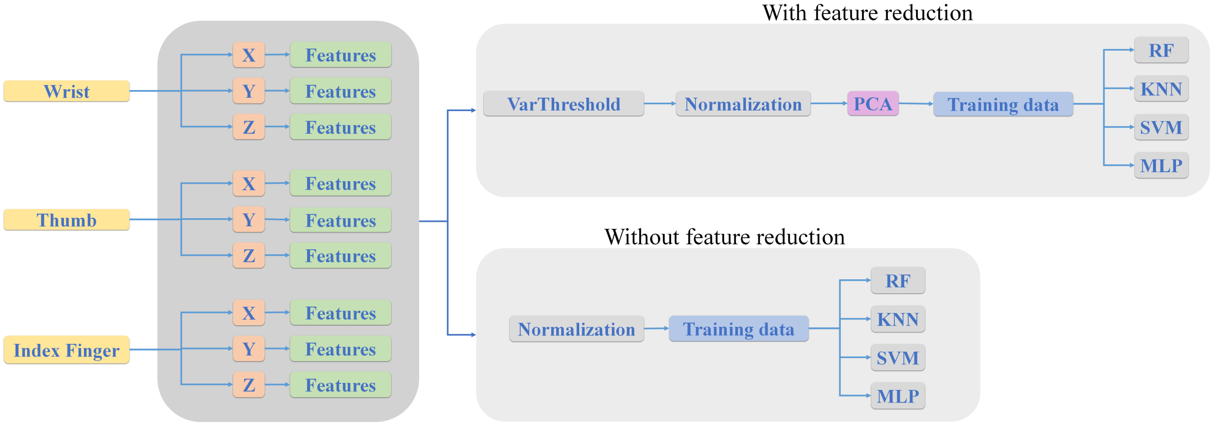


Figure 4.3: Data processing and training workflow for the classification methods for each cross-validation fold using the data from the sensors (a) with and (b) without feature reduction.

4.4.5 Results

The overall accuracies of the models with and without feature reduction are presented in Table 4.1. As can be seen, the four models performed poorly, with the RF model achieving the highest accuracy of 65% ± 0.56% and the KNN model achieving the lowest accuracy of 55% ± 0.2% when feature reduction was not applied. However, when feature reduction was applied, the accuracy of the MLP model increased from 59% ± 0.4% to 68% ± 0.12%, the RF model accuracy decreased by 1%, the SVM model accuracy increased to 60% ± 0.19% from 57% ± 0.26%, and the accuracy of the KNN model increased by 1%. The results also show that the models had low accuracies when identifying what proportion of positive identifications were actually correct (precision), and what proportion of actual positives were identified correctly (recall).

Given the poor classification accuracies of the four algorithms and based on the nonlinear nature of the data, the results showed that using a linear feature reduction approach did not yield models with high accuracies. As such, it was hypothesized that a nonlinear feature reduction approach should increase the performance of the models. The adopted nonlinear feature reduction approach is based on autoencoders, which will be discussed in the following section.

4.5 Autoencoders

Deep learning is a vast field that employs artificial neural networks to process data and train machine learning models. Auto-encoders [94] is an unsupervised learning approach where the neural network analyzes unlabelled datasets. They can learn patterns in the data and learn how to reconstruct the inputs as their outputs after significantly downsizing them. Autoencoders have three main parts, namely, the encoder, the bottleneck, and the decoder, along with the input and the reconstructed output. The encoder is a neural network that receives the input and downsizes it to a latent vector. The bottleneck is the latent vector, which is the compressed representation or the feature reduction representation of the encoded data (input). The decoder takes in the latent vector and tries to reconstruct the input as its output, as seen in Figure 4.4. As such, the goal of the autoencoder is to minimize the difference between the input and the reconstructed output thereby reducing the reconstruction loss. Autoencoders are used in many areas, such as image denoising, image compression, sequence-to-sequence prediction, data imputation, anomaly detection, and feature reduction [94, 95].

Table 4.1: The accuracies of the machine learning models trained with and without feature reduction (FR).

	Without FR				With FR		
	Class	**Precision**	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 Score**
RF	0	70%	88%	0.78	69%	89%	0.77
	1	61%	51%	0.56	61%	48%	0.53
	2	58%	20%	0.3	55%	18%	0.27
	3	57%	68%	0.62	56%	68%	0.61
	4	64%	58%	0.61	64%	57%	0.6
	Accuracy		65%			64%	
	STD		0.56%			0.28%	
	Class	**Precision**	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 Score**
MLP	0	66%	85%	0.74	77%	85%	0.81
	1	54%	35%	0.43	65%	63%	0.64
	2	63%	8%	0.15	56%	26%	0.36
	3	56%	61%	0.58	59%	67%	0.62
	4	46%	68%	0.55	63%	69%	0.66
	Accuracy		59%			68%	
	STD		0.40%			0.12%	
	Class	**Precision**	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 Score**
SVM	0	60%	90%	0.72	63%	89%	0.74
	1	49%	24%	0.32	54%	33%	0.41
	2	89%	5%	0.09	84%	5%	0.09
	3	54%	71%	0.61	53%	71%	0.61
	4	56%	48%	0.52	61%	53%	0.57
	Accuracy		57%			60%	
	STD		0.26%			0.19%	
	Class	**Precision**	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 Score**
KNN	0	61%	80%	0.69	64%	82%	0.72
	1	43%	42%	0.42	46%	42%	0.44
	2	45%	30%	0.36	42%	33%	0.37
	3	57%	52%	0.55	55%	51%	0.52
	4	55%	35%	0.43	58%	36%	0.44
	Accuracy		55%			56%	
	STD		0.20%			0.24%	

An autoencoder model was trained to reconstruct the input, which consists of the features calculated in Section 4.4.1. Once the model was trained, the decoder was discarded, and the encoder was used to generate the compressed feature vector (latent vector), that represents the reduced features (encoded features). The compressed feature vector was then used as the input to RF, KNN, SVM, and MLP models. Despite autoencoders being a robust nonlinear feature

Table 4.2: The accuracies of the four models after using the autoencoder for feature reduction.

Model	**Accuracy**	**SD**
RF	68.4%	1.12%
KNN	60.7%	2.82%
SVM	65.3%	0.78%
MLP	70.4%	1.10%

reduction approach, the performance of the four models slightly increased, as shown in Table. 4.2. The RF model accuracy increased from 64% to 68.4% ± 1.12%, the KNN model accuracy increased from 56% to 60.7% ± 2.82%, the SVM model accuracy increased from 60% to 65.3% ± 0.78%, and the MLP model accuracy increased from 68% to 70.4% ± 1.10%.

The results presented in Section 4.4.5 and in this section demonstrate that feature engineering for classifying the task performed by individuals with tremor is not a good approach and that regardless of how good the machine learning models are in different areas, without good data as an input, they cannot produce good results. As a result, the following section proposes a new solution that relies on automatic feature extraction using convolutional neural networks.

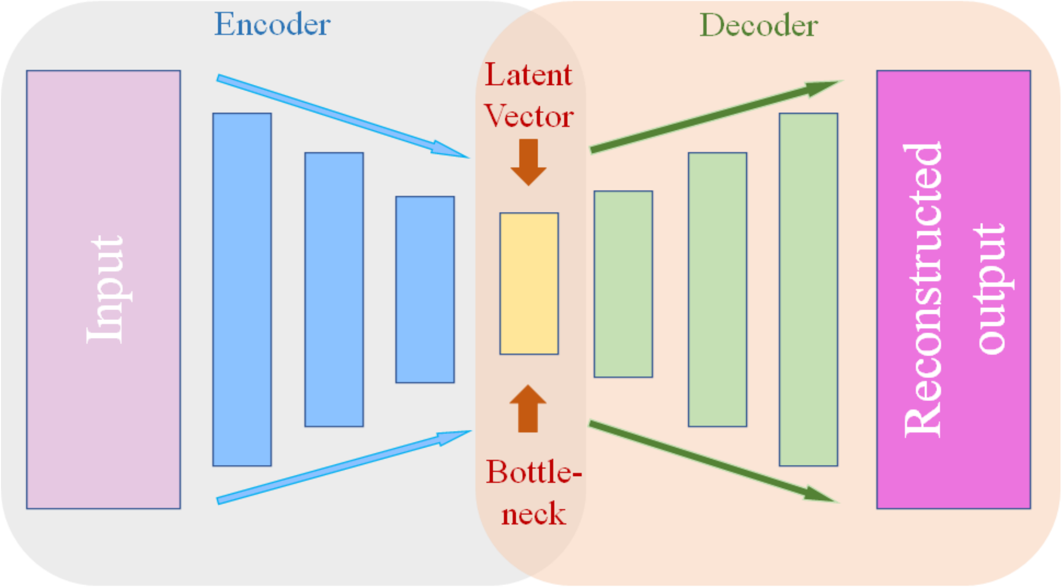


Figure 4.4: Sample visualization of the autoencoder architecture. The architecture shows the main parts of the model, the encoder, the bottleneck, and the decoder, along with the input and the reconstructed output.

4.6 Convolutional Neural Network Classification

Given the results presented in Section 4.4.5 and in Section 4.5, Convolutional Neural Networks (CNN) were investigated to achieve a higher classification accuracy. CNN is a type of neural network that has had groundbreaking results in the last few years in various fields involving pattern recognition. One main advantage of using CNNs is that they reduce the number of parameters in a neural network. This allows CNNs to solve complex problems that are not possible to solve with classical neural networks and other machine learning approaches. CNNs work under the assumption that the features of a specific problem should not be spatially dependent. For example, to detect an animal in an image, the network does not need to know where the animal in the image is. Another important aspect of a CNN is that it learns to abstract features, then uses these higher-order features as the input that propagates into deeper layers [44, 96]. CNNs do not only learn to extract features automatically but also learn what relevant information to extract from the input compared to the manual feature extraction approach, where the user specifies the features that should go into a machine learning model.

In order to work with CNN, the numeric data collected must be transformed into images. The Continuous Wavelet Transform (CWT) and the Hilbert-Huang Transform (HHT) were used for this purpose. These transformations are applied in order to extract a high-resolution representation of the time-frequency space of each signal recording, as will be discussed in the following section.

4.6.1 Continuous Wavelet Transform

The continuous wavelet transform (CWT) is a generalization of the Short-Time Fourier Transform (STFT). The STFT has a fixed resolution over the entire time–frequency space, whereas the CWT allows the analysis of nonstationary signals such as tremor signals at multiple scales. The CWT makes use of windows called wavelets to extract signal segments. The CWT wavelet gets dilated and contracted depending on the scale of activity under study, where the wavelet dilation increases the sensitivity of the CWT to long time-scale events and the contraction increases the sensitivity of the CWT to short time-scale events [97]. As a result, the CWT provides a time-scale map called the scalogram [98] that has a higher time–frequency resolution than the STFT.

The CWT wavelet is defined as a function $\psi(t) \in L^2\ (R)$ with a zero mean that is localized in both time and frequency. Dilating and translating the wavelet $\psi(t)$, a family of wavelets are produced, as follows:

$$\psi_{\upsilon,\tau}(t) = \frac{1}{\sqrt{\upsilon}}\psi(\frac{t-\tau}{\upsilon}), \tag{4.3}$$

where $\upsilon, \tau \in R$, and υ is the dilation parameter and τ is the position of the wavelet in the time domain. The CWT is then defined as the inner product of the family of the wavelets $\psi_{\upsilon,\tau}(t)$ with the signal $f(t)$, and computed as follows:

$$C_W(\upsilon,\tau) = \int_{-\infty}^{\infty} f(t)\frac{1}{\sqrt{\upsilon}}\bar{\psi}(\frac{t-\tau}{\upsilon})dt, \tag{4.4}$$

where t is the time variable, $\bar{\psi}$ is the conjugate of ψ, and $C_W(\upsilon,\tau)$ is the scalogram.

