UNIVERSITY OF PADOVA

DEPARTMENT OF INFORMATION ENGINEERING
Degree Course: Bioengineering

A Multimodal Deep Learning Framework for Improved Parkinson's Disease Detection Using Biosignals: A BioVRSea Paradigm Study

Supervisor:
Emanuela Formaggio

Master candidate:
Riccardo Brun

Co-supervisors:
Federico Del Pup
Paolo Gargiulo

Academic Year 2024 - 2025
Graduation date 17/10/2025

Abstract

Electroencephalography (EEG), electromyography (EMG), and center of pressure (CoP) signals capture complementary aspects of motor and postural control. These modalities have recently emerged as promising biomarkers for early detection of Parkinson's disease. Despite encouraging progress with deep learning models applied to single modalities, existing approaches often suffer from high inter-subject variability and limited robustness. This limits their generalization to real-world clinical scenarios. This thesis aims to improve early-stage PD classification by developing a multimodal deep learning framework that effectively integrates EEG, EMG, and CoP signals while preserving their temporal structure. Literature-established architectures for EEG, EMG, and CoP were adapted into modality-specific encoders by modifying their structures to retain temporal dynamics within learned representations. These encoders were combined through fusion strategies, emphasizing sequence-level integration. Additionally, a novel Multiple Instance Learning (MIL) paradigm was introduced to assess its potential advantages over supervised learning. The proposed framework was evaluated on the BioVRSea dataset comprising 300 subjects (29 Parkinson's disease patients and 271 healthy controls) using a stratified nested cross-validation scheme to ensure unbiased subject-level evaluation and prevent information leakage. In the supervised setting with data augmentation, multimodal fusion achieved a median balanced accuracy of 81.67% and an F1-score of 75.00%, with a [1st–99th] balanced accuracy percentile range of 23.54%. MIL-based experiments did not consistently outperform the supervised framework and showed strong dependence on architectural design, suggesting limited reliability under current dataset constraints. These results demonstrate that preserving temporal dynamics within modality-specific representations and integrating them through supervised multimodal fusion substantially improves classification performance. The findings highlight the potential of multimodal deep learning as a robust approach for early Parkinson's disease detection from EEG, EMG, and CoP signals.

Sommario

L'elettroencefalografia (EEG), l'elettromiografia (EMG) e il centro di pressione (CoP) catturano aspetti complementari del controllo motorio e posturale. Queste modalità hanno recentemente mostrato un notevole potenziale come biomarcatori per la diagnosi precoce del morbo di Parkinson. Nonostante i progressi incoraggianti del deep learning applicato alle singole modalità, gli approcci esistenti presentano spesso un'elevata variabilità inter-soggetto ed una limitata robustezza, riducendo così la loro capacità di generalizzare a scenari clinici. L'obiettivo della tesi è migliorare la classificazione della patologia attraverso lo sviluppo di un framework basato sul deep learning multimodale in grado di integrare efficacemente i segnali EEG, EMG e CoP, preservandone la struttura temporale. Architetture derivate dalla letteratura sono state adattate, per ciascuna modalità, in encoder, modificandone la struttura per mantenere la dinamica temporale nelle rappresentazioni estratte. Tali encoder sono stati successivamente combinati mediante strategie di fusione, con particolare attenzione ad integrazioni a livello di sequenza temporale. Inoltre, è stato introdotto un nuovo paradigma di Multiple Instance Learning (MIL) per valutarne i potenziali vantaggi rispetto all'apprendimento supervisionato. Il framework proposto è stato valutato sul dataset BioVRSea, composto da 300 soggetti (29 soggetti affetti da morbo di Parkinson e 271 soggetti sani), utilizzando uno schema di nested cross-validation stratificata per garantire una valutazione imparziale a livello di soggetto ed evitare fenomeni di data leakage. Nell'approccio supervisionato con data augmentation, la fusione multimodale ha raggiunto valori mediani di accuratezza bilanciata dell'81,67% e F1-score del 75,00%, con un intervallo [1°–99°] percentile di accuratezza bilanciata pari a 23,54%. Gli esperimenti basati sul paradigma MIL non hanno superato in modo consistente il framework supervisionato, mostrando una forte dipendenza dal design architetturale e quindi una limitata affidabilità considerati i vincoli del dataset attuale. Questi risultati dimostrano che preservare la dinamica temporale nelle rappresentazioni per le singole modalità ed integrarle tramite fusione multimodale supervisionata migliora significativamente le prestazioni. Nel complesso, i risultati confermano il valore del deep learning multimodale come strumento robusto per la diagnosi precoce del morbo di Parkinson da segnali EEG, EMG e CoP.

Contents

Nomenclature

Table 1: Nomenclature of common deep learning terminology

Term	Definition
Loss function	A function that quantifies the difference between predicted outputs and ground-truth labels
Gradient	The vector of partial derivatives of a loss function with respect to the model parameters
Gradient descent	An optimization algorithm that updates parameters in the opposite direction of the gradient of the loss
Optimizer	The algorithm used to update model parameters based on gradients
Regularization	Techniques to prevent overfitting by constraining the model
Forward pass or Forward propagation	The computation of model outputs from inputs by propagating data through the network
Backward pass or Backpropagation	The process of propagating gradients back through the network using the chain rule
Batch	A subset of the training dataset processed simultaneously in one forward and backward pass.
Epoch	One complete pass through the entire training dataset
Overfitting	When a model learns training data too closely and fails to generalize to unseen data
Generalization	The ability of a model to perform well on new, unseen data
Embedding	A low-dimensional representation of data that preserves meaningful informations
Neural Network	An artificial intelligence algorithm inspired by the human brain
Encoder	A neural network designed to extract an embedding of the input data

Term	Definition
Stride	The step size used to slide a filter or window over a signal or feature map
Convolutional kernel	A trainable vector/matrix used for extracting information through convolutions, determined by a kernel size and a stride
Pooling	A downsampling operation determined by a kernel size and a stride but with no learnable weights
Token	Individual units of data representing letters, words of phrases of natural language[1]
Tokenizer	A Neural Network designed to extract a series of tokens from its input data

[1] Despite this thesis not being about language modeling, the deep learning community still uses the word token to refer to the typical unit of representation used as input to transformers

Chapter 1

Introduction

1.1 The Importance of Early-Stage Parkinson's Disease Detection and Challenges

Parkinson's Disease (PD) is a progressive neurodegenerative disorder and the second most common after Alzheimer's. It affects movement, balance, and coordination, with early symptoms often so subtle that diagnosis is often delayed until significant degeneration has occurred. The World Health Organization reports a doubling in PD prevalence over the past 25 years, now affecting more than 8.5 million people around the word. It affects primarily individuals over the age of 65, with both incidence and prevalence steadily increasing with age [1].

Early diagnosis is crucial to initiate treatment plans that can slow disease progression and support patient quality of life. Levodopa-based therapy remains the gold standard, along with exercise and Deep Brain Stimulation (DBS) [2]. Although treatment can slow symptoms, motor and cognitive impairments progress, making early detection and classification essential for long-term disease management.

Among the techniques being explored for early detection, Electroencephalography (EEG) has emerged as a promising non-invasive tool. Quantitative EEG (qEEG) studies have shown differences in spectral power in PD patients; during resting state acquisitions these are typically characterized by reduced activity in the alpha and beta bands and increased power in the delta and theta bands [3], in contrast, recordings from the BioVRSea task-based paradigm showed reduced activity in both the theta and alpha bands [4]. However, EEG-based automated classification remains technically challenging. Deep Learning (DL) models trained on EEG data have emerged as a powerful tool. However, despite their ability to extract highly nonlinear patterns embedded within the signal [5], they often suffer from limited robustness and generalizability, due to inter-subject variability, noise, and relatively small datasets.

Electromyography (EMG) has also been proposed as a diagnostic tool, and it is more prominent in

gait analysis studies, where it has been shown that PD patients exhibit decreased distal and increased proximal lower limb muscle activity during walking, with greater variability compared to healthy controls. There remains insufficient evidence directly linking these muscle activation changes to specific gait impairments [6]. Deep learning models applied to EMG data for PD detection have also shown promising results, with several studies reporting strong performance using common architectures. However, they face similar challenges, including limited data, variability across subjects, and a lack of clinical validation [5].

Center of Pressure (CoP) measurements during quiet standing have long been used to assess postural control in PD. Studies consistently report increased sway magnitude and greater instability, particularly in the Medio-Lateral (ML) direction, which reflects compensatory and less efficient balance control strategies [7]. However, despite the high potential for deep learning applications in this domain, no reliable models have yet been applied to PD classification using CoP data.

1.2 Experimental Context: The BioVRSea Dataset and Its Constraints

To investigate the neurophysiological basis of postural control in early-stage PD we use the BioVRSea paradigm, a unique and multimodal experimental protocol that integrates high-density EEG, EMG, CoP, and heart rate (HR) recordings. Participants are exposed to a sequence of phases designed to mimic maritime balance challenges, progressing from rest to visually and physically perturbed conditions. These perturbations are delivered through a moving platform synchronized with an immersive Virtual Reality (VR) environment, enabling a realistic simulation of being at sea [8]. As such, BioVRSea provides a valuable framework for exploring postural control impairments in neurological conditions such as PD [4].

The dataset includes healthy control (HC) individuals and subjects diagnosed with early-stage Parkinson's Disease, all of whom are under Levodopa treatment. Although the BioVRSea setup provides a rich multimodal perspective on postural control, a key limitation lies in the imbalance between the two classes. This imbalance may affect the robustness of the classification results and limits the generalizability of findings to the broader PD population. Therefore, classification performance should be interpreted with caution.

1.3 Thesis Objective: Signal Fusion for Early Parkinson's Disease Detection During Postural Control Challenge

The core contribution of this work lies in the fusion of EEG, EMG, and CoP signals during a postural challenge to improve early-stage PD classification. Although EEG alone has been extensively studied

for PD-related abnormalities, few studies have incorporated muscular and biomechanical signals into the classification pipeline, particularly using modern deep learning architectures. The rationale for this multimodal approach is grounded in the idea that these signals are complementary: EEG reflects central nervous system activity and cortical processing, EMG captures peripheral neuromuscular activity and CoP provides biomechanical information on balance and postural control. Previous studies using BioVRSea data have analyzed these modalities separately to characterize group-level differences between PD and healthy controls [4]. However, the integration of these signals into a unified predictive model remains an open challenge.

In this thesis, we propose deep learning architectures that can learn both intra-modality patterns and cross-modality interactions during the MOVE phase of the protocol. We evaluated different fusion strategies, ranging from simple concatenation to single token fusion and transformer-based mechanisms, with the goal of identifying patterns that distinguish healthy controls from early PD subjects. Particular attention is paid to preserving the temporal structure and physiological meaning of each signal type.

Although the small sample size of the PD group prevents strong claims about generalizability to the broader population, the methods developed here aim to support future multimodal research, especially when larger datasets become available. Moreover, the techniques explored in this work may be applied in other contexts involving different neurological disorder classification tasks.

1.4 Thesis Structure

This thesis is organized into six chapters. Chapter 1 introduces the motivation and importance of early Parkinson's disease detection, the experimental context, and the objectives of this work. Chapter 2 presents the state of the art in deep learning approaches for biomedical signal analysis, with dedicated sections on EEG-based, EMG-based, and CoP-based Parkinson's disease detection, as well as strategies for multimodal signal fusion. Chapter 3 describes the methodology, including the dataset, preprocessing pipelines, data partitioning, augmentation strategies, model architectures, and training procedures. Chapter 4 reports the experimental results, comparing supervised learning and multiple instance learning approaches, analyzing discrepancies, and evaluating different fusion strategies. Chapter 5 discusses the findings in light of the existing literature and addresses the limitations of the study. Finally, Chapter 6 concludes the thesis and outlines possible directions for future work.

Chapter 2

State of the Art - Deep Learning

2.1 Introduction

Deep learning has emerged as a powerful subset of machine learning, capable of modeling complex patterns in large-scale data through architectures inspired by the structure and function of the human brain [9]. Unlike traditional machine learning approaches, which require handcrafted features, deep learning models can learn hierarchical feature representations directly from raw or minimally preprocessed data, enabling significant performance improvements across a wide range of tasks, from computer vision to speech recognition and biomedical signal analysis.

At the core of deep learning are artificial neural networks (ANNs), particularly deep neural networks (DNNs), which consist of multiple layers of interconnected nodes, called neurons, typically involving sequences of linear transformations followed by nonlinear activation functions. These layers enable the modeling of increasingly abstract features, which makes deep networks particularly suitable for tasks involving unstructured data such as images, audio, and time-series signals. Training these models typically involves optimization techniques such as stochastic gradient descent and regularization methods such as dropout to prevent overfitting [10].

During the past decade, deep learning has driven substantial breakthroughs in domains such as image classification, natural language processing [11], and reinforcement learning. More recently, its applications have been extended to the biomedical domain, including disease classification from biosignals, medical image segmentation, and patient outcome prediction. These applications have highlighted the potential of deep learning to outperform classical techniques, particularly when large amounts of high-dimensional data are available.

Despite its successes, deep learning remains an active area of research, with ongoing challenges including model interpretability, generalizability across datasets, and the need for large annotated datasets. The remainder of this chapter explores the current state of the art deep learning methods relevant to this thesis, with a focus on architectures and techniques applied in biomedical signal processing and neurodegenerative disease classification.

2.2 EEG-based Parkinson's Disease Detection

Reference [5] provides a comprehensive review that analyzed 63 DL-based PD detection studies across various modalities and found that only a small subset used EEG as the primary input modality. Despite the limited number of EEG-focused studies, reported performances were generally high, often exceeding 80–90% accuracy in binary classification tasks. Convolutional Neural Networks (CNNs) were the dominant architecture, accounting for roughly 57% of the models across all modalities, with hybrid CNN–recurrent designs also represented. However, the review highlighted persistent challenges, including small dataset sizes, limited reproducibility, and inconsistent reporting of preprocessing steps.

Two of the most influential EEG-specific architectures are EEGNet and ShallowConvNet. EEGNet [12] introduced a compact architecture using depthwise and separable convolutions to efficiently capture temporal and spatial EEG features, making it suitable for small datasets and real-time applications. ShallowConvNet, introduced alongside DeepConvNet [13], was designed to extract features through large temporal convolution kernels followed by spatial filtering, achieving strong performance in Brain Computer Interface (BCI) tasks. Both have been widely adopted as benchmarks in EEG classification studies, including those targeting PD detection.

Subsequent work has extended these foundational designs by integrating recurrent layers, such as Long Short-Term Memory (LSTM) networks, attention mechanisms, and graph-based learning. CNN–LSTM hybrids have been applied to PD EEG classification, where convolutional layers learn spatial filters and LSTM layers capture temporal dynamics.

Graph-based approaches have also emerged. In [14] researchers proposed an attention-based sparse graph convolutional neural network (ASGCNN) for PD detection, which learned channel relationships while promoting sparsity to highlight the most relevant brain regions.

Transformer architectures, originally developed for natural language processing, have been adapted to EEG classification to model long-range temporal dependencies. By treating EEG channels or segments as tokens, transformers can capture complex spatial–temporal interactions without relying on recurrence [15].

Preprocessing strategies vary considerably across EEG-based PD studies. Some use only minimal steps such as bandpass filtering and standardization, while others incorporate artifact removal, such as Independent Component Analysis (ICA), spatial filtering, like Common Spatial Patterns (CSP), or wavelet-based denoising. Although complex preprocessing can improve signal quality, several high-performing studies have successfully trained on minimally preprocessed data, suggesting that modern DL models may internally suppress noise and artifacts [5]. This variability in preprocessing makes direct cross-study performance comparisons difficult.

EEG-based DL approaches for PD detection have progressed from compact CNNs like EEGNet to hybrids, graph neural networks, and transformers. Although performance reports are promising, the field remains limited by small datasets, inconsistent preprocessing, and scarce reproducibility, all of

which must be addressed to enable clinical deployment.

2.3 EMG-based Parkinson's Disease Detection

According to reference [5], only a small number of deep learning studies have investigated EMG for Parkinson's disease detection compared to other modalities such as gait or speech. While fewer in number than EEG-based approaches, EMG-based studies benefit from directly capturing motor unit activity and muscle activation patterns, which are often altered in PD. These alterations can manifest as changes in tremor frequency components, muscle activation timing, and coordination between muscle groups, making EMG a valuable modality for PD assessment.

