UNIVERSITY OF VALLADOLID

FACULTY OF MEDICINE
SCHOOL OF INDUSTRIAL ENGINEERING

FINAL DEGREE PROJECT
BIOMEDICAL ENGINEERING DEGREE

Exploración de diferencias en la actividad cerebral entre movimientos ejecutados e imaginados para optimizar terapias de neurofeedback en pacientes con accidente cerebrovascular

Exploring differences in brain activity between executed and imagined movements to optimize neurofeedback therapies in stroke patients

AUTHOR:
Mr. Pablo Paisan Garcia

SUPERVISORS:
Dr. Roberto Hornero Sánchez
Dr. Eduardo Santamaría-Vázquez

Valladolid, September of 2024

TITLE:	**Exploración de diferencias en la actividad cerebral entre movimientos ejecutados e imaginados para optimizar terapias de neurofeedback en pacientes con accidente cerebrovascular.** **Exploring differences in brain activity between executed and imagined movements to optimize neurofeedback therapies in stroke patients.**
AUTHOR:	**Mr. Pablo Paisán García**
SUPERVISORS:	**Dr. Roberto Hornero Sánchez** **Dr. Edurado Santamaría-Vázquez**
DEPARTMENT:	**Signal Theory and Communications and Telematics Engineering** **Brain-Computer Interface Department**

COMMITEE	
PRESIDENT:	**Dr. Roberto Hornero Sánchez**
SECRETARY:	**Dr. Javier Gómez Pilar**
VOCAL:	**Dr. Jesús Poza Crespo**
ALTERNATE 1:	**Dr. Carlos Gómez Peña**
ALTERNATE 2	**Dr. María García Gadañón**

DATE:	**September 2024**
GRADE:	

Resumen

Los sistemas de Interfaz Cerebro-Computadora (BCI: Brain-Computer Interface) basados en la imaginación motora (MI: Motor Imagery) tienen un gran potencial para la rehabilitación de pacientes con discapacidades motoras, como las personas que han sufrido de accidentes cerebrovasculares. Sin embargo, las metodologías actuales necesitan de una extensa calibración, que puede no ser efectiva en todos los casos debido a la variabilidad entre sujetos y la calidad de la ejecución de la tarea de MI. Esta limitación reduce la accesibilidad y eficacia de los sistemas BCI en aplicaciones clínicas.

El objetivo de este Trabajo de Fin de Grado es desarrollar una estrategia innovadora para la clasificación de actividad cerebral medida mediante el EEG mientras el sujeto hace una tarea de MI. Con ese propósito, se han aplicado técnicas de aprendizaje por transferencia con el fin de mejorar la efectividad y accesibilidad de los sistemas BCI para la neurorehabilitación, mediante la reducción del tiempo de calibración, la realimentación más eficaz mediante el aprendizaje de patrones de movimientos ejecutados, o la creación de programas de entrenamiento personalizados. Para ello, se ha realizado un experimento de caracterización donde se han analizado las diferencias y similitudes en la actividad neuronal durante la ejecución motora (ME: Motor Execution) y la MI mediante técnicas espectrales, como la potencia relativa, y técnicas temporales, como la desincronización/sincronización relacionada con eventos (ERD/ERS: Event Related Desynchronization/Event Related Synchronization). Posteriormente, se ha diseñado una estrategia de aprendizaje por transferencia basado en modelos que consiste en aplicar modelos preentrenados con datos de EEG durante ME para la clasificación de MI, sin necesidad de extensas calibraciones.

Los resultados de caracterización muestran que existen patrones similares en la actividad neural asociados a ME y MI, lo cual valida la posibilidad de utilizar estos datos para la clasificación de MI. Los resultados de clasificación demostraron que la estrategia de aprendizaje por transferencia permite el control de BCI, logrando una precisión del 80,31% en sujetos no entrenados al utilizar solo dos segundos de registros de EEG. Además, la utilización de registros de mayor duración o la calibración específica para cada sujeto optimiza aún más el rendimiento del sistema.

La implementación de la estrategia diseñada en este trabajo podría disminuir significativamente el tiempo de calibración de los sistemas BCI basados en MI, lo cual es crucial para su aplicación en la rehabilitación de pacientes con discapacidades motoras. Esta mejora en la accesibilidad y eficiencia de los sistemas BCI podría optimizar los protocolos de rehabilitación de accidentes cerebrovasculares y mejorar la calidad de vida de los pacientes.

Palabras clave: Imaginación motora (MI), Clasificación, Aprendizaje por transferencia, Interfaz Cerebro-Computadora (BCI), Neurorehabilitación, Ejecución motora (ME).

Abstract

Motor imagery (MI)-based Brain-Computer Interface (BCI) systems hold great potential for the rehabilitation of patients with motor disabilities, such as those who have suffered from strokes. However, current methodologies require extensive calibration, which may not be effective in all cases due to intersubject variability and the quality of MI task execution. This limitation reduces the accessibility and efficacy of BCI systems in clinical applications. The objective of this Final Degree Project is to develop an innovative strategy for the classification of brain activity measured by EEG while the subject performs an MI task. To that end, transfer learning techniques have been applied to improve the effectiveness and accessibility of BCI systems for neurorehabilitation, by reducing calibration time, providing more effective feedback through the learning of executed movement patterns, or creating personalized training programs. To achieve this, a characterization experiment was conducted to analyze the differences and similarities in neural activity during motor execution (ME) and MI using spectral techniques, such as relative power, and temporal techniques, such as event-related desynchronization/synchronization (ERD/ERS). Subsequently, a model-based transfer learning strategy was designed, which involves applying pretrained models with EEG data during ME for MI classification without the need for extensive calibration.

Characterization results show similarities in neural activity patterns associated with ME and MI, validating the feasibility of using these data for MI classification. Additionally, classification results demonstrated that the transfer learning strategy enables BCI control achieving an accuracy of 80.31% in untrained subjects using only two seconds of EEG recordings, with the possibility of using longer recordings or subject-specific calibration further optimizing system performance.

The implementation of the strategy designed in this work could significantly reduce the calibration time of MI-based BCI systems, which is crucial for their application in the rehabilitation of patients with motor disabilities. This improvement in the accessibility and efficiency of BCI systems could optimize stroke rehabilitation protocols and enhance patients' quality of life.

Key words: Motor imagery (MI), Classification, Transfer learning, Brain-Computer Interface (BCI), Neurorehabilitation, Motor execution (ME).

Acknowledgments

I would like to begin by expressing my deepest gratitude to both my supervisors, Eduardo and Roberto, not only for providing me with the opportunity to work with them in this project, but also for the support I received since I first made contact with the department two years ago. Furthermore, I would also like to thank all the members of the Biomedical Engineering Group for the warm welcome and great treatment I received through this years, specially to Sergio and Diego.

To my friends, specially maulas, thank you for bearing with me all these years and sharing this journey. To my Boston New Generation & Company friends and Leon, thank you for being part of the best experience of my life. To my hockey family, I am immensely grateful for the countless memories and the unwavering support, providing a safe space over the years with a special thanks to the 'new member' who has been by my side always with a smile dealing with an emotional rollercoaster of events in recent times.

Finally, I am forever grateful to my family. Your unconditional love and support have been my bedrock, giving me the courage to pursue my dreams. I love you and I would not be where I am today without you.

Thank you all.

Table of Contents

List of Figures

List of Tables

Acronyms

Acronym	English	Spanish
AI	Artificial Intelligence	Inteligencia Artificial
BCI	Brain-Computer Interface	Interfaz Cerebro-Computadora
CAR	Common Average Reference	Referencia Promedio Común
CNN	Convolutional Neural Network	Red Neuronal Convolucional
CNS	Central Nervous System	Sistema Nervioso Central
c-VEP	Code-modulated Visually Evoked Potential	Potencial Evocado Visual Modulado por Código
DL	Deep Learning	Aprendizaje Profundo
EEG	Electroencephalography	Electroencefalografía
EMG	Electromyography	Electromiografía
ERD	Event-Related Desynchronization	Desincronización Relacionada con un Evento
ERP	Event-Related Potential	Potencial Evocado con Eventos
ERS	Event-Related Synchronization	Sincronización Relacionada con un Evento
FES	Functional Electrical Stimulation	Estimulación Eléctrica Funcional
fMRI	Functional Magnetic Resonance Imaging	Imágenes por Resonancia Magnética Funcional
ICA	Independent Component Analysis	Análisis de Componentes Independientes
LOSO	Leave-One-Subject-Out	Dejar un sujeto fuera
ME	Motor Execution	Ejecución Motora
MEG	Magneto-encephalogram	Magnetoencefalografía
MI	Motor Imagery	Imaginación Motora
ML	Machine Learning	Aprendizaje Automático
NFB	Neurofeedback	Neuroretroalimentación
PSD	Power Spectral Densisty	Densidad Espectral de Potencia
RNN	Recurrent Neural Networks	Red Neuronal Recurrente
RP	Relative Power	Potencia Relativa
SCP	Slow Cortical Potential	Potencial Cortical Lento
SMR	Sensorimotor Rhythm	Ritmo Sensorimotor
SMA	Supplementary Motor Area	Area Motor Suplementaria
SSEP	Steady State Evoked Potentials	Potencial Evocado en Estado Estable
TMS	Transcranial Magnetic Stimulation	Estimulación Magnética Transcraneal
VR	Virtual Reality	Realidad Virtual

1. INTRODUCTION

1.1. Stroke

Stroke is a medical emergency, second leading cause of death and third cause of disability in adults worldwide (Feigin *et al.*, 2018). It is characterized by a sudden disruption of blood flow to the brain, leading to the rapid loss of brain function. This interruption deprives brain tissue of essential oxygen and nutrients, causing cells to begin dying within minutes (Murphy and Werring, 2020). The implications of a stroke are profound and multifaceted, affecting motor control, sensory perception, language abilities, and cognitive functions. The extent and location of brain damage determine the specific symptoms and their severity. Common symptoms include sudden numbness or weakness in the face, arm, or leg, particularly on one side of the body; confusion; trouble speaking or understanding speech; vision problems in one or both eyes; difficulty walking; dizziness; and loss of balance or coordination (NIH, 2024).

Rapid diagnosis and intervention are crucial in stroke treatment. The medical community uses the acronym F.A.S.T. (Face drooping, Arm weakness, Speech difficulties, and Time to call emergency services) to help recognize and respond to the signs of stroke promptly. Treatment options vary depending on the type of stroke and the time elapsed since symptoms began. They may include medications to break up or remove clots, surgical interventions, and other medical therapies to manage complications and support recovery (NIH, 2024).

Despite advancements in acute stroke management, many survivors face long-term challenges, calling for comprehensive rehabilitation efforts. This is where innovative approaches like brain-computer interfaces (BCIs) can play a significant role, particularly in improving neurorehabilitation outcomes. By exploiting BCIs, researchers and clinicians aim to better understand brain activity associated with stroke and develop more effective therapeutic interventions, ultimately improving the quality of life for stroke patients.

As seen on Figure 1.1, stroke can be classified into two main types: ischemic and hemorrhagic. Ischemic stroke, which accounts for about 85% of all cases (Murphy and Werring, 2020), occurs when a blood clot obstructs a blood vessel supplying the brain. This blockage can be caused by thrombosis (a clot forming in an artery within the brain) or embolism (a clot forming elsewhere in the body and traveling to the brain). Hemorrhagic stroke, on the other hand, happens when a blood vessel in the brain bursts, leading to bleeding within or around the brain. This type can be further divided into intracerebral hemorrhage (bleeding within the brain tissue) and subarachnoid hemorrhage (bleeding in the space between the brain and the surrounding membrane) (Unnithan *et al.*, 2023).

Early diagnosis and treatment are vital to minimize brain damage and improve outcomes. Medical imaging techniques such as CT scans and MRIs play a crucial role in diagnosing the type and location of a stroke.

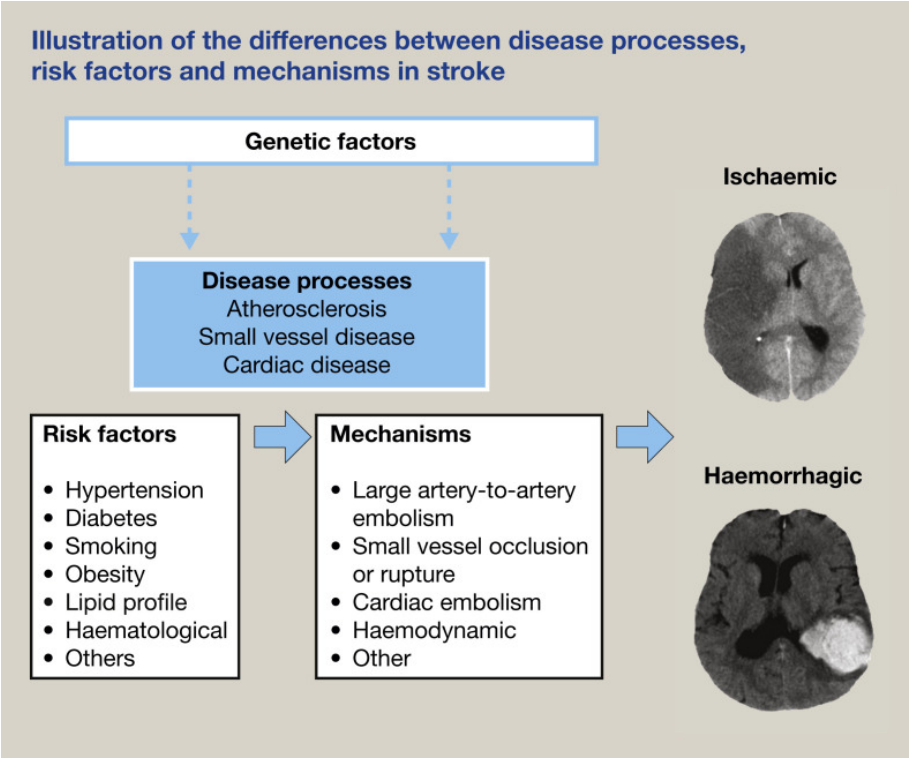


Fig. 1.1. Differences between Ischaemic and Hemorrhagic stroke (Murphy and Werring, 2020).

1.1.1. Neurorehabilitation

Neurorehabilitation is a therapeutic process aimed at aiding recovery and improving the functional abilities of stroke survivors. It encompasses a range of strategies, including physical therapy, occupational therapy, speech and language therapy, and psychological support. The goal is to help patients regain independence and increase their quality of life.

Physical therapy focuses on restoring movement and strength through exercises and activities tailored to the patient's specific needs. Occupational therapy helps patients relearn daily activities and adapt to their physical limitations by using adaptive tools and techniques. Speech and language therapy addresses difficulties with communication and swallowing, often involving exercises to improve speech clarity, language skills, and cognitive functions related to communication. Given that many stroke survivors experience aphasia and dysphagia, this therapy is fundamental for improving their ability to safely consume food and liquids (Kelley and Borazanci, 2009).

Psychological support is also a crucial component of neurorehabilitation, as stroke survivors often experience emotional and cognitive challenges such as depression, anxiety, and memory loss. Counseling, cognitive behavioral therapy, and community integration can help patients and their families cope with these changes and improve their mental well-being (Perna and Harik, 2020).

In addition to these traditional therapies, innovative techniques are being integrated into neurorehabilitation programs to strengthen their effectiveness. These approaches play upon the

concept of neural plasticity, the brain's capacity to alter, reorganize, or develop new neural connections. The brain's ability to reorganize itself allows undamaged areas to take over functions lost due to stroke, or to 'reconnect' affected or isolated neurons in the damaged areas. Some of the most promising approaches include (Gunduz *et al.*, 2023):

- **Mirror Therapy:** This technique involves using a mirror to create a reflection of the unaffected limb, giving the illusion that the affected limb is moving. This visual feedback can help retrain the brain and improve motor function in the affected limb. Mirror therapy has been particularly effective in reducing pain and improving movement in stroke patients with hemiparesis, implying weakness on one side of the body (Gandhi *et al.*, 2020).

- **Robotic-Assisted Therapy:** Robotic devices are increasingly used to provide precise, repetitive movements that aid in motor recovery and strength building. This technique can deliver a high-intensity, standardized therapy, which is considered crucial for motor cortex reorganization and improving recovery. These devices can be tailored to the patient's needs, offering support and resistance as necessary. It has has shown to improve outcomes in both upper and lower limb rehabilitation in post-stroke patients, improving muscle strength, coordination, and range of motion (Chang and Kim, 2013).

- **Virtual Reality (VR):** VR provides an immersive environment where patients can engage in simulated activities that promote motor and cognitive recovery, specially of the recovery of the motor function in upper body limbs. VR has also shown to improve motor function, balance, and spatial awareness in stroke patients (Laver *et al.*, 2018).

- **Transcranial Magnetic Stimulation (TMS):** TMS involves using magnetic fields to stimulate specific areas of the brain. This non-invasive technique can reinforce neural plasticity and promote recovery in stroke patients by modulating brain activity. TMS has been found to be effective in improving motor function, language abilities, and mood in stroke survivors (Sheng *et al.*, 2023).

- **Transcranial Direct Current Stimulation**: It involves applying a low electrical current to the scalp, which modulates neuronal activity. This technique alters the excitability of neurons in the brain. Anodal stimulation generally increases cortical excitability, while cathodal stimulation decreases it (Schlaug *et al.*, 2008).

- **Functional Electrical Stimulation (FES):** FES uses electrical currents to stimulate muscles affected by stroke, promoting muscle contraction and movement. Also as an alternative to TMS, electrical stimulation can also be applied to the brain, with less pinpoint accuracy but at a much cheaper cost. This technique can be used to improve motor function, reduce spasticity, and prevent muscle atrophy. Moreover, FES can be combined with other stroke rehabilitation therapies to improve overall outcomes (Sinha *et al.*, 2021).