4.6.2 Hilbert-Huang Transform

The HHT, on the other hand, is used to obtain the instantaneous frequency and amplitude of each movement. The HHT shows the instantaneous energy and frequency over time, as well as the measure of the energy contribution for each frequency value [85, 99]. The high resolution of the time–frequency representation differentiates it from the STFT [100]. To provide a more physically meaningful time–frequency–energy description of a time series, Empirical Mode Decomposition (EMD) is used to generate intrinsic mode functions (IMFs) of the time series instantaneous frequency and energy [100]. The Hilbert Spectral Analysis (HSA) is then applied to the components produced by the EMD in order to produce meaningful definitions of the instantaneous frequency and energy.

The HHT of the input signal, *u(t)*, is calculated using Eq. (5.2):

$$H(u(t)) = \sum_{j=1}^{n} a_j(t)e^{i\int v_j(t)dt}, \tag{4.5}$$

where i is $\sqrt{-1}$, and $a_j(t)$ and $v_j(t)$ are the instantaneous amplitude and frequency of the j^{th} IMF obtained by the EMD. The frequency is obtained using Eq. (5.3), as follows:

$$v = \frac{d\kappa}{dt}, \tag{4.6}$$

where κ is the instantaneous phase function, calculated as follows:

$$\kappa(t) = \tan^{-1}\frac{q(t)}{u(t)}, \tag{4.7}$$

and a is obtained using Eq. (5.4), as follows:

$$a(t) = (u(t)^2 + q(t)^2)^{\frac{1}{2}}. \tag{4.8}$$

Here, *q(t)* is the Hilbert transform obtained using Eq. (4.9), as follows:

$$q(t) = \frac{1}{\pi}\int_{-\infty}^{\infty} Pc\frac{u(\tau)}{t-\tau}d\tau, \tag{4.9}$$

where Pc is the Cauchy principal value of the singular integral, and τ is the time lag of a sliding window. The HSA is obtained using Eq. (5.5), as follows:

$$h(v(t)) = \int_0^T H(u(t))dt. \tag{4.10}$$

The CWT scalograms and the HSA are used as the input for the CNN as discussed in the following section.

4.6.3 Network Architecture and Cross Validation

The optimal architecture of the CNN consisted of two hidden layers, a fully connected layer, and the output layer, as shown in Figure 4.5. The first layer contained 128 filters with a kernel size of 3 × 3, followed by a *MaxPooling* layer of size 2 × 2. The second layer contained 64 filters with a kernel size of 3 × 3, followed by a *MaxPooling* layer of size 2. The CNN layers were followed by a hidden fully connected layer that contained eight neurons. The network was trained using the *Adam* optimizer [52], and the activation function that was used in all of the hidden layers was the Rectified Linear Unit (ReLU) [69]. The activation function of the output layer was the softmax function [93]. The CNN model was also trained using the 5-fold cross-validation method repeated five times.

4.6.4 Results

The CNN model achieved an overall accuracy of 91.1% ±1.9%, as shown in Table 4.3. This is an increase of at least 20% compared to the eight classifiers trained and presented in Section 4.4.5

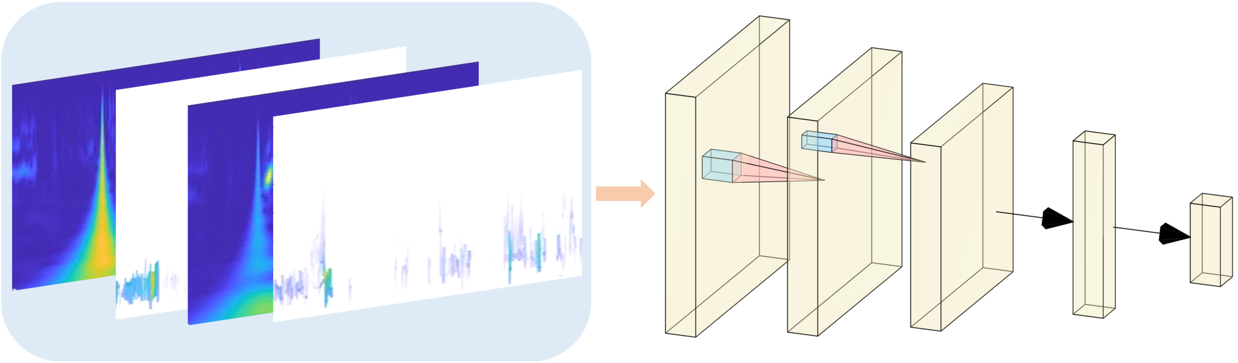


Figure 4.5: Visualization of the CNN architecture. The input of the proposed model are images of size 200 × 200 pixels representing the CWT and the HHT spectrums of the five classes. It is followed by two hidden CNN layers, one fully connected hidden layer and the output layer.

Table 4.3: The accuracy of the CNN when classifying five tasks.

Class	**Precision**	**Recall**	**F1 Score**
0	94.1%	92.8%	0.935
1	87.6%	83.6%	0.860
2	89.1%	87.4%	0.879
3	91.8%	94.6%	0.921
4	93.6%	91.5%	0.922
Accuracy		91.1%	
STD		1.9%	

and the four classifiers presented in Section 4.5.

The same architecture was then used to develop a tremor classification model as discussed in Section 4.4. The overall accuracy of the model was 91.2% ± 0.02%, as shown in Table 4.4.

Looking at Tables 4.3 and 4.4, the precision and recall accuracies are relatively balanced, except for Class 1 (postural tremor), where the ability of the model to correctly identify the proportion of actual positives was relatively low compared to the other classes. This may be attributed to the class imbalance, where the data of Class 1 (postural tremor) have fewer data points than Classes 0 (resting tremor) and 2 (action tremors), which affects the performance of the classifier.

4.7 Discussion

Given the results in Section 4.4.5, the overall accuracy of the traditional machine learning approaches was low for both datasets (with and without feature reduction). The low accuracies

Table 4.4: The accuracy of the CNN when classifying three tasks based on the three types of tremor.

Class	**Precision**	**Recall**	**F1 Score**
0	90.7%	91.5%	0.944
1	88.1%	79.6%	0.828
2	92.3%	95.2%	0.940
Accuracy		91.2%	
STD		0.02%	

might be due to two main factors. The first factor is that tremor is nonlinear and stochastic in nature; even when using a nonlinear feature reduction method, as shown in Section 4.5, accuracies barely improved. The results from Sections 4.4 and 4.5 show that the traditional machine learning models are susceptible to noisy data (tremor data). The second factor is that feature engineering is not the right approach to developing models for tremor and task classification. As a result, a new approach was proposed. The new approach was based on transforming the time-series data into high-resolution images using the CWT and HHT, and then using these as input to the CNN. The results showed that the machine learning model could perform well with the proper input, as shown in Section 4.6, where CNNs showed again that they are robust for task classification and for identifying tremor types that individuals with PD experience, regardless of how stochastic and nonlinear the tremor signals are. The results achieved by the CNN models were relatively high (91%), demonstrating an increase of more than 20%, compared to the accuracies achieved by the models in Sections 4.4 and 4.5.

4.8 Conclusion

This work proposes novel user-independent tremor and task classification models that are based on deep learning. The results showed that parkinsonian tremor could be classified into three different tremor types (resting, postural, and action tremors), and further showed that the tasks performed by individuals could also be classified with high accuracy when tremor is active. The information provided in this work is valuable for improving the performance of active wearable tremor suppression devices. By adapting them to consider both the tremor type and the task that is being performed, they can achieve higher percentages of tremor suppression.

This work also showed that manual feature engineering is not always successful despite having knowledge about the data. Furthermore, this work showed that even robust feature reduction methods did not help to increase the accuracy of the models. This has been confirmed by the results presented in Sections 4.4 and 4.5. In contrast, automatic feature extraction based on high-resolution images that are generated using the CWT and HHT is well suited for this problem.

Future work will focus on collecting more data from individuals with tremor when performing activities of daily living so that the models can be trained on data arising from real-life situations.

Chapter 5

Tremor and Voluntary Motion Generative Model

This chapter is adapted from ”PT-Net: Pathological Tremor and Voluntary Motion Generative Adversarial Network”, will be submitted to Elsevier - Engineering Applications of Artificial Intelligence.

5.1 Abstract

Pathological tremor is nonlinear and stochastic in nature. A great deal of work has been done to study and analyze tremor data in order to develop tremor estimators and smart algorithms to help with tremor suppression. Most solutions require large amounts of data that are difficult to collect in the quantities required for training. Existing methods to generate synthetic data fail to preserve the temporal dynamics, characteristics, patterns, and distribution of tremor across time, and do not adequately attend to the temporal correlations that are unique to tremor time series data. This work introduces a framework that is based on a one-dimensional convolutional generative adversarial network, named pathological tremor network (PT-Net), for generating realistic tremor data. The new approach combines the flexibility of the unsupervised learning paradigm with the control afforded by supervised learning. Qualitative and quantitative results show that the proposed framework consistently produces realistic and reliable synthetic tremor data that can be used to develop algorithms to help suppress tremor.

5.2 Introduction

Pathological tremor is an involuntary movement disorder that is characterized by rhythmic oscillatory movements that affect different parts of the body [21]. Pathological tremor is mainly classified into essential tremor (ET) and parkinsonian tremor (PT) [17, 18, 21, 29]. Within each type of pathological tremors, they can be further classified into resting tremors and action tremors [21, 29] based on when they occur. Resting tremors are visible when a body part is at rest, while action tremors manifest when a body part is moved voluntarily or kept at a certain posture [17, 21, 101, 102]. ET and PT affect more than fifty million people worldwide [101, 102], impacting people's mobility and their ability to perform activities of daily living (ADLs) [20–23].

Many studies have been conducted to study and analyze pathological tremor [13, 20–23, 59] in order to develop tremor estimators and smart algorithms to help with tremor suppression. However, most solutions are heavily dependent on machine learning algorithms that require large amounts of data for training. Nevertheless, tremor datasets are scarce and are not easily accessible, in the sense that data, when collected, are not always enough, and at times they are difficult to collect. This is due to many reasons, such as not being able to recruit participants with tremor for data collection, or for reasons that are out of researchers' control, such as the spread of COVID-19, which prevented many research groups from going forward with their recruitment and data collection plans. Another reason that data are not easily accessible to many is due to security and confidentiality concerns that could intrude on participants' privacy.

To overcome the limitations presented, synthesized tremor data have been used alongside real tremor datasets [38, 103] to evaluate the performance of the tremor estimators and the algorithms developed. Synthesized data consist of sine and cosine waves that are periodic, with noise added to the signals to mimic tremor. However, studies have shown that tremor is stochastic and nonlinear in nature [19], and working with tremor time series data under the assumption that they are periodic and can be modelled as a series of sine and cosine components hindered the accuracy of the tremor estimators developed to help with tremor suppression [13, 59]. Another method that is commonly used to create more samples is the synthetic minority oversampling technique (SMOTE) [104]. SMOTE is used to tackle the problem of class imbalance in a dataset for a classification problem. SMOTE works by selecting a minority class instance at random and then finding its k nearest minority class neighbours. A random synthetic instance is then created by randomly choosing one of the k nearest neighbours of the minority class instance selected, then connecting the instance selected and the instance created to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances [105, 106]. However, this method creates single instances of data and not sequences, and it is unable to attend to the temporal correlations and characteristics that are unique to time series data. Thus, the synthesized data do not integrate and represent tremor patterns, characteristics, and distribution that are caused by human pathology. Moreover, the synthesized data do not mimic and emulate real tremor, which manifests in different intensities, severity, and stochasticity when the person with tremor moves voluntarily. As a result, it is quite challenging to create a generic mathematical model that can effectively produce synthesized tremor signals that are similar to the real ones.