The majority of EMG-based PD detection studies reviewed in [5] employed CNNs, leveraging their ability to extract localized temporal–frequency features from raw or transformed EMG signals. In some cases, CNNs were applied directly to spectrograms or time–frequency representations, allowing the network to learn frequency-specific PD biomarkers such as pathological tremor activity.

Some studies extended baseline CNNs with Recurrent Neural Networks (RNNs) to better capture temporal dependencies in EMG sequences, particularly during repetitive or tremor-related movements, resulting in the development of hybrid CNN-RNN architectures.

Although relatively underexplored compared to EEG, EMG-based deep learning studies demonstrate that convolutional architectures can effectively capture pathological motor patterns in PD. However, the limited number of works highlights the need for larger-scale, standardized datasets and comparative benchmarks to fully establish the utility of EMG in PD detection.

2.4 CoP-based Parkinson's Disease Detection

Center of Pressure signals, derived from force platforms, provide an objective measure of postural stability and balance control. Since postural instability is a hallmark symptom of Parkinson's Disease, CoP has been investigated as a non-invasive biomarker for PD assessment. Traditional analysis has relied heavily on handcrafted features, including sway area, path length, velocity, and frequency-domain descriptors, which capture balance deficits associated with PD. The application of deep learning in this context is still at an early stage. Unlike EEG and EMG, no widely adopted neural network architecture has been established as a benchmark for CoP-based PD detection.

A recent contribution in this direction is the work by [16], in which researchers proposed a Temporal Convolutional Neural Network (TCNN) to detect Freezing of Gait (FoG) episodes from plantar-pressure data collected via wearable insoles. Their model demonstrated promising performance in characterizing FoG dynamics, highlighting the feasibility of using temporal deep learning approaches for balance- and gait-related PD symptoms.

In summary, while CoP offers a physiologically meaningful and non-invasive window into postural instability, deep learning applications remain limited and exploratory. Current studies, such as those employing TCNNs, illustrate the potential of this modality but also underline the need for larger-scale validation and the establishment of standardized neural architectures for CoP-based PD detection.

2.5 Multimodal Signal Fusion

Multimodal signal fusion refers to the integration of complementary information from different data sources into a unified deep learning framework. The motivation stems from the observation that single modalities often capture only a partial view of complex biomedical phenomena, whereas combining multiple physiological signals can provide a richer and more discriminative representation. In the context of Parkinson's Disease, modalities such as EEG, EMG, and CoP highlight different aspects of motor and neural dysfunction, ranging from cortical activity to muscle activation and postural control, and their joint use holds promise for improving diagnostic accuracy.

From a machine learning perspective, multimodal fusion is not unique to biomedical applications; it has been extensively studied in domains such as speech, vision, and natural language processing. A comprehensive taxonomy of multimodal fusion approaches was outlined by [17], which distinguishes methods based on the stage at which modalities are integrated within the learning pipeline. This taxonomy is widely adopted across disciplines and can be directly applied to biosignal processing, where signal heterogeneity, temporal alignment, and varying signal-to-noise ratios present unique challenges.

2.5.1 Early Fusion

Early fusion, also referred to as feature-level fusion, integrates modalities at the input stage, either by concatenating raw signals or by combining hand-engineered features before passing them to a learning model. In the context of biosignals, this might involve aligning EEG, EMG, and CoP time series and feeding them jointly into a shared network. Although conceptually simple, early fusion is often suboptimal because raw physiological signals differ in sampling rate, dimensionality, and noise characteristics. This can lead to challenges in synchronizing modalities and may obscure modality-specific features that are critical for PD classification.

2.5.2 Intermediate Fusion

Intermediate fusion strategies, sometimes called representation-level fusion, first process each modality with a dedicated encoder and then combine the resulting latent representations. This approach allows networks to exploit modality-specific characteristics before integration, making it particularly suitable for heterogeneous biosignals. According to [17], the representations can be stacked or summed,

with or without modality weights, or even combined by leveraging cross-attention through the decoder part of the transformer architecture [11]. Intermediate fusion can be further categorized based on the similarity of the modality-specific representations in output from the encoders.

Homogeneous fusion is characterized by having the encoders for different modalities produce representations with compatible structure. In practice, this corresponds to the fusion of representations without applying any type of alignment strategy.

Heterogeneous fusion consists in having encoders produce modality-specific embeddings that differ in scale, dimensionality, or structure. These must be transformed into a common space through pooling, projection, or cross-modal attention before being combined. For example, in order for the representations to be compatible for fusion, signals with different numbers of embedded channels need to be projected to a common embedding dimension.

In general, intermediate fusion offers a balance between capturing modality-specific detail and learning cross-modal interactions, and represents the most flexible strategy for multimodal biosignal learning.

2.5.3 Late Fusion

Late fusion, also known as decision-level fusion, integrates modalities only at the output stage, typically by combining predictions from unimodal classifiers. Methods range from simple averaging or majority voting to more sophisticated meta-classifiers that learn optimal weighting schemes. Late fusion is attractive when modalities are difficult to align or when unimodal pipelines are already well-optimized. However, it generally fails to capture fine-grained cross-modal interactions, since integration occurs only after independent decisions are made.

Chapter 3

Methods

Notation

The following table summarizes the notation used.

Notation	Description
C	The number of channels of a signal
L	The number of samples of a signal or window
W	The window size in seconds
O	The overlap percentage $\in [0, 1)$
S	The stride of a signal in seconds
N	The batchsize, the number of examples given as input to the model
$\hat{z}_i$	The logit for example i
p_i	The probability for example i
$\hat{y}_i$	The predicted label for example i
y_i	The ground truth label for example i
$\mathscr{L}$	A generic loss function
ℓ	The learning rate
$[\, n \,]$	General integer set representing $\{1, 2, \ldots, \mathrm{n}\}$
$x_{c,t}$	Scalar data sample for channel c at instant t, where $c \in [\, C \,]$ and $t \in [\, L \,]$
$\odot$	The Hadamard product, the element-wise multiplication between matrices
$[\mathbf{x}_1 \,;\, \mathbf{x}_2]$	The concatenation of vectors

Notation	Description
$[X_1 ; X_2]_n$	The concatenation of tensors/matrices along dimension n
$\mathcal{N}(0, 1)$	Standard normal distribution
$\mathcal{U}(\{a, b, ..., n\})$	Discrete uniform distribution

Table 3.1: Notation table

3.1 Dataset

In the following subsections, the dataset, equipment and hardware, and the BioVRSea protocol are explained, highlighting the age confounder issue and the difference in equipment used, which were carefully considered during data partitioning in *Subsection 3.3.1*.

3.1.1 Dataset Description

The BioVRSea dataset originally includes 549 subjects, spanning multiple diagnostic groups, such as healthy controls, Parkinson's Disease patients, subjects with a history of concussion and whiplash, and healthy sailors. The pie chart in *Figure 3.1a* illustrates the cohort proportions in the dataset.

In this thesis we explore Parkinson's Disease detection, therefore from the whole dataset we include only Healthy Controls (HC) and affected patients. A further selection criterion is the availability of the recordings for all three signals with less than 30% of missing data for each subject. These constraints reduce the subject count from 380 to 271 Healthy Controls and from 42 to 29 Parkinson's Patients, a visual representation of the proportion is illustrated in *Figure 3.1b*.

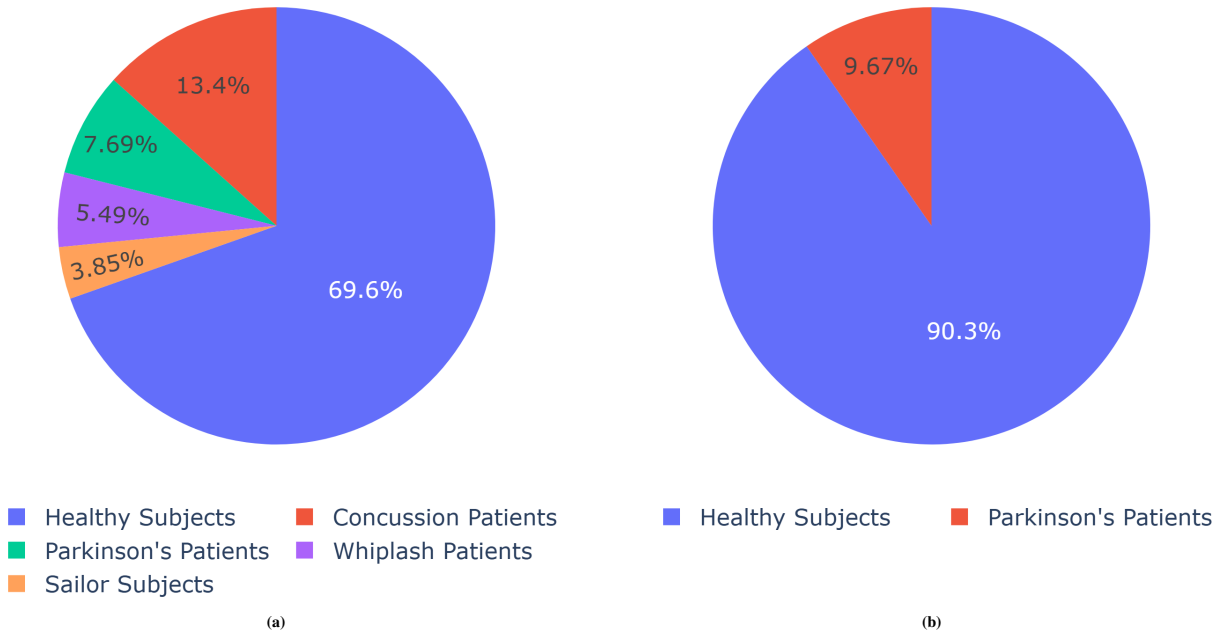

Figure 3.1: Pie charts representing the proportions of dataset cohorts (a) and the selected subset (b)

3.1.2 Subject Demographics

After applying the selection criteria, we analyzed the demographics of the remaining subjects. In particular, we focus on the age and gender distributions. Inspection of the healthy cohort revealed an age confounder with respect to Parkinson's patients, evident in *Figure 3.2*. To address this, we set a threshold at 45 years, chosen based on the youngest PD patient, distinguishing between younger and older healthy controls. Among the 271 selected HCs, 58 subjects are aged 45 years or older, while the remaining 213 are younger than 45 years. The age distribution after thresholding is illustrated in *Figure 3.3*. Finally, in *Figure 3.4* the proportion of patients and healthy groups can be observed. Analyzing gender distributions in *Figure 3.5* and *Figure 3.6* we can observe that the ratio between male and female subjects is balanced and remains stable after age thresholding.

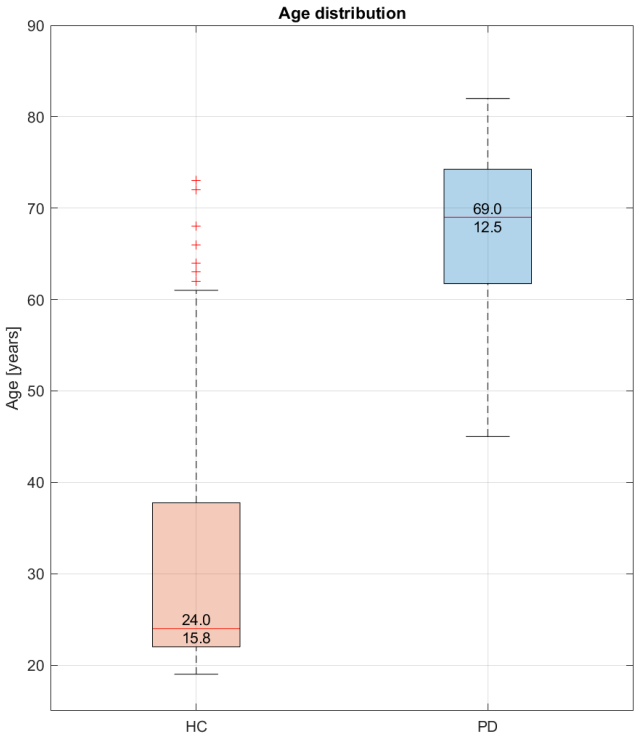

Figure 3.2: Boxplot of the age distributions HC vs PD for the dataset

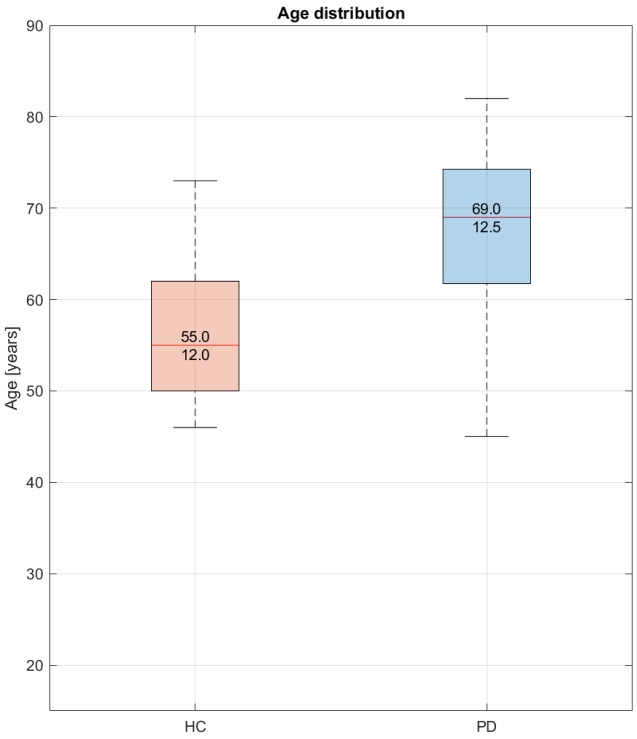

Figure 3.3: Boxplot of the age distributions HC vs PD for the subset after age thresholding

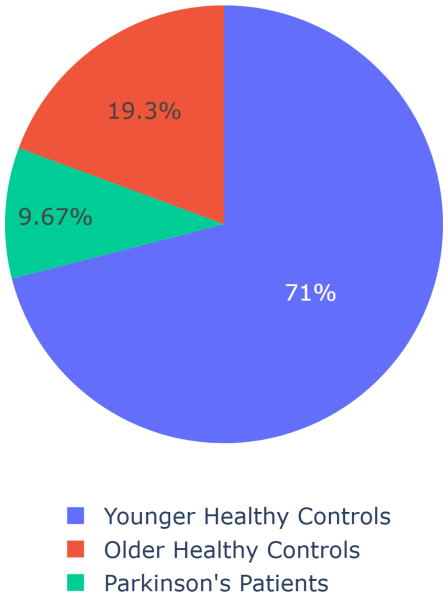

Figure 3.4: Pie chart of the dataset proportions with age grouping

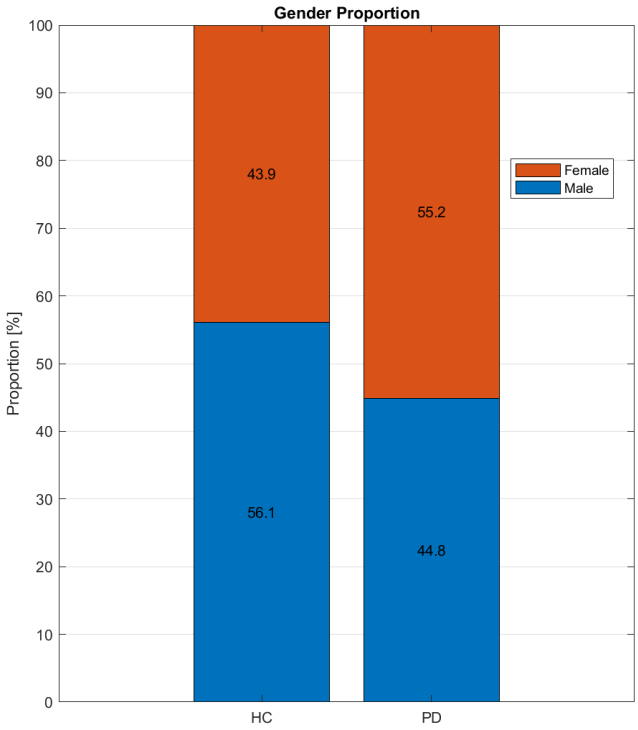

Figure 3.5: Barplot of the gender distributions HC vs PD for the dataset

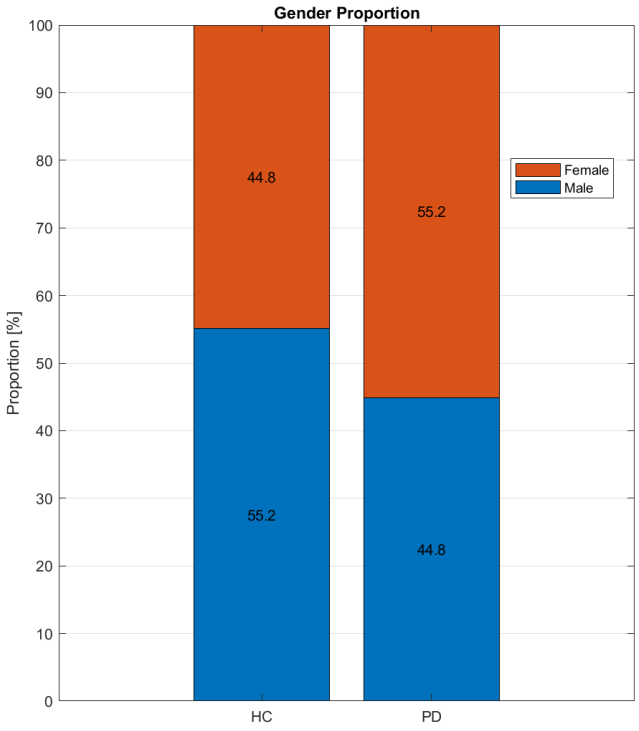

Figure 3.6: Barplot of the gender distributions HC vs PD for the subset after age thresholding

3.1.3 Equipment and Hardware

In this section, the hardware used to record the data is detailed, highlighting the differences between platforms and EEG caps used.