- **Pharmacological and Cellular Therapies**: These therapies include antidepressants, stimulants, dopamine agonists, niacin, memantine, growth factors, monoclonal antibodies,

and stem cells. Additionally, selective serotonin reuptake inhibitors are among the most extensively studied pharmacological agents for stroke recovery due to their antidepressant effects and possible increase in synaptic plasticity following ischemic brain injury. Furthermore, stem cell therapies offer potential neuroprotection and neuroregeneration. They aim to limit initial damage during the acute stroke phase and promote neuronal replacement (Mahla, 2016).

- **Brain-Computer Interfaces:** BCIs enable direct communication between the brain and external devices. By monitoring brain activity and providing real-time feedback, BCIs can help retrain the brain and improve motor function, particularly for patients with severe impairments. BCIs show great promise and their application in stroke rehabilitation is still being researched and developed (Gunduz *et al.*, 2023).

Through these innovative approaches, researchers have also focused on predicting individual stroke recovery outcomes more accurately. For example, a multimodal biomarker-based algorithm has been designed to integrate clinical evaluations with neurophysiological and brain imaging data. When patients demonstrate significant impairment in a basic bedside test of shoulder abduction and finger extension, TMS is used to assess the functional integrity of the corticospinal tract. If TMS produces a motor evoked potential, indicating that the tract remains intact, these patients are expected to achieve notable recovery despite their initial severe impairment. For patients who do not show motor evoked potentials, diffusion tensor imaging is employed to further evaluate the corticospinal tract, aiding in the classification of those with limited recovery potential (Anaya and Branscheidt, 2019).

In the context of stroke rehabilitation, BCIs can be used to monitor brain activity during imagined movements, providing feedback to help retrain the brain and improve motor function. Playing upon the concept of neural plasticity and while repeatedly imagining specific movements, patients can strengthen neural pathways and strengthen their motor recovery (Khan *et al.*, 2020). This work delves into the application of BCIs as a promising tool for optimizing neurorehabilitation after stroke. The following sections will introduce this technology, providing context for the proposed study.

1.2. Brain Computer Interfaces (BCIs)

The field of BCI represents a revolutionary intersection of neuroscience, engineering, and computer science, aiming to create a direct communication pathway between the human brain and external devices. This technology has the potential to transform how we interact with the digital world, offering new alternatives for communication and device control for individuals with severe disabilities, improving cognitive and sensory experiences, and providing innovative solutions for neurorehabilitation and mental health monitoring. The concept of BCIs has its roots in the mid-20th century, starting with Jacques Vidal's work on monkeys (Vidal, 1973), but recent advances in neuroimaging, signal processing, and machine learning have acceler-

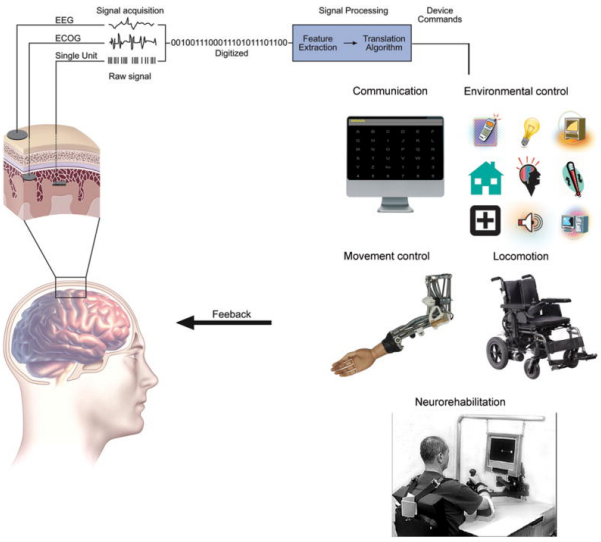


Fig. 1.2. Simplified scheme of tpyical components of a BCI system and communication methods (Kawala-Sterniuk *et al.*, 2021).

ated its development and application. On Figure 1.2, we can view a simplified scheme of the functioning of a BCI system.

1.2.1. Brain activity signal acquisition methods

Understanding and interfacing with brain activity begins with the precise acquisition of neural signals. There is a wide variety of methods to acquire the brain activity signal, classified between invasive and non-invasive techniques.

Invasive methods require previous surgical intervation in order to implant microelectrodes that record brain signals directly from the cortex's surface. The recorded signal captures high-fidelity brain activity without damaging neurons. Since the electrodes are placed inside the skull, it bypasses the filtering effects of the skull and several layers of tissue, leading to superior signal quality. Additionally, invasive methods boast high signal-to-noise ratio, high spatial and temporal resolution, larger signal amplitude, improved artifact rejection, and a broader frequency range. However, due to the surgical nature of this technique and the risk factors involved, it is less often applied in humans losing the edge against non-invasive methods (J. Wolpaw and E. W. Wolpaw, 2012).

Non-invasive techniques don't require any surgical intervention, measuring the brain activity signal through sensors placed on the scalp. The most common non-invasing techniques are (Shih *et al.*, 2012):

- **Near Infra-Red Spectroscopy (NIRS):** Uses near-infrared light to measure changes in blood oxygenation and blood volume in the brain. By shining light into the scalp and detecting the reflected light, NIRS provides information about brain activity based on the hemodynamic response. It offers good spatial resolution and is portable, making it

suitable for real-world applications, though it has limited temporal resolution compared to other methods.

- **Functional magnetic resonance imaging (fMRI):** fMRI measures brain activity by detecting changes in blood flow using magnetic fields and radio waves. This technique provides excellent spatial resolution and allows for the detailed mapping of brain function. However, fMRI is expensive, has poor temporal resolution, and requires the subject to remain still within a large, noisy scanner, limiting its practicality for everyday BCI applications.

- **Magnetoencephalogram (MEG):** Records the magnetic fields produced by neural electrical activity using highly sensitive magnetometers. It offers better spatial resolution than EEG and excellent temporal resolution. MEG is noninvasive and provides precise localization of brain activity. However, it is costly and requires a magnetically shielded room, limiting its accessibility and use in everyday settings.

- **Electroencephalogram (EEG):** EEG is a widely used noninvasive method that measures electrical activity in the brain through electrodes placed on the scalp. It provides good temporal resolution and is relatively inexpensive and easy to set up. While EEG offers limited spatial resolution and is susceptible to noise and artifacts, its portability and ease of use make it a popular choice for many BCI applications.

Particularly for BCI interest, we require a portable, low cost, non-invasive and simple to operate system, hence the common and widespread use of EEG in BCI applications and its selection for the development of our project.

1.2.2. Electroencephalography (EEG)

As previously mentioned, EEG is a non-invasive method used to record the electrical activity of the brain by placing electrodes on the scalp. This electrical activity is the result of the combined signals generated by millions of neurons firing simultaneously, producing rhythmic and oscillatory patterns known as brain waves or rhythms. These rhythms are classified based on their frequency range and relative amplitude. The amplitude of EEG signals, which ranges from a few microvolts to 200 μV, is associated with the synchronization of neuronal activity. High amplitude signals occur when large groups of neurons fire in synchrony, creating constructive interference. Conversely, asynchronous firing results in lower amplitude signals due to destructive interference (J. Wolpaw and E. W. Wolpaw, 2012). The advantages and limitations of this signal acquistion technique can be seen on Table 1.1.

EEG rhythms are divided into several frequency bands, each associated with different brain states and functions. They are usually divided in (Anilkumar and Nayak, 2023):

- Delta (0.5-4 Hz): Dominant during deep sleep and involved in restorative processes.

Advantages	Limitations
Non-invasiveness: EEG does not require surgical procedures, making it safe and relatively easy to use.	Spatial Resolution: EEG has relatively low spatial resolution due to the diffusion of electrical signals through the skull and scalp. This blurring effect makes it difficult to pinpoint the exact source of brain activity.
Temporal Resolution: EEG provides excellent temporal resolution, capturing rapid changes in brain activity with millisecond precision.	Susceptibility to Noise: EEG signals can be contaminated by various artifacts, such as muscle movements, eye blinks, and external electrical interference. This noise can complicate the interpretation of the data.
Portability: EEG equipment is relatively portable, allowing for studies in various settings outside the laboratory.	Depth Limitation: EEG primarily captures activity from the cortical surface and is less effective at detecting signals from deeper brain structures.
Cost-effectiveness: Compared to other neuroimaging techniques like fMRI or MEG, EEG is more affordable and widely accessible.	

Table 1.1. Advantages and Limitations of EEG

- Theta (4-8 Hz): Linked to light sleep, relaxation, and meditation, as well as memory and navigation.
- Alpha (8-13 Hz): Prominent when the eyes are closed and the mind is at rest, particularly over the occipital lobe.
- Beta (13-30 Hz): Associated with active thinking, focus, and problem-solving, as well as anxiety and arousal.
- Gamma (30-100 Hz): Involved in high-level cognitive processing, such as perception, attention, and consciousness.

Additionally, for this study, we are going to make use of an additional band withing the frequency of the SMR wave known as the mu band (12-15 Hz), in between alpha and beta, due to its significance in EEG systems based on Motor Imagery (MI). We will further explore this in chapter 2.

Ultimately, EEG electrodes can be classified into two main categories (Nunez and Srinivasan, 2006):

- Active Electrodes: These electrodes have a preamplifier located near the scalp, which amplifies weak signals before transmitting them to a distant amplifier. Active electrodes reduce noise and improve signal quality but require careful placement to avoid electromagnetic interference.

- Passive Electrodes: Simpler and easier to place, these electrodes connect directly to a distant amplifier without a preamplifier. However, they are more susceptible to noise due to the longer distance the signal must travel.

1.2.3. Types of control signals in BCIs

The efficiency and functionality of BCIs rely heavily on the nature and quality of control signals extracted from the brain. Control signals are crucial because BCIs cannot detect any arbitrary brain activity; they can only recognize specific patterns of neural activity. Therefore, engineering tools are necessary to identify these patterns and use them for practical applications. As stated on subsection 1.2.1, EEG is the method of choice to obtain said signals. Depending on the control signal of interest, the location and number of electrodes necessary for acquisition may vary. These control signals are classified into two main categories (Ramadan and Vasilakos, 2017):

1. **Exogenous Signals:** These signals are generated in response to external stimulation. It's an unconscious response to an exterior stimuli that could in occasions be of discomfort for the subject.

    - Steady State Evoked Potentials (SSEP): SSEPs are brain responses that occur at a fixed frequency in response to a continuous or repetitive stimulus, such as visual, auditory, or tactile stimuli. When a user focuses on a stimulus that repeats at a constant rate, the brain's electrical activity synchronizes with this frequency, producing detectable SSEPs withing the visual cortex in the occipital region.
    - Code-modulated Visually Evoked Potential (c-VEP): In this situation, various visual stimuli are modulated using a pseudorandom code. When a person focuses on a specific stimulus, a c-VEP response is elicited, which can then be utilized to operate the BCI.
    - P300 Evoked Potentials: The P300 signal is an event-related potential (ERP) that occurs approximately 300 milliseconds after the presentation of a rare or significant stimulus, known as the "oddball" paradigm. The P300 is characterized by a positive deflection in the EEG signal, and it is commonly used in BCIs due to its reliability and ease of detection. The structure of a P300 BCI system can be observed in Figure 1.3.

2. **Endogenous signals:** Spontaneous signals refer to brain activity that occurs naturally and continuously without the need for external stimuli. Unlike evoked signals, which

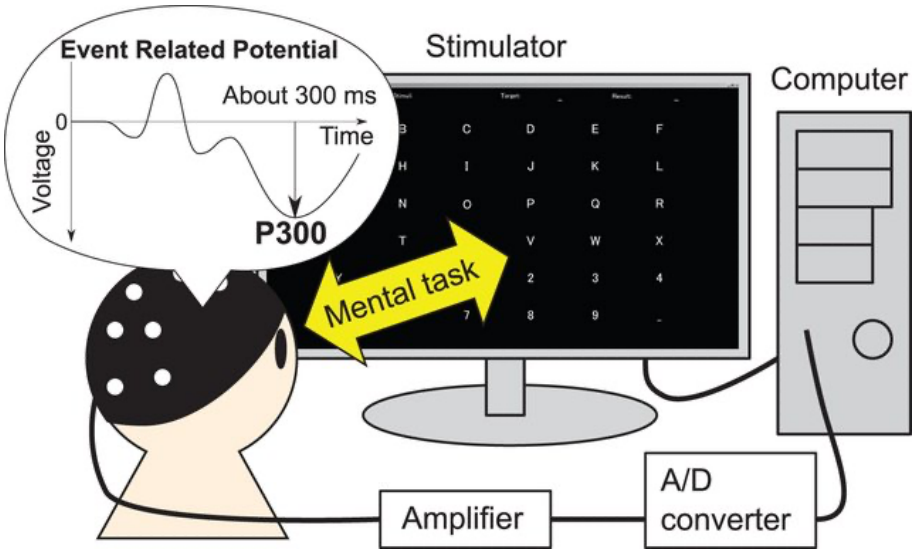


Fig. 1.3. P300 based BCI system structure. The subject is performing a mental task for the BCI system to predict a letter (Onishi and Natsume, 2014)

are triggered by specific events or stimuli, spontaneous signals are intrinsic to the brain's ongoing neural processes.

- Sensorymotor Rhytms (SMRs): SMRs are oscillatory brain activities that are primarily observed in the alpha (8-13 Hz) and beta (13-30 Hz) frequency bands over the sensorimotor cortex. These rhythms are modulated by motor imagery (the mental simulation of movement) or actual motor activity.
- Slow Cortical Potentials (SCP): SCPs are low-frequency EEG signals that fall below 1 Hz. These potentials are detected primarily in the frontal and central regions of the cortex and result from shifts in the depolarization levels of upper cortical dendrites. SCPs are characterized by very slow variations, either positive or negative, in cortical activity, lasting from milliseconds (500ms) to several seconds (10s).
- Non-motor cognitive tasks: Non-motor cognitive tasks refer to the use of cognitive processes, rather than physical or motor activities, to drive the BCI. Various cognitive tasks can be employed, including music imagination, visual counting, mental rotation, and mathematical computation. Back in 1990, Keirn and Aunon (1990) evaluated on 7 subjects a mental arithmetic, mental letter composing, geometric figure rotation and visual counting task in order to prove that they could be distinguishable through the analysis of the subjects brain waves.

1.2.4. Stages of BCI systems

The end goal of a BCI system, to interpret and convert EEG brain signals into commands that fulfill the user's intentions in real-time, is accomplished through a series of through a series of stages: (1) signal acquisition; (2) signal processing, which includes feature extraction, selection and classifying (Shih *et al.*, 2012).

1.2.4.1 Signal Acquisition

Signal acquisition involves capturing brain signals using EEG sensors placed on the scalp. As previously stated, EEG measures the electrical activity of the brain, providing valuable data on brain function. These signals are amplified to levels suitable for electronic processing and may undergo filtering to remove electrical noise or other unwanted signal characteristics, such as 60-Hz or 50-Hz power line interference. After amplification and cleaning, the signals are digitized and transmitted to a computer for further processing.

Regarding the placement of electrodes for EEG recordings, several standardized systems are used to ensure consistent electrode positioning. The 10-20 System is the international standard for EEG electrode placement, where electrodes are spaced 10% or 20% apart relative to the distance between anatomical landmarks (the nasion and the inion). Electrodes are labeled with letters and numbers indicating their location (Fp for frontopolar, F for frontal, C for central, P for parietal, T for temporal, O for occipital). Even numbers belong to the right hemishphere, while odd numbers belong to the left counterpart, while electrodes in the mid-line receive the label 'z'. Additionally, the 10-10 and 10-5 Systems offer higher electrode density, with placements at 10% and 10%, or 10% and 5% intervals, respectively, providing greater spatial resolution (Nunez and Srinivasan, 2006).

1.2.4.2 Signal processing

Signal processing involves two main steps: feature extraction and classification (J. Wolpaw and E. W. Wolpaw, 2012):

- Feature extraction is the process of analyzing digitized EEG signals to identify relevant characteristics that correlate with the user's intentions. These features must be compact and suitable for translation into output commands. Since relevant brain activity is often transient or oscillatory, commonly extracted features in current BCI systems include power within specific EEG frequency bands or EEG response amplitudes and latencies.
- Classification algorithms translate the extracted features into application commands. To do so, you can use direct translations (i.e. a decrease in power within a specific frequency band might be translated into an upward movement of a computer cursor), thresholds, or classification algorithms. The translation algorithm must be dynamic, meaning it can adapt to spontaneous or learned changes in the signal features. This ensures that the full range of the user's potential feature values is adequately covered for device control.

Machine Learning (ML) is a branch of Artificial Intelligence (AI) focused on developing algorithms that can learn from and make decisions based on data. The most common ML methods used in BCIs will be further explained in chapter 2. However, classical ML techniques, present several limitations such as the extensive and labor-intensive manual feature engineering, or the high computational complexity not powerful enough to discover complex patterns in data (Kamath *et al.*, 2018).

Deep learning (DL) is an advanced subset of ML that utilizes neural networks with multiple layers (deep neural networks) to analyze complex data patterns. In the context of signal processing for BCIs, deep learning techniques are increasingly being used to improve feature extraction and processes (Iftikhar *et al.*, 2018). DL models in BCIs, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are distinguished by their ability to autonomously learn and extract intricate features directly from raw EEG data. Unlike traditional ML approaches that rely on manual feature engineering, DL optimizes the extraction and classification of features in an end-to-end manner. This capability is particularly advantageous for interpreting complex patterns within EEG signals, such as temporal dynamics and spatial dependencies associated with user intentions (Miao *et al.*, 2019).

By using DL, BCI systems can achieve better accuracy and robustness in translating brain signals into meaningful commands. This technology also allows the BCI system to adapt more effectively to individual users, as deep learning models can be trained on personalized datasets, ensuring that the system's responses are finely tuned to each user's unique brain signal patterns.

In summary, DL represents a significant advancement in the development of more sophisticated and efficient BCI systems, offering the potential for more intuitive and responsive interfaces.