Due to the challenges related to using real data and the limitations of using the traditional methods to generate synthesized data, this work introduces a novel approach to generate realistic data, that is based on deep neural networks [58]. Deep learning has shown its robustness and ability to make neural networks learn from high dimensional data in different areas, including computer vision, anomaly detection, healthcare, robotics and assistive devices that use motion data as their input. In order to effectively develop smart algorithms for tremor suppression, this work introduces a novel framework called pathological tremor network (PT-Net) and implements the network architecture that is based on one-dimensional convolutional neural networks (1D-CNNs) and generative adversarial networks (GANs). The GAN model architecture will be mainly based on the 1D-CNNs, where the 1D-CNNs robustness for learning the temporal behaviour and characteristics of voluntary motion and tremor time-series data was shown in previous studies by the authors [13, 59]. The results of PT-Net were evaluated using four different methods, which will be discussed in greater detail in Section 5.5.4, to verify and validate that the data generated by the proposed model are usable, reliable, and mimic the real data collected.

As such, the main contributions of this work are summarized as follows:

1. A framework that is based on one-dimensional convolutional neural networks (1D-CNN) and generative adversarial networks (GANs) that leverages the advantages of the supervised and unsupervised learning paradigms of machine learning.

2. A 1D-CNN GAN model “PT-Net” that generates realistic motion that reflects a combination of voluntary motion and parkinsonian tremor.

3. An open access Python library that can be installed by any research group or team around the world to generate and use realistic synthetic parkinsonian tremor data.

4. A framework that can be trained and used with any type of biosignals, without being limited to tremors, e.g., electromyography (EMG) data, electroencephalogram (EEG) data, and electrocardiogram (ECG) data.

5.3 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [107] are an interesting innovation in machine learning. GAN modelling is an unsupervised deep learning task where the neural network automatically discovers and learns the patterns, regularities, and structure present in the data. As a result, given a training dataset, GANs are capable of generating entirely new data with similar characteristics and distribution to that of the real data [108, 109].

Unlike traditional and vanilla neural networks, a GAN is inspired by the two-player zero-sum game, where the total gains and losses of the two players are zero, and each player's gain or loss of utility is balanced by the loss or gain of the utility of another player [107]. A GAN consists of two neural networks as shown in Figure 5.1, a generator and a discriminator. The generator competes against its adversary, the discriminator, by trying to capture the distribution of the real samples and generating new data with similar characteristics. The discriminator on the other hand, is a classifier that learns to determine whether the sample it receives is real or fake, or in other words, it tries to determine if the sample data it receives are from the model (generator) distribution or the real data distribution. As a result, the competition between the generator and the discriminator drives both of them to improve until the data generated by the generator are indistinguishable from the real data.

The generator takes in a random input, also called noise or latent vector, and produces fake data that will be referred to as synthetic data throughout this paper. The discriminator then receives two inputs, the synthetic data that are produced by the generator and the real data, and outputs probabilities that indicate whether the samples are real or not. The discriminator D and the generator G play a min–max game with the value function $V(G, D)$:

$$\min_{\mathrm{G}} \max_{\mathrm{D}} V(D, G) = \mathbb{E}_{x \sim p_{\mathrm{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))], \tag{5.1}$$

where z is the input noise received by G to produce the synthetic data. D is trained to learn the

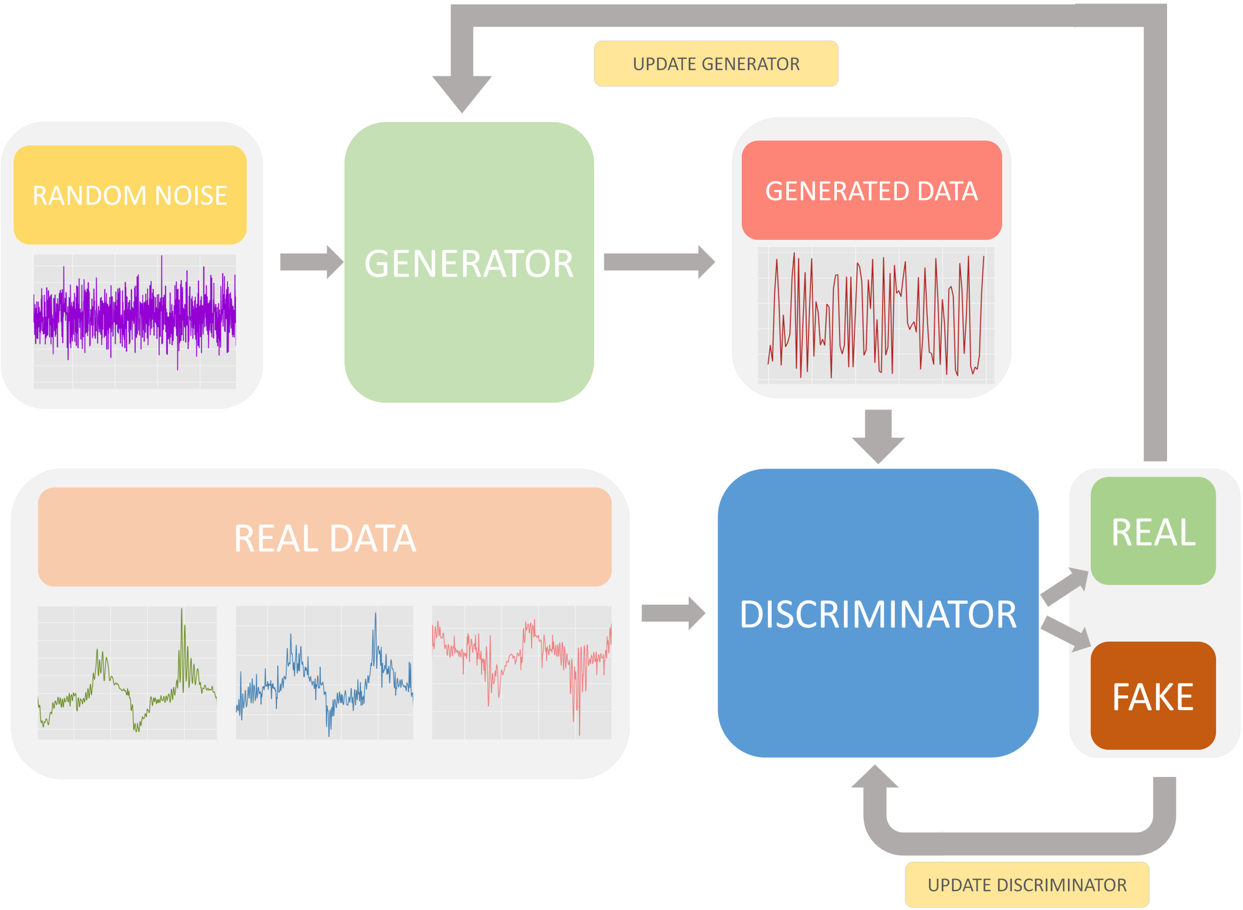


Figure 5.1: Structure of a Generative Adversarial Network. The generator takes in a vector of randomly generated numbers, and is trained to produce real-like data. The discriminator is a classifier that takes in real data samples and samples produced by the generator, where it learns to differentiate which ones are real and which ones are fake. The training stops when the generator is able to output data that have similar characteristics to the real data, and the discriminator is not able to tell whether the data from the generator are fake or not.

probability distribution p and maximize the probability of assigning the correct label to the real samples x received from the training dataset and the generated samples received from G. G is trained to minimize $\log(1 - D(G(z)))$ in order to produce realistic data.

Thus far, the most successful applications of GANs have been in computer vision areas such as video generation, image synthesis, image super-resolution, image translation, video completion, colouring black and white images [110–114]. GANs have also been successful in applications related to natural language processing, security, medicine, and health care [115–118].

5.4 One-Dimensional Convolutional Neural Networks

Recently, one-dimensional convolutional neural networks (1D-CNN) have been shown to produce remarkable results on challenging activity recognition tasks, as well on time-series estimation and prediction tasks [13, 44, 59]. A CNN is a type of neural network architecture that takes advantage of the concept of convolutions to learn higher-order features [13, 44, 59, 65]. CNNs are mainly used in computer vision problems such as object detection and object recognition, image classification, and image segmentation. They are also used for tasks such as sound and text recognition [66–68]. Automatic feature extraction is one of the main processes that differentiate CNNs from the classical machine learning algorithms, in the sense that automatic feature extraction allows them to learn an internal representation of n-dimensional data without the need for feature engineering that relies on human expertise and knowledge about the data. The same process is exploited by the authors [13, 59] for one-dimensional sequences of motion data.

As such, and based on the fact that PT is nonlinear, nonstationary, and stochastic in nature, 1D-CNNs were considered as the backbone of the GAN network, which will be discussed in greater detail in Section 5.5.3.

5.5 Methods

To train the 1D-CNN-GAN model, the data collected need to be processed, curated, and adjusted in order to accommodate the 1D-CNN architecture. In this section, data collection, data processing and preparation, implementation, and evaluation methods are presented.

5.5.1 Data Collection

Data from 18 subjects with PD were used in this study. The data were recorded at the Wearable Biomechatronics Laboratory at Western University [20]. The participants were recruited by a movement disorders neurologist, and the study was approved by the Health Sciences Research Ethics Board (#106172). The PD group included eleven males and seven females with varying tremor severity.

Inertial measurement units (IMUs) were used to record the data, and they were placed as shown in [13]. Data of the wrist joint, the thumb metacarpal bone, the index finger metacarpal bone (dorsal side), and the metacarpophalangeal (MCP) joints of the thumb and the index finger, were recorded at a sampling rate of 100 Hz.

The data recorded consisted of six different tasks, as follows:

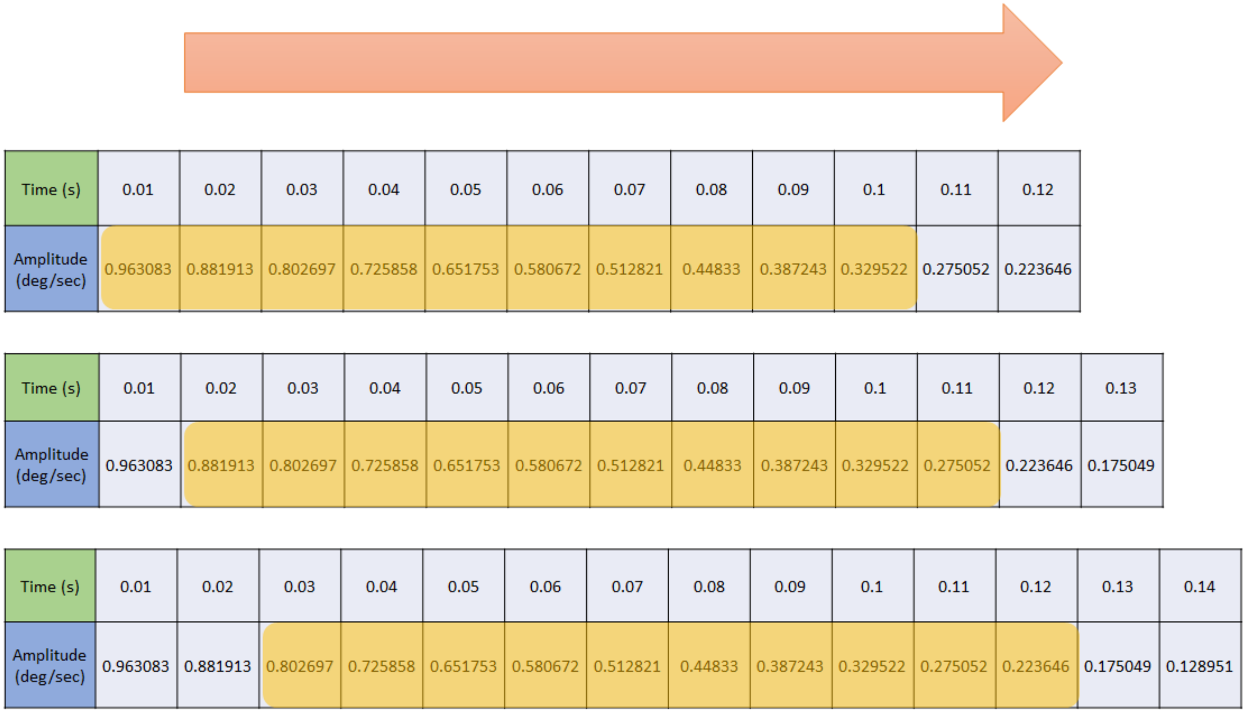
- In Tasks 1 and 2, the participant's hand was in a resting position with the palm facing down and up, respectively.
- In Task 3, the participant's hand was held in a postural position at approximately 45 degrees against gravity.
- In Task 4, the participants were asked to extend and flex their wrist, and pinch a pencil when extending the wrist joint.
- In Task 5, the participants were asked to move a pencil with thumb and index finger when extending and flexing the wrist joint.
- In Task 6, the participants were asked to draw a spiral.