Virtual Reality headset

Visual stimulation was delivered through virtual reality HTC Vive goggles, portrayed in *Figure 3.7*, from the HTC Corporation (Taoyuan, Taiwan), which provided an immersive virtual environment. The system included two external tracking stations for head position monitoring and displayed scenes at a resolution of 1080 × 1200 pixels per eye with a 90 Hz refresh rate.

Figure 3.7: HTC Vive virtual reality headset with the two tracking stations

Motion Platforms

Throughout the dataset two different motion platforms, designed and built by Virtualis (Clapiers, France), were used for the acquisitions of both HCs and PD patients. The first one, referred as the old platform and represented in *Figure 3.7*, used a rigidly connected safety bar to its moving section, the second or new platform, instead, had the safety bar connected to its fixed part making the safety bar static with respect to the surroundings, visible in *Figure 3.8*.

Figure 3.8: Old platform with the moving safety bar

Figure 3.9: New platform with the fixed safety bar

Heart Rate sensor

Heart rate was measured using a chest heart sensor with a 3-lead system from Polar Electro (Kempele, Finland) and a sampling frequency of 1000 Hz.

EEG Caps

Similarly to the platforms, two different EEG caps were used throughout the dataset for both cohorts. Both caps have 64 channels but differ in layout and electrode type. In *Figure 3.9*, the ANT Neuro wet cap is shown; this follows the 10–20 system and uses Cz as the reference electrode. In *Figure 3.10*, the semi-dry cap from the same manufacturer is shown; this employs an equidistant layout with 4z as the reference. Data were recorded with a sampling frequency of 4096 Hz.

Figure 3.10: CA-204-64 ANT Neuro EEG wet cap, 10-20 system

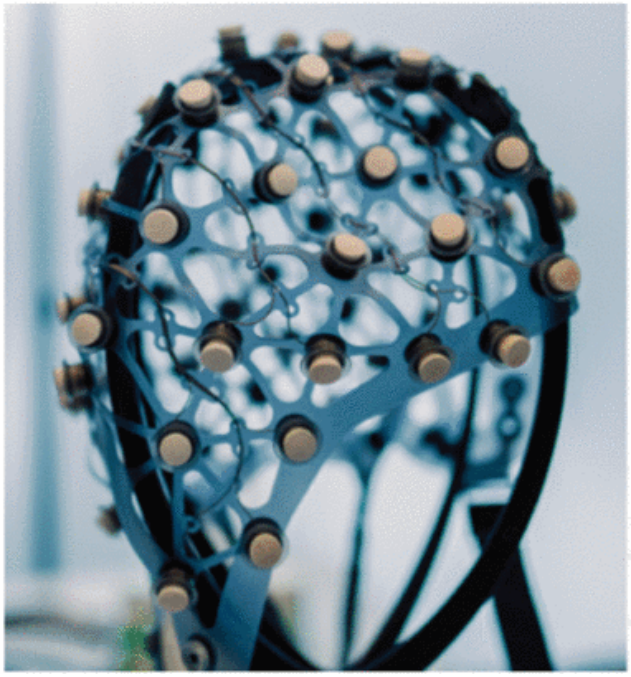

Figure 3.11: NA-261 ANT Neuro EEG semi-dry cap, equidistant system

EMG sensors

Wireless EMG surface electrode sensors, represented in *Figure 3.12*, were used to acquire muscle activity, produced from Kiso (Reykjavik, Iceland), and recorded at a sampling frequency of 1600 Hz.

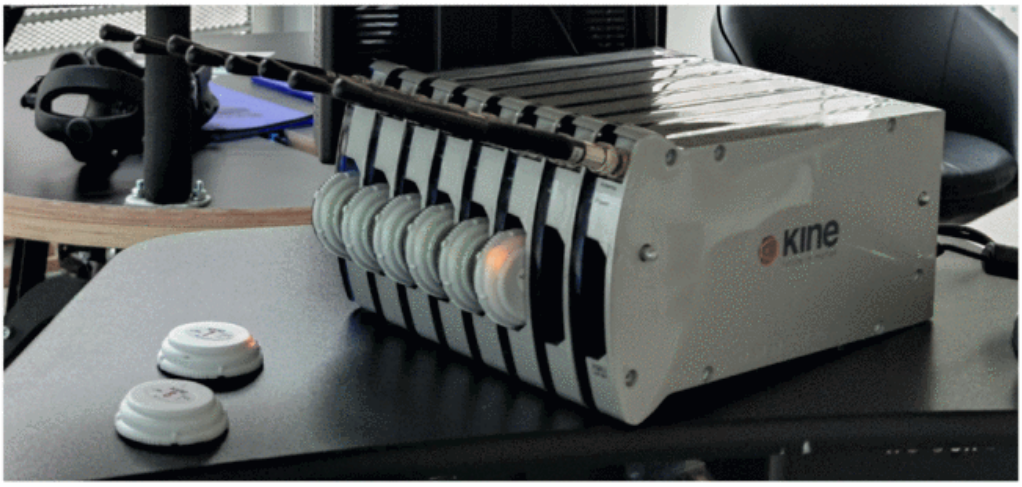

Figure 3.12: EMG surface electrode sensors

CoP force plates

Force plates, provided by the same manufacturer of the platform, are portrayed in *Figure 3.13*. These were used to record Anterior-Posterior (AP) and Medio-Lateral (ML) displacements of the center of pressure, at a sampling frequency of 90 Hz.

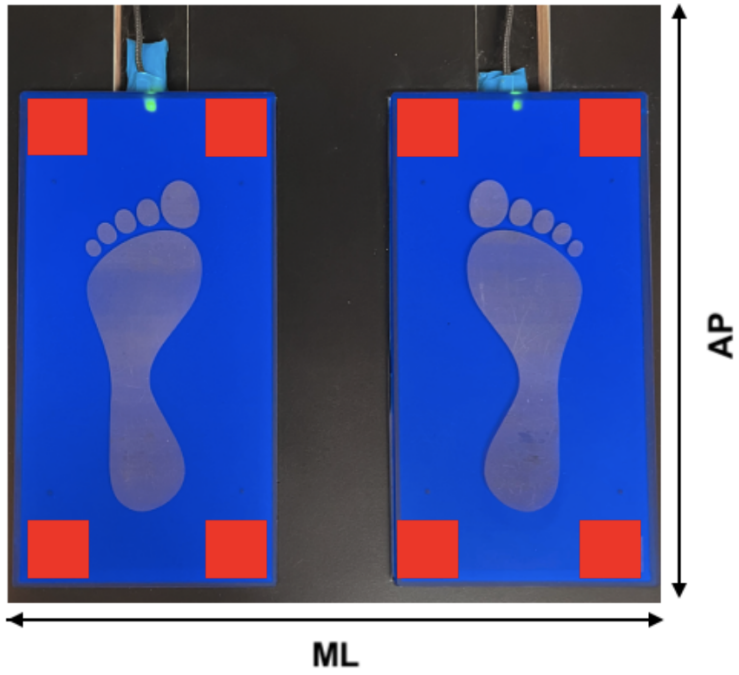

Figure 3.13: CoP force plates with AP and ML directions

Computing Hardware

All deep learning trainings were performed on a computing cluster with up to four Nvidia A100 Graphics Processing Units (GPUs) and 64 GB of available Random Access Memory (RAM).

3.1.4 Experimental Protocol - The BioVRSea Paradigm

The BioVRSea protocol is structured into four sequential phases, Baseline, PRE, MOVE, and POST. Its design aims to simulate maritime balance challenges and assess postural control under varying sensory conditions. Originally developed to identify individuals prone to motion sickness by emulating the experience of navigating at sea on a small boat, the paradigm leverages perturbations at approximately 0.2 Hz, a frequency known to induce motion sickness symptoms [18]. More recently, it has been extended to investigate postural control impairments in patients with Parkinson's Disease, given the strong impact of the disorder on balance regulation. In general, the BioVRSea paradigm enables a systematic evaluation of postural responses to controlled sensory and motor perturbations, allowing the

characterization of adaptation and recovery mechanisms while facilitating direct comparisons between healthy controls and patients with early-stage Parkinson's Disease [4].

Experiment Phases

- *Baseline*: In this initial 60-second phase, participants view a static mountain landscape in the VR environment, represented in *Figure 3.14*. They are instructed not to hold the safety bar. This condition provides a resting state measure of postural stability in the absence of visual or physical perturbations.
- *PRE*: In this phase participants are exposed to visual stimuli representing the motion of the boat within the VR environment, portrayed in *Figure 3.15*. During this 40-second period, subjects are instructed not to hold the safety bar in front of them. This phase establishes a baseline postural response to visual perturbations alone.
- *MOVE*: Lasting 120 seconds, this phase introduces physical and visual perturbations through the motion platform and is synchronized with the VR environment. Participants are allowed to hold the safety bar for support. The MOVE phase is subdivided into three sequential 40-second segments with increasing perturbation amplitudes: 25%, 50%, and 75% of the platform's maximum power. Both visual and platform-induced perturbations oscillate at approximately 0.2 Hz.
- *POST*: This phase mirrors the PRE phase in duration and safety bar rules, but occurs after exposure to the physical perturbations. It is generally used to assess postural adjustments and recovery dynamics after the MOVE phase.

Figure 3.14: View of the mountains during the initialization and Baseline phase of the experiment

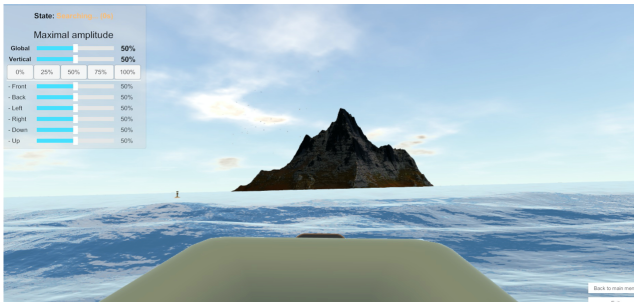

Figure 3.15: View of the sea during the PRE, MOVE, and POST phases of the experiment

Recorded Signals

The heart rate (HR) recording starts at the beginning of the experiment and continues until the end of the acquisition, providing cardiac activity information across the entire timeline. The EEG recording

follows the same timing, offering key insights into brain activity. For the EMG, sensors are placed on each leg over the tibialis anterior, gastrocnemius lateralis, and soleus muscles, with acquisition beginning at the baseline phase to capture lower-limb muscle activity. Finally, the force plates embedded in the platform, each equipped with four sensors, measure anterior–posterior and medio–lateral displacements, with data collection starting at the PRE phase of the protocol.

Platform timelines

In *Subsection 3.1.3*, two different platforms were presented, and the change in platform type came with a modification in the acquisition timeline. Specifically, for the newer device, the transition between the baseline and the PRE phase lasted thirty seconds, as shown in *Figure 3.16*, whereas for the older device it lasted fifteen seconds, as shown in *Figure 3.17*.

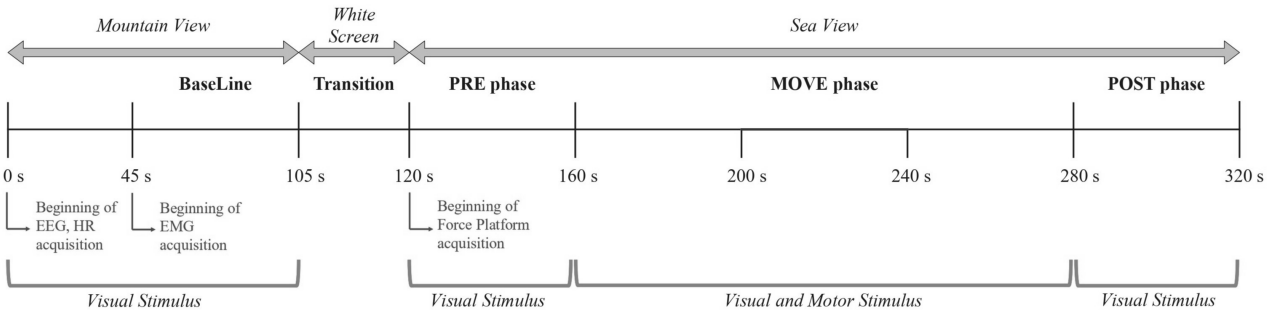

Figure 3.16: Timeline of the old platform

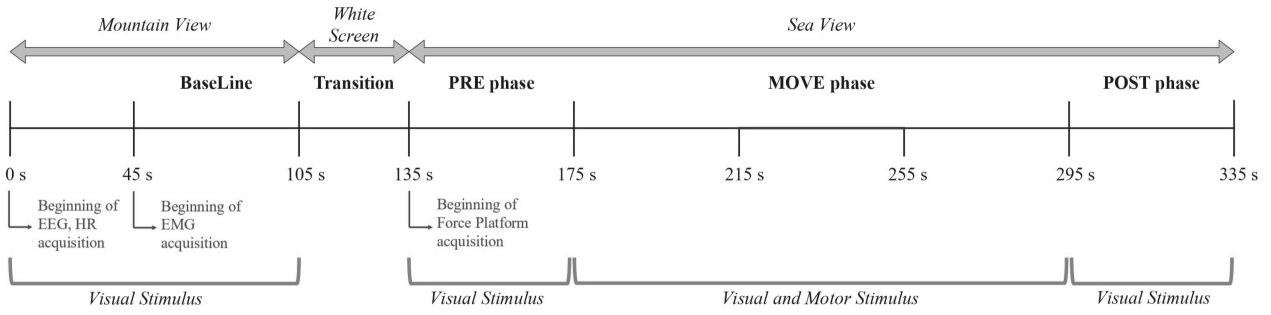

Figure 3.17: Timeline of the new platform

3.2 Preprocessing

Standard preprocessing steps were performed in Matlab©, whereas Python was used for deep learning–specific preprocessing, allowing modification of hyperparameters, if required. During data loading, all signals were split into temporal windows, with signal-specific hyperparameters, to increase the number of examples available for training.

3.2.1 EEG

The preprocessing of EEG signals was performed primarily with the EEGLAB toolbox, according to the results found in [19].