1.2.5. Applications of BCI

BCI systems have a wide range of applications across various fields, translating brain activity into actionable commands. These applications span from medical and therapeutic uses to non-medical implementations, improving the quality of life and providing new interaction paradigms. BCI systems can be based on how they utilize brain activity, categorized as active or passive. An active BCI involves direct user engagement through intentional modifications in brain activity to control the system. In contrast, a passive BCI decodes brain states by analyzing ongoing brain activity without requiring active user involvement in controlling the BCI system. Additionally, active BCIs, rely on brain waves produced in response to external stimuli (Sibilano *et al.*, 2024). We will briefly discuss some of the most relevant uses of this technology (Padfield *et al.*, 2019; J. Wolpaw and E. W. Wolpaw, 2012):

- **Assistive technologies for disabled individuals:** It is the most common and intuitive perception of application of BCIs in the medical field. BCIs can increase the independence of individuals with severe physical disabilities by providing alternative means to interact with their environment. For example, BCIs can enable users to control wheelchairs, prosthetic limbs, or home automation systems using their brain signals. This empowerment leads to greater autonomy and integration into society, improving both physical and mental well-being. Of particular interest is the application of BCI for individuals within the locked-in syndrome, where voluntary muscle control is severely limited or absent, BCIs offer a vital communication channel (Branco *et al.*, 2021). By detecting brain signals associated with specific thoughts or intentions, BCIs can enable users to select letters,

words, or phrases on a computer screen, allowing them to communicate with others. This application significantly improves the quality of life for patients who otherwise have no means of interaction (Hamzelou, 2022).

- **Endogenous cerebral stimulation or Neurofeedback:** Passive BCIs can be used for managing mental workload. By monitoring brain activity, BCIs can provide insights into a user's cognitive state, such as attention levels or fatigue. This information can be used to optimize work environments, improve learning outcomes, or increase performance in various tasks. For instance, adaptive learning systems can adjust the difficulty of educational content based on the learner's cognitive state, providing personalized and effective learning experiences (Baldwin and Penaranda, 2012). In educational settings, BCIs can be used to monitor students' engagement and comprehension levels. Educators can adjust teaching methods in real-time to better suit the needs of the class, fostering a more effective learning environment. Similarly, when consistently combined with various pedagogical teaching, BCIs can be applied in training programs to improve skill acquisition and performance resulting in better academic results (Jamil *et al.*, 2021). Furthermore, several studies have explored the use of active BCIs for memory training. T. S. Lee *et al.* (2013) developed a BCI training method that utilized EEG patterns to enhance cognitive skills such as immediate and delayed memory, as well as visuospatial and constructional attention. Other studies have also targeted cognitive improvements in older adults. For instance, Gomez-Pilar *et al.* (2014) investigated tasks involving MI, which resulted in improvements in visual perception, expressive speech, and immediate memory.

  More particularly for stroke, Neurofeedback (NFB) techniques aimed at increasing neural plasticity within motor regions of the brain to foster functional improvement are being studied as a promising technique due to its potential as a non-pharmacological and non-invasive alternative for treating non-degenerative brain disorders (Franc *et al.*, 2022). Furthermore, NFB modalities include visual, VR-based, haptic, FES, robotic and multimodal strategies. This innovative technique constitutes the basis of our hyptohesis that will be further developed in section 1.3.

- **Entertainment, gaming and VR:** The integration of BCIs into gaming and VR offers immersive and interactive experiences. By using brain signals to control game characters or interact with virtual environments, users can experience a new level of engagement. The integration of BCIs into entertainment opens up possibilities for innovative gaming experiences and interactive virtual environments that respond directly to users' cognitive states and intentions. This application not only improves entertainment but also has potential therapeutic benefits (Yadav *et al.*, 2020).

1.2.5.1 BCI for Neurorehabilitation in Stroke

BCI systems for neurorehabilitation were initially evaluated using brain signals in stroke patients, where cortical reorganization was observed following BCI-based training (Buch *et*

al., 2008). Since then, Janis J. Daly and J. R. Wolpaw (2008) have categorized these systems according to the learning strategy (see Figure 1.4):

1. Training for normal brain activity: This strategy involves training patients to generate more typical brain activity patterns to control motor functions. The premise is that normalizing brain activity will lead to improved central nervous system (CNS) function and thereby improve motor control. Studies with stroke patients have shown that they can learn to control specific brain activity patterns. BCIs can measure and extract EEG features, translating them into feedback to help patients gain control. Bai *et al.* (2020) systematic review corroborated how numerous clinical studies have shown that BCI technology can induce functional recovery in stroke patients. Additionally, Enzinger *et al.* (2008) reported significant motor function improvements through sensorimotor rehabilitation using BCI training combined with motor imagery.

2. Device-assisted movement: This approach uses BCI output to activate devices that assist movement, such as FES or robotic aids. The hypothesis is that the sensory input generated by the strengthened motor function facilitated by these devices will induce CNS plasticity, leading to improved motor control. Studies have demonstrated the effectiveness of neurorehabilitation training with robotic devices in stroke patients. For example, Janis J Daly *et al.* (2008) conducted promising preliminary work combining BCIs with FES or assistive robotics to aid motor relearning in stroke patients. BCI-based therapy could complement traditional neurorehabilitation methods and potentially reduce costs by minimizing the need for constant supervision by a rehabilitation therapist.

Despite the promising potential of BCIs for neurorehabilitation, several challenges must be addressed (Simon *et al.*, 2021):

- User control and learning: Achieving effective control over the BCI requires significant practice, with individual differences in motor imagery and learning strategies affecting success rates. Multiple practice sessions are often needed for users to gain effective control, affecting to the motivation of the patient and reducing the effectivness of the therapy and some even may never achieve it.

- Technological and practical barriers: Current BCI setups, which often involve cumbersome EEG systems, require skilled operators and are not user-friendly. Advances in wireless, high-impedance dry electrode systems show promise but are still in development. The heterogeneity of brain lesions among patients complicates the development of universally applicable BCI solutions. Adaptive algorithms are needed to tailor BCIs to individual neural activity patterns.

- Mechanistic understanding: There is a lack of understanding of the specific neural mechanisms through which BCIs facilitate functional improvement, complicating the prediction of patient outcomes.

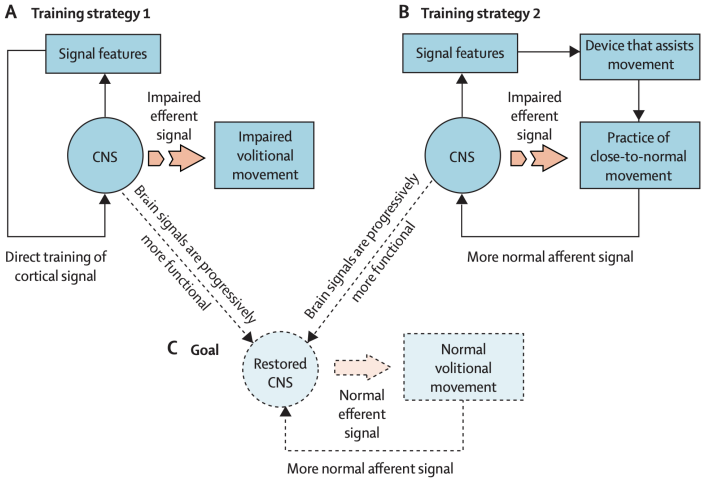


Fig. 1.4. Two BCI-based training strategies aim to promote and guide CNS plasticity to increase motor function: (A) Converts specific brain activity features into actions (e.g., cursor movement) and uses these actions as feedback to train patients to produce more normalized brain activity. The plasticity resulting in this more normal activity will also restore CNS function, thereby improving motor control. (B) Uses specific brain activity features to activate a device that assists movement, compensating for the patient's impaired neuromuscular control during motor tasks. By increasing motor function, the assistance will generate sensory input that induces CNS plasticity, leading to the restoration of more normal motor control. (C) The first strategy focuses on normalizing brain activity with the expectation that this will lead to improved motor function. In contrast, the second strategy uses brain activity to assist in practicing more normal neuromuscular control, with the expectation that the sensory input generated by improved motor function will induce plasticity, thereby improving neuromuscular control (Janis J. Daly and J. R. Wolpaw, 2008).

1.3. Hypothesis of the study

The motivation behind this study stems from the urgent need to improve rehabilitation outcomes for stroke survivors, who often face significant challenges in regaining lost motor functions and achieving optimal recovery. Additionally, this same scenario is applied to other pathologies like progressive diseases, including amyotrophic lateral sclerosis, multiple sclerosis, or Parkinson's Disease. In these others scenarios, motor recovery presents an incredible challenge at present. Despite promising results from novel rehabilitation methods in clinical trials, existing techniques often fall short of restoring normal or near-normal motor function and quality of life for many patients (Janis J. Daly and J. R. Wolpaw, 2008). While traditional rehabilitation approaches are effective to a degree, they may not consistently achieve optimal recovery. Therefore, there is a critical demand for innovative and personalized therapeutic strategies capable of accelerating and improving the rehabilitation process.

Through NFB training, users can learn to control the neural substrates related to specific behavior patterns. Given that strokes frequently trigger upper limb hemiparesis, studying the differences in brain activity patterns generated by executed and imagined movements represents an interesting research area to drive significant advances in NFB-based neurorehabilitation therapy. Currently, these therapies generate brain plasticity in the affected area through Motor Imagery. The main problem with these therapies, especially for stroke patients, is that the patient imagines the movement due to the inability to control the hand or limb. Although there are

certain similarities, the areas and the activity genereted in those areas during the imagination and execution of the movement are not exactly the same.

An accurate characterization of the differences between the execution and imagination of movements would allow for the optimization of these therapies and reduce the training time of classification models through the use of transfer learning. Additionally, by identifying the similarities between both tasks, the system aims to guide the user with feedback to better emulate the patterns of the executed movement, endogenously stimulating more relevant areas.

The project would contribute to continuing the active national project "Platform for neuromotor and cognitive rehabilitation through active therapies in individuals who have suffered a stroke -M3Rob" (Cisnal *et al.*, 2022), which is already being carried out in the Brain Computer Interface research line at the Biomedical Engineering Group at the University of Valladolid (Spain). This project aims to design, develop, and validate a platform for neuromotor and cognitive rehabilitation for stroke survivors, incorporating neurofeedback techniques for cognitive function rehabilitation.

Regarding its application beyond the department, the project's outcomes could be directly applied in hospitals or rehabilitation centers with faster and less costly methods. Furthermore, they could stimulate future research in the field of NFB therapies.

1.4. Aim of the study

The general objective of this project is to better understand the differences and similarities of EEG features for executed and imagined movements and evaluate a transfer learning strategy that allows the application of a ME trained model to a MI database. To achieve this, we have the following specific objectives:

1. To conduct a state-of-the-art review on BCI rehabilitation in stroke and to search public databases containing records of both execution and imagination of movements for the proposed analyses.

2. To characterize of EEG during the execution and imagination of movements. Time-frequency analysis techniques and power topography generation will be used to identify specific patterns associated with each type of task This characterization will allow for a detailed understanding of differences in brain activity between both types of movement.

3. To train of a DL model using records of executed movements. Subsequently, the possibility of applying various transfer learning techniques to adapt the model to recognize imagined movements will be explored.

4. To discuss, draw conclusions from the obtained results, and suggest possible lines of future research.

1.5. Structure of the study

This Final Degree Project is organized into seven chapters.

The first chapter, Introduction, provides a comprehensive overview of stroke and its significant impact on motor functions, which necessitates effective neurorehabilitation strategies. This chapter introduces the concept of BCIs and their potential role in enhancing neurorehabilitation outcomes. The methods for brain activity signal acquisition, particularly EEG, are discussed in detail. Additionally, the chapter presents the different types of control signals used in BCI systems and outlines the various stages involved in the operation of BCI systems. The introduction concludes with a discussion on the application of BCIs in the field of neurorehabilitation, followed by the hypothesis and aim of the study. This section frames the key research questions that guide the investigation, effectively setting the stage for the chapters that follow.

The second chapter, Motor Imagery Based BCIs, delves into the specifics of sensorimotor rhythms and their significance during motor behaviors. It explores the methodologies for detecting MI through EEG signals and discusses the preprocessing, feature extraction, and classification techniques essential for MI-based BCIs. A comparative ME vs. MI analysis is provided, highlighting the use of deep learning techniques for the effective decoding of EEG signals.

In the third chapter, Dataset & Signals, the EEG Motor Movement/Imagery Dataset (Physionet BCI2000) is described in detail. This chapter covers the protocols for signal acquisition and the experimental setup used in the study. It also explains the EEG montage utilized, ensuring a clear understanding of the dataset's structure and the signals analyzed.

The fourth chapter, Methods, outlines the procedures for characterizing and preprocessing the dataset. This chapter presents the MEDUSA framework and the various neural network models employed in the study. Detailed descriptions of the preprocessing steps, the structure of the neural networks, and the process of fine-tuning pre-trained models are provided, offering insights into the methodological approach adopted in the research.

The fifth chapter, Results, presents the findings from the study. It includes an analysis of the characterization of ME and MI, focusing on relative bandpower and event-related desynchronization/synchronization. The classification performance of the neural network models is evaluated, and the impact of transfer learning and fine-tuning on the results is discussed.

In the sixth chapter, Discussion, the results of the study are interpreted, and their implications for stroke rehabilitation are considered. This chapter also addresses the limitations of the research.

The final chapter, Conclusion, summarizes the key findings of the study and presents the overall conclusions drawn from the research. It encapsulates the contributions of the Final Degree Project to the field of BCI-based neurorehabilitation and outlines potential avenues for future research.

2. MOTOR IMAGERY BASED BCIs

2.1. Paradigm

The MI-based BCIs use imagined movements that trigger specific patterns of brain activity that can be detected. By interpreting the brain's electrical activity during imagined motor tasks, these BCIs enable users to interact with technology in innovative ways, bypassing traditional neuromuscular pathways. This section explores the principles, applications, and advancements in MI-based BCIs, highlighting their potential to transform lives through neurorehabilitation and assistive technology.

2.1.1. Brain dynamics during motor execution

ME involves a complex interplay of neural processes that initiate, coordinate, and regulate movement. Particularly, neuronal activation is the cornerstone of ME, involving complex interactions among various brain regions seen on Figure 2.1, to produce and regulate movement.

The main regions which which play integral roles in these neural processed are (Purves *et al.*, 2001; J. Wolpaw and E. W. Wolpaw, 2012):

- Primary Motor Cortex (M1): located in the precentral gyrus of the frontal lobe, is the principal brain region responsible for generating neural impulses that control the execution of voluntary movements. When an individual plans and executes a movement, pyramidal neurons in M1 exhibit a burst of rapid firing. These neurons are characterized by their ability to send long axons down the corticospinal tract to synapse onto motor neurons in the spinal cord. Moreover, M1 is somatotopically organized, meaning different regions of M1 correspond to different parts of the body. For instance, the lateral part of M1 controls movements of the face and hands, while the medial part controls the legs. Moreover, the hands are located in the upper part of the lateral region. During ME, neurons in the relevant area of M1 increase their firing rates to initiate and control the movement of the corresponding body part. This event causes a desynchronization, apparent through a power decrease when activity is increased. Ultimately, neurons in M1 integrate a multitude of excitatory and inhibitory synaptic inputs to produce a coherent motor output. The precise timing and balance of these inputs are crucial for the smooth execution of movements. Excitatory postsynaptic potentials and inhibitory postsynaptic potentials summate to modulate the firing rate of motor neurons, translating into the precise control of muscle contractions.

- Premotor Cortex and Supplementary Motor Area (SMA): The Premotor Cortex, located anterior to the primary motor cortex, is involved in the planning and coordination of

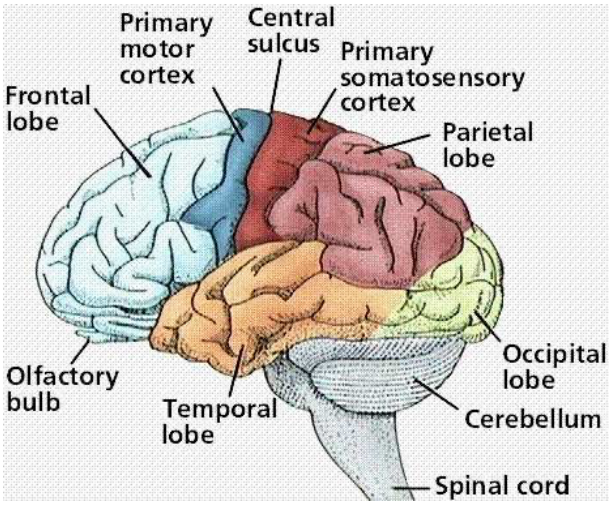


Fig. 2.1. Cortical Areas of Human Brain (Zippo, 2011)

movements. It helps in the preparation of movements by integrating sensory information with motor commands. Meanwhile the SMA, located on the medial aspect of the frontal lobe, is crucial for the initiation and coordination of complex movements, especially those involving sequences or bilateral coordination. This region is essential for tasks that require the simultaneous use of both sides of the body, with neurons that exhibit preparatory activity even before the movement begins, indicating their role in movement planning.

- Basal Ganglia and Cerebellum: The Basal Ganglia is a group of subcortical nuclei, essential for movement initiation, modulation, and the inhibition of unwanted movements. It plays a critical role in motor learning and executing smooth, purposeful movements. Neuronal activity in the basal ganglia reflects the complex decision-making processes involved in initiating movements. Alternatively, the cerebellum is involved in the fine-tuning of movements, coordination, and balance. It receives input from sensory systems and other brain regions to adjust motor output in real-time.

- Motor Pathways: The corticospinal tract is the primary pathway for motor signals from the brain to the spinal cord. It tract transmits impulses from the motor cortex to the motor neurons in the spinal cord, which then innervate the muscles. Additionally, the corticobulbar tract transmits motor signals from the cortex to the brainstem, controlling movements of the face, head, and neck (Biga *et al.*, 2019).