Studies have shown that the frequencies of voluntary motion range from 0 to 2 Hz [20, 23], and the frequencies of parkinsonian tremor range from 3 to 17 Hz [20, 23]. Since the goal is to generate synthetic data that contains a combination of tremor and voluntary motion, the raw data of Tasks 4, 5, and 6 were used, while the data from Tasks 1, 2, and 3 were discarded in this study.

5.5.2 Data Preparation

Unlike the data processing procedure that is followed in [13, 59], the raw recorded motion was used in this work. This is due to the fact that the objective of this work is to generate data that mimic and capture the characteristics, distribution, and patterns of the motion that contains both voluntary and tremor time-series motion. Moreover, the authors showed in [13, 59] that, although feature engineering is a recommended step to train machine learning models, it does not always yield good results for PT data. This work demonstrated that the automatic feature extraction done by the CNNs based models was able to learn the nonlinear patterns and structure in the data with promising results.

The raw data collected were sliced using the sliding window of size m approach as shown in Figure 5.2. The sliding window approach works by taking a sub-array of size m and moving the window within the larger array creating the effect of a sliding window. Three different sizes of m were used in this work, $m = 2, 5,$ and 10 seconds, respectively. Larger sizes of m were used,



Time (s)	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1	0.11	0.12
Amplitude (deg/sec)	0.963083	0.881913	0.802697	0.725858	0.651753	0.580672	0.512821	0.44833	0.387243	0.329522	0.275052	0.223646

Time (s)	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1	0.11	0.12	0.13
Amplitude (deg/sec)	0.963083	0.881913	0.802697	0.725858	0.651753	0.580672	0.512821	0.44833	0.387243	0.329522	0.275052	0.223646	0.175049

Time (s)	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1	0.11	0.12	0.13	0.14
Amplitude (deg/sec)	0.963083	0.881913	0.802697	0.725858	0.651753	0.580672	0.512821	0.44833	0.387243	0.329522	0.275052	0.223646	0.175049	0.128951

Figure 5.2: Visualization of an example of slicing the data using the sliding window approach. Starting at index 0, take a sub-array of size m and move the window within the larger array, which creates the effect of a sliding window.

however, the data generated by the model were not realistic enough when they were validated using the methods described in Section 5.5.4. As a result, three different datasets were created based on $m = 2,\ 5,$ and 10 seconds, respectively. Each dataset contained more than two million window samples. The three sizes of m were used to train three different 1D-CNN-GAN models that are capable of generating samples of different sizes.

5.5.3 Implementation

Data processing and the implementation of the PT-Net were performed using Python 3.7 and TensorFlow 2.4. PT-Net was trained on an Intel® Core$^{\text{TM}}$ i7-9700 CPU @ 4.7 GHz PC, using an NVIDIA GeForce RTX 2060 to speed up the training process, as it is computationally expensive.

In order for the generator to be able to learn to map the internal features learned, based on the feedback from the discriminator, the automatic feature extraction process in each of the convolutional layers [59] is performed as follows:

$$x_i^l = b_i^l + \sum_{j=1}^{N_{l-1}} \text{conv1D}(w_{ji}^{l-1}, o_j^{l-1}), \tag{5.2}$$

where N is the number of variables to be predicted, x_i^l is the input, b_i^l is the bias of the i^{th} neuron at layer l, o_j^{l-1} is the output of the j^{th} neuron at layer $l-1$, w_{ji}^{l-1} is the kernel weight from the j^{th} neuron at layer $l-1$ to the i^{th} neuron at layer l. conv1D(...) is used to perform a one-dimensional convolution, and as a result, the dimension of the input array, x_i^l, is less than the dimension of the output arrays, o_j^{l-1}. The output, y_i^l, is then obtained by passing the input through the activation function f:

$$y_i^l = f(x_i^l) \tag{5.3}$$

For the generator to learn to produce realistic time-series data, it has to adjust its weights and biases. This process is done by backpropagation. Based on the feedback from the discriminator, the generator back propagates the error starting from the output layer.

The derivative of the accuracy metric or the loss function with respect to the weights and biases should be calculated to minimize the contribution of the network parameters to the error E. The delta error, Δ_i^l, of the i^{th} neuron at layer l is calculated to update the bias of that neuron and all of the weights of the neurons in the preceding layer that are connected to that neuron, as follows:

$$\frac{\partial E}{\partial w_{ji}^{l-1}} = \Delta_i^l y_j^{l-1} \quad \text{and} \quad \frac{\partial E}{\partial b_i^l} = \Delta_i^l. \tag{5.4}$$

After the first back propagation is performed from layer $l+1$ to layer l, the backpropagation is then carried to the input delta of the convolutional layer l, Δ_i^l, and the zero-order up-sampled map is $uo_i^l = \mathrm{up}(o_i^l)$, and the delta error is then calculated as follows:

$$\Delta_i^l = \frac{\partial E}{\partial y_i^l}\frac{\partial y_i^l}{\partial x_i^l} = \frac{\partial E}{\partial uo_i^l}\frac{\partial uo_i^l}{\partial y_i^l} f'(x_i^l) = \mathrm{up}(\Delta o_i^l) f'(x_i^l), \tag{5.5}$$

where the back propagation of the delta error ($\Delta o_i^l \leftarrow \sum \Delta_j^{l+1}$) is obtained as follows:

$$\Delta o_i^l = \sum_{j=1}^{N_{l+1}} \mathrm{conv1Dz}(\Delta_j^{l+1}, \mathrm{rev}(w_{ji}^l)), \tag{5.6}$$

where conv1Dz(...) is used to perform full one dimensional convolution with zero-padding, and the rev(...) is used to reverse the array.

The generator consisted of a dense layer and three 1D-CNN layers. The output of the dense layer was reshaped to properly accommodate the three 1D-CNN layers that followed it. The Leaky

ReLU [119] activation function was used for each of the hidden layers. Each of the 1D-CNN layers were followed by upsampling functions in order to produce the proper output dimension from the output layer. The input of each of the generators that produced data of lengths 2, 5, and 10 seconds was the latent vector that consisted of the random noise data of sizes 150, 1000, and 2000, respectively. The size of the latent vector is a hyperparameter that was tuned during the training process. The chosen latent vector sizes resulted in PT-Net producing high resolution time-series data that are similar to the real data, as will be shown in the following sections. The output of the generator is time-series data of size m.

The discriminator consisted of three 1D-CNN layers. The Leaky ReLU [119] activation function was also used for each of the hidden layers. The third convolutional layer is followed by a dense layer with a dropout layer that had a dropout rate of 0.4. The dropout rate is a tunable hyperparameter that helps a neural network to generalize instead of memorizing the data. The inputs to the discriminator were samples of real data and generated data of the same size m, and the output was a binary output which is either 1 for real, or 0 for fake.

5.5.4 Evaluation

The objective of this study was to generate synthetic data that mimic human motion combined with tremor to be used alongside collected data, to increase the data points of a small dataset that then can be used to study tremor, or to be used by research groups that are unable to collect data. Therefore, the synthetic data must be similar to real data in terms of characteristics, distribution, and power density spectrum. As such, the PT-Net is evaluated using four different approaches, as follows:

5.5.4.1 Train on Real Data, Test on Synthetic Data (TRTS)

An estimation model is trained with real data and tested on synthetic data. The estimation model used is based on the model proposed by the authors in [13], where the reported accuracy was 99%. TRTS serves as an evaluation of the ability of the PT-Net to generate realistic looking data, since the goal is to generate data that can be used for various purposes, as mentioned above.

5.5.4.2 Train on Synthetic Data, Test on Real Data (TSTR)

An estimation model is trained using synthetic data, then tested on real data. TSTR evaluates the ability of the PT-Net to generate data that can be reliably used alone or with real collected data for tremor and voluntary motion estimation and tremor suppression.

The estimation accuracy (EA) of the models trained in TRTS and TSTR was computed as follows:

$$\left(1 - \frac{\mid \text{True Value} - \text{Predicted Value} \mid}{\mid \text{True Value} \mid}\right) \times 100. \tag{5.7}$$

5.5.4.3 Power Spectrum Density (PSD)

Zhou *et al.*, reported in their studies [20, 23] that the frequency characteristics of parkinsonian hand tremor ranged between 3 and 18 Hz and consists of multiple harmonics, with the dominant harmonic lying between 4 and 7 Hz, the second harmonic lying between 7 and 12 Hz and third harmonic lying between 12 and 17 Hz. Other harmonics were also reported, however their power decreased as the frequency range increased. As a result, calculating the PSD [27] of the synthetic data evaluates the ability of the PT-Net to also generate realistic and reliable data.

5.5.4.4 T-Distribution Stochastic Neighbouring Embedding (t-SNE)

T-SNE is a technique that is well suited for visualizing high-dimensional data by giving each data point, in this case a window of size m, a location in a two or three dimensional map [120]. T-SNE is a powerful, important, and a nonlinear method that creates a single map that reveals structure and patterns in data of different but related classes (synthetic and real), and groups the data points in different cluster(s) based on their similarity. In other words, t-SNE is a nonlinear dimensionality reduction method, that uses the concept of similarity of data points to identify observed clusters by finding patterns with multiple features in the data. This method evaluates and compares synthetic and real data and visualizes them in a 2D or 3D graph that shows data points of clusters. If the generated data have similar patterns, structure, distribution and characteristics to real data, then the output should be one cluster.

5.6 Results

In this section, the generated synthetic data of the PT-Net were evaluated using the four different methods discussed in the previous section. The results are as follows:

5.6.1 Train on Real, Test on Synthetic Data (TRTS)

The estimation model proposed by the authors in [13] was trained on all of the real action tremor data of the 18 participants. Figs. 5.3, and 5.4 show four samples of data generated by PT-Net for window sizes of 5 and 10 seconds. Images of data generated in 2 seconds windows are not shown since 5 and 10 seconds windows show a better visualization of the raw motion over a more extended period of time, where the raw motion (red) consists of voluntary motion and tremor. To compare the ground truth voluntary motion (blue) to the estimated voluntary motion (black), a low-pass filter with cutoff frequency of 2 Hz [13, 20] was applied to the generated motion data (red). The estimated voluntary motion is the output of the estimation model when given the raw data (red) as an input, compared to the truth voluntary motion that is extracted using the low-pass filter. The three PT-Net models generated 2000 samples each, each of them is trained based on the three window sizes, and Table 5.1 presents the estimation accuracy (EA) of the estimation model on the outputs of each PT-Net model. The number 2000 is a relatively large random number that

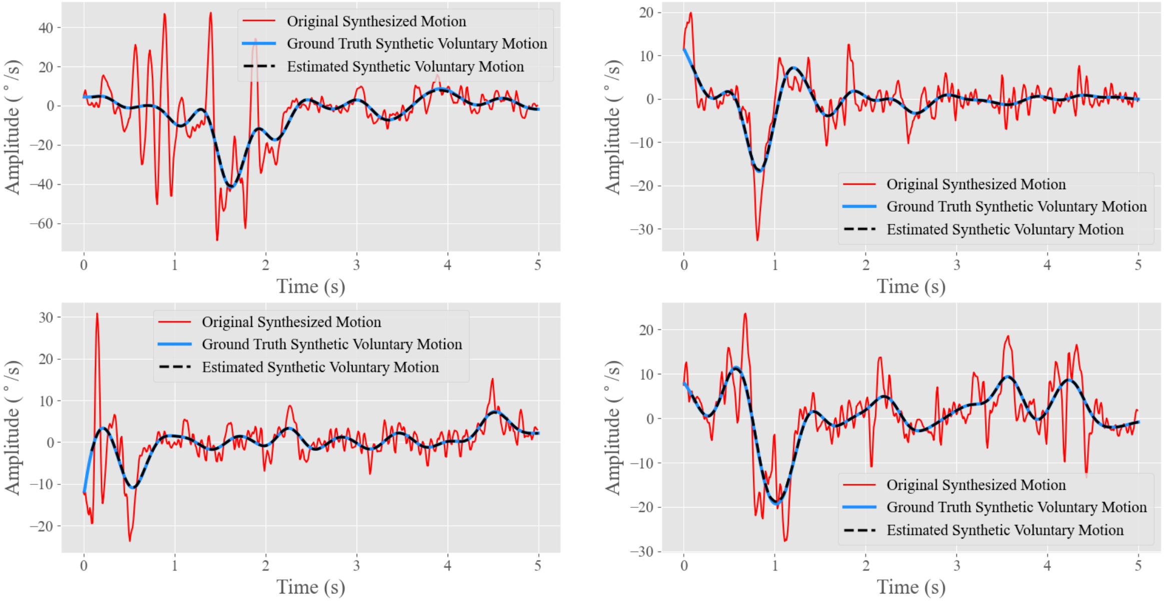


Figure 5.3: Visualization of four sample outputs of the PT-Net model of 5 seconds, and the ground truth and estimated voluntary motion — TRTS.