Preprocessing performed in Matlab©:

- *Removal of the EOG channel* (only for the CA-204-64 cap): the electrooculogram (EOG) channel was removed.
- *Resampling to* 128 Hz: from the original 4096 Hz we downsampled the signal to ease the computational load while still retaining spectral information.
- *Cut from beginning of PRE to end of POST*: the signal was cut in this range to have all signals starting from the same instant.
- *Removal of the DC component*: the channel-wise mean of the signal was computed and removed.
- *Line noise removal*: a notch filter at 50 Hz and harmonics was applied to remove noise related to the AC power supply.
- *Bandpass filtering*: to retain only important spectral properties, the signal was filtered in the 1-30 Hz range with a FIR filter. The upper bound of the filter (30 Hz) is due to a relevant number of faulty acquisitions that resulted in damped power spectral density over this threshold.
- *Removal of channels for cap alignment*: from the two caps specific channels were considered and removed to have a more similar distribution, the result of this operation can be visualized in *Figure 3.18* and *Figure 3.19*.
- *Addition and interpolation of channels to 52*: for both caps some channels were added at specific positions to obtain a set of channels of the same number and similar coverage. A visual representation can be found in *Figure 3.20* and *Figure 3.21*.
- *Interpolation from equidistant to 10-20* (only for the NA-261 cap): to finalize the alignment of the two caps, the channels of the NA-261 were spherically interpolated to the locations of the 10-20 reference system.
- *Re-referencing*: the channels were re-referenced to a Common Average Reference (CAR).
- *Independent Component Analysis (ICA)*: an ICA decomposition of the channels was performed, with the 'runica' algorithm, and bad components removed through automatic labeling. The labeling thresholds for the brain class were set between 0 and 0.1, while for the other six artifacts it was set between 0.9 to 1.

Preprocessing performed in Python:

- *Cut in the MOVE phase*: the signal was cut to only retain the MOVE phase.
- *Z-score*: a channel-wise z-score transformation $z_{c,t} = \frac{x_{c,t} - \mu_c}{\sigma_c}$ was applied to standardize channels.
- *Splitting in windows*: recordings were split into non-overlapping segments of length 4 s.

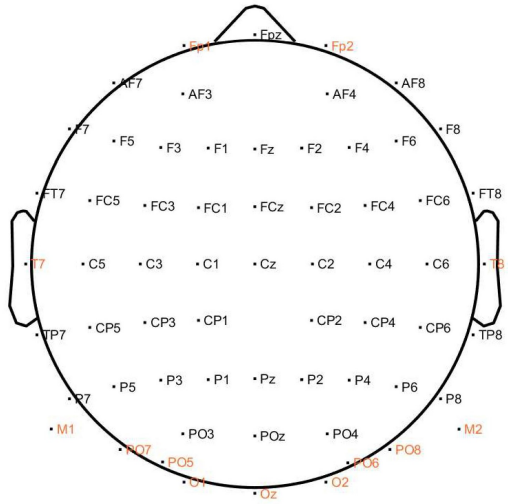

Figure 3.18: CA-204-64 cap, channels removed are highlighted in red

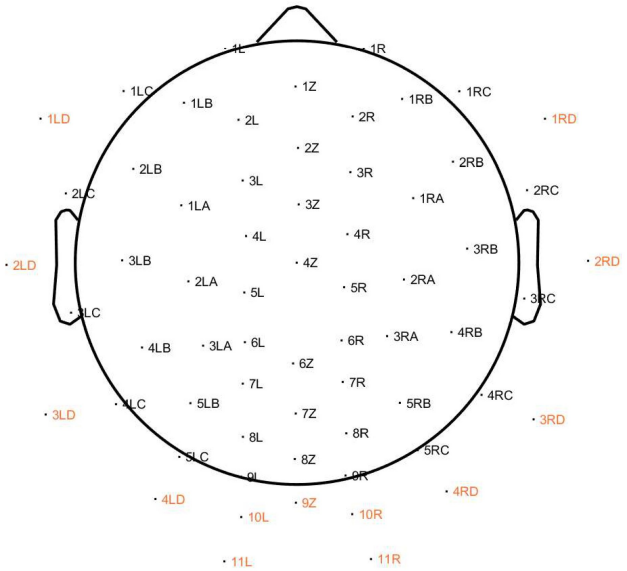

Figure 3.19: NA-261 cap, channels removed are highlighted in red

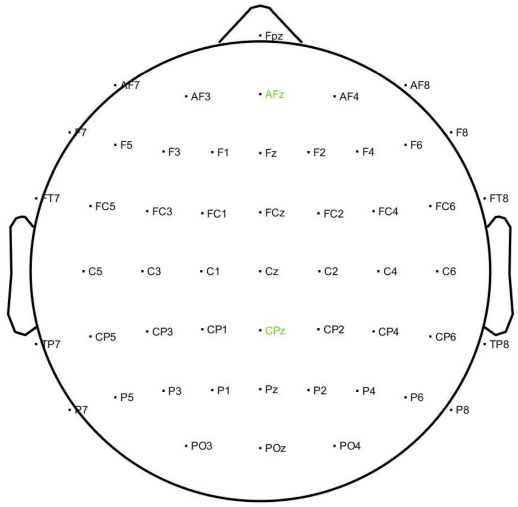

Figure 3.20: CA-204-64 cap, channels added are highlighted in green

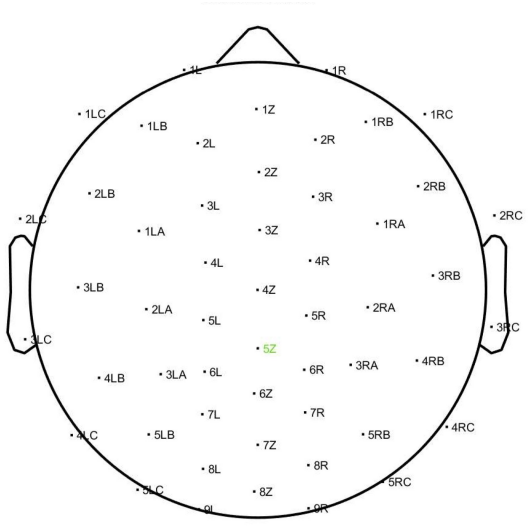

Figure 3.21: NA-261 cap, channels added are highlighted in green

3.2.2 EMG

The preprocessing of EMG signals was performed based on [8, 20]. It comprises applying a non-uniform anti-aliasing downsampling method found in [21].
Preprocessing performed in Matlab©:

- *Cut from beginning of PRE to end of POST*: the signal was cut in this range for consistency with the EEG preprocessing pipeline.
- *Fill missing values*: the missing values were filled with the Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) method. This approach was preferred to spline interpolation, considering its susceptibility to undershooting and overshooting data points.
- *Line noise removal*: a notch filter at 50 Hz and harmonics was applied to remove noise related to the AC power supply.
- *Bandpass filtering*: to retain only important spectral properties, the signal was filtered in the 20-500 Hz range with zero-phase filtering using a Butterworth filter of order 6.
- *Anti-aliasing resampling to* 500 Hz: from the original 1600 Hz we applied a downsampling algorithm explained in *3.1*.

The following set of steps and equations describe the downsampling method used.

Lowpass filtering with cutoff at f_c

$$x_m^{low} = x_m * h_m^{low} \tag{3.1a}$$

Highpass filtering with cutoff at f_c and downshifting by f_{shift}

$$x_m^{high} = \Re\left\{(x_m * h_m^{high})\, e^{-j2\pi f_{\text{shift}} \frac{m}{f_s}}\right\} \tag{3.1b}$$

Resampling of both signals from f_s to f_d

$$x_n^{low} = \mathscr{R}_{f_s \to f_d}\{x_m^{low}\} \tag{3.1c}$$
$$x_n^{high} = \mathscr{R}_{f_s \to f_d}\{x_m^{high}\} \tag{3.1d}$$

Upshifting of the high signal by f_{shift} and reconstruction

$$x_n = x_n^{low} + \Re\left\{x_n^{high}\, e^{j2\pi f_{\text{shift}} \frac{m}{f_d}}\right\} \tag{3.1e}$$

Where:

- f_s is the sampling frequency of the signal (1600 Hz),
- f_d is the downsampling target frequency (500 Hz),
- f_c is the cutoff frequency for both filters (250 Hz)
- f_{shift} is the shifting frequency (250 Hz),
- x_m is the original signal,
- h_m^{low} is a Butterworth lowpass filter, with cutoff at f_c, of order 6, applied with zero-phase filtering,
- x_m^{low} is the signal filtered in the low band,
- h_m^{high} is a Butterworth highpass filter, with cutoff at f_c, of order 6, applied with zero-phase filtering,
- $\Re\{\cdot\}$ is the real part of a complex number,
- x_m^{high} is the signal filtered in the high band downshifted to the low band,
- $\mathscr{R}_{f_s \to f_d}$ is a resampling operation from f_s to f_d with an internal lowpass FIR filter,
- x_n is the final resampled signal.

Preprocessing performed in Python:

- *Cut in the MOVE phase*: the signal was cut to only retain the MOVE phase.
- *Envelope extraction*: the envelope was extracted from the signal with the Root Mean Square (RMS) method by applying a moving average filter with a 50 ms window.
- *Z-score*: a channel-wise z-score transformation was applied.
- *Splitting in windows*: recordings were split into non-overlapping 2 s segments.

3.2.3 CoP

The preprocessing of COP signals was based on [22] for the filtering parameters.
Preprocessing performed in Matlab©:

- *Cut from beginning of PRE to end of POST*: the signal was cut in this range as the other two signals.

- *Fill missing values*: the missing values were filled with the PCHIP method.
- *Lowpass filtering*: the signal was filtered in the 0-7 Hz range with zero-phase filtering using a Butterworth filter of order 6.

Preprocessing performed in Python:

- *Cut in the MOVE phase*: the signal was cut to only retain the MOVE phase.
- *Concatenation of first and second time derivatives*: the first and second numerical time derivatives of the AP and ML displacements are computed and concatenated, resulting in a total of 6 channels. This was proven to be effective in classical machine learning for Parkinson's Disease detection [23].
- *Z-score*: a channel-wise z-score transformation was applied.
- *Splitting in windows*: recordings were split into 4 s segments with 50% of overlap.

3.3 Data Partitioning and Evaluation

In the following subsections, we explain how the hardware differences and age confounder issues found in *Section 3.1* were mitigated, how we retrieved a subject-wise prediction from the windowing approach explained in the previous section, and finally the metrics used to evaluate the performance of the models proposed in *Section 3.5*.

3.3.1 Stratified Nested Cross-Validation

To ensure an unbiased representation of performance and reproducibility of the results, we set the random seed at 42 (the standard value used by the machine learning community) and implemented a 5-inner 5-outer Cross-Validation (CV) at the subject level, shown in *Figure 3.22*. We also ensured that data from the same subject was assigned to a single partition between training, validation or test sets. This was done specifically to avoid training and testing on windows of the same subject, which would otherwise largely inflate performance and not generalize across the population [24]. The total number of trainings for each set of hyperparameters can be computed, with the number of outer folds n_{outer} and the number of inner folds n_{inner}, as $n_{outer} \cdot n_{inner} = 25$.

As highlighted in the previous sections, the age distribution of the subjects was found to be a major issue. To solve this problem, we included in the cross-validation only the older HCs and all the PD patients. Subsequently, we assigned all younger HCs to the training set, ensuring that the validation and test sets did not present any age-related bias.

Stratification of the partitions was applied based on the cohort of subjects and taking into account the difference in the types of EEG caps and platforms, explained in *Subsection 3.1.3*. We stratified the partitions according to the inputs, limiting, as much as possible, the bias introduced by different hardware and protocols across the three sets. We separated subjects according to cap type, if EEG was used as input, or based on platform type, if EMG or CoP were involved. The use of both stratification methods was employed if EEG and EMG and/or CoP were used together at the same time. The modified version of cross-validation used introduced a suboptimal distribution of classes across sets; therefore, Receiver Operating Characteristic (ROC) threshold correction was also applied, described in *Subsection 3.3.2*.

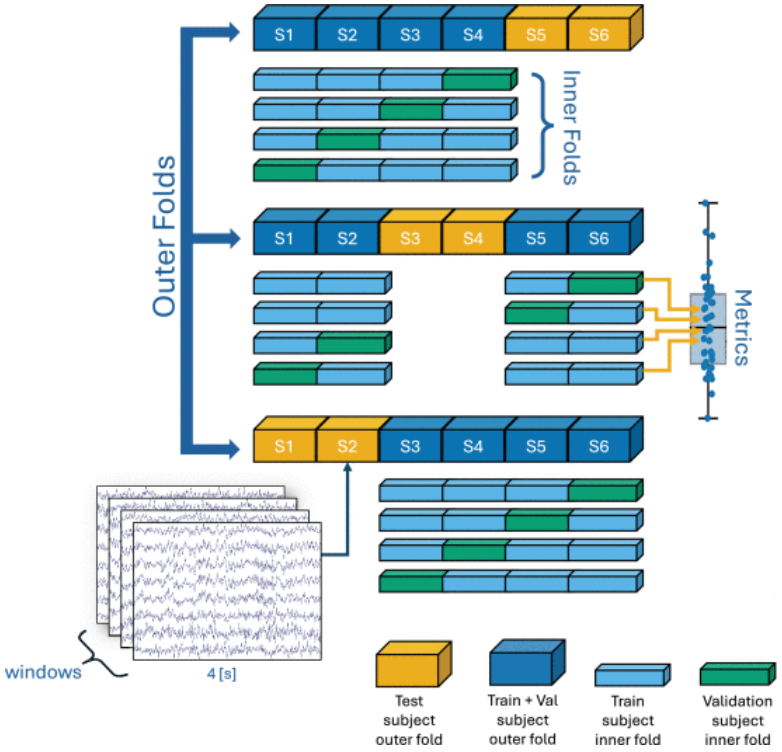

Figure 3.22: Nested Cross-Validation visual representation

3.3.2 Window-level and Subject-level Performance

The evaluation of the model produced predictions on the subjects' windows, which is not clinically relevant and is addressed in this section. As pointed out in the previous subsection, ROC correction was applied by estimating the best threshold for the output probabilities, maximizing balanced accuracy, on the validation set, to avoid data leakage, the same threshold was then used on the outer test set.

Predicting temporal windows was not the objective; therefore, to obtain a subject-level prediction, we estimated a second threshold, optimized for balanced accuracy, that specifies the proportion of windows that must be classified as a patient for the subject to be labeled positive. This threshold was also estimated on the validation set, subsequently to the ROC correction on the windows, and included

a rejection of uncertain predictions by not considering windows predicted with a probability within a margin of ±5% around the 50% mark, improving the robustness of the estimation.

3.3.3 Performance Metrics

To evaluate model performance, we report balanced accuracy, recall, precision, and F1-score, all defined in terms of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right)$$

$$\text{Recall} = \frac{TP}{TP+FN} \qquad \text{Precision} = \frac{TP}{TP+FP}$$

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Balanced accuracy accounts for class imbalance by averaging sensitivity and specificity, while recall and precision capture complementary aspects of positive class prediction. The F1-score summarizes precision and recall as their harmonic mean.

3.4 Data Augmentation

Increasing the sample size, during training, through data augmentation, proved to be essential for the model to effectively generalize its representations and improve performance. We applied the transformations detailed in the following subsections to all signals included in the model input. The hyperparameters used are detailed in *Subsection 3.6.3*.

3.4.1 Signal to Noise Ratio Scaling

This augmentation adds random noise to a signal sample, scaled to reach a specified signal-to-noise ratio (SNR). Given a target SNR, the augmented sample can be defined as:

$$\tilde{x}_{c,t} = x_{c,t} + k\varepsilon_t \tag{3.2}$$

where:

- $\varepsilon_t \sim \mathcal{N}(0,1)$ is Gaussian noise with unitary variance,
- k is a scaling factor calculated as:

$$k = 10^{\left(-\frac{\text{SNR}}{20}\right)} \sqrt{\frac{1}{NC} \sum_{c} \sum_{t} x_{c,t}^2} \tag{3.3}$$

The same noise is added to all channels.