The coordinated firing of neurons in motor-related areas and the modulation of cortical rhythms are captured by EEG, offering a window into the brain's motor control mechanisms. Of particular interest during the execution of movement, and prominent above the sensorimotor cortex, are the mu and beta rhythms previosuly mentioned due to their involvement with the current motor state and the inhibition of new movements.

2.1.2. Sensorimotor Rhytms

Sensorimotor rhythms (SMRs) are oscillations recorded over sensorimotor cortices, including the posterior frontal and anterior parietal areas (J. Wolpaw and E. W. Wolpaw, 2012). EEG commonly captures mu and beta activities, while ECoG and MEG can detect higher-frequency activities. Within these frequency bands, the amplitudes of local rhythms in the pericentral regions decrease significantly or are almost entirely inhibited by movements of the corresponding body part. This phenomenon occurs regardless of whether the movements are active, passive, or reflexive. These blocking effects are observable on both sides of the brain, although they are more pronounced contralateral to the moving limb. This concept regarding the desynchronization during movement through attenuation of brain rhythms is coined Event-Related Desyncronization (ERD). This event is accompanied by a corresponding increase of brain rhythms after the movement, the Event-Related Syncronization (ERS) (Pfurtscheller *et al.*, 2006).

2.1.3. SMR During Motor Behaviors

Studies have shown that voluntary movements are associated with mu and beta ERD localized over sensorimotor cortical areas. Mu rhythms exhibit two ERD patterns: lower frequency mu ERD, which is widespread and reflects general motor preparation and attention, and higher frequency mu ERD, which is topographically and functionally task-specific. Beta rhythms show desynchronization during movement and a brief synchronization (beta rebound) post-movement (J. Wolpaw and E. W. Wolpaw, 2012).

For instance, prior to finger flexion, mu ERD appears over the contralateral Rolandic region and becomes bilaterally symmetrical during the execution of the movement. This phenomenon is illustrated by a clear desynchronization pattern several seconds before movement onset, indicating motor preparation and attentional processes (Pfurtscheller *et al.*, 1996).

This localized mu ERD does not usually occur alone while related to a specific sensorimotor event, meaning it will usually be accompanied by additional ERS in the surrounding cortical areas. This concept is known as the *focal ERD/surround ERS*, as it can be seen in studies such as Gerloff *et al.* (1998), where it was reported that the mu ERD observed during repetitive finger movements was accompanied by a synchronization of the visual alpha rhythm in the parieto-occipital (visual) cortex.

MI, the mental simulation of movement without actual execution, also induces changes in SMRs similar to those observed during actual movements. Imagined movements of the limbs produce ERD patterns in the mu and beta frequency bands in the contralateral sensorimotor cortex, closely mirroring those seen during physical movement. These similarities have been corroborated through studies involving both healthy individuals and patients with impaired motor function (Neuper *et al.*, 2009; Pfurtscheller *et al.*, 2006).

2.1.4. EEG-Based strategies to detect MI

Over the years, diverse EEG-based strategies have been developed to reliably detect these SMR signals. These strategies primarily encompass Operant Conditioning, Machine Learning, and Adaptive Strategies, each with distinct approaches and applications (Ang and Guan, 2017).

Operant conditioning, as seen on Figure 2.2, involves training users to modulate specific EEG features through repeated practice. Users learn to control their SMR by receiving feedback from the BCI system, which helps them understand how their mental imagery affects EEG signals. This strategy utilizes a fixed model comprising parameters for EEG feature extraction, feature selection, and translation into control commands. The operant conditioning approach has been effectively applied in various scenarios, such as two-dimensional cursor control and robotic limb movement. For example, controlling horizontal and vertical cursor movement using mu and beta rhythms, respectively, has been demonstrated in research studies (J. R. Wolpaw and McFarland, 2004).

Alternatevely, the machine learning strategy visible on Figure 2.3 addresses the variability in EEG signals and reduces the extensive training required for operant conditioning. This approach involves creating a subject-specific model during a calibration phase, which is then used for MI detection. Machine learning strategies tailor the BCI model to individual users based on their calibration data. Algorithms such as CSP are used to improve the signal-to-noise ratio (Blankertz *et al.*, 2008), improving the detection of MI.

Ultimately, adaptive strategies continuously update the subject-specific model during use, addressing non-stationarity properties of EEG signals. This approach, provided by Figure 2.4 ensures that the BCI system remains effective even as the user's EEG patterns change over time. Adaptation can be achieved through supervised or unsupervised learning, targeting aspects like spatial filters, feature selection, or classifier parameters. A co-adaptive approach allows both the user and the model to adjust together, improving the discrimination of brain activity over time (Faller *et al.*, 2014). Although adaptive strategies have shown potential in improving accuracy, particularly in real-time applications (Nicolas-Alonso *et al.*, 2015), their effectiveness in BCI rehabilitation remains limited.

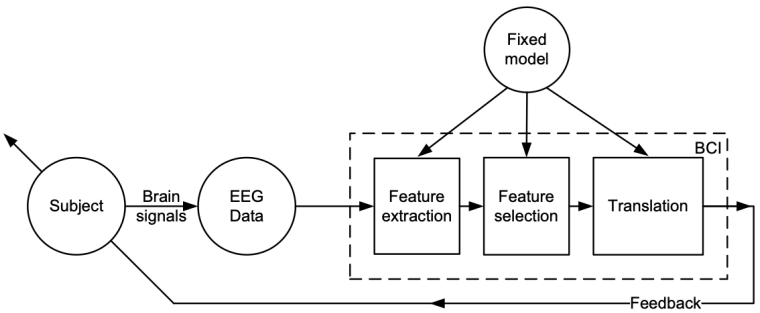


Fig. 2.2. Operant Conditioning strategy scheme (Ang and Guan, 2017).

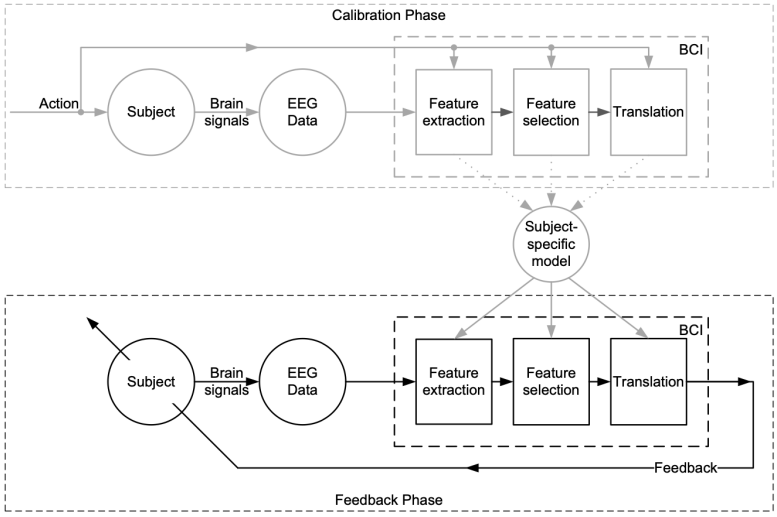


Fig. 2.3. Machine Learning strategy scheme (Ang and Guan, 2017).

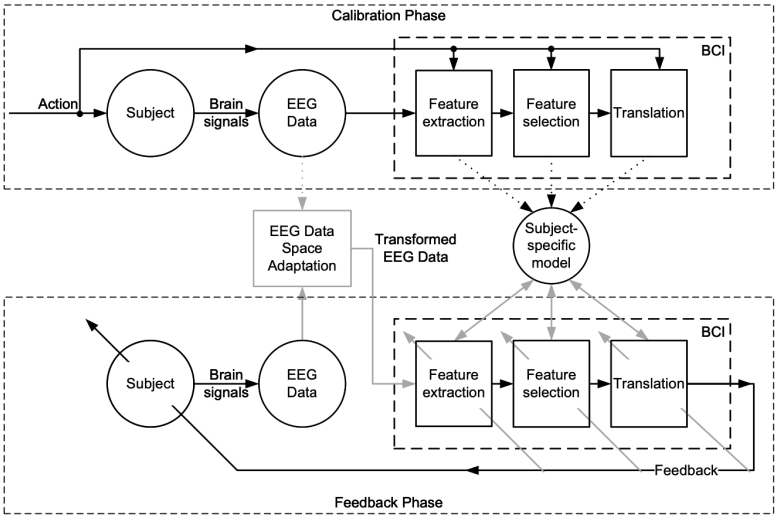


Fig. 2.4. Adaptative strategy scheme (Ang and Guan, 2017).

2.1.5. Signal processing pipeline for MI-Based BCIs

SMR EEG signals are popular in BCI systems due to their naturally occurring discriminative properties and the relatively low cost of signal acquisition. These signals provide valuable information about the user's intended movements without the need for actual physical activity, making them ideal for various applications, including the rehabilitation of stroke patients. However, processing SMR EEG data presents several challenges. The signals, typically obtained from a small training since the training process is time consuming and demanding for the subjects, are inherently unstable and susceptible to noise and artifacts, making accurate classification difficult. Furthermore, the high dimensionality of multichannel EEG data requires effective feature extraction and selection techniques to ensure reliable system performance. Aditionally, many classifiers do not adequately consider time-series information, which is crucial for improving classification accuracy, and the neurophysiological and psychological state of the user significantly affects BCI performance. Factors such as attention, fatigue, and individual differences in

brain activity patterns can impact the detection of SMRs (Lotte *et al.*, 2007).

Regarding the handling of raw EEG data, EEG signal events are frequently identified in grouped data that has been averaged across subjects or trials. However, it can obscure poor performance by presenting averaged results. Other studies focus on single-trial data, avoiding the averaging across trials. This approach is valuable, as it allows for the analysis of variability in performance across trials and offers unique insights into brain activity (Padfield *et al.*, 2019).

2.1.5.1 Pre-processing

Pre-processing of EEG signals usually requires several steps:

- Artifact Removal: targeting the elimination of non-neural signals such as eye blinks, muscle movements, and other artifacts. Common techniques include (Jiang *et al.*, 2019):
  - Independent Component Analysis (ICA): ICA is widely used to separate the mixed signals into independent components, allowing the identification and removal of artifacts. This method leverages the statistical independence of the source signals to decompose the EEG into components that can be individually inspected and filtered out if identified as artifacts. However, ICA is suitable for offline analysis as it cannot be applied in real time.
  - Regression-based methods: These methods use reference signals from known artifacts, such as EOG (electrooculography) for eye movements, to subtract the artifact contribution from the EEG signals.
- Filtering: Artifact removal is accompanied by filtering in order to remove frequency components that are not of interest. This includes bandpass filtering to focus on selected frequency bands of interest depending on the objective of the study. Typically for SMR detection, these frequency filters encompass the mu and beta bands. More importantly a notch filter at 50Hz or 60Hz to remove line noise from electrical sources. Furthermore, in EEG data spatial filters are commonly applied, for processing signals recorded from multiple electrodes placed on the scalp, improving the signal-to-noise ratio and isoleting signals originating from specific brain regions. Particularly, one of the most common spatial filters used in EEG signal processing is the Common Average Reference (CAR). It involves subtracting the average of all electrodes' signals from each electrode's signal, reducing common noise (J. Wolpaw and E. W. Wolpaw, 2012).
- Segmentation: This stage involves dividing the continuous EEG signal into epochs or time windows corresponding to specific events or tasks. This step is essential for analyzing the EEG data in a context-dependent manner. For MI-based BCIs, segmentation typically aligns with the onsets and end periods when the user is instructed to perform motor imagery tasks. These tasks may be executed during different time intervals to extract different segments, as seen in Zich *et al.* (2015), lasting between 2 and 4 seconds so segmentation of the signal is performed in intervals of 2.5-4.5 seconds after the onset.

Additionally, it could include segmentation of the recording 3.5 to 0.5 s before fixation of the onset (Zich *et al.*, 2015).

- Normalization: Not always performed, as it depends on the particularity of the study. It standardizes the EEG data, often by z-score normalization, to ensure that the data from different channels and trials are on a comparable scale. Since EEG signals are non-stationary and can vary with each recording session, direct data comparison is hard. Normalization enables the comparison of EEG data across different recording sessions and subject by transforming the EEG data into a standardized distribution with a standard deviation of 1 and a mean of 0. This step is particularly important in multichannel EEG setups to mitigate the effects of varying signal amplitudes across channels (J. Wolpaw and E. W. Wolpaw, 2012).

Aditionally to the standard EEG pre-processing techniques, different studies have suggested other advanced methods in order to improve the accuracy and strength of BCI systems. For example, Multiscale Principal Component Analysis (MSPCA) is a technique which combines the strengths of PCA and wavelet transform to decompose the EEG signals at multiple scales, improving artifact removal while preserving the sharp transitions characteristic of neural activity (Kevric and Subasi, 2017; Padfield *et al.*, 2019).

2.1.5.2 Feature Extraction

After pre-processing the signal, our extracted features must capture prominent signal characteristics that can serve as a basis for distinguishing between task-specific brain states. Some of the most used techniques in MI-based BCIs are (Padfield *et al.*, 2019):

- Fime-Domain Features: Autoregressive (AR) modeling is frequently used for feature extraction in EEG analysis. This technique involves fitting an AR model to EEG data segments, using the resulting AR coefficients or spectrum as features. Additional methods include root-mean-square and integrated EEG analysis.

- Frequency-Domain Features: They involve transforming the EEG signal into the frequency domain to analyze its spectral properties. Common methods include:

  - Power Spectral Density (PSD): This measures the power distribution of the EEG signal across different frequency bands. It helps in identifying the dominant frequencies associated with specific brain activities.

  - Band Power: The power within specific frequency bands (e.g., delta, theta, alpha, beta, gamma) is computed. For MI tasks, the alpha, mu, and beta bands are particularly informative.

  - Fourier Transform: This is used to convert the time-domain signal into its frequency components, providing insights into the signal's spectral content.

- Time-Frequency Domain Features: These features provide a combined view of time and frequency information, which is particularly useful for non-stationary EEG signals.

  - Short-Time Fourier Transform (STFT): This method divides the signal into short segments and applies the Fourier transform to each, offering a time-resolved frequency analysis.
  - Wavelet Transform: Wavelet transform decomposes the EEG signal into time-frequency representations, enabling the detection of transient features and changes at different scales.
  - Empirical Mode Decomposition (EMD) is a powerful data-driven method used to decompose EEG signals into intrinsic mode functions (IMFs). Each IMF represents a simple oscillatory mode embedded within the signal. The EMD process is particularly suited for non-linear and non-stationary signals like EEG.

- Spatial Features: Spatial features take advantage of the multi-channel nature of EEG recordings to extract information about the spatial distribution of brain activity.

  - Common Spatial Patterns (CSP): CSP is a popular method for spatial filtering that maximizes the variance difference between two classes of EEG data. It is widely used in MI-based BCIs.
  - Filter Bank Common Spatial Patterns (FBCSP): FBCSP improves the traditional CSP method by applying it to multiple frequency bands. This approach captures more detailed information by filtering the EEG signal into several sub-bands before applying CSP to each one. The features from all sub-bands are then concatenated, improving classification accuracy.
  - Riemannian Geometry: This method involves representing the covariance matrices of EEG signals on a Riemannian manifold. By treating these matrices as points on the manifold, advanced mathematical tools can be used to compute distances and extract features.

- Event-Related Features: ERD and ERS provide valuable insights into the temporal and spectral characteristics of brain activity during motor tasks. Temporal analysis of ERD and ERS examines the time course of EEG power changes, offering insights into the timing of neural processes associated with motor preparation, execution, and termination. Spectral power analysis identifies reductions in mu and beta bands during motor imagery (ERD) and increases in power following the task (ERS). Time-frequency representations, such as spectrograms, visualize how the power in different frequency bands evolves over time, aiding in identifying periods of ERD and ERS and understanding their temporal dynamics.

2.1.5.3 Feature selection and classification

Feature selection involves choosing the most relevant features from the extracted data to improve the classifier's performance. This step reduces computational complexity and increases classification accuracy. Techniques like Principal Component Analysis, filter bank selection, and evolutionary algorithms are commonly employed (Padfield *et al.*, 2019). These methods help in identifying and retaining the most informative features while discarding redundant or non-informative data.

The classification process in MI EEG-based BCIs involves using machine learning algorithms to categorize the selected features into specific motor imagery tasks. Various classifiers are used, including (Padfield *et al.*, 2019):

- Linear Discriminant Analysis: Known for its simplicity and effectiveness in low-dimensional feature spaces.
- Support Vector Machines: Effective in high-dimensional spaces and robust against overfitting.
- k-Nearest Neighbors: Simple but computationally intensive, often used when the number of features is small.
- Recurrent Neural Networks: Suitable for time-series data, capturing temporal dependencies in EEG signals.
- Deep Learning Approaches: CNNs integrate feature extraction, selection, and classification into a single processing block, providing a comprehensive solution for handling complex EEG data. We will further explore this approach in the next section.

2.1.6. BCI uses of SMRs

The usage of SMRs for BCI purposes have been developed in various ways, primarily exploiting the observation that SMRs change with motor imagery as well as with actual movements. Users typically imagine specific actions, and the resultant changes in SMR amplitudes are translated into outputs such as cursor movement or control commands.

SMR-based BCIs generally use motor imagery, but they vary significantly in how they incorporate this imagery. Some BCIs use imagery primarily as a starting point, allowing the user to gain initial control. Over time, these BCIs rely on the mutual adaptation of the user and the system, with the user's control becoming more automatic and less dependent on explicit imagery. Other BCIs maintain reliance on specific imagery or mental tasks throughout their operation, requiring continuous recalibration to accommodate any changes in SMR patterns (J. Wolpaw and E. W. Wolpaw, 2012). The different approaches to incorporating motor imagery in SMR-based BCIs are reflected in various applications:

- Cursor Movement in One or More Dimensions (J. R. Wolpaw *et al.*, 1991): SMR-based BCIs have been widely used for cursor control on computer screens. This application often begins with users imagining movements that modulate the amplitude of their mu-rhythm, which is recorded over sensorimotor areas. Over time, the control becomes more automatic and less reliant on specific imagery. However, this use is now obsolete in advantage of better paradigms for this application.