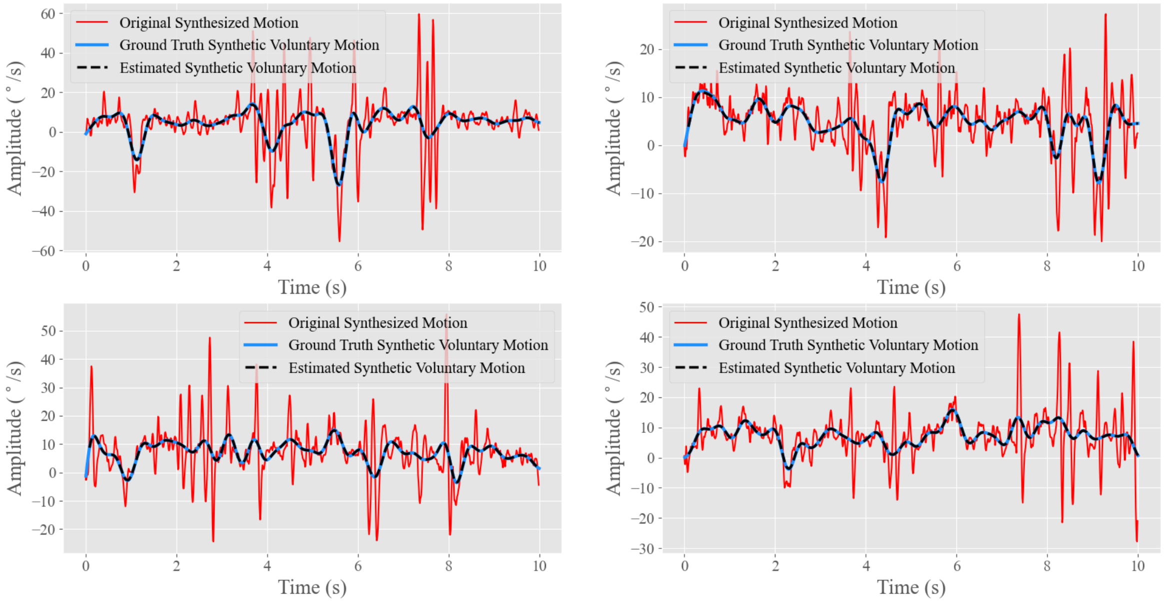


Figure 5.4: Visualization of four sample outputs of the PT-Net model of 10 seconds, with the ground truth and estimated voluntary motion — TRTS.

was chosen to generate enough samples for each dataset. The estimation model had an average EA of 99.2%, 99%, and 99% for the three window sizes, respectively.

Table 5.1: The estimation accuracy of the model for estimating voluntary motion for TRTS for the three window sizes.

Model	**Voluntary Motion - EA**	**SD ±**
PT-Net 2	99.2%	0.18%
PT-Net 5	99%	0.23%
PT-Net 10	99%	0.21%

5.6.2 Train on Synthetic, Test on Real Data (TSTR)

Three datasets were created using the PT-Net model. Each dataset contained 100,000 data samples of 2, 5, and 10 seconds, respectively. A second estimation model that is also based on [13] was then trained on each of the datasets separately, and was then tested on the real data of the 18 participants. The average accuracy of the estimation model when tested on real data was 97.6% ± 0.35%. Figure 5.5 shows four randomly selected samples from the real data, with their respective ground truth voluntary motion and the estimated voluntary motion. It is important to note here that the testing dataset contained the entire length of the recorded motions and was not sliced to

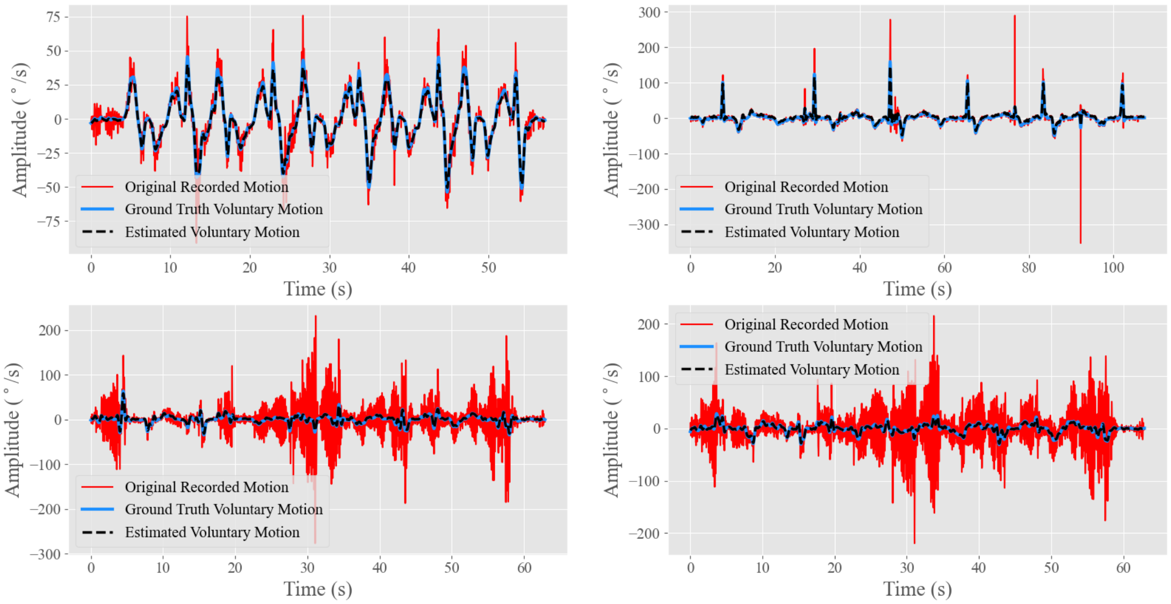


Figure 5.5: Visualization of four random real data samples selected from the real dataset, with the ground truth and estimated voluntary motion — TSTR.

different window sizes. The reasoning behind this was to evaluate the performance of the model on unprocessed real data.

5.6.3 Power Spectrum Density (PSD)

As the third evaluation method of PT-Net, the PSD was calculated for the samples in each of the testing datasets created for TRTS. As discussed in Section 5.5.4, the dominant harmonic reported for the real data ranged between 3 and 7 Hz, and the mean power proportion of the 2^{nd} and 3^{rd} harmonics to the 1^{st} harmonic is 60.9% and 38.9% [20], respectively, as shown in Table. 5.2. Applying the same method to the generated samples, the frequency of the dominant harmonic ranged between 4 and 9 Hz, the frequency of the 2^{nd} harmonic ranged between 10 and 13 Hz, and the frequency of the 3^{rd} harmonic ranged between 13 and 19 Hz. Moreover, the mean power proportion of the 2^{nd} and 3^{rd} harmonics to the 1^{st} harmonic of the generated samples were 67.4% and 42.3%, respectively. It is important to note that there were some outliers in the generated samples, as one would expect, where the dominant harmonic for some of the samples ranged between 17 and 25 Hz. However, in practice, one can check the frequencies of the harmonics and their ratios. If they do not fall within the range of reported values of the real parkinsonian tremor data, the generated sample(s) can be discarded and new ones can be generated instantly on the fly.

Table 5.2: Comparison of the mean power proportion of 2nd and 3rd harmonics of tremor to the 1st harmonic of tremor in real and synthetic data.

Tremor type	**2nd to 1st harmonic**	**3rd to 1st harmonic**
Real	60.9%	38.9%
Synthetic	67.4%	42.3%

5.6.4 T-Distribution Stochastic Neighbouring Embedding (t-SNE)

t-SNE was used as the final evaluation method to validate the PT-Net model. t-SNE is a popular method for exploring high-dimensional data that is able to create two or three dimensional maps from nonlinear data with hundreds of dimensions. The goal of using t-SNE is to take a set of points in a high-dimensional space and find their representation in a lower dimension, typically, a 2D or a 3D plot. The t-SNE algorithm is nonlinear and adapts to the underlying nonlinearities and patterns in a dataset, where it performs different transformations on different regions of the data to find the clusters in the dataset. t-SNE has several tunable parameters, with perplexity being the main one. Perplexity is related to the number of nearest neighbours that are used for each data point, and acts as a way to balance the attention between the local and global aspects of the data. It serves to find how many neighbours each data point has. Other key hyperparameters to consider are the number of iterations and the learning rate.

Different perplexity values were used to evaluate the similarity of the real and synthetic data. The perplexity values ranged between 5 and 60, with three components to generate a 3D plot. The maximum number of iterations was set to 5000, and the algorithm converged at iteration 1000. Regardless of the perplexity value used, the results were similar.

Figure 5.6 shows the results of visualizing the similarities between the real and synthetic data. As can be seen, both the real and synthetic data were clustered together, which strongly indicates a high correlation between both datasets, and that the synthetic data share similar patterns, distribution, and characteristics with the real data.

5.7 Discussion

In this section, the findings and results presented in Section 5.6 are discussed in further detail, and conclusions are drawn based on these findings.