3.4.2 Signal Masking

This augmentation sets portions of the signals to zero. Specifically, given a masking percentage, p, and a predefined number of masking blocks, k, the augmented signal can be defined as:

$$\tilde{\mathrm{X}} = \mathrm{X} \odot (\mathbf{1}_C \mathbf{m}^T) \tag{3.4}$$

where:

- $\odot$ is the Hadamard product, representing the element-wise product of two matrices.
- $\mathbf{1}_C$ is a column vector of ones with length equal to the number of channels C,
- $\mathbf{m}$ is a column mask vector of length equal to the signal length, constructed as a concatenation of blocks of ones and zeros, i.e.,

$$\mathbf{m} = [\mathbf{1}_{b_1} ; \mathbf{0}_{b_2} ; \mathbf{1}_{b_3} ; \mathbf{0}_{b_4} ; \ldots ; \mathbf{0}_{b_{2k}} ; \mathbf{1}_{b_{2k+1}}] \tag{3.5}$$

with:

- each b_i representing the length of the corresponding block,
- $\mathbf{0}_b$ being a column vector of zeros with length b,
- $[\mathbf{x}_1 ; \mathbf{x}_2]$ representing the concatenation of vectors.

The total length of $\mathbf{m}$ is:

$$L = \sum_{i=1}^{2k+1} b_i$$

and the total number of masked samples, corresponding to the sum of the zero blocks, is

$$L \cdot p = \sum_{i=2,\,4,\,6,\ldots,\,2k} b_i$$

The length of each masked portion b_i is randomly generated, while ensuring that the total masked proportion equals the hyperparameter p.

3.4.3 Channel Dropout

This augmentation sets a predefined number of channels to zero. Specifically, given the number of channels to drop, C_d, and a bijective function $\sigma : S \to S$ that defines a random permutation of the elements of a vector, the augmented signal is computed as

$$\tilde{\mathrm{X}} = \mathrm{X} \odot (\mathbf{s}_C \mathbf{1}_N^T) \tag{3.6}$$

where

$$\mathbf{s}_C = \sigma([\mathbf{0}_{C_d}; \mathbf{1}_{C-C_d}]) \tag{3.7}$$

In this context:

- The permutation σ shuffles the elements of the concatenated $\mathbf{0}_{C_d}$ and $\mathbf{1}_{C-C_d}$.
- $\mathbf{s}_C$ is the random column vector composed of zeros and ones, produced by the σ function.
- C_d is the number of channels dropped

3.4.4 Augmentation composition

We leveraged these augmentations by applying signal masking or channel dropout with a 50% chance each and always applied SNR ratio scaling sequentially. This augmentation setup, aimed to increase model robustness across both channels and time samples, with channel dropout and signal masking respectively, and included SNR ratio scaling for the benefits of sequential augmentations found in [15].

3.5 Model Architectures

In this section, we discuss the signal-specific models and how they were adapted for the implementation of the proposed fusion models, highlighting how multiple streams of data were handled. The models are described using tables and diagrams that summarize the number of parameters, the input and output shapes of the tensor for each layer, and the relevant hyperparameters used.

3.5.1 Signal-specific Models

For each signal, a specific neural network was chosen based on its ability to extract information from the specific type of data. When having only a single signal as input, the base models, taken from the scientific literature, were used. These neural networks were then slightly modified to maintain an embedded temporal dimension, turning them into tokenizers, instead of collapsing the extracted representation to a set of features. The use of both modified and base signal models for fusion methods is explored in the next subsection.

EEG - ShallowNet

For EEG signals, the fully convolutional model proposed in [13] was chosen. A summary table of this model is shown in *Table 3.2*. To retain the temporal dimension, we modified the kernel size and stride of the average pooling layer, dividing both by 3, and avoid flattening the tensor. The tokenizer version of ShallowNet is shown in *Table 3.3*.

Layer (type)	Input Shape	Output Shape	Kernel Shape	Stride	Param#
ShallowNet	[N, 52, 512]	[N, 1]	–	–	21 921
+ShallowNet Encoder	[N, 52, 512]	[N, 540]	–	–	21 380
| +Unsqueeze	[N, 52, 512]	[N, 1, 52, 512]	–	–	–
| +Conv2d	[N, 1, 52, 512]	[N, 20, 52, 488]	[1, 25]	[1, 1]	520
| +Conv2d	[N, 20, 52, 488]	[N, 20, 1, 488]	[52, 1]	[1, 1]	20 820
| +BatchNorm2d	[N, 20, 1, 488]	[N, 20, 1, 488]	–	–	40
| +Square	[N, 20, 1, 488]	[N, 20, 1, 488]	–	–	–
| +AvgPool2d	[N, 20, 1, 488]	[N, 20, 1, 27]	[1, 90]	[1, 15]	–
| +Log	[N, 20, 1, 27]	[N, 20, 1, 27]	–	–	–
| +Dropout (0.3)	[N, 20, 1, 27]	[N, 20, 1, 27]	–	–	–
| +Flatten	[N, 20, 1, 27]	[N, 540]	–	–	–
+Linear	[N, 540]	[N, 1]	–	–	541

Table 3.2: Summary Table of ShallowNet

Layer (type)	Input Shape	Output Shape	Kernel Shape	Stride	Param#
ShallowNet Tokenizer	[N, 52, 512]	[N, 20, 92]	–	–	21 380
+Unsqueeze	[N, 52, 512]	[N, 1, 52, 512]	–	–	–
+Conv2d	[N, 1, 52, 512]	[N, 20, 52, 488]	[1, 25]	[1, 1]	520
+Conv2d	[N, 20, 52, 488]	[N, 20, 1, 488]	[52, 1]	[1, 1]	20 820
+BatchNorm2d	[N, 20, 1, 488]	[N, 20, 1, 488]	–	–	40
+Square	[N, 20, 1, 488]	[N, 20, 1, 488]	–	–	–
+AvgPool2d	[N, 20, 1, 488]	[N, 20, 1, 92]	[1, 30]	[1, 5]	–
+Log	[N, 20, 1, 92]	[N, 20, 1, 92]	–	–	–
+Squeeze	[N, 20, 1, 92]	[N, 20, 92]	–	–	–

Table 3.3: Summary Table of ShallowNet Tokenizer

EMG - XiaNet

To analyze EMG signals we used the model proposed in [25], referred as XiaNet. Despite the original work applied it to human activity recognition, leveraging Inertial Measurement Units (IMUs) data, the potential for applications of hybrid LSTM-CNN networks to EMG is highlighted in [5]. A model summary can be found in *Table 3.4*. To retain a temporal dimension we removed the last part of the network, after the global average pooling, and multiplied by 5 both kernel size and stride of the last convolutional layer, the result of these modifications can be found in *Table 3.5*.

Layer (type)	Input Shape	Output Shape	Parameter	Value	Param#
XiaNet	[N, 6, 1000]	[N, 1]	–	–	27 169
+XiaNet Encoder	[N, 6, 1000]	[N, 64]	–	–	25 056
| +Permute	[N, 6, 1000]	[N, 1000, 6]	–	–	–
| +LSTM	[N, 1000, 6]	[N, 1000, 32]	Hidden Size	32	13 568
| |			Layer Number	2	
| |			Bidirectional	False	
| +Permute	[N, 1000, 32]	[N, 32, 1000]	–	–	–
| +Unsqueeze	[N, 32, 1000]	[N, 32, 1000, 1]	–	–	–
| +Conv2d	[N, 32, 1000, 1]	[N, 32, 498, 1]	Kernel	[5, 1]	5 152
| |			Stride	[2, 1]	
| +Squeeze	[N, 32, 498, 1]	[N, 32, 498]	–	–	–
| +MaxPool1d	[N, 32, 498]	[N, 32, 249]	Kernel	2	–
| |			Stride	2	
| +Unsqueeze	[N, 32, 249]	[N, 32, 249, 1]	–	–	–
| +Conv2d	[N, 32, 249, 1]	[N, 64, 247, 1]	Kernel	[3, 1]	6 208
| |			Stride	[1, 1]	
| +Squeeze	[N, 64, 247, 1]	[N, 64, 247]	–	–	–
| +AdaptiveAvgPool1d	[N, 64, 247]	[N, 64, 1]	–	–	–
| +Squeeze	[N, 64, 1]	[N, 64]	–	–	–
| +BatchNorm1d	[N, 64]	[N, 64]	–	–	128
+Head	[N, 64]	[N, 1]	–	–	2 113
+Linear	[N, 64]	[N, 32]	–	–	2 080
+LeakyReLU	[N, 32]	[N, 32]	–	–	–
+Linear	[N, 32]	[N, 1]	–	–	33

Table 3.4: Summary Table of XiaNet

CoP - ResNet1D

Since no established neural network was found for CoP analysis, a 1D version of ResNet was chosen, referred as ResNet1D, from [26]. A summary of its structure is found in *Table 3.6*. For this architecture to extract a tokenized representation, we removed the global average pooling layer and reduced the stride of convolutions, for both Layer3 and Layer4, to unitary, meaning that no reduction of the last dimension of the tensor occurs after Layer2, a summary of this version is found in *Table 3.7*.

Layer (type)	Input Shape	Output Shape	Parameter	Value	Param#
XiaNet Tokenizer	[N, 6, 1000]	[N, 64, 47]	–	–	49 632
+Permute	[N, 6, 1000]	[N, 1000, 6]	–	–	–
+LSTM	[N, 1000, 6]	[N, 1000, 32]	Hidden Size	32	13 568
|			Layer Number	2	
|			Bidirectional	False	
+Permute	[N, 1000, 32]	[N, 32, 1000]	–	–	–
+Unsqueeze	[N, 32, 1000]	[N, 32, 1000, 1]	–	–	–
+Conv2d	[N, 32, 1000, 1]	[N, 32, 498, 1]	Kernel	[5, 1]	5 152
|			Stride	[2, 1]	
+Squeeze	[N, 32, 498, 1]	[N, 32, 498]	–	–	–
+MaxPool1d	[N, 32, 498]	[N, 32, 249]	Kernel	2	–
|			Stride	2	
+Unsqueeze	[N, 32, 249]	[N, 32, 249, 1]	–	–	–
+Conv2d	[N, 32, 249, 1]	[N, 64, 47, 1]	Kernel	[15, 1]	30 784
|			Stride	[5, 1]	
+Squeeze	[N, 64, 47, 1]	[N, 64, 47]	–	–	–

Table 3.5: Summary Table of XiaNet Tokenizer

Layer (type)	Input Shape	Output Shape	Kernel Shape	Stride	Param#
ResNet1D	[N, 6, 360]	[N, 1]	–	–	81 721
+ResNet1D Encoder	[N, 6, 360]	[N, 48]	–	–	80 520
| +Unsqueeze	[N, 6, 360]	[N, 1, 6, 360]	–	–	–
| +Conv2d	[N, 1, 6, 360]	[N, 8, 6, 180]	[1, 7]	[1, 2]	56
| +BatchNorm2d	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	16
| +ReLU	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	–
| +Layer1	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	928
| | +Skip	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	–
| | +Conv2d	[N, 8, 6, 180]	[N, 8, 6, 180]	[1, 7]	[1, 1]	448
| | +BatchNorm2d	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	16
| | +ReLU	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	–
| | +Conv2d	[N, 8, 6, 180]	[N, 8, 6, 180]	[1, 7]	[1, 1]	448
| | +BatchNorm2d	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	16
| | +ReLU	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	–
| +Layer2	[N, 8, 6, 180]	[N, 16, 6, 90]	–	–	3 680
| | +Skip (+ Downsample)	[N, 8, 6, 180]	[N, 16, 6, 90]	–	–	928
| | | +Conv2d	[N, 8, 6, 180]	[N, 16, 6, 90]	[1, 7]	[1, 2]	896
| | | +BatchNorm2d	[N, 16, 6, 90]	[N, 16, 6, 90]	–	–	32
| | +Conv2d	[N, 8, 6, 180]	[N, 16, 6, 90]	[1, 7]	[1, 2]	896
| | +BatchNorm2d	[N, 16, 6, 90]	[N, 16, 6, 90]	–	–	32
| | +ReLU	[N, 16, 6, 90]	[N, 16, 6, 90]	–	–	–
| | +Conv2d	[N, 16, 6, 90]	[N, 16, 6, 90]	[1, 7]	[1, 1]	1 792
| | +BatchNorm2d	[N, 16, 6, 90]	[N, 16, 6, 90]	–	–	32
| | +ReLU	[N, 16, 6, 90]	[N, 16, 6, 90]	–	–	–
| +Layer3	[N, 16, 6, 90]	[N, 32, 6, 45]	–	–	14 528
| | +Skip (+ Downsample)	[N, 16, 6, 90]	[N, 32, 6, 45]	–	–	3 648
| | | +Conv2d	[N, 16, 6, 90]	[N, 32, 6, 45]	[1, 7]	[1, 2]	3 584
| | | +BatchNorm2d	[N, 32, 6, 45]	[N, 32, 6, 45]	–	–	64
| | +Conv2d	[N, 16, 6, 90]	[N, 32, 6, 45]	[1, 7]	[1, 2]	3 584
| | +BatchNorm2d	[N, 32, 6, 45]	[N, 32, 6, 45]	–	–	64
| | +ReLU	[N, 32, 6, 45]	[N, 32, 6, 45]	–	–	–
| | +Conv2d	[N, 32, 6, 45]	[N, 32, 6, 45]	[1, 7]	[1, 1]	7 168
| | +BatchNorm2d	[N, 32, 6, 45]	[N, 32, 6, 45]	–	–	64
| | +ReLU	[N, 32, 6, 45]	[N, 32, 6, 45]	–	–	–
| +Layer4	[N, 32, 6, 45]	[N, 64, 6, 23]	–	–	57 728
| | +Skip (+ Downsample)	[N, 32, 6, 45]	[N, 64, 6, 23]	–	–	14 464
| | | +Conv2d	[N, 32, 6, 45]	[N, 64, 6, 23]	[1, 7]	[1, 2]	14 336
| | | +BatchNorm2d	[N, 64, 6, 23]	[N, 64, 6, 23]	–	–	128
| | +Conv2d	[N, 32, 6, 45]	[N, 64, 6, 23]	[1, 7]	[1, 2]	14 336
| | +BatchNorm2d	[N, 64, 6, 23]	[N, 64, 6, 23]	–	–	128
| | +ReLU	[N, 64, 6, 23]	[N, 64, 6, 23]	–	–	–
| | +Conv2d	[N, 64, 6, 23]	[N, 64, 6, 23]	[1, 7]	[1, 1]	28 672
| | +BatchNorm2d	[N, 64, 6, 23]	[N, 64, 6, 23]	–	–	128
| | +ReLU	[N, 64, 6, 23]	[N, 64, 6, 23]	–	–	–
| +Conv2d	[N, 64, 6, 23]	[N, 8, 6, 17]	[1, 7]	[1, 1]	3 584
| +AdaptiveAvgPool2d	[N, 8, 6, 17]	[N, 8, 6, 1]	–	–	–
| +Reshape	[N, 8, 6, 1]	[N, 48]	–	–	–
+Head	[N, 48]	[N, 1]	–	–	1 201
+Linear	[N, 48]	[N, 24]	–	–	1 176
+LeakyReLU	[N, 24]	[N, 24]	–	–	–
+Linear	[N, 24]	[N, 1]	–	–	25

Table 3.6: Summary Table of ResNet1D, all four residual layers include a skip connection from input to output, when needed, a downsampling operation is applied to the input and summed before the final ReLU