- Communication Applications: SMR-based BCIs facilitate communication for individuals with conditions like ALS (Kübler *et al.*, 2005) by enabling them to select letters or words through imagined movements. This use is also obsolete in advantage of better paradigms.

- Control of Robotic Limbs: SMR-based BCIs can control robotic limbs, allowing users to perform tasks despite paralysis or limb loss. Pfurtscheller *et al.* (2000) developed an orthosis that closed and opened the subject's paralyzed hand. Users imagine specific movements of the paralyzed or missing limb, and the BCI translates these imaginations into movements of a robotic arm or leg. This application improves the independence and quality of life for individuals with amputations or paralysis by enabling them to perform daily activities such as eating and manipulating objects.

- Virtual Reality and Gaming: VR environments and gaming are promising areas for SMR-based BCIs, merging rehabilitation with entertainment. It also facilitates an environment for testing BCI applications that would otherwise be very expensive, hazardous or currently physically not possible. Users navigate virtual environments or control game characters through motor imagery, with the BCI interpreting SMR changes to move avatars or interact within the virtual world. This use provides therapeutic advantages by offering engaging environments for rehabilitation. It also offers recreational benefits, improving mental well-being and quality of life for users with motor disabilities (Coogan and He, 2018).

- Neurorehabilitation: As stated in subsubsection 1.2.5.1, MI has been extensively utilized for BCI control by decoding user intentions from changes in the SMR rhythms of the EEG. SMR-based BCI, have garnered increasing attention for rehabilitation purposes by enhancing the natural output of the CNS (Yuan and He, 2014). Real-time feedback from these devices reinforces correct MI patterns. This approach is particularly beneficial for stroke survivors and individuals with spinal cord injuries, as it aids in regaining motor functions and improving overall quality of life (Cantillo-Negrete *et al.*, 2023).

2.1.7. Motor Imagery vs Motor Execution

MI and ME are fundamental components in the study of BCIs) and neurorehabilitation. The interplay between both paradigms is particularly significant for SMR-based BCIs, as both processes elicit similar sensorimotor rhythms, albeit with varying magnitudes and spatial distributions.

There are different types of motor imagery (Yang *et al.*, 2021; Ladda *et al.*, 2021):

- Visual Motor Imagery (VMI): VMI involves the mental visualization of movements, which can be further categorized based on the perspective of the imagery: first-person and third-person perspectives (Yu *et al.*, 2016).
  - First-Person Perspective (Internal Visual Imagery): individuals imagine the movement as if they are performing it themselves, seeing the actions through their own eyes. For instance, when visualizing lifting an arm, they would see their arm moving up from their own vantage point. First-person VMI primarily engages the motor cortex, premotor areas, and SMA, suggesting a strong overlap with the neural processes underlying actual movement.
  - Third-Person Perspective (External Visual Imagery): individuals imagine watching themselves perform the movement from an outsider's viewpoint, akin to viewing a video of themselves. For example, they would visualize their entire body performing a task as if they were observing from a few meters away. Third-person VMI activates visual processing areas, such as the occipital cortex, in addition to the motor-related regions. This broader activation pattern reflects the involvement of visual spatial processing and observation of movement dynamics.
- Kinesthetic Motor Imagery (KMI): Kinesthetic Motor Imagery focuses on the internal sensations associated with movement, including muscle tension, joint position, and spatial orientation. Unlike visual imagery, kinesthetic imagery emphasizes "feeling" the movement rather than "seeing" it. For instance, when imagining lifting a weight, an individual would focus on the sensation of muscle contraction and the weight's resistance. KMI engages the somatosensory cortex, which processes sensory information from the body, along with the motor cortex (Ridderinkhof and Brass, 2015).

In this study, we focus on KMI due to it's extensive research and available studies regarding the MI paradigm and its similarity to the actual execution of movement, supporting our project regarding the application of ME to predict MI.

In Miller *et al.* (2010), they quantitatively establish that the spatial distribution of local neuronal population activity during motor imagery mimics the spatial distribution of activity during actual motor movement. By comparing responses to electrocortical stimulation with imagery-induced cortical surface activity, they demonstrate the role of primary motor areas in movement imagery. The magnitude of imagery-induced cortical activity change was around 25% of that associated with actual movement. However, when subjects learned to use this imagery to control a computer cursor in a simple feedback task, the imagery-induced activity change was significantly augmented, even exceeding that of overt movement. In Chen *et al.* (2021), they came to the conclusion that there were no significant difference in ERD and laterality index between the motor attempt and MI tasks in the 8–30 Hz frequency bands. Batula *et al.* (2017) performed a fNIRS study to explore cortical activation differences between ME and MI for both upper and

lower limb, coming to the conclusion that ME related with higher activation levels, a faster response, and a different spatial distribution, which should be taken into account in the design of a MI-based BCI. Additionally, when comparing right versus left tasks, upper limb movements are the most clearly distinct. Dekleva *et al.* (2024) conducted a study to record intracortical activity from the motor cortex of two people with residual wrist function following incomplete spinal cord injury to evaluate the specific activity patterns and temporal dynamics within the motor cortex. Their results suggest that during MI, motor cortex maintains the same overall population dynamics as during ME by recreating the missing components related to motor output and/or feedback within a unique imagery-only subspace. Hardwick *et al.* (2018) conducted a meta-analysis comparison of the neural correlates of action between ME, MI and Action Observation with over 18000 participants. The study observed that ME and Action Observation recruited similar premotor-parietal cortical networks. Additionally, MI and ME recruited a similar subcortical network, while Action Observation did not consistently recruit any subcortical areas. Another study performed by Kraeutner *et al.* (2014), recorded MEG with electromyography (EMG) to further establish MI as a secondary modality of skill acquisition by providing electrophysiological evidence of an overlap between brain areas recruited for motor execution and imagery. Their results at source-level analysis showed that MI has similar patterns of spatial activity as ME, including activation of contralateral primary motor and somatosensory cortices. Yi *et al.* (2013) performed a study to investigate the differences of the EEG patterns between simple limb motor imagery and compound limb motor imagery, and discuss the separability of multiple types of mental tasks. Coming to the conclusion that there exists separable differences between simple limb motor imagery and compound limb motor imagery, which can be utilized to build a multimodal classification paradigm in MI-based BCI systems. In Lubbe *et al.* (2021) a version of the Go/NoGo discrete sequence production paradigm, which typically involves presenting participants with a series of stimuli to which they must respond (Go) or withhold a response (NoGo), was employed to compute band ERD in ME, MI and Motor Preparation (MP). Their findings suggest that MI showed increased frontal theta power, which could be caused by increased effort, and ME and MP showed decreased posterior alpha power, which could indicate increased visual attention. The end results indicate positive results for the motor-cognitive model, as it requires extra involvement of frontal executive processes during MI.

Consequently caused by all the research available, it requires tools for monitoring and controlling imagery performance. Some of the most relevant are mentioned on Table 2.1 and include:

Method/Tool	Explanation
KVIQ [a]	Evaluates the clarity and vividness of mental visualizations and sensory experiences related to general movements, particularly useful for elderly individuals or patients
TKBV [b]	Assesses the capability to mentally simulate and manipulate motor representations by performing six specified movements mentally
MiScreen	A mobile application designed to evaluate an individual's ability to perform motor imagery
Chronometry	Measures the time taken for imagined and actual movements to evaluate their synchronization
Autonomic Nervous System Measures	Records physiological responses such as heart rate, respiration, and skin conductance to gauge the physical effort associated with imagined movements
Central Nervous System Measures	Utilizes transcranial magnetic stimulation to compare motor-evoked potentials in the target muscle with reference muscles
EMG	Electromyography recordings monitor muscle activity to detect unwanted movements during motor imagery tasks in experimental settings
Neurofeedback via EEG	Uses EEG to assess brain activity related to motor imagery and provides real-time feedback for performance improvement

Table 2.1. Methods and tools for assessing motor imagery (Ladda *et al.*, 2021).

[a] KVIQ: The Kinesthetic and Visual Imagery Questionnaire

[b] TKBV: Test zur Kontrollierbarkeit der Bewegungsvorstellungsfähigkeit (Testing the Assessability of Movement Imagery Ability)

2.2. Deep Learning

Deep Learning (DL) is a subfield of Artificial Intelligence (AI) and ML that focuses on using neural networks with many layers, known as deep neural networks, to model complex patterns in large datasets. AI encompasses a broad range of techniques that simulate human intelligence, including learning, reasoning, and self-correction, through well-defined algorithms (Ávila-Tomás *et al.*, 2021). Over the past decade, AI has gained substantial attention in both the scientific community and the mainstream media. It falls within the realm of data science, which also includes classical programming and machine learning techniques (Choi *et al.*, 2020).

DL, a specific type of machine learning, allows computers to improve incrementally with data and experience. It achieves impressive results by representing real-world problems as computational models, building complex concepts from simpler ones. These end-to-end models

streamline the process by combining feature extraction and classification into a single system. As a result, they can automatically optimize features during training, eliminating the need for extensive domain knowledge to manually design characteristics (Alzubaidi *et al.*, 2021). In an upcoming section we will discuss the most relevant architectures regarding our project.

2.2.1. Neural-Networks

Artificial Neural Networks (ANNs) are mathematical models inspired by the structure of the human brain. They replicate how biological neurons transmit information to each other. ANNs consist of layers of nodes or artificial neurons, including an input layer, one or more hidden layers, and an output layer. Each artificial neuron, is connected to others with associated weights and thresholds. Finally, if a neuron's output exceeds its threshold, it activates, sending data to the next layer (Walczak and Cerpa, 2003).

CNNs are a powerful architecture within neural networks, particularly effective for human activity recognition. As it can be seen on Table 2.2, CNN are the most common approach within studies to evaluate deep learning models for EEG decoding within the BCI framework. A CNN consists of an input layer, one or more convolutional and pooling layers, followed by fully connected layers. The convolutional layer in a CNN is responsible for extracting various features from input data. It achieves this by applying multiple filters (or kernels) to the input data through a process called convolution, computing a weighted sum of the values within the convolutional kernel, adding a bias, and passing it through an activation function to form feature maps (Indolia *et al.*, 2018).

Following the convolutional layers, pooling layers are used to reduce the spatial dimensions of the feature maps while retaining important information. Pooling operates by dividing the feature map into non-overlapping or overlapping regions (e.g. 2x2 window) and applying a pooling function such as max pooling or average pooling within each region. Max pooling selects the maximum value from each region, thereby emphasizing the most prominent features, while average pooling computes the average. This downsampling reduces the computational load and the number of parameters in the network, leading to improved efficiency and some degree of translation invariance. The fully connected layer, also known as the dense layer, follows the convolutional and pooling layers in a CNN architecture. It serves to integrate the high-level features learned by the convolutional and pooling layers into class scores or predictions. Each neuron in the fully connected layer is connected to every neuron in the previous layer, forming a dense network. The input to each neuron is multiplied by a weight matrix and then summed with a bias term before passing through an activation function. This process allows the network to learn complex relationships between features and produce final output predictions (Indolia *et al.*, 2018).

2.2.2. Inception modules

Inception modules, introduced in the GoogLeNet architecture (Szegedy *et al.*, 2015), represent a significant advancement in CNN design. They address the challenge of choosing the appropriate kernel size for convolutional layers by incorporating multiple kernel sizes within a single layer, capturing features at various scales simultaneously. An inception module includes several parallel convolutional and pooling layers with different kernel sizes, followed by concatenation of their outputs. This structure enables effective learning and processing of spatial hierarchies of features. The variety of operations withing this method allows the network to handle information at multiple scales, which is advantageous for identifying objects with significant scale variation.

This design has demonstrated an ability to produce more detailed feature maps with reduced computational expense, improving performance while maintaining acceptable training and testing durations (Francois Chollet, 2017). In this context, EEG comprises transient and oscillatory patterns of varying temporal scales that reflect the brain's active processes, thus making multi-scale analysis with Inception modules particularly appropriate for these data.

The inception architecture has evolved through several iterations, each improving performance and efficiency (Raj, 2019):

- Inception v1: The original design, employing 1x1 convolutions for dimensionality reduction.
- Inception v2 and v3: Introduced batch normalization and factorized convolutions to reduce computational cost and enhance training speed.
- Inception v4 and Inception-ResNet: Combined inception modules with residual connections to mitigate the vanishing gradient problem and facilitate deeper network training.

2.2.3. Regularization Techniques

In deep learning, constructing and training models often leads to overfitting or underfitting due to the numerous parameters involved. Regularization techniques aim to mitigate these issues and improve model performance. The most frequently used are (Wang *et al.*, 2024):

- Dropout: Prevents overfitting by randomly and temporarily deactivating neurons in hidden layers, reducing parameter estimation and network complexity.
- Batch Normalization: Adds an extra step of normalizing activations between mini-batches, accelerating network training.
- L1 and L2 Regularization: Updates the loss function with additional parameters. L2 regularization adds the sum of squared weight norms, while L1 adds the sum of absolute weight values, reducing the influence of less important weights.

- Early Stopping: Stops training when the validation loss starts increasing, preventing overfitting. It can be done by either restarting the model with all training data or continuing training until a new minimum error is achieved.

- Reduce Loss on Plateau: Reduces the learning rate when the metric of interest stagnates, helping the model to fine-tune its performance.

- Data Augmentation: it artificially creates new data from an existing source, in order to train new models.

2.2.4. Validation techniques

Neural networks are trained with datasets to learn and improve their accuracy over time. The data is divided into three sets: the training set, the validation set, and the test set. Each set has a specific purpose. The training set is used for learning, where the model adjusts its parameters to minimize prediction errors. The validation set is used to tune hyperparameters and mitigate overfitting by evaluating the model's performance during training. It helps ensure the model generalizes well to new data. The test set is reserved for evaluating the model's final performance after training and validation, providing an unbiased assessment of its accuracy. To ensure robust model evaluation, different validation schemes are employed. Validation schemes for neural networks include within-subject (intra-subject) and cross-subject (inter-subject) approaches. Within-subject validation uses part of a subject's data for training and the remaining data of the same subject for testing, repeating this process for each subject in the dataset. Cross-subject validation, on the other hand, can follow different schemes (Wang *et al.*, 2024):

- Leave-n-subjects-out: Also known as subject-wise data split in the medical field, this method selects data from a specific number of subjects for the training set and uses data from the remaining subjects for the test set. This scheme is ideal for scenarios where BCI systems need to predict unseen subjects' data such our case in the study, assessing the model's ability to generalize to new subjects.

- Mix-up-all-subjects: This method involves combining training data from all subjects and using the remaining data for testing. Though less frequently used, it is considered an easier classification task compared to leave-n-subjects-out.

- Random-selection: Also called record-wise data split, this method randomly selects a specific ratio of data from the whole dataset for training and the remaining data for testing. This can result in training and test sets containing data from different subjects, both sets containing data from all subjects, or a mix of seen and unseen subjects.

Furthermore, k-fold cross-validation involves splitting the data into k parts, training the model k times , each time using $k - 1$ parts for training and the remaining part for validation. This method reduces the variance of the model's performance estimation (Jung and Hu, 2015).

Finally, evaluating the performance of deep learning algorithms requires a suite of metrics to provide a comprehensive view. Key metrics include (Wang *et al.*, 2024):

- Confusion Matrix: A matrix representation of prediction results for unseen data, effective for multi-class problems. Offers a detailed breakdown of prediction results for multi-class problems, showing the frequency of true positives, false positives, true negatives, and false negatives.

- Accuracy: The percentage of correct predictions. It may not be suitable for imbalanced datasets.

- Recall (Sensitivity): The proportion of correctly classified positive cases (true positives).

- Specificity: The proportion of correctly classified negative cases (true negatives).

- Precision: The number of true positives compared to the total number of positive predictions.

- F1-Score: it is the harmonic mean of recall and precision, is particularly useful for imbalanced datasets as it balances the trade-off between precision and recall.

2.2.5. Neural Networks for EEG decoding

One major counterpart of BCIs is the decoding accuracy of the EEG. Classical Machine Learning approaches usually require calibration runs for the session being performed due to the inter-subject variability and the intersession variability (Saha and Baumert, 2020). On the other hand, DL models outperform the latter, while reducing the inter-subject and intersession variability due to the ability to apply transfer learning techniques. Schirrmeister *et al.* (2017) and Lawhern *et al.* (2018) proved across a variety of paradigms the ability of CNN architectures for EEG decodification.

Table 2.2 provides a comprehensive review of the most relevant models for decoding EEG.

Model	Type	Architecture & Key Features	Advantages
CNN-BLSTM [a]	CNN + RNN	1D Convolutional layers for spatial feature extraction followed by 2 Bidirectional Long Short-Term Memory (BLSTM) layers for temporal pattern detection. Combines RNN with CNN.	Captures both spatial and temporal features, good for sequential data analysis.
DeepConvNet [b]	CNN	4 convolutional blocks with max-pooling layers, with a special first block designed for EEG input. Ends with a dense softmax classification layer.	Ability to decode task-related information from the raw EEG without handcrafted features.
EEGNet [c]	CNN	Compact CNN tailored for EEG with depthwise and separable convolutions to tie spatial filters directly to a temporal filter. Includes batch normalization, dropout, and average pooling for dimensionality reduction.	Compact and efficient, robust against overfitting, suitable for real-time applications.
ShallowConvNet [d]	CNN	Shallow Convolutional Network focusing on frequency-related features. Includes fewer layers, suitable for low-complexity tasks. In comparison with DepthConvNet, the temporal convolution of the shallow ConvNet had a larger kernel size (25 vs 10), allowing a larger range of transformations in this layer, followed by a squaring nonlinearity, a mean pooling layer and a logarithmic activation function.	Ability to decode band power features, learning a temporal structure of the band power changes within the trial.
EEG-Inception [e]	CNN	Incorporates inception modules within a CNN architecture, facilitating the temporal multi-scale extraction of feature maps.	Captures features at various scales, robust, improved performance over traditional and simpler CNNs.
EEGSym [f]	CNN	Includes inception modules and residual connections to improve spatial feature extraction. Incorporates brain symmetry through mid-sagittal plane in architecture. Includes patch perturbation, hemisphere perturbation and random shift for data augmentation.	Improved accuracies on inter-subject MI binary classification.