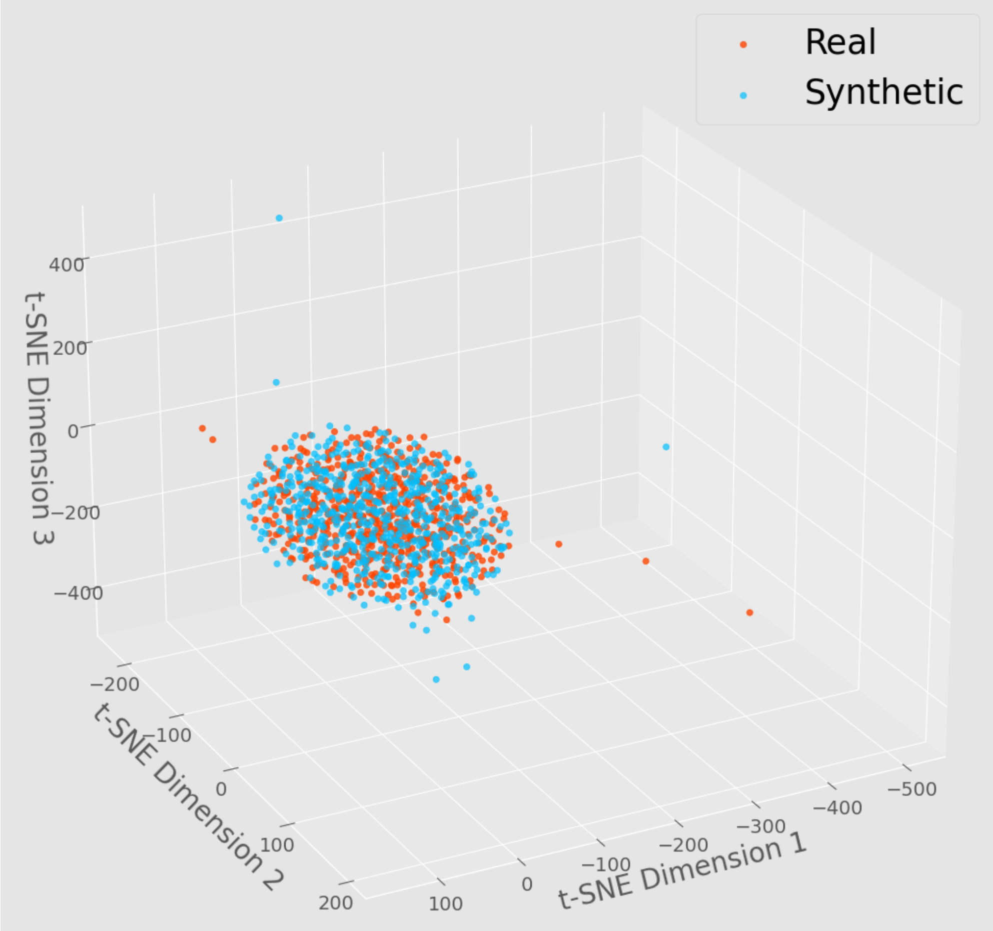


Figure 5.6: Visualization of the cluster(s) created by t-SNE of the real and synthetic data.

The high accuracies show that the estimation models were able to generalize from real data to generated samples and from generated samples to real data, and that the generated data are similar to the real ones. However, although both accuracies are high, the results show that the accuracy of TRTS is higher than the accuracy of TSTR, which indicates that the estimation model trained on synthetic data was not able to capture all of the patterns and characteristics of the real data. As a result there is room for improvement for the generative model. This is further supported by the results shown when PT-Net was evaluated using the PSD method. The PSD results showed that the frequencies of the samples generated by PT-Net fall within the range of frequencies of the real data with slight differences, where real data frequencies ranged between 3 and 17 Hz and the frequencies of the generated samples ranged between 4 and 19 Hz. Since the output of PT-Net, which is time-series data, cannot be assessed visually, unlike computer vision GANs, the t-SNE is a powerful nonlinear dimensionality reduction algorithm that is well suited for visualizing high-dimensional datasets. As can be seen in Figure 5.6, both data types are inseparable, which indicates that the generated time-series data are from the same distribution as the real time-series data.

PT-Net (with its three different variations and their weights, where each variation generates data of different window size) can be downloaded from `https://github.com/ibanas/PT-Net` and used with a public Python module `https://pypi.org/project/TremorGan` to generate realistic samples.

5.8 Conclusion

This paper presented the framework and implementation of a parkinsonian and voluntary motion generative adversarial network called PT-Net that is based on 1D-CNN and GANs, and an open access Python library and models that are available for anyone to use. The results of the proposed work show its potential in generating realistic data that can be used if data collection is not possible, if datasets are not available, or to augment existing datasets. PT-Net can be used to analyze tremor in order to develop smarter algorithms that can be integrated with wearable tremor suppression devices to increase tremor suppression while allowing individuals with tremor to perform activities of daily living. To evaluate the reliability of the data generated by PT-Net, four evaluation methods were used. Synthetic data produced by PT-Net was used to train an estimation model, then the trained model was tested on real data, and vice versa. Results showed that the model trained on synthetic data achieves very high accuracy that is close to the accuracy achieved when an estimation model was trained on real data. The PSD and t-SNE evaluation methods also showed that the generated data had characteristics that are similar to real data. Moreover, while it is not within the scope of this work, and given the promising results presented, the proposed framework and model architecture can be adapted and used for any type of biomedical signals, without being limited to tremors, e.g., electromyography (EMG) data, electroencephalogram (EEG) data, and electrocardiogram (ECG) data.

Future work will explore the extension of this model to a conditional GAN in order to be able to generate data for specific joints and tasks. A limitation of PT-Net is that it is currently limited to generating sequences of data that are a maximum of 10 seconds long, thus, future work will also focus on modifying the PT-Net architecture in order to generate longer sequences of reliable realistic data.

Chapter 6

Conclusion

There have been significant advances in the field of machine learning, particularly in deep learning and computer vision, in recent years. It all started with ImageNet, a large-scale visual recognition challenge, where the top-5 error rate for image classification decreased from 27% in 2010 to 3% in 2016. The performance of the deep learning models emulates human recognition capabilities, and sometimes they even surpass human performance on some subcategories, such as the ability to differentiate different breeds of dogs and cats [121].

These results have encouraged many researchers to unlock the potential of deep learning in a wide variety of complex problems that once seemed impossible to solve, and are now within reach. In particular, in this dissertation, models and techniques were developed that push the frontier of PD tremor multi-step ahead motion prediction, tremor and task classification, and for the first time, the ability to generate synthetic data that resemble real tremor and voluntary motion data. This allows researchers to use the developed model to generate data to study, analyze and develop better and smarter parkinsonian tremor suppression devices when they are unable to access real data recorded from people with PD tremor. It was argued that 1) adding two layers of data processing for a dual-head model for prediction and classification is a stepping stone towards a smarter and more capable data-driven wearable tremor suppression device, and 2) an open source generative model to make PD tremor data widely available is desirable.

Specifically, in Chapter 3, a prediction model that can overcome the drawbacks of the tremor estimators currently used for wearable tremor suppression devices was developed. The proposed solution allowed the problem of the phase delay introduced by existing tremor estimators to be solved and predict up to 100 steps of motion with high accuracy.

In Chapter 4, it was shown that feature engineering is not suited for tremor and task classification. A model was then introduced that can take motion data that have been converted to images and classify, with high accuracy, the type of tremor that is active and the task being performed.

Finally, in Chapter 5, the first parkinsonian tremor generative model that is also open access, and a framework with the network architecture that can be used for training to generate any type of biosignals was introduced. This work provides three models that can generate tremor data of 2 seconds, 5 seconds, and 10 seconds long, respectively.

From a modeling perspective, the developed models fall under the category of deep learning approaches, where each model defines a single differentiable function from raw data to raw outputs. The function parameters are trained by optimizing the final objective of interest end-to-end on data. This approach offers many practical benefits: it relies on relatively simple and homogeneous operations that are repeated in layers, which decreases code complexity. As such, it is hoped that many of the architectural elements featured in this dissertation can be reused and help inspire future work.

6.1 Contributions

In this dissertation, PT-Net was proposed, a multi data-driven model, arguably a giant step forward to help in the study, analysis, and suppression of pathological tremors, specifically parkinsonian tremor. The specific contributions of the work presented in this dissertation are as follows:

1. Proposed a predictive model that can learn directly from data, can adapt to the various tremor frequencies and amplitudes and can learn the nonlinear relationships between voluntary motion and tremor with predictive ability of 10, 20, 50 and 100 steps ahead with high accuracy. The predictive model showed that it can help increase tremor suppression up to 93.8%, which is an improvement of 25% over the tremor reduction achieved by the WFLC.
2. Showed that traditional machine learning algorithms and feature engineering are not well suited to solving the problem of tremor and task classification.
3. Introduced classification models that are based on CNNs that outperform the traditional machine learning methods to detect PD tremor type and identify the task being performed.
4. Introduced a framework that is based on one-dimensional convolutional neural networks

(1D-CNN) and generative adversarial networks (GANs) that leverages the advantages of the supervised and unsupervised learning paradigms of machine learning. They can be used with any type of biosignals, without being limited to tremors, e.g., electromyography (EMG) data, electroencephalogram (EEG) data, and electrocardiogram (ECG) data.

5. Showed that the 1D-CNN GAN model generates realistic motion data that reflect a combination of voluntary motion and parkinsonian tremor.

6. Made the generative model open source, where it can be downloaded and used by research groups or teams worldwide to generate and use realistic synthetic parkinsonian tremor data.

6.2 Limitations and Future Work

The results presented in this dissertation showed the robustness of "PT-Net" for tremor estimation and multi-step ahead prediction, tremor and task classification, and generating realistic tremor and motion data. However, PT-Net is a data-driven model, where data are the main component of this successful approach. As such, it is important to recognize that a necessary condition is that the information about motion data and tremor must be made available to the machine. PT-Net is data hungry and without data it would not have achieved such results. The rest of the limitations that have been identified with the potential solutions and future directions are as follows:

1. The predictive and classification models have been trained on data that represent a finite number of tasks, which makes these models brittle when presented with data that represent different tasks from those on which the models have been trained. As such, to overcome this limitation, a potential solution is to collect more data with a wider variety of tasks, such that the models can have the ability to generalize over more scenarios when used in a real-life setting.

2. The predictive and classification models will need to run on chips with sufficient processing capabilities when they are integrated with the WTSG in order to work in real time and achieve the performance expected. This need can be addressed by using single-board computers with enough floating-point operations, such as the Nvidia Jetson Nano, Google Coral, and Rasberry Pi 4.

3. The generative model is limited to producing data that resemble the tasks that were present in the data, and cannot generate data that relate to eating, writing, or dressing up, for example. This limitation can be addressed by training the generative model on a dataset that contains data from a wider variety of tasks.

4. The generative model is limited to generating data of 2, 5, and 10 seconds long. This limitation was mainly caused by the limited size of the RAM when processing the data and the small size of the RAM of the GPU used. This can be easily addressed in two steps, first by having access to a larger RAM, and second by scaling up the size of the neural network to be able to learn from larger sequences of input data in order to generate usable synthetic data.

5. The generative model does not have the ability to generate data by specifying the type of task the user wants (e.g., generate data similar to drawing a spiral only). To overcome this limitation, the generative model can be enhanced by making it a conditional generative model, such that users can have the option of specifying the kind of data that they want the model to produce.

6. All of the three sub models of PT-Net require high computational power and time to train, which is unavoidable given the nature of neural networks and how they learn.

In addition to the proposed steps for future work, a key future direction is to make all three models of PT-Net open source and available to the public, allowing research groups to generate realistic tremor datasets, and then use or build on the classification and prediction models to develop smarter algorithms for tremor suppression devices, which in turn would help expedite the advancement of WTSDs.

Bibliography

[1] W. J. Elias and B. B. Shah, "Tremor," *JAMA*, vol. 311, no. 9, pp. 948–954, 2014.

[2] Public Health Agency of Canada (PHAC), Neurological Health Charities Canada (NHCC), "Mapping connections: An understanding of neurological conditions in canada," 2019. [Online]. Available: https://www.canada.ca/en/public-health/services/reports-publications/mapping-connections-understanding-neurological-conditions.html

[3] B. D. Adelstein and M. J. Rosen, "The effect of mechanical impedance on abnormal intention tremor," in *Bioengineering*, W. Welkowitz, Ed. Pergamon, 1981, pp. 205–209. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080272078500473

[4] A. As'arry, M. Z. Md Zain, M. Mailah, and M. Hussein, "Hybrid learning control for improving suppression of hand tremor," *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 227, no. 11, pp. 1171–1180, 2013.

[5] B. Taheri, D. Case, and E. Richer, "Theoretical development and experimental validation of an adaptive controller for tremor suppression at musculoskeletal level," in *Proceedings of the ASME Dynamic Systems and Control Conference*, Palo Alto, CA, USA, October 21–23, 2013, pp. 1–8.

[6] Y. Zhou, M. E. Jenkins, M. D. Naish, and A. L. Trejos, "Development of a wearable tremor suppression glove," in *Proceedings of the IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, Enschede, Netherlands, August 26–29, 2018, pp. 640–645.