Layer (type)	Input Shape	Output Shape	Kernel Shape	Stride	Param#
ResNet1D Tokenizer	[N, 6, 360]	[N, 48, 90]	–	–	80 520
+Unsqueeze	[N, 6, 360]	[N, 1, 6, 360]	–	–	–
+Conv2d	[N, 1, 6, 360]	[N, 8, 6, 180]	[1, 7]	[1, 2]	56
+BatchNorm2d	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	16
+ReLU	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	–
+Layer1	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	928
| +Skip	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	–
| +Conv2d	[N, 8, 6, 180]	[N, 8, 6, 180]	[1, 7]	[1, 1]	448
| +BatchNorm2d	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	16
| +ReLU	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	–
| +Conv2d	[N, 8, 6, 180]	[N, 8, 6, 180]	[1, 7]	[1, 1]	448
| +BatchNorm2d	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	16
| +ReLU	[N, 8, 6, 180]	[N, 8, 6, 180]	–	–	–
+Layer2	[N, 8, 6, 180]	[N, 16, 6, 90]	–	–	3 680
| +Skip (+ Downsample)	[N, 8, 6, 180]	[N, 16, 6, 90]	–	–	928
| | +Conv2d	[N, 8, 6, 180]	[N, 16, 6, 90]	[1, 7]	[1, 2]	896
| | +BatchNorm2d	[N, 16, 6, 90]	[N, 16, 6, 90]	–	–	32
| +Conv2d	[N, 8, 6, 180]	[N, 16, 6, 90]	[1, 7]	[1, 2]	896
| +BatchNorm2d	[N, 16, 6, 90]	[N, 16, 6, 90]	–	–	32
| +ReLU	[N, 16, 6, 90]	[N, 16, 6, 90]	–	–	–
| +Conv2d	[N, 16, 6, 90]	[N, 16, 6, 90]	[1, 7]	[1, 1]	1 792
| +BatchNorm2d	[N, 16, 6, 90]	[N, 16, 6, 90]	–	–	32
| +ReLU	[N, 16, 6, 90]	[N, 16, 6, 90]	–	–	–
+Layer3	[N, 16, 6, 90]	[N, 32, 6, 90]	–	–	14 528
| +Skip	[N, 16, 6, 90]	[N, 32, 6, 90]	–	–	3 648
| | +Conv2d	[N, 16, 6, 90]	[N, 32, 6, 90]	[1, 7]	[1, 1]	3 584
| | +BatchNorm2d	[N, 32, 6, 90]	[N, 32, 6, 90]	–	–	64
| +Conv2d	[N, 16, 6, 90]	[N, 32, 6, 90]	[1, 7]	[1, 1]	3 584
| +BatchNorm2d	[N, 32, 6, 90]	[N, 32, 6, 90]	–	–	64
| +ReLU	[N, 32, 6, 90]	[N, 32, 6, 90]	–	–	–
| +Conv2d	[N, 32, 6, 90]	[N, 32, 6, 90]	[1, 7]	[1, 1]	7 168
| +BatchNorm2d	[N, 32, 6, 90]	[N, 32, 6, 90]	–	–	64
| +ReLU	[N, 32, 6, 90]	[N, 32, 6, 90]	–	–	–
+Layer4	[N, 32, 6, 90]	[N, 64, 6, 90]	–	–	57 728
| +Skip	[N, 32, 6, 90]	[N, 64, 6, 90]	–	–	14 464
| | +Conv2d	[N, 32, 6, 90]	[N, 64, 6, 90]	[1, 7]	[1, 1]	14 336
| | +BatchNorm2d	[N, 64, 6, 90]	[N, 64, 6, 90]	–	–	128
| +Conv2d	[N, 32, 6, 90]	[N, 64, 6, 90]	[1, 7]	[1, 1]	14 336
| +BatchNorm2d	[N, 64, 6, 90]	[N, 64, 6, 90]	–	–	128
| +ReLU	[N, 64, 6, 90]	[N, 64, 6, 90]	–	–	–
| +Conv2d	[N, 64, 6, 90]	[N, 64, 6, 90]	[1, 7]	[1, 1]	28 672
| +BatchNorm2d	[N, 64, 6, 90]	[N, 64, 6, 90]	–	–	128
| +ReLU	[N, 64, 6, 90]	[N, 64, 6, 90]	–	–	–
+Conv2d	[N, 64, 6, 90]	[N, 8, 6, 90]	[1, 7]	[1, 1]	3 584
+Reshape	[N, 8, 6, 90]	[N, 48, 90]	–	–	–

Table 3.7: Summary Table of ResNet1D tokenizer, all four residual layers include a skip connection from input to output, when needed, a downsampling operation is applied to the input and summed before the final ReLU

3.5.2 Fusion Models

The fusion of signals was performed using either the encoder part of the base neural network or the tokenizer version of it. For both cases, we established and used an easily scalable framework capable of handling signals that were split with different window sizes and with or without overlap. Starting from the idea that signal windows must be temporally coherent during fusion, we established two new global hyperparameters, the global batchsize, N, and the global window, W.

We start by computing the stride of each signal with the following formula:

$$S_i = W_i \cdot (1 - O_i) \tag{3.8}$$

Where:

- S_i is the stride of signal i, in seconds,
- W_i is the window of signal i, in seconds,
- $O_i \in (0, 1)$ is the overlap between windows of signal i.

We then compute how many sub-windows, M_i, are needed to retrieve enough temporal information to span the global time window as:

$$M_i = \frac{W}{S_i} \tag{3.9}$$

Finally, we compute the batch size to use for each signal-wise neural network, N_i, as:

$$N_i = M_i \cdot N = \frac{WN}{W_i(1 - O_i)} \tag{3.10}$$

To ensure the mathematical and practical validity of the batch size computation, we constrain the values of W_i and O_i such that:

$$M_i = \frac{W}{W_i(1 - O_i)} \in \mathbb{Z}^+ \tag{3.11}$$

This ensures that $N_i = M_i \cdot N$ is also a positive integer, and specifically a multiple of the global batch size N.

Given the number of sub-windows for each signal, we ensure that M_i subsequent signal windows are kept close together in the batchsize dimension of the input tensors. Thus, shuffling is performed on a global window basis.

In the following models, the softmax function was also used to normalize fusion weights (or logits) across signals to $(0, 1)$:

$$\text{softmax}(z_i; \tau) = \frac{\exp(z_i/\tau)}{\sum_{j=1}^{K} \exp(z_j/\tau)}, \quad i = 1, \dots, K \tag{3.12}$$

With:

- z_i are the logits to normalize,
- τ is the temperature parameter, used to control if the distribution should be more uniform or selective.

To balance the small initialization values of the fusion logits, drawn from $\mathcal{N}(0, 0.01)$, chosen due to the small gradients observed, we set a temperature value matching the standard deviation of the distribution. Therefore, we used a standard softmax function to normalize fusion weights.

Concatenated Fusion - StackNet

For concatenated fusion, we removed the classifier head from the base models, for ShallowNet the final linear projection, for XiaNet and ResNet1D the Head block, retaining only the encoders of each network. We then projected the output features, through a learned linear projection, to a common dimension to prevent a signal from dominating the final representation. Finally, we applied a softmax learnable weight for each set of features and employed branch dropout before concatenating along the feature dimension. To better understand the flow of gradients, we show a diagram of StackNet in *Figure 3.23*; note that the "W" block contains both branch dropout and softmax weighting and that the batchsize for the single signals is related to the global one with the relation found in *Equation 3.10*. The reshape block concatenates, in the second dimension, enough features to span the global time window. In *Table 3.8* a summary of the model can be found.

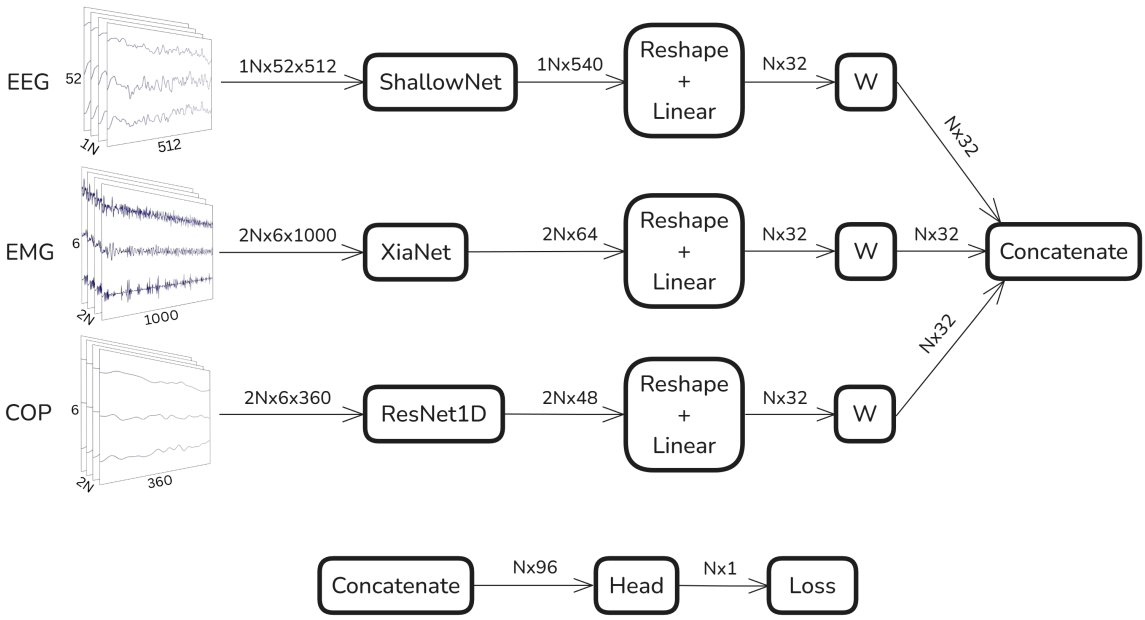

Figure 3.23: Network diagram of StackNet with all signals

Layer (type)	Input Shape	Output Shape	Param#
	[1N, 52, 512]		
StackNet	[2N, 6, 1000]	[N, 1]	156 208
|	[2N, 6, 360]		
|			
+ShallowNet Encoder	[1N, 52, 512]	[1N, 540]	21 380
+XiaNet Encoder	[2N, 6, 1000]	[2N, 64]	25 056
+ResNet1D Encoder	[2N, 6, 360]	[2N, 48]	80 520
|			
|	[1N, 540]	[N, 540]	–
+Reshape	[2N, 64]	[N, 128]	–
|	[2N, 48]	[N, 96]	–
|			
|	[N, 540]	[N, 32]	17 312
+Linear	[N, 128]	[N, 32]	4 128
|	[N, 96]	[N, 32]	3 104
|			
|	[N, 32]	[N, 32]	–
+Branch Dropout (0.1)	[N, 32]	[N, 32]	–
|	[N, 32]	[N, 32]	–
|			
|	[N, 32]	[N, 32]	1
+Softmax weighting	[N, 32]	[N, 32]	1
|	[N, 32]	[N, 32]	1
|			
|	[N, 32]		
+Concatenation	[N, 32]	[N, 96]	–
|	[N, 32]		
|			
+Head	[N, 96]	[N, 1]	4 705
+Linear	[N, 96]	[N, 48]	4 656
+LeakyReLU	[N, 48]	[N, 48]	–
+Linear	[N, 48]	[N, 1]	49

Table 3.8: Summary Table of StackNet with all signals

Embedding Fusion - TokenSumNet

The embedding fusion leveraged the tokenizer versions of the signal NNs. A similar process to StackNet was applied in order to have compatible representations. In addition to the reshape and linear projection, a pooling layer was added to ensure alignment to the minimum sample dimension among the three signals. The branch dropout and softmax weighting of each representation was also applied, in four different ways, weighting: signals, channels, samples, and both channels and samples. After this layer, instead of concatenation, we summed the weighted representations into a single one and applied a transformer architecture before a Global Average Pooling (GAP) and classification head. The reshape operation differs with respect to the one implemented in StackNet, in this case, we concatenated along the last dimension, representing time, if no overlap was applied during preprocessing, or concatenated by averaging the overlapping samples between windows, if instead overlap was present. A diagram of this network can be found in *Figure 3.24* and a summary in *Table 3.9*.

Layer (type)	Input Shape	Output Shape	Parameter	Value	Param#
	[1N, 52, 512]				
TokenSumNet	[2N, 6, 1000]	[N, 1]	–	–	166 631[a]
|	[2N, 6, 360]				
|					
+ShallowNet Tokenizer	[1N, 52, 512]	[1N, 20, 92]	–	–	21 380
+XiaNet Tokenizer	[2N, 6, 1000]	[2N, 64, 47]	–	–	49 632
+ResNet1D Tokenizer	[2N, 6, 360]	[2N, 48, 90]	–	–	80 520
|					
|	[1N, 20, 92]	[N, 20, 92]			–
+Reshape	[2N, 64, 47]	[N, 64, 94]	Average overlap	True	–
|	[2N, 48, 90]	[N, 48, 90]			–
|					
+AdaptiveMaxPool	[N, 20, 92]	[N, 20, 90]	–	–	–
+AdaptiveMaxPool	[N, 64, 94]	[N, 64, 90]	–	–	–
+AdaptiveAvgPool	[N, 48, 90]	[N, 48, 90]	–	–	–
|					
|	[N, 20, 90]	[N, 90, 20]			–
+Permute	[N, 64, 90]	[N, 90, 64]	–	–	–
|	[N, 48, 90]	[N, 90, 48]			–
|					
|	[N, 90, 20]	[N, 90, 32]			672
+Linear	[N, 90, 64]	[N, 90, 32]	–	–	2 080
|	[N, 90, 48]	[N, 90, 32]			1 568

Layer (type)	Input Shape	Output Shape	Parameter	Value	Param#
|	[N, 90, 32]	[N, 32, 90]			–
+Permute	[N, 90, 32]	[N, 32, 90]	–	–	–
|	[N, 90, 32]	[N, 32, 90]			–
|					
|	[N, 32, 90]	[N, 32, 90]			–
+L2 Norm (3rd dim)	[N, 32, 90]	[N, 32, 90]	–	–	–
|	[N, 32, 90]	[N, 32, 90]			–
|					
|	[N, 32, 90]	[N, 32, 90]			–
+Branch Dropout (0.1)	[N, 32, 90]	[N, 32, 90]	–	–	–
|	[N, 32, 90]	[N, 32, 90]			–
|					
|	[N, 32, 90]	[N, 32, 90]			w^b
+Softmax weighting	[N, 32, 90]	[N, 32, 90]	–	–	w^b
|	[N, 32, 90]	[N, 32, 90]			w^b
|					
|	[N, 32, 90]				
+Sum	[N, 32, 90]	[N, 32, 90]	–	–	–
|	[N, 32, 90]				
|					
+Permute	[N, 32, 90]	[N, 90, 32]	–	–	–
+Transformer Encoder (+Pos Encoding)	[N, 90, 32]	[N, 90, 32]	Model Size	32	10 234
|			Heads	1	
|			Feedforward	90	
|			Dropout	0.3	
|			Activation	Hardswish	
+Permute	[N, 90, 32]	[N, 32, 90]	–	–	–
+AdaptiveAvgPool	[N, 32, 90]	[N, 32, 1]	–	–	–
+Squeeze	[N, 32, 1]	[N, 32]	–	–	–
+L2 Norm (2nd dim)	[N, 32]	[N, 32]	–	–	–
+Head	[N, 32]	[N, 1]	–	–	545
+Linear	[N, 32]	[N, 16]	–	–	528
+LeakyReLU	[N, 16]	[N, 16]	–	–	–
+Linear	[N, 16]	[N, 1]	–	–	17

Table 3.9: Summary Table of TokenSumNet with all signals

[a] The parameter count is missing the softmax weights.

[b] The parameter count changes based on how we apply tensor weights, $w \in [1, 32, 90, 2880]$, for signal, channel, samples and both respectively.

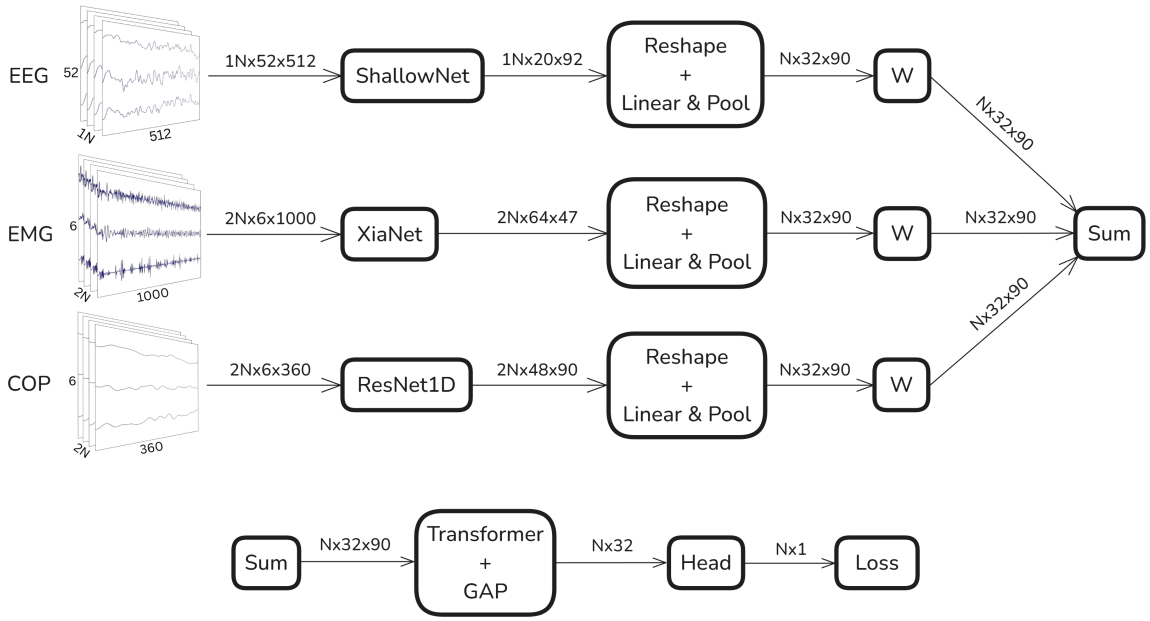

Figure 3.24: Network diagram of TokenSumNet with all signals

3.6 Training and Hyperparameters

In this section, we explore the learning paradigms used to train the aforementioned models and report a complete list of training and augmentation hyperparameters. For both learning paradigms, we used ADAM optimizer with an exponential learning rate scheduler and leveraged an early stopping criterion, based on validation set loss, to understand when the model started to overfit and stop training. We then restored the model weights corresponding to the minimum value, across epochs, of the validation loss.