Table 2.2. Review of neural networks decoding EEG

[a] CNN-BLSTM: Santamaría-Vázquez *et al.* (2020)
[b] DeepConvNet: Schirrmeister *et al.* (2017)
[c] EEGNet: Lawhern *et al.* (2018)
[d] ShallowConvNet: Schirrmeister *et al.* (2017)
[e] EEGInception: Santamaria-Vazquez *et al.* (2020)
[f] EEGSym: Perez-Velasco *et al.* (2022)

As seen on Perez-Velasco *et al.* (2022), EEGSym and EEGInception achieved the highest metrics on inter-subject MI binary classification between all the models provided above. Additionally, due to their robustness, ability to capture features at multi-scale level through inception modules and ability for cross subject transfer learning and datasets, we selected EEG-Inception and EEGSym for testing in our project.

2.2.6. Transfer learning from ME to MI

In order to increase efficiency and in search of better real-word applications of these neural networks, we seek to apply transfer learning techniques to our selected model. The objective is to obtain a pre-trained neural architecture trained with ME data, to be applied to MI subjects.

As previously seen in studies in Table 2.2, numerous studies have demonstrated that deep CNNs are capable of effectively extracting features from MI EEG data and achieving high classification and recognition accuracy. However, the benefits of neural networks are contingent on the availability of large training datasets. In the field of BCI, the application of CNNs is challenging due to the low SNR and the limited size of EEG data samples. Alternatively, traditional machine learning methods usually necessitate the collection of a substantial amount of EEG signals before MI-EEG recognition can be performed. This process significantly increases the user's calibration time and diminishes the overall user experience of the system (Zheng and Lin, 2024). Furthermore, EEG signals exhibit individual differences and possess a low SNR, making them highly susceptible to external environmental influences, other physiological signals (eye movements, electrocardiograms, and electromyography), and the user's mental state. Consequently, classification models trained on data from different users or at different times often lack generalizability, presenting a major challenge for the practical implementation of MI-BCI systems. Transfer learning originated as a solution to this problem, applied to certain scenarios in which obtaining training data that matches the feature space and predicted data distribution characteristics of the test data is difficult and expensive (Weiss *et al.*, 2016).

Simply explained, transfer learning aims to make use of the available knowledge/data from a particular source domain in order to help solve a task in another target source. Similarly, deep transfer learning (DTL) aims to use the knowledge obtained from another task/dataset to reduce the costs of learning (Iman *et al.*, 2023).

According to Weiss *et al.* (2016) there are various different classifications of transfer learning, most of which can also be applied to DTL (Iman *et al.*, 2023).

As seen in figure Figure 2.5 these categories are divided amongst Problem Categorization and Solution/Approach Categorization.

Furthermore, the Solution Categorization is further subdivided in four main categories (Weiss *et al.*, 2016):

- Instance-Based Approaches: These methods utilize selected instances from the source data, applying various weighting strategies to adapt them for use with the target data.

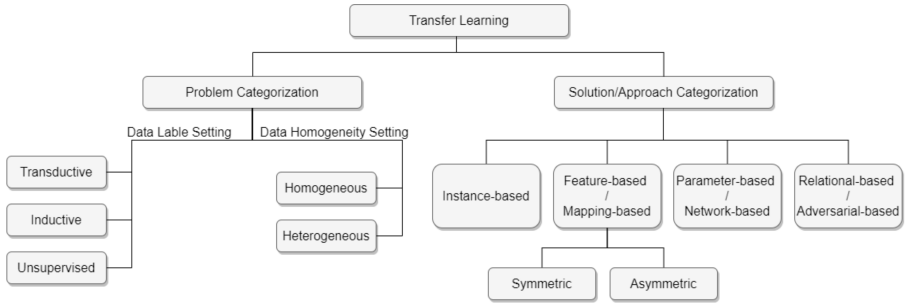


Fig. 2.5. Taxonomy of transfer learning (Iman *et al.*, 2023)

This approach is useful when there is some overlap in the distributions of source and target data.

- Feature-Based/Mapping-Based Approaches: These techniques involve mapping instances or features from both source and target data into a more homogeneous feature space. This category is further divided into:
  - Asymmetric Approaches: These transform source features to match the target features.
  - Symmetric Approaches: These find a common latent feature space and transform both source and target features into a new feature representation that is more comparable.
- Model-Based/Parameter-Based Approaches: These methods leverage the knowledge obtained from pre-trained models. They involve various strategies, such as freezing certain layers, fine-tuning others, or adding new layers to the pre-trained model, to adapt it to the target task. This approach is particularly beneficial when the source and target tasks are closely related. The pre-trained model's learned features can be directly used to process new data. The model's layers act as a fixed feature extractor, and a new classifier is trained on these features for the target task. This approach is quick and useful when the target dataset is small.
- Relational-Based/Adversarial-Based Approaches: These focus on extracting transferable features by leveraging logical relationships or rules from the source domain. Methods inspired by generative adversarial networks are often used to achieve this, as they can generate realistic data representations that bridge the gap between source and target domains.

Through our characterization of the dataset, we aim to prove the close relation between the source (ME) and target (MI) tasks in order to apply a Model/Parameter-Based Approach transfer learning. Due to the domain adaptation between the data, adjusting the model, this method is the most applied technique in DTL (Iman *et al.*, 2023).

Within this Model-Based approach transfer learning, there are several different types of algorithms applied, which are generally a fusion between pre-training, freezing, fine-tuning

and/or adding of layers (Weiss *et al.*, 2016).

A deep learning network trained on source data is referred to as a pre-trained model. This model comprises pre-trained layers that have already learned features from the source data. The term 'Freezing' involves keeping the parameters or weights of certain layers constant. These layers retain the knowledge gained from the source data without any updates during the training on target data. This technique is particularly handy when the initial layers capture general features that are likely to be relevant to the new task. Moreover, 'finetuning' entails initializing the parameters of the pre-trained model with the learned values instead of random initialization. These parameters are then adjusted during training on the target data. Typically, higher layers are more tuned to the specifics of the original task, so fine-tuning might involve re-training only the deeper layers or the entire network, depending on the target task's similarity to the original task. This allows the model to refine its knowledge and better adapt to the specifics of the new task (François Chollet, 2021).

According to Wan *et al.* (2021), the benefits of transfer learning in the analysis of EEG signals include:

1. Accounting for individual variability: In EEG signal processing, there is a significant difference between training and testing data such as subjects, sampling times, and task objectives, making analysis more challenging. Transfer learning enables the model to adapt to different individuals and tasks through adjustments.

2. Minimizing data requirements: Data scarcity and insufficient labeling in EEG signal analysis impede the learning of the target task. Transfer learning leverages prior knowledge from a related domain using a small amount of target domain data to adjust the classifier, thereby reducing the data needed.

Many transfer learning techniques have been applied within the MI paradigm: in a similar case to our study Dose *et al.* (2018) developed a DL approach for an EEG-Based BCI to improve current stroke rehabilitation techniques. Contrastly, they applied TL only to a MI dataset to adapt the global classifier to single individuals, achieving improved accuracy. In a recent study, Zheng and Lin (2024) performed a MI-BCI study where they applied simultaneously two domain adaptation modules to a deep transfer neural network to effectively use labeled EEG data from previous times, hence achieving good recognition performance for a small number of labeled EEG signals at the current time. The study showed a higher classification accuracy to the compared models. Another study, Zhang *et al.* (2021) aimed to apply adaptative TL for EEG-MI classificacion with deep CNNs by training with pre-existing data from other subjects and evaluating the model on new target subjects. Other studies can also be found where they combine ME with MI in order to decode the MI, as it can be seen D. Y. Lee *et al.* (2020), where they collected EEG data regarding arm reaching within 3 directions. They developed a transfer learning method based on a Relation Network architecture to decode MI while using ME as a supporting tool.

Regarding the application of transfer learning from ME to MI, there are scarce articles

available, with multiple avenues still to explore, such as the use of CNNs, optimizing cross-task transfer learning frameworks, and improving the interpretability and efficiency of these models in practical BCI applications. This method aims to take advantage of the information acquired during the initiation phases of the movement in order to facilitate the adaptation process for the new task.

In a study by D.-Y. Lee *et al.* (2022) focused on decoding various forearm movements from EEG signals, ten healthy participants performed 4 ME and MI tasks of the right upper limb (forearm extension, hand grasp, wrist supination, and rest). The research introduced a convolutional neural network using a channel-wise variational autoencoder (CVNet) based on inter-task transfer learning. The proposed CVNet was validated on the subjects achieving a cross task classification accuracy of 0.83, and on the public BNCI Horizon 2020 dataset applying a combination of ME and MI for training, achieving a MI classification accuracy of 0.69. In another example, Miao *et al.* (2023) introduce a novel explainable cross-task adaptive TL method for MI EEG decoding. The method begins with similarity analysis and data alignment for EEG data from ME and MI tasks. Subsequently, a MI EEG decoding model is developed through pre-training with extensive ME EEG data and fine-tuning with a subset of MI EEG data. Finally, an expected gradient-based post-hoc explainability analysis is performed to visualize significant temporal-spatial features. The method achieved the highest average classification accuracy of 80.00% for OpenBMI dataset and 72.73% for GIST dataset. Additionally, the explainability analysis results further confirmed the correlation between ME and MI EEG data and the effectiveness of ME/MI cross-task adaptation.

In chapter 2, we explored the foundational aspects of MI-based BCIs, focusing on the underlying paradigms, signal processing techniques, and the use of neural networks for decoding EEG signals. We also examined the distinctions between MI and ME, as well as the application of transfer learning to improve BCI performance. In the next chapter, chapter 3, we will delve into the specifics of the dataset and signals used in this study. This will include a detailed overview of the EEG Motor Movement/Imagery dataset, the experimental protocol, and the EEG montage, setting the stage for the subsequent analysis and methodology.

3. DATASET & SIGNALS

In order to perform our study, we searched for available public databases regarding healthy subjects performing ME and MI. We performed our search in diverse repositories before making a selection for PhysioNet. We selected the public database available *EEG Motor Movement/Imagery Dataset v1.0.0* because of the large number of subjects available (109) in comparison with others and due to being one of the scarce databases with EEG recording of the same subject during both ME and MI (Gerwin Schalk *et al.*, 2022; G. Schalk *et al.*, 2004; Goldberger *et al.*, 2000).

PhysioNet is a comprehensive repository of physiological and clinical data, managed by the MIT Laboratory for Computational Physiology, and aimed at supporting the research and development of computational medicine (Goldberger *et al.*, 2000). It was established in 1999 as part of the National Institutes of Health's (NIH) Research Resource for Complex Physiologic Signals, providing free access to large collections of recorded physiologic signals and related open-source software. This platform serves as a critical resource for the scientific community, fostering the development of innovative algorithms and tools for analyzing complex physiological signals.

3.1. EEG Motor Movement/Imagery Dataset Physionet BCI2000

This dataset (Gerwin Schalk *et al.*, 2022) was created by Gerwin Schalk and his colleagues at the BCI R&D Program, Wadsworth Center, New York State Department of Health, Albany, NY. It was created by the developers of BCI2000. This platform presents a versatile and open-source software option designed for BCI research and development. It provides a comprehensive framework for data acquisition, signal processing, and stimuli and feedback presentation in real-time. It's a valuable resource that supports a variety of data acquisition systems and can be used for diverse BCI applications, including neurorehabilitation, communication, and control tasks (G. Schalk *et al.*, 2004).

Particularly to our dataset, it was published in 2009 with a set of 64-channel EEGs from subjects who performed a series of motor/imagery tasks for the research in brain computer interfaces. It contains over 1500 1 or 2 minute EEG recordings from 109 subjects. Sociodemographic data of the individuals that were subject to the recordings is not available.

3.1.1. Acquisition software

The EEG recordings to be analyzed were recorded using the BCI2000 system (G. Schalk *et al.*, 2004). The system is based on a model consisting of four modules:

- Source Module: Digitizes and stores brain signals, passing them to the signal processing module without preprocessing. Consists of data acquisition and data storage components, with a documented file format including an ASCII header followed by binary signal samples and event marker values.

- Signal Processing Module: Converts brain signals into control signals through two stages: feature extraction (e.g., extracting firing rates, amplitudes) and feature translation (e.g., translating features into control signals). Uses independent signal operators that can be combined or interchanged flexibly.

- User Application Module: Uses control signals to drive applications, which can be visual (e.g., selection of targets, letters) or other feedback forms (e.g., auditory, haptic). Supports various applications including BCI control of neuroprostheses or orthoses.

- Operator Module: Defines system parameters and controls the onset and offset of operations. Displays real-time information from any module, enabling experiment control and real-time monitoring.

Furthermore, these modules communicate via a documented network-capable protocol based on TCP/IP, allowing flexibility in programming languages and networked machine usage. Components are designed to be interchangeable and independent, allowing different implementations to be used without affecting the system. Each module performs specific functions and communicates standardized information, reducing dependencies. Moreover, BCI2000 is designed to minimize the effects of operating system and hardware interruptions on response time, ensuring timely processing and response. Finally, BCI2000 incorporates three types of system variables:

- Parameters: Constant throughout a data file, such as signal sampling rates.

- Event Markers: Record events during operation, enabling full session reconstruction and comprehensive data analysis. Modules can modify or monitor event markers.

- Signals: Functions of brain signals modified by modules. Modules can request the creation of various system parameters and event markers.

3.1.2. Experimental Protocol

As previously stated, in this experimental protocol subjects performed various motor and motor imagery tasks while their brain activity was recorded using a 64-channel EEG system and the BCI2000 software. The experiment included baseline measurements and four distinct tasks, focusing on both actual and imagined movements (Gerwin Schalk *et al.*, 2022).

The baseline runs consist of two separate runs, one with eyes closed and one with eyes opened, of 1 minute each. As for the four tasks:

1. **Motor Execution of one hand:** A visual target appeared on either the left or right side of a screen. Subjects opened and closed the corresponding fist until the target disappeared, then relaxed.

2. **Motor Imagery of one hand:** A visual target appeared on either the left or right side of a screen. Subjects imagined opening and closing the corresponding fist until the target disappeared, then relaxed.

3. **Motor Execution of both fists or feet**: A visual target appeared on either the top or bottom of a screen. Subjects opened and closed both fists if the target was on top, or both feet if the target was on the bottom, until the target disappeared, then relaxed.

4. **Motor Imagery of both fists or feet**: A visual target appeared on either the top or bottom of a screen. Subjects imagined opening and closing both fists if the target was on top, or both feet if the target was on the bottom, until the target disappeared, then relaxed.

Regrading the tasks runs order, three repetitions were performed of each Task 1, Task 2, Task 3, and Task 4, in the following sequence:

1. Baseline, eyes open

2. Baseline, eyes closed

After these recordings, three repetitions were performed of each Task 1, Task 2, Task 3, and Task 4, in the following sequence:

3. Task 1 (left or right fist movement)

4. Task 2 (left or right fist imagined movement)

5. Task 3 (both fists or both feet movement)

6. Task 4 (both fists or both feet imagined movement)

After the task sequence is repeated three times, we obtain a total of 14 experimental runs. The resulting data obtained is provided in EDF+ format. Each annotation file of each recordings contains the following event markers:

- T0: Indicates rest periods.
- T1: Indicates the onset of motion (real or imagined) of:
  - The left fist in specific runs (3, 4, 7, 8, 11, and 12)
  - Both fists in other specific runs (5, 6, 9, 10, 13, and 14)
- T2: Indicates the onset of motion (real or imagined) of:

– The right fist in specific runs (3, 4, 7, 8, 11, and 12)

– Both feet in other specific runs (5, 6, 9, 10, 13, and 14)

It is worth noting that inside each task, the most common number of onsets for both ME and MI paradigms that the subjects received were 30 (including T0, T1 and T2). However, some particular subjects performing ME may have received 24, 38 or 44 repetitions, while subjects performing MI may have received 24, 26 or 38 repetitions.

3.1.3. EEG Montage

The EEGs were recorded using the BCI2000 system previously explained with 64 electrodes following the international 10-10 system, from which the following electrodes were excluded Nz, F9, F10, FT9, FT10, A1, A2, TP9, TP10, P9, and P10. The resulting recordings contain 64 EEG signals, sampled at 160 or 128Hz. The montage can be seen on Figure 3.1.