[7] J. Á. Gallego, E. Rocon, J. M. Belda-Lois, and J. L. Pons, "A neuroprosthesis for tremor management through the control of muscle co-contraction," *Journal of Neuroengineering and Rehabilitation*, vol. 10, no. 1, pp. 1–13, 2013.

[8] B. Taheri, "Real-time pathological tremor identification and suppression in human arm via active orthotic devices," Ph.D. dissertation, Dept. of Mechanical Engineering, Southern Methodist University, 2013.

[9] D. Case, B. Taheri, and E. Richer, "Design and characterization of a small-scale magnetorheological damper for tremor suppression," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 1, pp. 96–103, 2011.

[10] Y. Zhou, M. D. Naish, M. E. Jenkins, and A. L. Trejos, "Design and validation of a novel mechatronic transmission system for a wearable tremor suppression device," *Robotics and Autonomous Systems*, vol. 91, pp. 38–48, 2017.

[11] S. Kazi, A. As'Arry, M. Z. Md Zain, M. Mailah, and M. Hussein, "Experimental implementation of smart glove incorporating piezoelectric actuator for hand tremor control," *WSEAS Transactions on Systems and Control*, vol. 5, no. 6, pp. 443–453, 2010.

[12] Y. Zhou, A. Ibrahim, K. G. Hardy, M. E. Jenkins, M. D. Naish, and A. L. Trejos, "Design and preliminary performance assessment of a wearable tremor suppression glove," *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 9, pp. 2846–2857, 2021.

[13] A. Ibrahim, Y. Zhou, M. E. Jenkins, A. L. Trejos, and M. D. Naish, "Real-time voluntary motion prediction and Parkinson's tremor reduction using deep neural networks," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 29, pp. 1413–1423, 2021.

[14] Y. Zhou, Z. Habibollahi, A. Ibrahim, M. E. Jenkins, M. D. Naish, and A. L. Trejos, "Real-time performance assessment of high-order tremor estimators used in a wearable tremor suppression device," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 30, pp. 2856–2865, 2022.

[15] A. Ibrahim, Y. Zhou, M. E. Jenkins, A. L. Trejos, and M. D. Naish, "The design of a Parkinson's tremor predictor and estimator using a hybrid convolutional-multilayer perceptron neural network," in *Proceedings of the International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, Montreal, QC, Canada, July 20–24, 2020, pp. 5996–6000.

[16] Parkinson's News Today, "Parkinson's disease statistics," 2020. [Online]. Available: https://parkinsonsnewstoday.com/parkinsons-disease-statistics

[17] C. A. Davie, "A review of Parkinson's disease," *British Medical Bulletin*, vol. 86, no. 1, pp. 109–127, 2008.

[18] C. Marras, J. Beck, J. Bower, E. Roberts, B. Ritz, G. Ross, R. Abbott, R. Savica, S. Van Den Eeden, A. Willis, *et al.*, "Prevalence of Parkinson's disease across North America," *NPJ Parkinson's Disease*, vol. 4, no. 1, pp. 1–7, 2018.

[19] J. Timmer, S. Häußler, M. Lauk, and C.-H. Lücking, "Pathological tremors: Deterministic chaos or nonlinear stochastic oscillators?" *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 10, no. 1, pp. 278–288, 2000.

[20] Y. Zhou, M. E. Jenkins, M. D. Naish, and A. L. Trejos, "The measurement and analysis of parkinsonian hand tremor," in *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, Las Vegas, NV, USA, February 24–27, 2016, pp. 414–417.

[21] G. Deuschl, P. Bain, M. Brin, and A. H. S. Committee, "Consensus statement of the movement disorder society on tremor," *Movement Disorders*, vol. 13, no. S3, pp. 2–23, 1998.

[22] E. R. De Lima, A. O. Andrade, J. L. Pons, P. Kyberd, and S. J. Nasuto, "Empirical mode decomposition: A novel technique for the study of tremor time series," *Medical and Biological Engineering and Computing*, vol. 44, no. 7, pp. 569–582, 2006.

[23] Y. Zhou, M. E. Jenkins, M. D. Naish, and A. L. Trejos, "Characterization of parkinsonian hand tremor and validation of a high-order tremor estimator," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, no. 9, pp. 1823–1834, 2018.

[24] R. C. Loureiro, J. M. Belda-Lois, E. R. Lima, J. L. Pons, J. J. Sanchez-Lacuesta, and W. S. Harwin, "Upper limb tremor suppression in ADL via an orthosis incorporating a controllable double viscous beam actuator," in *Proceedings of the 9th International Conference on Rehabilitation Robotics (ICORR)*, 2005, pp. 119–122.

[25] J. Kotovsky and M. J. Rosen, “A wearable tremor-suppression orthosis,” *Journal of Rehabilitation Research and Development*, vol. 35, pp. 373–387, 1998.

[26] B. Taheri, D. Case, and E. Richer, “Robust controller for tremor suppression at musculoskeletal level in human wrist,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 2, pp. 379–388, 2013.

[27] E. Rocon, J. Belda-Lois, A. Ruiz, M. Manto, J. C. Moreno, and J. L. Pons, “Design and validation of a rehabilitation robotic exoskeleton for tremor assessment and suppression,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 15, no. 3, pp. 367–378, 2007.

[28] K. C. Veluvolu and W. T. Ang, “Estimation of physiological tremor from accelerometers for real-time applications,” *Sensors*, vol. 11, no. 3, pp. 3020–3036, 2011.

[29] B. Taheri, D. Case, and E. Richer, “Adaptive suppression of severe pathological tremor by torque estimation method,” *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 717–727, 2014.

[30] C. A. Vaz and N. V. Thakor, “Adaptive Fourier estimation of time-varying evoked potentials,” *IEEE Transactions on Biomedical Engineering*, vol. 36, no. 4, pp. 448–455, 1989.

[31] C. N. Riviere, R. S. Rader, and N. V. Thakor, “Adaptive cancelling of physiological tremor for improved precision in microsurgery,” *IEEE Transactions on Biomedical Engineering*, vol. 45, no. 7, pp. 839–846, 1998.

[32] K. C. Veluvolu, U.-X. Tan, W. T. Latt, C. Shee, and W. T. Ang, “Bandlimited multiple fourier linear combiner for real-time tremor compensation,” in *Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Lyon, France, August 22–26, 2007, pp. 2847–2850.

[33] S. Wang, Y. Gao, J. Zhao, and H. Cai, “Adaptive sliding bandlimited multiple fourier linear combiner for estimation of pathological tremor,” *Biomedical Signal Processing and Control*, vol. 10, pp. 260–274, 2014.

[34] S. F. Atashzar, M. Shahbazi, O. Samotus, M. Tavakoli, M. S. Jog, and R. V. Patel, “Characterization of upper-limb pathological tremors: application to design of an augmented haptic rehabilitation system,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 5, pp. 888–903, 2016.

[35] K. C. Veluvolu, S. Tatinati, S. Hong, and W. T. Ang, “Multi-step prediction of physiological tremor for robotics applications,” in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Osaka, Japan, July 3–7, 2013, pp. 5075–5078.

[36] K. C. Veluvolu, S. Tatinati, S.-M. Hong, and W. T. Ang, “Multistep prediction of physiological tremor for surgical robotics applications,” *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 11, pp. 3074–3082, 2013.

[37] S. Shahtalebi, S. F. Atashzar, R. V. Patel, and A. Mohammadi, “Wake: Wavelet decomposition coupled with adaptive Kalman filtering for pathological tremor extraction,” *Biomedical Signal Processing and Control*, vol. 48, pp. 179–188, 2019.

[38] 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," *Scientific Reports*, vol. 10, no. 1, pp. 1–19, 2020.

[39] P. Sharma and M. Kaur, "Classification in pattern recognition: A review," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 4, pp. 1–9, 2013.

[40] Y. Ouali, C. Hudelot, and M. Tami, "An overview of deep semi-supervised learning," *arXiv preprint arXiv:2006.05278*, 2020.

[41] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *Technologies*, vol. 9, no. 1, p. 2, 2020.

[42] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* Cambridge, MA, USA: MIT Press, 2018.

[43] A. E. Eiben, J. E. Smith, *et al.*, *Introduction to Evolutionary Computing.* New York, NY, USA: Taylor and Francis Group, 2003.

[44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* Cambridge, MA, USA: MIT Press, 2016.

[45] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[46] J. Gou, H. Ma, W. Ou, S. Zeng, Y. Rao, and H. Yang, "A generalized mean distance-based k-nearest neighbor classifier," *Expert Systems with Applications*, vol. 115, pp. 356–372, 2019.

[47] B. Kamiński, M. Jakubczyk, and P. Szufel, "A framework for sensitivity analysis of decision trees," *Central European Journal of Operations Research*, vol. 26, no. 1, pp. 135–159, 2018.

[48] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, *et al.*, "Xgboost: Extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.

[49] T. K. Ho, "Random decision forests," in *Proceedings of the International Conference on Document Analysis and Recognition*, Montreal, QC, Canada, August 14–16, 1995, pp. 278–282.

[50] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of Machine Learning Research*, vol. 12, no. 7, pp. 2121–2159, 2011.

[51] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[53] Y. Wu, S. Zhang, Y. Zhang, Y. Bengio, and R. R. Salakhutdinov, "On multiplicative integration with recurrent neural networks," in *Proceedings of the 30th Annual Conference on Advances in Neural Information Processing Systems*, Barcelona, Spain, December 5–10, 2016, pp. 2856–2864.

[54] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[55] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[56] A. Graves, A. R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada, May 26–31, 2013, pp. 6645–6649.

[57] K. E. Lyons and R. Pahwa, "Deep brain stimulation and tremor," *Neurotherapeutics*, vol. 5, no. 2, pp. 331–338, 2008.

[58] M. Van Gerven and S. Bohte, "Artificial neural networks as models of neural information processing," *Frontiers in Computational Neuroscience*, vol. 11, pp. 114–115, 2017.

[59] A. Ibrahim, Y. Zhou, M. E. Jenkins, A. L. Trejos, and M. D. Naish, "The design of a Parkinson's tremor predictor and estimator using a hybrid convolutional-multilayer perceptron neural network," in *Proceedings of the 42nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Montreal, QC, Canada, July 20–24, 2020, pp. 1–4.

[60] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges, Belgium, April 22–24, 2015, pp. 89–94.

[61] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[62] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific Reports*, vol. 8, no. 1, pp. 1–12, 2018.

[63] X. Zhang, F. Shen, J. Zhao, and G. Yang, "Time series forecasting using GRU neural network with multi-lag after decomposition," Guangzhou, China, November 14–18, 2017, pp. 523–532.

[64] A. Graves, N. Jaitly, and A. R. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, Olomouc, Czech Republic, December 8–12, 2013, pp. 273–278.

[65] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[66] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," *AT&T Labs* [Online]. Available: http://yann.lecun.com/exdb/mnist, vol. 2, p. 18, 2010.

[67] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, August 6–11, 2017, pp. 933–941.

[68] S. Shon, A. Ali, and J. Glass, "Convolutional neural networks and language embeddings for end-to-end dialect recognition," *arXiv preprint arXiv:1803.04567*, 2018.

[69] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, Haifa, Israel, June 21–24 2010, pp. 807–814.

[70] N. I. of Neurological Disorders and Stroke, "Tremor fact sheet," 2020. [Online]. Available: https://https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Fact-Sheets/Tremor-Fact-Sheet

[71] A. M. Woods, M. Nowostawski, E. A. Franz, and M. Purvis, "Parkinson's disease and essential tremor classification on mobile device," *Pervasive and Mobile Computing*, vol. 13, pp. 1–12, 2014.