3.6.1 Supervised Learning

Supervised learning is the classical approach used in the scientific literature to train DL models that process temporal windows instead of the entire signal. When splitting a subject's signal, the label is generally distributed across the windows, meaning that if one belongs to a healthy control, it will be labeled as such. This approach comes with a trade-off, we are able to largely increase dataset size, extremely useful in DL, but we might introduce label noise. The latter happens when, for example, a window labeled as belonging to a PD presents only healthy control patterns.

We used this approach with some specific implementation details. Taking into account the great imbalance between classes, a ratio of 9:1 for healthy controls to PD patients, an adjustment was made to the loss function to force the model into recognizing differences between classes and not just classify all as healthy. We used the Binary Cross-Entropy (BCE) loss function, and set the positive weight, α, to the ratio between the number of healthy controls and the number of patients.

$$\alpha = \frac{n_{HC}}{n_{PD}} \tag{3.13}$$

$$\mathscr{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\left[\alpha\, y_i\, \log(\hat{y}_i) + (1 - y_i)\, \log(1 - \hat{y}_i)\right] \tag{3.14}$$

Where:

- α is the positive class weight,
- n_{HC} is the number of healthy controls,
- n_{PD} is the number of PD patients,
- $\mathscr{L}_{\text{BCE}}$ is the loss function,
- N is the batchsize,
- $y_i \in \{0, 1\}$ is the ground truth,
- $\hat{y}_i \in [0, 1]$ is the model prediction, after having applied the sigmoid function to the logits.

Two different versions of this loss function were used. For the training loss, the α was determined for each batch, meaning it was computed each evaluation, representing the specific ratio in the current batch. For the validation loss, instead, the α was computed with the class counts of the entire validation set.

3.6.2 Multiple Instance Learning

The supervised approach presents a large dataset increase but the possible introduction of label noise. On the other hand, training the model by minimizing a loss function on each subject would represent no dataset increase but guaranteed trustworthy labels. With the approach explained in this section, we wanted to develop a paradigm between the two.

We started by defining a new parameter, called n_{split}, which determines in how many chunks or groups of windows a single subject, composed of n_{subj} windows, should be divided into and assigned the label to the chunk of windows, instead of the single ones. This means that we could consider a subject not as a set of windows but as a set of groups of windows. The number of windows present in each chunk, n_{win}, can be calculated with the following formula:

$$n_{win} = \frac{n_{subj}}{n_{split}} \tag{3.15}$$

With this formula, we can then determine how many chunks are present in a batch, N_{chunks}, with:

$$N_{chunks} = \frac{N}{n_{win}} \tag{3.16}$$

This restricts the split number values we can use because, without the introduction of window overlap between chunks, the number of windows of each subject must be divisible by the split number. This kind of overlap was ultimately not introduced for the possible inflation of validation loss. Another constraint this approach introduces is the batchsize value restrictions, since all n_{win} windows of a chunk must be present in a single batch, allowed batchsizes are only multiples of n_{win}.

Similarly to the approach used to train on multiple signals, explained in *Subsection 3.5.2*, we structured the batch, even with oversampling of the minority class if needed, so that n_{win} subsequent windows were kept in the same order; this way, after a simple reshape of the logit tensor, the following focal loss function can be applied.

$$\hat{z}_i = \frac{1}{n_{\text{win}}} \sum_{j=1}^{n_{\text{win}}} \hat{z}_{s,i} \tag{3.17}$$

$$p_i = \sigma(\hat{z}_i) = \frac{1}{1 + e^{-\hat{z}_i}} \tag{3.18}$$

$$q_i = y_i p_i + (1 - y_i)(1 - p_i) \tag{3.19}$$

$$\mathscr{L}_{\text{MIL}} = -\frac{1}{N_{chunks}} \sum_{i=1}^{N_{chunks}} \left[\alpha\, y_i + (1 - y_i)\right] (1 - q_i)^{\gamma} \log(q_i) \tag{3.20}$$

Where:

- $\hat{z}_{i,j}$ are the logits, with i chunks and j windows,
- p_i is the sigmoid probability of the chunk logits,
- $\sigma(\cdot)$ is the sigmoid function,
- q_i is the sigmoid probability of the correct label,
- $y_i \in \{0, 1\}$ is the ground truth for the chunks,
- $\mathscr{L}_{\text{MIL}}$ is the final loss function,
- N_{chunks} is the number of chunks in the batch,
- α is the same positive class weight defined in *Equation 3.13*,
- γ is the focal parameter, down-weights the easy-to-classify examples.

As mentioned before, to ensure that enough chunks of windows were present to have both classes in every batch, we introduced oversampling of the minority class. We computed an oversampling ratio, r, with the following formula:

$$r = \frac{n_{HC} + n_{PD}}{n_{PD}} \cdot \frac{n_{win}}{N} \tag{3.21}$$

Where:

- r is the oversampling ratio,
- n_{HC} is the number of healthy controls in the training set,
- n_{PD} is the number of PD patients in the training set,
- n_{win} is the number of windows in each chunk,
- N is the batchsize.

Oversampling led to faster overfitting; therefore, we scaled linearly the base learning rate, ℓ_{base}, to mitigate the behavior with:

$$\ell_{scaled} = \frac{\ell_{base}}{r} \tag{3.22}$$

This relationship was far from optimal, but sufficient to prevent fast overfitting.

3.6.3 Hyperparameters

In this subsection we report all the training and augmentation hyperparameters.

Augmentations

During model training, for each batch, the probability of augmenting the input tensor was 15%. The hyperparameters of the single augmentations can be found in *Table 3.10*.

Augmentation	Parameter	EEG	EMG and CoP
SNR scaling	Target SNR	$\mathcal{U}(\{10, 15, 20\})$	$\mathcal{U}(\{10, 15, 20\})$
Signal masking	Masking percentage p	$\mathcal{U}(\{0.05, 0.1\})$	$\mathcal{U}(\{0.05, 0.1\})$
	Masking blocks k	$\mathcal{U}(\{1, 2\})$	$\mathcal{U}(\{1, 2\})$
Channel dropout	Dropout channels C_d	$\mathcal{U}(\{1, 2, 3, 4, 5, 6\})$	$\mathcal{U}(\{1, 2\})$

Table 3.10: Augmentation hyperparameters

Training

The hyperparameters of the training setup can be found in *Table 3.11*.

Component	Parameter	Value
MIL loss function	γ	2.0
Learning rate	ℓ	2.5×10^{-5}
ADAM optimizer	β_1 β_2	0.9 0.999
Exponential learning rate scheduler	γ	0.99
Early stopping	Patience Minimum delta	10 1×10^{-4}

Table 3.11: Training setup hyperparameters

Chapter 4

Results

The metrics reported in this chapter are calculated on the distribution obtained by pooling all the performances for a full CV. Specifically, we report median, Inter-Quartile Range (IQR) and the difference between the 99th and the 1st quartiles, for both balanced accuracy and F1-score. Moreover, for the TokenSumNet results we only report the version that includes fusion weights for both channels and tokens.

4.1 Classification Performance

In this section, we report the results of the relevant configurations for both learning paradigms. In all tables the best summary statistics for each input are highlighted, regarding tie-breaking rules, we considered each evaluation metric separately; when breaking median ties, the best $[1^{st} - 99^{th}]$ was used, and vice versa, for breaking IQR ties the best $[1^{st} - 99^{th}]$ was used. The IQR was not used as a tie-breaking rule for its instability, considering the large skewness observed in some distributions.

4.1.1 Supervised Learning

For this subsection, the results are divided into two tables, in *Table 4.1* the monomodal metrics and in *Table 4.2* the multimodal metrics, the reported configurations include the input type, model, and whether data augmentation was applied or not.

Input (Model)	DA	Balanced Accuracy			F1-Score		
		Median	IQR	$[1^{st}-99^{th}]$	Median	IQR	$[1^{st}-99^{th}]$
EEG (ShallowNet)		71.67	**14.02**	**29.67**	61.54	**17.25**	**51.01**
	✓	**75.00**	18.18	31.67	**66.67**	20.45	52.38
EMG (XiaNet)		58.33	18.56	39.91	50.00	35.90	72.56
	✓	**65.15**	**14.39**	**36.22**	**52.63**	**33.33**	**68.06**
COP (ResNet1D)		73.48	**11.36**	32.60	**66.67**	**13.45**	**43.27**
	✓	**75.00**	15.00	**32.60**	66.67	15.58	44.34

Table 4.1: Supervised monomodal classification performance

Input	Model	DA	Balanced Accuracy			F1-Score		
			Median	IQR	$[1^{st}-99^{th}]$	Median	IQR	$[1^{st}-99^{th}]$
EEG + EMG	StackNet		**78.03**	15.15	27.13	**71.43**	20.00	39.62
		✓	75.00	12.50	**22.65**	66.67	15.38	**32.24**
	TokenSumNet		78.03	10.61	38.67	71.43	**8.33**	41.83
		✓	78.03	**5.00**	41.85	71.43	8.33	74.76
EEG + COP	StackNet		75.00	11.06	28.71	66.67	15.00	**27.20**
		✓	77.27	**9.17**	**25.44**	66.67	15.58	29.20
	TokenSumNet		75.83	9.85	32.15	66.67	**10.26**	43.24
		✓	**79.17**	10.83	27.12	**72.73**	13.33	33.31
EMG + COP	StackNet		69.17	11.67	26.99	60.00	12.12	37.18
		✓	70.83	**10.61**	**26.51**	61.54	**12.12**	**32.47**
	TokenSumNet		70.83	11.44	34.34	61.54	16.04	49.99
		✓	**75.00**	13.33	33.34	**66.67**	18.18	45.13
EEG + EMG + COP	StackNet		77.27	12.50	29.37	66.67	16.92	36.76
		✓	77.27	**8.33**	28.52	66.67	**12.73**	35.47
	TokenSumNet		78.03	10.83	30.76	71.43	13.33	43.27
		✓	**81.67**	10.61	**23.54**	**75.00**	13.33	**28.91**

Table 4.2: Supervised multimodal classification performance

In the supervised setting, we can observe that data augmentation generally improved median performance but sometimes slightly increased variability. Regarding monomodal experiments, we can observe higher performance using EEG and CoP, with respect to EMG, for both median and $[1^{st}-99^{th}]$ values. When comparing multimodal approaches, the relevance of data augmentation becomes even more evident. The best-performing model (TokenSumNet with all signals) achieved not only a higher median balanced accuracy and F1-score but also a reduction in variability compared to other models, and especially compared to monomodal approaches.

4.1.2 Multiple Instance Learning

In this subsection, the DA is replaced by the n_{split} parameter, further dividing tables into monomodal without DA, *Table 4.3*, monomodal with DA, *Table 4.4*, multimodal without DA, *Table 4.5*, and multimodal with DA, *Table 4.6*.

Input (Model)	n_{split}	Balanced Accuracy			F1-Score		
		Median	IQR	$[1^{st}-99^{th}]$	Median	IQR	$[1^{st}-99^{th}]$
EEG (ShallowNet)	1	75.00	16.52	32.88	66.67	17.86	55.53
	3	**75.00**	16.67	**27.35**	**66.67**	19.78	**44.40**
	5	74.24	14.02	33.26	66.67	18.10	51.01
	10	74.24	**12.88**	34.61	66.67	**15.00**	46.92
EMG (XiaNet)	1	59.85	21.67	39.27	**53.33**	40.00	65.82
	3	58.33	16.74	**32.97**	44.44	26.37	**61.87**
	5	57.58	18.94	33.06	47.06	25.49	71.43
	10	**59.85**	**11.74**	34.00	52.63	**21.32**	70.29
COP (ResNet1D)	1	70.83	10.61	27.12	61.54	13.45	39.26
	3	**73.33**	5.83	**25.51**	62.50	**6.67**	**29.26**
	5	71.67	8.33	35.59	**63.16**	12.12	42.51
	10	70.83	**5.83**	**23.22**	61.54	9.52	32.18

Table 4.3: MIL monomodal classification performance without data augmentation

Input (Model)	n_{split}	Balanced Accuracy			F1-Score		
		Median	IQR	$[1^{st}-99^{th}]$	Median	IQR	$[1^{st}-99^{th}]$
EEG (ShallowNet)	1	75.00	18.18	34.96	66.67	19.78	61.76
	3	**75.00**	14.02	**30.49**	**66.67**	**9.52**	**45.20**
	5	74.24	14.02	36.19	62.50	14.29	48.38
	10	74.24	**12.12**	34.32	66.67	18.18	49.20
EMG (XiaNet)	1	54.55	22.35	39.30	**52.63**	41.54	**70.29**
	3	58.33	**16.67**	40.39	52.63	33.82	77.94
	5	**62.50**	16.67	40.99	50.00	**26.67**	71.24
	10	59.85	20.00	**35.94**	50.00	29.17	74.14
COP (ResNet1D)	1	68.94	14.39	29.70	62.50	18.18	50.20
	3	**75.00**	9.09	27.92	**66.67**	11.43	**29.94**
	5	71.67	**5.30**	24.63	62.50	**6.67**	37.49
	10	73.48	12.50	**21.12**	66.67	12.61	33.48

Table 4.4: MIL monomodal classification performance with data augmentation

Input	Model	n_{split}	Balanced Accuracy			F1-Score		
			Median	IQR	$[1^{st}-99^{th}]$	Median	IQR	$[1^{st}-99^{th}]$
EEG + EMG	StackNet	1	75.00	10.30	29.08	66.67	11.19	41.05
		3	73.48	11.82	25.02	66.67	12.73	**30.00**
		5	77.27	**6.67**	31.12	**70.59**	**8.33**	45.69
		10	**77.27**	13.33	**24.00**	66.67	15.00	36.75
	TokenSumNet	1*	50.00	0.00	13.94	50.00	6.72	52.17
		3*	50.00	0.00	23.63	50.00	6.72	31.49
		5	77.27	15.00	51.63	70.59	17.86	81.69
		10	75.00	13.33	45.31	66.67	22.73	81.43
EEG + COP	StackNet	1	73.48	15.00	23.42	66.67	15.58	33.48
		3	75.00	**8.33**	**21.23**	66.67	**8.93**	**24.83**
		5	75.00	9.17	29.52	66.67	11.43	43.27
		10	75.00	10.00	23.44	66.67	12.73	36.75
	TokenSumNet	1	59.85	12.50	43.62	53.33	23.53	47.40
		3	75.83	10.61	28.72	66.67	10.26	34.49
		5	**77.27**	10.91	28.53	**70.59**	14.42	40.60
		10	77.27	11.67	31.91	70.59	16.92	42.74
EMG + COP	StackNet	1	66.67	8.33	30.55	57.14	11.54	48.51
		3	68.18	8.33	**23.24**	58.82	9.52	34.53
		5	68.94	9.17	27.07	57.14	8.61	34.70
		10	**69.17**	12.50	27.21	**61.54**	12.12	34.53
	TokenSumNet	1	54.17	10.83	35.49	42.86	21.43	71.27
		3	58.33	11.36	31.38	50.00	16.97	60.67
		5	54.17	11.36	30.98	44.44	**7.89**	**32.78**
		10	57.58	**7.58**	32.97	44.44	10.00	43.78
EEG + EMG + COP	StackNet	1	74.24	13.33	39.38	66.67	16.92	46.93
		3	77.27	13.33	27.90	66.67	18.46	37.47
		5	77.50	8.94	30.52	70.59	**6.06**	40.85
		10	73.48	8.33	26.35	66.67	9.52	38.53
	TokenSumNet	1	77.50	**7.58**	27.12	70.59	8.33	35.71
		3	74.24	10.15	**23.73**	66.67	13.46	**28.91**
		5	**79.17**	10.61	25.46	**72.73**	10.26	33.25
		10	78.03	12.50	28.59	70.59	16.92	38.56

Table 4.5: MIL multimodal classification performance without data augmentation

* Results marked were ignored because the model collapsed.