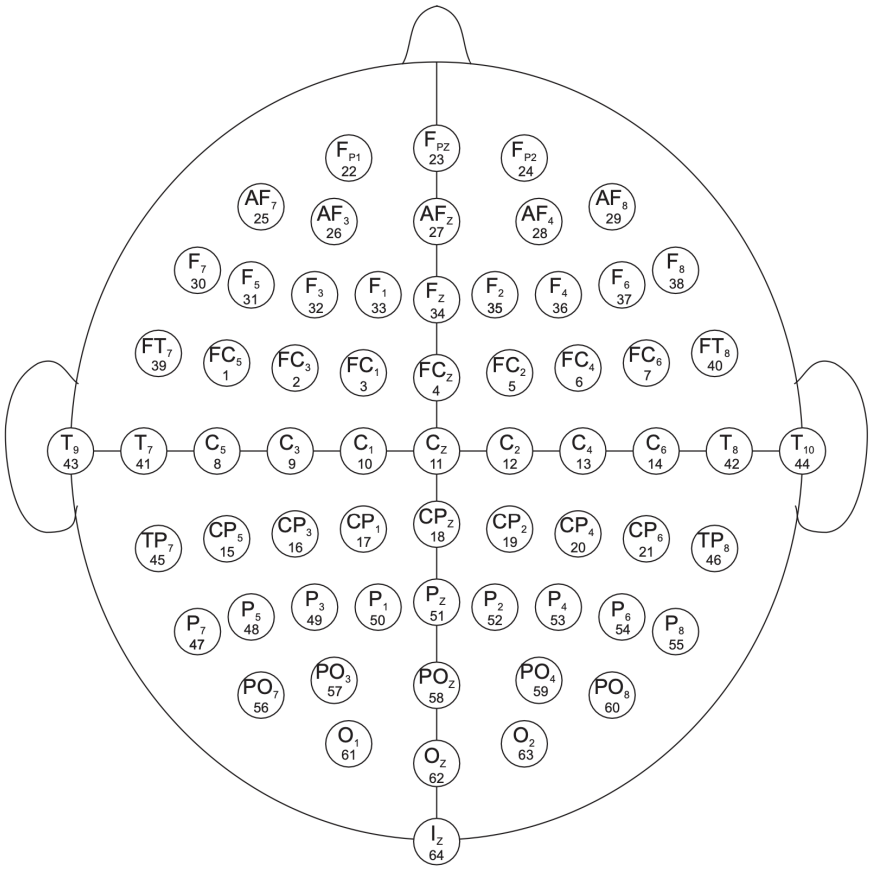


Fig. 3.1. EEG montage. The number under the electrode's name corresponds to the order in which they appear in the recordings (Gerwin Schalk *et al.*, 2022).

4. METHODS

As our platform of choice, we have made use of MEDUSA©. We will be using the kernel available within MEDUSA© in order to perform both the characterization as well as the classification and transfer learning methods.

4.1. MEDUSA

MEDUSA© is a new versatile platform for contemporary non-invasive BCIs developed at the Biomedical Engineering Group of Universidad de Valladolid (Santamaría-Vázquez *et al.*, 2023). Key features of this open-source solution include: (1) broad compatibility with various signal acquisition systems, supported by advanced functionalities built on the lab-streaming layer (LSL) protocol, allowing for simultaneous recording of multiple signals; (2) a comprehensive suite of BCI paradigms and cognitive neuroscience experiments, such as ERP spellers based on c-VEP and P300, MI, neurofeedback, and neuropsychological tasks; (3) cutting-edge signal processing methods and models for both offline and online analysis, including deep neural networks and connectivity analysis; (4) developer tools to facilitate the creation of custom open-loop and closed-loop experiments; and (5) specific features to enable the sharing of experiments and promote open science, reproducibility, and community collaboration, including an app market and multiple discussion forums on our website.

This platform contains two different and independent services:

- MEDUSA© Kernel is a Python package that provides ready-to-use methods for analyzing brain signals. It includes advanced techniques in signal processing, machine learning, deep learning, and other high-level analyses. Additionally, it offers functions and classes for managing various biosignals, including EEG and MEG, as well as for saving experimental data or creating standalone processing pipelines.
- MEDUSA© Platform, a Python-based desktop application, delivers high-level functionalities for experimental purposes. It features a modern graphical user interface (GUI) with advanced signal acquisition capabilities and real-time charts. A key aspect of this platform is its ability to install and create apps that implement neuroscience and BCI experiments or paradigms. All these functionalities are built on MEDUSA© Kernel for the required real-time signal processing.

The kernel provides several modules for different functionalities, such as artifact removal, temporal and spatial filters, or local activation metrics. Even though we made use of most of the modules available, of particular interest for our project is the Motor Imagery modules, providing comprehensive classification pipelines that can be utilized in both offline and online modes; MI analysis charts; and specialized data structures designed for MI decoding;

For this project, we required the use of two similar, one for the characterization and one for the neural network model, but different datasets with slightly different pre-processing.

4.2. Characterization

In order to better understand the difference and similarities between ME and MI, we created specific datasets for the characterization experiments.

4.2.1. Dataset

As previously explained on subsection 2.1.5 for the characterization to be solid and reliable, the raw signals provided have to be pre-processed. The following steps were applied iteratively to all runs of each task of each subject.

4.2.1.1 Convert to MEDUSA Recording

Given that the recordings from the dataset are provided in EDF+ format, they require to be converted to a Medusa Biosignal Data Recording class in order to be compatible with the rest of the functionalities withing the Medusa Kernel package.

As upper limb movements seem to be the most clearly distinct (Batula *et al.*, 2017), we will only make use of tasks 1 and 2. Additionally, these tasks are also the most relevant for stroke rehabilitation, regarding the execution or imagination of the left or right fist. The pipeline for the creation of the dataset is the following:

1. Download ZIP file with all available recordings, containing 109 folders, with 28 files each. These files correspond to the 14 runs performed by each subject, which information available in .edf and .edf.event format.

2. Once downloaded, and imported into python using the MNE python package, we transform the data of a RawArray data structure from mne to a MedusaData data structure from MEDUSA. While doing so, we transfer all relevant information of the recording, including sampling frequency, onset times, labels of the paradigm, montage used during the acquisition, subject or task and run executed during the recording.

   The ME and MI recordings are imported separately, with a generalized python function we created that also allows you to import tasks 3 and 4 if wanted by the user. Additionally, as we will be primarely working with the left and right tasks, the label annotation within the recordings is changed for better and more effective use in future sections. In the provided dataset T0 corresponds to the rest time period, T1 to the "left" time period and T2 to the "right" time period. In this new configuration, label 0 corresponds to the "left" time period, label 1 to the "right" time period and label 2 to the "rest" time period.

3. The ME and MI MIData files of each run and subject are exported separately into the corresponding "PhysionetMI" or "PhysionetME" folders in BSON format in order to reduce the data storage size as well as optimize as speed the creation different processed datasets.

4.2.1.2 Preprocessing and artifact removal

We applied a Infinte-Impulse Response Butterworth bandpass filter of 4th order in the frequencies 1Hz-58Hz in order to remove the baseline drift of the recordings, avoid aliasing and remove power-line interference. This bandpass filter also allows us to focus on the main frequencies of interest within the EEG without removing relevant data. Moreover, a CAR spatial filter was applied. This filter is frequently applied to EEG recordings to help detect small signal sources in otherwise very noisy recordings, improving the Signal to Noise Ratio (Binnie *et al.*, 2003).

For this dataset, we selected all the available labels: left, right and rest. Even though the time interval between onsets in the recordings was of 4 seconds, in the following epoch extraction section, it was observed how some subjects didn't have enough samples on the last onset in order to extract the full interval. For this reason, the maximum duration available for extraction was 3400ms. The 3400 ms window was the maximum extractable duration that neither required discarding trials due to insufficient samples nor necessitated artificial signal padding. Additionally, as some recordings had a sampling frequency (*fs*) of 128Hz and others of 160Hz, all recordings were subsampled to the lower fs to homogenize the dataset, while reducing the computational cost of the neural network..

After being subsampled, all filtered and epochs recordings were concatenated and exported as an .h5 file. Along with the EEG recordings, the stored data with the corresponding paradigm labels (0, 1, 2) , run indexes (corresponding to ME or MI), subjects (from 1 to 109) and channels selected was also exported as it will be needed for the posterior classification and characterization of the signals.

The final technique applied in order to remove any additional artifacts present in the dataset was an epoch rejection function. This technique was applied per subject after applying first z-score normalization to each subjects epoch.

The epoch rejection technique applied is a simple thresholding method to reject noisy epochs. It discards epochs with n samples greater than k*std in ncha channels. In our early tests, we tried applying a k value of 4 in within 2 samples of 2 channales, but it was too strict, forcing us to discard more than 7 subjects. Finally, we found satisfactory results applying a k value of 5 within 3 samples and 5 channels, allowing us to keep all available subjects. This technique resulted in the rejection of 5.15 % of epochs available within the MI paradigm, and the rejection of 5.96 of % epochs available in ME.

Additionally to the explained pre-processing for this dataset, after performing some tests on the datasets, it was observed that the periferial electrodes of the scalp, specially the frontal electrodes, were interfering signifcantly with the results of the characterization. For this reason,

we decided to remove the periferial electrodes since, as stated in subsection 2.1.3, our main region of interest is located over the sensorimotor cortical areas.

- Removed Electrodes (23): ['FPZ', 'FP2', 'FP1', 'AF7', 'F7', 'FT7', 'T7', 'T9', 'TP7', 'P7', 'PO7', 'O1', 'OZ', 'O2', 'PO8', 'P8', 'TP8', 'T8', 'T10', 'FT8', 'F8', 'AF8', 'IZ']
- Remaining Electrodes (41): ['FC5' 'FC3' 'FC1' 'FCZ' 'FC2' 'FC4' 'FC6' 'C5' 'C3' 'C1' 'CZ' 'C2' 'C4', 'C6' 'CP5' 'CP3' 'CP1' 'CPZ' 'CP2' 'CP4' 'CP6' 'AF3' 'AFZ' 'AF4' 'F5', 'F3' 'F1' 'FZ' 'F2' 'F4' 'F6' 'P5' 'P3' 'P1' 'PZ' 'P2' 'P4' 'P6' 'PO3', 'POZ' 'PO4']

It is worth noting that even through this pre-processing method, while manually observing the recordings of certain subjects, some artifacts such as ECG may have persisted.

The final output of the exported dataset can be seen on Table 4.1

	Features	**Labels**	**Run Index**	**Subjects**	**Channels**
Motor imagery	(9836, 435, 41)	(9836)	(9836)	(9836)	(41)
Motor Execution	(9854, 435, 41)	(9854)	(9854)	(9854)	(41)

Table 4.1. Size of the exported data on the created dataset. The first dimension of the features indicates the number of epochs available for each paradigm, the second dimension shows the number of samples per epoch (corresponding to 3400ms at a sampling frequency of 128Hz), and the last dimension contains all the channels available. Features contains the values of the EEG signal; Labels contains values 0,1 and 2 representing each task; Subjects contains values from 1 to 109 corresponding to each subject's epoch; Channels contains the name of all the electrodes selected

4.2.2. Relative Bandpower

The relative power (RP) is a linear measurement that provides information on the normalized weight of each spectral band in the EEG signal distribution. The selected spectral bands for assessing induced EEG changes were delta (1–4 Hz), theta (4–8 Hz), alpha (8–12 Hz), mu (12-15 Hz), beta (15–30 Hz) and gamma (30-50Hz).

In order to compute the RP from the extracted dataset, we split the dataset according to the different tasks. To each epoch of each task we computed the power spectral density (PSD) of the signal by means of the Welch method, applying a Dirichlet window, default option on Medusa Kernel, with 80% of the signal (348 samples) and 50% overlap. The RP for each frequency band is estimated (Marcos-Martínez *et al.*, 2021):

$$\mathrm{RP}_i = \frac{\sum_{f=l_f^i}^{u_f^i} \mathrm{PSD}(f)}{\sum_{f=0.1\mathrm{Hz}}^{50\mathrm{Hz}} \mathrm{PSD}(f)}, \quad i = \{\mathrm{Delta}, \mathrm{Theta}, \mathrm{Alpha}, \mathrm{Mu}, \mathrm{Beta}, \mathrm{Gamma}\} \tag{4.1}$$

where f is the frequency, l is the lower frequency of each band, u is the upper frequency of each band and i is the band of interest.

4.2.3. ERD/ERS

The classical method to compute the time course of ERD includes the following steps (Pfurtscheller, 2001):

1. Bandpass filtering of all event-related trials;

2. Squaring of the amplitude samples in each trial to obtain power samples;

3. Averaging of power samples across all epochs;

4. Averaging over time samples to smooth the data and reduce the variability.

To calculate percentage values for ERD/ERS, let A represent the power within the frequency band of interest during the activity period, and R represent the power during the preceding baseline or reference period. ERD or ERS is defined as the percentage decrease or increase in power, respectively, and is calculated using the formula:

$$\text{ERD}\% = \frac{(A - R)}{R} \times 100 \tag{4.2}$$

In our study, a moving average filter of 128 samples with a whole-sample symmetric setting was applied to the ERD/ERS of each channel to smooth the data. Averages were computed across trials per individual for each task and each frequency band. Conditions were compared for 200 ms time windows from 0 to 3375 ms. The average power in the time interval from - 300 to 0 ms from each onset was taken as baseline.

For this computation, the same pre-processing pipeline was followed, but with the bandpass filter being applied within the corresponding frequencies of interest (i.e theta, alpha, beta, etc.) and an additional 300 ms of signal recording extracted before the onset. Hence, we obtained one dataset for each of the bands of interest. However, since during epoch extraction the first epoch to be extracted did not contain 300 ms prior to the onset, we had to remove the first available onset for each subject. This did not affect the epoch rejection algorithm since we selected the same epochs per subject as the RP dataset in order to continue analysing the dataset with consistency.

4.3. Classification and transfer learning

As seen on subsection 2.2.5, our neural networks of choice for testing are EEG-Inception and EEGSym. These models have been previously tested with MI data, however, they have not been validated in cross-task transfer learning. In this study we will perform a preliminary test of both models and choose one to assess the performance of our hypothesis.

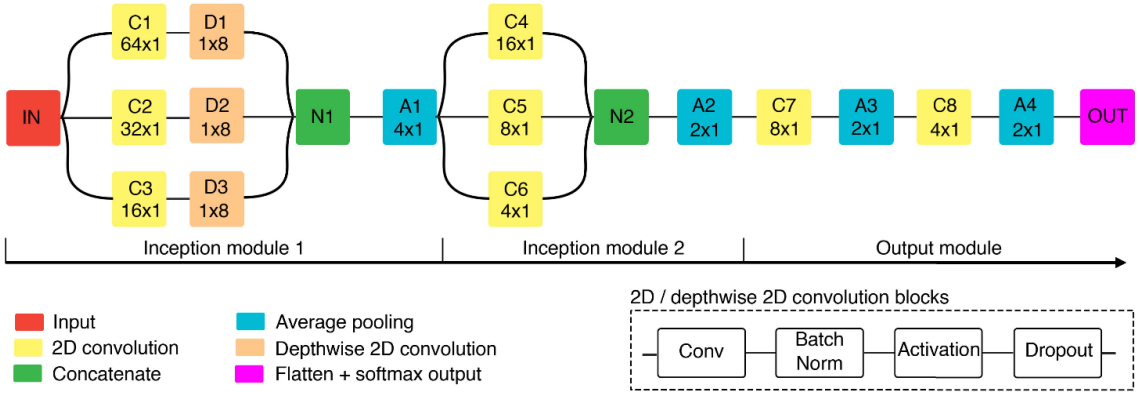


Fig. 4.1. Overview of EEG-Inception architecture: The architecture includes both 2D convolution blocks and depthwise 2D convolution blocks, each incorporating batch normalization, activation functions, and dropout regularization. The kernel sizes for convolutional and average pooling layers are specified (Santamaria-Vazquez *et al.*, 2020).

4.3.1. EEG-Inception

EEG-Inception is a novel CNN created by Santamaria-Vazquez *et al.*, 2020, originally based on a NN for image classification, and particularly adapted for EEG processing and ERP detection. The architecture developed incorporates image classification concepts, such as Inception modules to capture relationships between features at different scales, and depthwise convolutions.

The EEG-Inception architecture was specifically crafted to prevent overfitting, featuring an output block that consolidates the data extracted by the Inception modules into a small number of high-level features. Specifically, only 24 features are inputted into the final classification layer. This strategy optimized the benefits of the fine-tuning process, which required only a minimal number of calibration trials to adapt the model to new subjects.

The full architecture includes three main blocks, which can be seen on Figure 4.1:

1. Inception Module 1: This module processes the signal at three different temporal scales for each EEG channel, using convolutional blocks C1, C2, and C3 with kernel sizes of 64 $\times$ 1, 32 $\times$ 1, and 16 $\times$ 1, respectively. Given the input sampling rate of 128 Hz, these sizes correspond to temporal windows of 500 ms, 250 ms, and 125 ms. Following these layers, the blocks D1, D2, and D3 process the signal in the spatial domain using depthwise convolutions. Depthwise convolutions, originally used in image classification to reduce the number of parameters by factorizing a convolution kernel into smaller kernels, are applied to each input channel separately. In EEG processing, they help learn optimal spatial filters for each temporal pattern extracted by the previous layer. The output features from D1, D2, and D3 are then merged by the concatenation layer N1, followed by average pooling for dimensionality reduction.

2. Inception Module 2: This module is structured similarly to the first one, with three branches processing the EEG signal at temporal scales of 500 ms, 250 ms, and 125 ms.

After the average pooling layer of the first block, these scales correspond to kernel sizes of 16×1, 8×1, and 4×1. This module extracts additional temporal features at a higher level of abstraction across all EEG channels. The outputs of convolutional blocks C4, C5, and C6 are concatenated, followed by average pooling for dimensionality reduction.

3. Output Module: The final two convolutional layers are designed to extract the most significant patterns for classification, compressing the information into a few features. The number of filters is progressively decreased to reduce dimensionality and avoid overfitting, resulting in only 24 features being fed into the final classification layer. The softmax output layer then estimates the probability for each class (target and non-target).

In Santamaria-Vazquez *et al.* (2020), the model was trained with the following configuration: Adam optimizer with default hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, categorical cross-entropy loss function, mini-batch size of 1024, and 500 epochs. To speed up training and prevent overfitting, early stopping was applied when the validation loss did not improve for 10 consecutive epochs, restoring the weights that minimized this metric. The learning rate (lr), activation function (f_{act}), and dropout rate (dr) were automatically optimized using grid search on the validation set. Specifically, the optimal set was $lr = 0.001$; $f_{act} = \text{ELU}$; $dr = 0.25$. Remaining hyperparameters, including the number of layers, number of branches in Inception modules, number of filters, kernel sizes, and pooling sizes, were chosen heuristically.

For our study, we will keep the same configuration, except due to our reduced number of available data, we reduced the mini-batch size to 32 epochs.