[72] S. Barrantes, A. J. Sánchez Egea, H. A. González Rojas, M. J. Martí, Y. Compta, F. Valldeoriola, E. Simo Mezquita, E. Tolosa, and J. Valls-Solè, "Differential diagnosis between Parkinson's disease and essential tremor using the smartphone's accelerometer," *PloS One*, vol. 12, no. 8, pp. 1–12, 2017.

[73] R. G. Garcia, A. H. Ballado, A. C. Paglinawan, C. C. Paglinawan, R. B. Gavino, B. A. J. Magcamit, J. C. S. Miranda, and M. F. Tiongson, "Hand tremor analyzer using accelerometry for preliminary diagnosis, classification and monitoring of selected movement disorders," in *Proceedings of the 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, Penang, Malaysia, November 25–27, 2016, pp. 392–396.

[74] K. P. Bhatia, P. Bain, N. Bajaj, R. J. Elble, M. Hallett, E. D. Louis, J. Raethjen, M. Stamelou, C. M. Testa, G. Deuschl, *et al.*, "Consensus statement on the classification of tremors. From the Task Force on Tremor of the International Parkinson and Movement Disorder Society," *Movement Disorders*, vol. 33, no. 1, pp. 75–87, 2018.

[75] L. Ai, J. Wang, and R. Yao, "Classification of parkinsonian and essential tremor using empirical mode decomposition and support vector machine," *Digital Signal Processing*, vol. 21, no. 4, pp. 543–550, 2011.

[76] P. Palmes, W. T. Ang, F. Widjaja, L. C. Tan, and W. L. Au, "Pattern mining of multichannel sEMG for tremor classification," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 12, pp. 2795–2805, 2010.

[77] S. Fahn, "Classification of movement disorders," *Movement Disorders*, vol. 26, no. 6, pp. 947–957, 2011.

[78] F. Gövert and G. Deuschl, "Tremor entities and their classification: An update," *Current Opinion in Neurology*, vol. 28, no. 4, pp. 393–399, 2015.

[79] A. B. Oktay and A. Kocer, "Differential diagnosis of Parkinson and essential tremor with convolutional LSTM networks," *Biomedical Signal Processing and Control*, vol. 56, pp. 101 683–101 689, 2020.

[80] E. Rocon, A. Ruiz, J. L. Pons, J. M. Belda-Lois, and J. Sánchez-Lacuesta, "Rehabilitation robotics: A wearable exo-skeleton for tremor assessment and suppression," in *Proceedings of the International Conference on Robotics and Automation*, Barcelona, Spain, April 18–22, 2005, pp. 2271–2276.

[81] S. Dosen, S. Muceli, J. L. Dideriksen, J. P. Romero, E. Rocon, J. Pons, and D. Farina, "Online tremor suppression using electromyography and low-level electrical stimulation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 3, pp. 385–395, 2014.

[82] W. C. Pinheiro, B. E. Bittencourt, L. B. Luiz, L. A. Marcello, V. F. Antonio, P. H. A. de Lira, R. G. Stolf, and M. C. F. Castro, “Parkinson’s disease tremor suppression,” in *Proceedings of the 10th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC)*, Porto, Portugal, February 21–23, 2017, pp. 21–23.

[83] J. Jankovic, “Parkinson’s disease: Clinical features and diagnosis,” *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 79, no. 4, pp. 368–376, 2008.

[84] A. A. Cook, G. Misirli, and Z. Fan, “Anomaly detection for IoT time-series data: A survey,” *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, 2019.

[85] A. Ibrahim, Y. Zhou, M. E. Jenkins, M. D. Naish, and A. L. Trejos, “Parkinson’s tremor onset detection and active tremor classification using a multilayer perceptron,” in *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, London, ON, Canada, August 30–September 2, 2020, pp. 1–4.

[86] K. Englehart, B. Hudgin, and P. A. Parker, “A wavelet-based continuous classification scheme for multifunction myoelectric control,” *IEEE Transactions on Biomedical Engineering*, vol. 48, no. 3, pp. 302–311, 2001.

[87] J. Too, A. R. Abdullah, and N. M. Saad, “Classification of hand movements based on discrete wavelet transform and enhanced feature extraction,” *International Journal of Advanced Computer Science and Applications (JACSA)*, vol. 10, no. 6, pp. 83–89, 2019.

[88] F. Al Omari and G. Liu, “Analysis of extracted forearm sEMG signal using LDA, QDA, K-NN classification algorithms,” *The Open Automation and Control Systems Journal*, vol. 6, no. 1, pp. 108–116, 2014.

[89] S. Sadeghyan, “A new robust feature selection method using variance-based sensitivity analysis,” *arXiv preprint arXiv:1804.05092*, 2018.

[90] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1–3, pp. 37–52, 1987.

[91] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[92] S. Nembrini, I. R. König, and M. N. Wright, “The revival of the Gini importance?” *Bioinformatics*, vol. 34, no. 21, pp. 3711–3718, 2018.

[93] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation functions: Comparison of trends in practice and research for deep learning,” *arXiv preprint arXiv:1811.03378*, 2018.

[94] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” in *Proceedings of the 20th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, Vancouver, B.C., Canada, December 4–7, 2006, pp. 153–160.

[95] L. Girin, S. Leglaive, X. Bie, J. Diard, T. Hueber, and X. Alameda-Pineda, “Dynamical variational autoencoders: A comprehensive review,” *arXiv preprint arXiv:2008.12595*, 2020.

[96] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *Proceedings of the IEEE International Conference on Engineering and Technology (ICET)*, Antalya, Turkey, August 21–23, 2017, pp. 1–6.

[97] S. Sinha, P. S. Routh, P. D. Anno, and J. P. Castagna, "Spectral decomposition of seismic data with continuous-wavelet transform," *Geophysics*, vol. 70, no. 6, pp. 19–25, 2005.

[98] C. J. Ortiz-Echeverri, J. Rodríguez-Reséndiz, and M. Garduño-Aparicio, "An approach to STFT and CWT learning through music hands-on labs," *Computer Applications in Engineering Education*, vol. 26, no. 6, pp. 2026–2035, 2018.

[99] N. E. Huang, *Hilbert-Huang Transform and its Applications.* Hackensack, NJ, USA: World Scientific, 2014, vol. 16.

[100] N. E. Huang and Z. Wu, "A review on Hilbert-Huang transform: Method and its applications to geophysical studies," *Reviews of Geophysics*, vol. 46, no. 2, pp. 1–23.

[101] V. Shanker, "Essential tremor: diagnosis and management," *BMJ*, vol. 366, 2019. [Online]. Available: https://www.bmj.com/content/366/bmj.l4485

[102] R. C. Helmich, "The cerebral basis of parkinsonian tremor: A network perspective," *Movement Disorders*, vol. 33, no. 2, pp. 219–231, 2018.

[103] M. S. Faizan and M. Muzammil, "Hand tremor suppression device for patients suffering from Parkinson's disease," *Journal of Medical Engineering & Technology*, vol. 44, no. 4, pp. 190–197, 2020.

[104] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[105] H. Yuan, S. Liu, J. Liu, H. Lin, C. Yang, X. Cai, L. Zeng, and S. Li, "Detection and quantification of resting tremor in Parkinson's disease using long-term acceleration data," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–12, 2021.

[106] G. AlMahadin, A. Lotfi, M. M. Carthy, and P. Breedon, "Enhanced Parkinson's disease tremor severity classification by combining signal processing with resampling techniques," *SN Computer Science*, vol. 3, no. 1, pp. 1–21, 2022.

[107] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 28th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, Montreal, QC, Canada, December 8–13, 2014, pp. 2672–2680.

[108] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, August 6–11, 2017, pp. 214–223.

[109] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE international Conference on Computer Vision*, Venice, Italy, October 22–29, 2017, pp. 2794–2802.

[110] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July 21–26, 2017, pp. 1125–1134.

[111] Z. Yi, H. Zhang, P. Tan, and M. Gong, "DualGAN: Unsupervised dual learning for image-to-image translation," in *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, October 22–29, 2017, pp. 2849–2857.

[112] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional GANs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, June 18–23, 2018, pp. 8798–8807.

[113] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, Munich, Germany, September 8–14, 2018, pp. 63–79.

[114] R. Huang, S. Zhang, T. Li, and R. He, "Beyond face rotation: Global and local perception GAN for photorealistic and identity preserving frontal view synthesis," in *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, October 22–29, 2017, pp. 2439–2448.

[115] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun, "Adversarial ranking for language generation," in *Proceedings of the 31st Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, December 4–9, 2017, pp. 3158–3168.

[116] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," *arXiv preprint arXiv:1702.05983*, 2017.

[117] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *Proceedings of the International Conference on Information Processing in Medical Imaging*, Boone, NC, USA, June 28–30, 2017, pp. 146–157.

[118] N. Anand and P. Huang, "Generative modeling for protein structures," *Proceedings of the 32nd Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, pp. 7505–7516, December 3–8, 2018.

[119] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the International Conference on Machine Learning*, Atlanta, Georgia, USA, June 16–21, 2013, pp. 1–6.

[120] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.

[121] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

Curriculum Vitae

Name:	Anas Ibrahim
Post-secondary Education and Degrees:	2018–2022, Ph.D Electrical and Computer Engineering – Software Engineering Department of Electrical and Computer Engineering Western University London, ON, Canada
	2015–2017, M.E.Sc Electrical and Computer Engineering – Software Engineering Department of Electrical and Computer Engineering Western University London, ON, Canada
	2010–2015, B.Eng Computer Engineering Department of Electrical and Computer Engineering Beirut Arab University Beirut, Lebanon
Related Work Experience	Teaching & Research Assistant Western University 2018–2022
	Teaching & Research Assistant Western University 2015–2017

Publications:

1. Y. Zhou, Z. Habibollahi, A. Ibrahim, M. E. Jenkins, M. D. Naish, and A. L. Trejos, “Real-time performance assessment of high-order tremor estimators used in a wearable tremor suppression device,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 30, pp. 2856–2865, 2022.

2. A. Ibrahim, Y. Zhou, M. E. Jenkins, A. L. Trejos, and M. D. Naish, “Real-time voluntary motion prediction and Parkinson’s tremor reduction using deep neural networks,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 29, pp. 1413–1423, 2021.

3. Y. Zhou, A. Ibrahim, K. G. Hardy, M. E. Jenkins, M. D. Naish, and A. L. Trejos, “Design and preliminary performance assessment of a wearable tremor suppression glove,” *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 9, pp. 2846–2857, 2021.

4. Y. Zhou, A. Ibrahim, M. E. Jenkins, M. D. Naish, and A. L. Trejos, "Analysis of the effect of common disturbances on the safety of a wearable tremor suppression device," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2846–2853, 2021.

5. A. Ibrahim, Y. Zhou, M. E. Jenkins, A. L. Trejos, and M. D. Naish, "The design of a Parkinson's tremor predictor and estimator using a hybrid convolutional-multilayer perceptron neural network," in *Proceedings of the 42nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Montreal, QC, Canada, July 20–24, 2020, pp. 1–4.

6. A. Ibrahim, Y. Zhou, M. E. Jenkins, M. D. Naish, and A. L. Trejos, "Parkinson's tremor onset detection and active tremor classification using a multilayer perceptron," in *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE).* London, ON, Canada, August 30–Sept. 2, 2020, pp. 1–4.

7. J.G. Colli-Alfaro, A. Ibrahim, A. L. Trejos, "Design of user-independent hand gesture recognition using multilayer perceptron network and sensor fusion techniques," in *Proceedings of the 16th IEEE International Conference on Rehabilitation Robotics (ICORR).* Toronto, ON, Canada, June 23–28, 2019, pp. 1103–1108.

8. A. Ibrahim, A. Ouda, "A hybrid-based filtering approach for user authentication," in *Proceedings of the 30th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE).* Windsor, ON, Canada, April 30–May 3, 2017, pp. 1–5.

9. A. Ibrahim, A. Ouda, 'Innovative data authentication model," in *Proceedings of the 7th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON).* Vancouver, BC, Canada, October 13–15, 2016, pp. 1–7.