Input	Model	n_{split}	Balanced Accuracy			F1-Score		
			Median	IQR	$[1^{st}-99^{th}]$	Median	IQR	$[1^{st}-99^{th}]$
EEG + EMG	StackNet	1	75.00	12.12	**23.33**	66.67	13.46	38.89
		3	75.00	10.98	28.44	66.67	12.50	40.93
		5	**77.27**	11.74	28.58	**70.59**	12.50	**35.14**
		10	75.00	**9.17**	23.58	66.67	**11.84**	38.29
	TokenSumNet	1*	50.00	0.00	16.12	50.00	6.72	48.17
		3*	50.00	0.00	30.60	50.00	6.72	33.62
		5	73.48	15.91	40.00	66.67	22.83	45.02
		10	75.00	13.33	47.31	66.67	15.58	77.45
EEG + COP	StackNet	1	75.00	9.47	22.36	66.67	11.43	30.00
		3	75.00	10.23	**19.69**	66.67	11.43	**24.83**
		5	75.00	11.67	32.23	66.67	12.61	40.98
		10	75.00	8.33	21.46	66.67	9.89	24.83
	TokenSumNet	1	58.33	16.67	42.53	50.00	25.49	69.44
		3	75.83	11.74	26.00	66.67	13.77	33.43
		5	**77.27**	10.61	33.68	**70.59**	13.33	39.52
		10	75.83	**7.27**	27.36	66.67	**8.33**	40.93
EMG + COP	StackNet	1	**70.83**	10.61	**24.99**	**61.54**	12.12	36.87
		3	66.67	11.82	31.70	57.14	16.67	49.97
		5	70.83	13.33	28.71	61.54	12.12	38.36
		10	70.83	13.64	30.18	61.54	**6.67**	38.19
	TokenSumNet	1	56.82	11.67	35.32	47.06	20.00	66.67
		3	62.12	9.17	31.97	50.00	15.97	47.40
		5	55.00	**8.33**	35.22	40.00	16.67	58.93
		10	56.67	11.36	29.22	44.44	13.64	**30.90**
EEG + EMG + COP	StackNet	1	75.00	8.71	31.94	66.67	11.43	46.49
		3	74.24	8.03	29.76	66.67	9.89	**29.17**
		5	75.00	9.17	31.34	66.67	9.89	43.60
		10	75.00	8.03	33.96	66.67	11.43	43.90
	TokenSumNet	1	79.17	**6.82**	24.82	71.43	**8.33**	34.62
		3	77.27	12.65	**23.86**	70.59	16.18	35.56
		5	**81.67**	10.61	27.41	**72.73**	13.33	39.93
		10	75.83	10.15	27.32	66.67	15.00	37.47

Table 4.6: MIL multimodal classification performance with data augmentation

* Results marked were ignored because the model collapsed.

The MIL experiments display a heterogeneous dependence on both the hyperparameter n_{split} and the chosen fusion architecture. For monomodal inputs, $n_{split} = 3$ typically provided the highest median and lowest $[1^{st}-99^{th}]$ balanced accuracy. Multimodal results, however, do not admit a single optimal configuration. Considering only the results with data augmentation, the overall best median performance was obtained with TokenSumNet using all signals at $n_{split} = 5$ (median BA = 81.67%). On the other hand, the same configuration achieved the lowest variability with $n_{split} = 3$. Furthermore, for other modality combinations and n_{split} settings, TokenSumNet exhibited worse median performance than StackNet (e.g., StackNet achieved a median BA of 77.27% with $n_{split} = 5$ for EEG+EMG when TokenSumNet collapsed at low n_{split} values).

Consequently, no single combination of n_{split} and fusion architecture consistently outperformed the others across modalities. This variability highlights that model selection within the MIL framework requires a trade-off between peak performance and stability, where, instead, the supervised approach achieved better and more homogeneous results overall.

Chapter 5

Discussion

In this chapter we suggest possible interpretations of the results obtained and address the limitations of the present study.

5.1 Results Interpretation

In the supervised setting, data augmentation played a net beneficial role, particularly in multimodal experiments where it improved both central performance and stability. The superiority of EEG and CoP over EMG inputs is consistent with literature reporting that cortical activity and postural control are more sensitive markers of early PD than lower limb muscle activity [5, 16]. Multimodal fusion further enhanced performance, with TokenSumNet providing the most consistent improvements, suggesting that embedding fusion is more beneficial compared to concatenated fusion. TokenSumNet also benefits from the use of the transformer architecture, allowing more complex temporal modeling, that is only possible with embedding fusion approaches. Overall, supervised experiments yielded more homogeneous results, indicating a stronger robustness of the paradigm compared to the MIL framework.

MIL experiments showed a strong dependence on the n_{split} parameter, with no single value providing consistent superiority across modalities. This dependence is expected, as n_{split} directly influences the number of gradient updates per epoch, thereby altering the optimization dynamics. TokenSumNet demonstrated the capacity to reach the highest median balanced accuracy, but its sensitivity to both n_{split} and modality choice resulted in occasional performance collapses. In contrast, StackNet proved more stable for EMG-containing pairs, suggesting that EMG signals may introduce noise or require simpler network strategies within MIL approaches. These findings underline both the potential and the limitations of the proposed MIL framework: while it can match supervised performance under specific settings, it lacks robustness across all configurations.

Although supervised learning provided more stable and homogeneous improvements, the MIL framework revealed a possible alternative to traditional training. Supervised learning currently appears

more reliable, although MIL approaches remain promising for future work with larger datasets.

5.2 Limitations

This study, despite showing promising results, presents several limitations.

First, the dataset is limited in size and notably imbalanced, comprising only 29 Parkinson's patients compared to 271 healthy controls. This limitation restricts the generalizability of the findings and increases the risk of overfitting. Such challenges are common in clinical datasets, where inter-subject heterogeneity is the primary source of variability. The most effective way to mitigate this issue is to expand the cohort by including a larger number of patients.

Second, data were collected using two different acquisition platforms and EEG caps, which, despite correction efforts through stratified cross-validation, may introduce systematic variability unrelated to the underlying physiology. This increases the risk that models may, at least partially, learn to distinguish hardware-related characteristics rather than purely physiological differences.

Third, all evaluations were performed using nested cross-validation on the same dataset, and no independent test cohort was available, limiting conclusions about clinical validity.

Fourth, while data augmentation was generally beneficial, it may not perfectly capture natural physiological variability.

Finally, MIL experiments resulted in heterogeneous performance across modalities and architectures, a factor that could complicate model selection when using this paradigm.

Chapter 6

Conclusion and Future Work

Conclusion

This thesis investigated multimodal deep learning approaches for the early detection of Parkinson's Disease using EEG, EMG, and CoP signals from the BioVRSea dataset. By adapting literature-established architectures into modality-specific encoders that preserve temporal structure, we implemented both supervised and Multiple Instance Learning paradigms, with a special focus on multimodal fusion strategies such as TokenSumNet and StackNet.
The supervised learning experiments demonstrated that data augmentation improved central performance in most settings and yielded more stable and homogeneous results compared to MIL. Among monomodal inputs, EEG and CoP consistently outperformed EMG, while multimodal fusion improved both accuracy and stability, with TokenSumNet trained on all modalities achieving the best overall performance (median balanced accuracy 81.67%, F1-score 75.00%, $[1^{st}-99^{th}]$ balanced accuracy range 23.54%). In contrast, the MIL paradigm exhibited heterogeneous and inconsistent results across architectures and n_{split} values, with TokenSumNet occasionally achieving the best median performance but also displaying instability, while simpler architectures such as StackNet proved more robust in certain cases. Overall, supervised multimodal fusion emerged as the most reliable strategy within the current dataset.

Future Work

Several directions remain open for future research:

- **Larger and more balanced datasets.** Considering the small size of the patient cohort for the current study, reported performance is promising and extending evaluation to larger and more balanced datasets may bring the field closer to clinical translation.

- **MIL extensions.** Introducing overlap among window chunks in the MIL setting could extend the usable range of n_{split} values. While this may inflate validation performance, it may also offer insights into the observed heterogeneity across configurations.
- **Comprehensive architectural comparisons.** This work evaluated only one variant of TokenSumNet (with channel and token weights). Future work should include a systematic comparison of both TokenSumNet variants, StackNet, and other fusion models, under identical conditions.
- **Physiologically motivated augmentations.** Beyond generic signal transformations, augmentations inspired by neurophysiology and motor control may provide more realistic variability and improve generalization.

Bibliography

[1] T. Pringsheim, N. Jette, A. Frolkis, and T. D. Steeves, "The prevalence of Parkinson's disease: A systematic review and meta-analysis," *Mov. Disord.*, vol. 29, no. 13, pp. 1583–1590, 2014. doi: 10.1002/mds.25945 .

[2] M. J. Armstrong and M. S. Okun, "Diagnosis and treatment of Parkinson Disease: A review," *JAMA*, vol. 323, no. 6, pp. 548–560, 02 2020. doi: 10.1001/jama.2019.22360 .

[3] R. Soikkeli, J. Partanen, H. Soininen, A. Pääkkönen, and P. Riekkinen, "Slowing of EEG in Parkinson's disease," *Electroencephalogr. Clin. Neurophysiol.*, vol. 79, no. 3, pp. 159–165, 1991. doi: 10.1016/0013-4694(91)90134-P .

[4] D. Jacob, L. Guerrini, F. Pescaglia, S. Pierucci, C. Gelormini, V. Minutolo *et al.*, "Adaptation strategies and neurophysiological response in early-stage Parkinson's disease: BioVRSea approach," *Front. Hum. Neurosci.*, vol. 17, 2023. doi: 10.3389/fnhum.2023.1197142 .

[5] H. W. Loh, W. Hong, C. P. Ooi, S. Chakraborty, P. D. Barua, R. C. Deo *et al.*, "Application of deep learning models for automated identification of Parkinson's disease: A review (2011–2021)," *Sensors*, vol. 21, no. 21, 2021. doi: 10.3390/s21217034 .

[6] A. Islam, L. Alcock, K. Nazarpour, L. Rochester, and A. Pantall, "Effect of Parkinson's disease and two therapeutic interventions on muscle activity during walking: a systematic review," *npj Parkinson's Disease*, vol. 6, no. 1, p. 22, 2020. doi: 10.1038/s41531-020-00119-w .

[7] P. Wodarski, J. Jurkojć, J. Michalska, A. Kamieniarz, G. Juras, and M. Gzik, "Balance assessment in selected stages of Parkinson's disease using trend change analysis," *J. Neuroeng. Rehabil.*, vol. 20, p. 99, 2023. doi: 10.1186/s12984-023-01229-1 .

[8] M. Recenti, C. Ricciardi, R. Aubonnet, I. Picone, D. Jacob, H. Á. R. Svansson *et al.*, "Toward predicting motion sickness using virtual reality and a moving platform assessing brain, muscles, and heart signals," *Front. Bioeng. Biotechnol.*, vol. 9, 2021. doi: 10.3389/fbioe.2021.635661 .

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436–444, May 2015. doi: 10.1038/nature14539 .

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 06 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez *et al.*, "Attention is all you need," 2017. doi: 10.48550/ARXIV.1706.03762 .

[12] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, "EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces," *Journal of Neural Engineering*, vol. 15, no. 5, p. 056013, 2018. doi: 10.1088/1741-2552/aace8c .

[13] R. Schirrmeister, J. Springenberg, L. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann *et al.*, "Deep learning with convolutional neural networks for EEG decoding and visualization: Convolutional neural networks in eeg analysis," *Human Brain Mapping*, vol. 38, 08 2017. doi: 10.1002/hbm.23730 .

[14] H. Chang, B. Liu, Y. Zong, C. Lu, and X. Wang, "EEG-based Parkinson's disease recognition via attention-based sparse graph convolutional neural network," *IEEE Journal of Biomedical and Health Informatics*, vol. 27, no. 11, p. 5216–5224, Nov. 2023. doi: 10.1109/jbhi.2023.3292452 .

[15] F. Del Pup, R. Brun, F. Iotti, E. Paccagnella, M. Pezzato, S. Bertozzo *et al.*, "TransformEEG: Towards improving model generalizability in deep learning-based EEG Parkinson's disease detection," 2025. doi: 10.48550/ARXIV.2507.07622 .

[16] J.-M. Park, C.-W. Moon, B. C. Lee, E. Oh, J. Lee, W.-J. Jang *et al.*, "Detection of freezing of gait in Parkinson's disease from foot-pressure sensing insoles using a temporal convolutional neural network," *Frontiers in Aging Neuroscience*, vol. 16, 2024. doi: 10.3389/fnagi.2024.1437707 .

[17] S. R. Stahlschmidt, B. Ulfenborg, and J. Synnergren, "Multimodal deep learning for biomedical data fusion: a review," *Briefings in Bioinformatics*, vol. 2, 2022. doi: 10.1093/bib/bbab569 .

[18] J. Golding, A. Mueller, and M. Gresty, "A motion sickness maximum around the 0.2 hz frequency range of horizontal translational oscillation," *Aviation, space, and environmental medicine*, vol. 72, pp. 188–92, 04 2001. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/11277284/

[19] F. Del Pup, A. Zanola, L. Fabrice Tshimanga, A. Bertoldo, and M. Atzori, "The more, the better? evaluating the role of EEG preprocessing for deep learning applications," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 33, pp. 1061–1070, 2025. doi: 10.1109/TNSRE.2025.3547616 .

[20] M. Atzori, M. Cognolato, and H. Müller, "Deep learning with convolutional neural networks applied to electromyography data: A resource for the classification of movements for prosthetic hands," *Frontiers in Neurorobotics*, vol. 10, 2016. doi: 10.3389/fnbot.2016.00009 .

[21] P. P. Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: a tutorial," *Proc. IEEE*, vol. 78, pp. 56–93, 1990. [Online]. Available: https://api.semanticscholar.org/CorpusID:16142393

[22] A. Kamieniarz, J. Michalska, W. Marszałek, M. Stania, K. J. Słomka, A. Gorzkowska *et al.*, "Detection of postural control in early parkinson's disease: Clinical testing vs. modulation of center of pressure," *PLOS ONE*, vol. 16, no. 1, p. e0245353, 2021. doi: 10.1371/journal.pone.0245353 .

[23] S. Pardoel, G. Shalin, J. Nantel, E. D. Lemaire, and J. Kofman, "Early detection of freezing of gait during walking using inertial measurement unit and plantar pressure distribution data," *Sensors*, vol. 21, no. 6, p. 2246, 2021. doi: 10.3390/s21062246 .

[24] G. Brookshire, J. Kasper, N. M. Blauch, Y. C. Wu, R. Glatt, D. A. Merrill *et al.*, "Data leakage in deep learning studies of translational eeg," *Frontiers in Neuroscience*, vol. 18, 2024. doi: 10.3389/fnins.2024.1373515 .

[25] K. Xia, J. Huang, and H. Wang, "LSTM-CNN architecture for human activity recognition," *IEEE Access*, vol. 8, pp. 56 855–56 866, 2020. doi: 10.1109/ACCESS.2020.2982225 .

[26] Y. Zheng, Z. Liu, R. Mo, Z. Chen, W.-s. Zheng, and R. Wang, "Task-oriented self-supervised learning for anomaly detection in electroencephalography," 2022. doi: 10.48550/ARXIV.2207.01391 .