4.3.2. EEGSym

Akin EEGInception, EEGSym developed, by Perez-Velasco *et al.* (2022), incorporates effective techniques previously validated for EEG decoding. These include the use of inception modules early in the architecture and the adoption of grouped convolutions to emulate the success of EEGNet and EEG-Inception with depthwise convolutions. Each convolution operation is followed by batch normalization, 'elu' activation, and dropout regularization. As before, hyperparameters such as dr (0.4), number of filters in inception modules (24), and lr (0.001) were determined via grid search on the validation set.

EEGSym's architecture, illustrated in Figure 4.2 unfolds across 5 stages:

1. Symmetric Division: Creates virtual divisions within the model, optimizing spatial filter parameters for subsequent tempospatial analysis and reducing parameter redundancy.

2. Tempospatial Analysis: Captures detailed temporal relationships with two inception blocks and three residual blocks. Inception module sizes (64, 32, and 16) align with that of EEG-Inception, spanning temporal windows of 500 ms, 250 ms, and 125 ms. Results from each convolution are concatenated and added via residual connections, followed by temporal

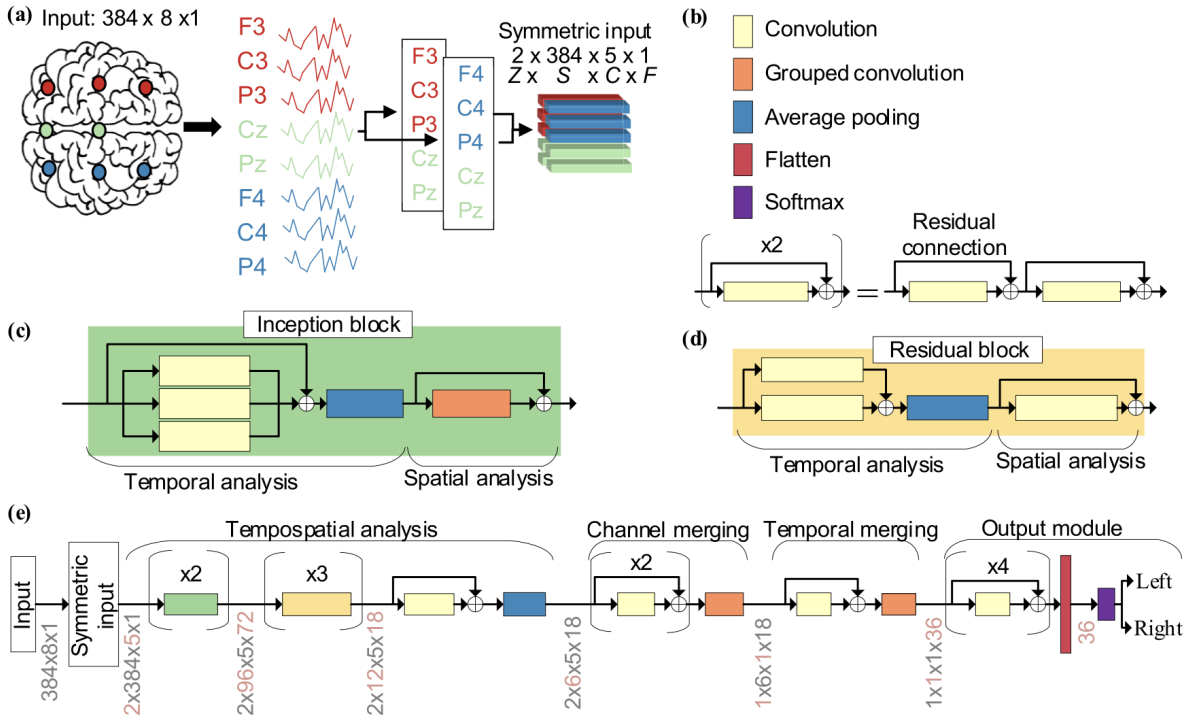


Fig. 4.2. Figure overview for EEGSym architecture. (a) Illustration showing the distribution of input electrodes in an 8-electrode configuration: Z denotes hemispheres (2), S denotes samples (384), C denotes electrodes per hemisphere (5), and F denotes the number of filters. (b) Key explaining the architecture overview. (c) Inception block. (d) Residual block. (e) Detailed representation of the EEGSym architecture. All convolution and grouped convolution operations are followed sequentially by batch normalization, 'elu' activation, and dropout regularization. Output sizes for each operation are displayed in gray, while affected dimensions after each stage are highlighted in red. (Perez-Velasco *et al.*, 2022)

dimension reduction with average pooling. Spatial extraction utilizes grouped convolution across hemisphere channels, reducing channels to 1, preserving residual connections. The residual block includes temporal and spatial analyses with convolution, maintaining residual connections for temporal relations post-spatial operation and average pooling.

3. Channel Merging: Reduces spatial dimensionality to 1 (Z and C) through two convolution layers with residual connections, merged via grouped convolution across hemispheres and channels (kernel size $2 \times 1 \times 5$).

4. Temporal Merging: Further reduces temporal dimensionality to 1 (S) with convolution and grouped convolution operations matching the temporal dimension.

5. Output Module: Final stage with features dependent on inception module filters (e.g., 24 filters per branch yield 36 features), includes four convolution layers with residual connections, flattening features for softmax classification across MI classes.

For our study, we will keep the same hyperparameter configuration, but contrary to the preprocessing applied to the data on Perez-Velasco *et al.* (2022), we will not be applying any data augmentation techniques.

4.3.3. Adaptation to Torch framework

The EEGInception and EEGSym models are currently available as open source on MEDUSA kernel in python using the Keras framework with Tensorflow backend, an open library package for the creation of machine learning models.

On the other hand, PyTorch which is an optimized tensor library designed for deep learning applications on both GPUs and CPUs, is another open library package for the creation of machine learning models (Paszke *et al.*, 2019). Given the advantages for of each framework that can be seen on Table 4.2, the advantages Torch presents for research, and the fact that Tenforflow is not going to continue providing updates for Windows nor support new python updates, we converted the models from Tensorflow to Pytorch. However, it is worth noting that Novac *et al.* (2022) observed that Pytorch framework falls slightly short of TensorFlow's training accuracy, showing a 1.16% disparity between the two frameworks.

In order to create a Neural Network model in Torch, it is required to create the model within a nn.Module class. The `__init__()` method is used to define the layers and other components of a model, while the `forward()` method is where the actual computation takes place. Additionally, you can print the model or any of its submodules to inspect their structure.

For example, in the `__init__()`) you would declare and establish all layers, including convolutional, fully connected, batch normalization, depthwise convolutional, pooling or softmax layers. Weight initialization is also established within this module.

'torch.nn.Module' is the foundational class in PyTorch designed to encapsulate the behaviors and functionalities specific to PyTorch models and their components. Alternatively to Keras, a 2D convolution within the torch module always receives and input of (N, C_{in}, H, W), being N the batch size, C_{in} the number of channels of the input, H the temporal axis (i.e. samples) and W the spatial axis (i.e. channels of EEG). Considering a 3 second input to our network, the input would be of size (N, 1, 384, 64).

In order to compute a 2D convolution required for EEGInception, we require the use of the function nn.Conv2d, which applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) * \text{input}(N_i, k)$$

In order to perform a specific 2D convolution in Keras with arguments:

```
unit = Conv2D(filters=self.filters_per_branch,
              kernel_size=(self.scales_samples[i], 1),
              kernel_initializer='he_normal',
              padding='same')(input_layer)
```

However, to perform a 2D convolution in PyTorch the input arguments are:

```
conv_layer = nn.Conv2d(in_channels=input_channels,
                       out_channels=self.filters_per_branch,
                       kernel_size=(self.scales_samples[i], 1),
                       stride=1,
                       padding='same')
nn.init.kaiming_normal_(conv_layer.weight)
```

As the torch equivalent can't receive a kernel initializer argument, we are required to separately initizialize the weights. Furthermore, padding='valid' is the same as no padding and padding='same' pads the input so the output has the same shape as the input.

Another consideration for the convolutional layers comes with the implementation of the Depthwise Convolutional layer. Contrary to Keras, which has a DepthwiseConv2D function available, a torch depthiwse convolution requires the use of the Conv2d function but with when the arguments `groups` == `in_channels` and `out_channels` == $K \times$ `in_channels`, where K is a positive integer known as depth multiplier.

In other words, for an input of size $(N, C_{\text{in}}, L_{\text{in}})$, a depthwise convolution with a depthwise multiplier K can be performed with the arguments ($C_{\text{in}} = C_{\text{in}}, C_{\text{out}} = C_{\text{in}} \times K, \ldots,$ `groups` $= C_{\text{in}}$). Furthermore, in order to be able to apply a depthwise constraint to our convolution, as it isn't established as an argument, we must manually 'clamp' the parameters. The PyTorch code for clamping the weights of the depthwise convolution layer is:

```
self.Depth.weight = nn.Parameter(torch.clamp(self.Depth.weight, min=-1., max=1.))
```

Regarding the training of the model, Keras has a built-in 'fit' method within the model. However, for torch you must manually hardcode the iteration training loop within a function. A training loop that performs one epoch performs the following actions:

1. Retrieves a batch of training data from the input.
2. Resets the optimizer's gradients.
3. Conducts an inference to obtain predictions from the model for the input batch.
4. Computes the loss by comparing the predictions to the actual labels in the dataset.
5. Computes the backward gradients for the learning weights.
6. Instructs the optimizer to execute one learning step, adjusting the model's weights based on the calculated gradients for this batch, according to the chosen optimization algorithm.
7. Reports the loss every 1000 batches.
8. Reports the average loss per batch for the last 1000 batches, for comparison with the validation run.

In per-epoch activity, while training we must make sure gradient tracking is on, to then set the model to evaluation mode, disabling dropout and using population statistics for batch normalization. Gradient tracking must also be disabled while testing the model on your target dataset, as we have found that it will cause memory leaks that will exacerbate runtime, even capable of causing your model to crash. Additionally, EarlyStopping handlers can be found in torch within complementary Pytorch libraries such as Pytorch-Ignite or Pytorch Lightning. Nonetheless, we created our own Earlystopping class to keep our model only torch based. The training loop and the Earlystopping class can also be found in chapter 7.3.

Finally, to work within the Pytorch framework, all arrays that interact with the framework must be converted to torch.tensor, a multi-dimensional matrix containing elements of a uniform data type. Most importantly if any GPUs are available, all tensors and models must be transferred from CPU to CUDA tensor in order to work within the GPU. CUDA (Compute Unified Device Architecture), developed by NVIDIA, is a programming model and computing toolkit that accelerates compute-intensive tasks by distributing them across GPUs in parallel. It stands as one of the predominant API leveraged in deep learning for increased performance.

Criteria	**PyTorch**	**Keras**
Key Differences	Deep integration with Python, preferred for research.	High-level API, user-friendly, and ideal for rapid prototyping.
Architecture	Dynamic computation graph enabling real-time graph construction, suitable for complex models.	High-level API that operates on top of TensorFlow, Theano, or CNTK, abstracts complex operations.
Ease of Use	Python based UI and intuitive, though it requires more code for defining models. Offers more control and flexibility, great for custom models and research	Simple and concise syntax, minimal code required for model definition. ideal for beginners and rapid development.
Impact on Practical Application	Facilitates quick iterations and detailed debugging, with interactive execution.	Allows for rapid prototyping and experimentation, with less control over low-level operations.
Speed and Efficiency	Efficient for small to medium-scale models, offering more control over optimization.	Performance is dependent on the backend (TensorFlow, Theano), optimized for ease of use.
Scalability	Suitable for experimental and research projects, effective for custom implementations.	Scales well for production through the TensorFlow backend, designed for high-level applications

Table 4.2. Comparison between PyTorch and TensorFlow

4.3.4. Classification Dataset

The Dataset used to train the NN, follows a similar pipeline as subsection 4.2.1. Alternatively, due to the nature of Neural Network models and their capability for feature learning, dimensionality reduction and noise handling, we made use of all 64 available channels with a lower bandpass filter frequency being situated at 0.05 Hz instead of 1Hz. The epoch rejection algorithm was not applied. Additionally, even though we also tested a 3 class classification network, we mainly focused on the distinction between 'left' and 'right', therefore we removed all the onsets corresponding to 'rest' periods, keeping only labels 0 and 1.

4.3.5. Model selection

In order to make a final selection between our two networks, we performed a test run while training the model with only MI the dataset. For computational purposes, as part of the test run, we only took 8 of the 64 channels available within the dataset. For this selection we made use of the optimal channels selected through eXplainable Artificial Inteligence (XAI) in Pérez-Velasco *et al.* (2024). After running a 5 fold, similar accuracy results were obtained from both models, 81% for EEGInception and 81.1% for EEGSym. Since both models provided similar accuracy, we made the final decision to select EEGInception, as it can be seen the summary of their arquitecture on Table 4.3 and Table 4.4, EEGSym presents a notable increase complexity and encompasses over 9 times the number of trainable parameters compared to EEG-Inception, thereby imposing a substantial burden on computational efficiency and runtime. Amongst all the code used for the development of this Final Degree Project, only the adapted EEGSym (hardcoded for 8 channels) and EEGInception torch models are provided in the Annex chapter.

4.3.6. Pre-trained model with transfer learning

As previously stated, the goal of the project is to train the neural network model with the dataset corresponding to the execution of movements from the subjects, to then transfered the acquired knowledge to predict the imagination of movement. A model-based approach was selected as our transfer learning technique of choice.

After importing the h5 files with the pre-processed MI and ME dataset, we performed a Leave-One-Subject-Out (LOSO) validation training strategy. Since the data imported from the file presents all the EEG information concatenated, we must first split the features and labels between the 109 subjects. The LOSO method implies that we pre-trained one model for each of the available subjects. In this model, we will use all the available ME data except for the ME data corresponding to said subject to train the model and validate the model. Once the model has been trained, we test the model with the MI data from the corresponding subject. This technique allows us to build a strong cross-subject model. We will further explain the pipeline for the prediction of movement for subject 0, this pipeline is iterated on a loop for each of the 109 subjects.

1. Split available data in train/development/test. Remove ME data corresponding to subject 0 from ME input data, including both features and labels. Once it has been removed, we perform a 90% train / validation split amongst the ME data, meaning ME data from 97 subjects is used for training of the model, and data pertaining to the remaining 11 subjects is used for the validation of the model. The subjects used for training is randomized and different for each iteration to avoid bias. Finally, all data must be converted to torch.tensor and transfered to CUDA, while also converting all the corresponding labels to one-hot encoding.

2. Training of the model while utilizing the Adam optimizer, configured with the default hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The categorical cross entropy loss function was employed, with a mini-batch size set at 32 and the model trained over 600 epochs. To expedite the training process and mitigate overfitting, the programmed early stopping was implemented. Specifically, training was halted if the validation set loss failed to show improvement over 20 consecutive epochs, subsequently restoring the weights to those which minimized the validation metric.

3. With gradient tracking disabled, the pre-trained model is tested on the corresponding MI data from the subject that had been left out.

The selected classification statistic to be tracked is the command decoding accuracy. In previous studies, it is considered that a user attains BCI control if he reaches accuracies higher than 70% in MI binary classification (M. H. Lee *et al.*, 2019).

4.3.6.1 Fine-Tuning

Once we have obtained a pre-trained model, we explore two different fine-tuning techniques in order to try to improve our current model.

- **Fine-tuning with MI data:** Once the pre-trained model has been obtained with ME data from available subjects, the premise of this technique, found on Figure 4.3, aims to further improve the trained model to adapt to the user and reduce the calibration time required for test subjects, as it only requires the subject to perform MI. However, since the available MI data per subject is considerably low, we don't have enough data to perform validation of the fine-tuning. For this reason, we have heuristically established the fine-tuning model to perform 10 epochs before arriving to a stop. In this fine tuning method. As stated on subsection 3.1.2, for most subjects we only have 30 repetitions onsets per run, half of which belong to 'rest' periods. Considering we have 3 total MI runs for each task, that makes a total of 45 repetitions available for each subject.

  Considering our limited available data, we evaluated each LOSO subject in the test set using a fine- tuning process with $N = [10, 14, 18, 22, 26, 30]$ trials. For $N = 0$, it is equivalent to testing directly in the pretained model simulating a plug and play device

and thus assessing their robustness to inter-subject variability. For $N > 0$, the fine-tuning process has 3 stages:

1. The algorithm picks N trials from each subject MI data.
2. The trained model is finetuned with the data from these trials, obtaining a subject-specific model.
3. The fine-tuned model is tested with the rest of trials for each subject.

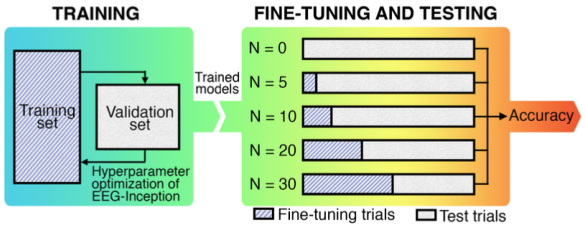


Fig. 4.3. Work flow of EEGInception. Fine-tuning process for each subject is applied with MI tials (Santamaria-Vazquez *et al.*, 2020).

- **Fine-tuning with ME data:** Since the available MI data for each subject is not resourceful enough to perform a validation approach during training, we also tried transfer learning to fine-tune the model with the subjects ME data to obtain a subject specific model. This approach has the disadvantage of requiring the subject to perform additional calibration runs, but it is an interesting approach since it would be reinforcing the model with data from actual execution of movement, which the subject will try to emulate during MI. Hence, the model becomes more adept at recognizing the neural patterns associated with actual movement, which should increase its ability to interpret similar patterns during motor imagery tasks. In this method, we are capable of performing a validation of the fine-tuning process, with early stopping also implemented. Withing this strategy visible in Figure 4.4, we evaluated each LOSO subject in the test set using a fine-tuning process with $N = [15, 21, 27, 33, 39, 45]$ trials. Again, for $N = 0$, it is equivalent to testing directly in the pretained model simulating a plug and play device. For $N > 0$, the fine-tuning process has 3 stages:

  1. The algorithm picks N trials from each subject's ME data.
  2. We perform a 80% train / dev split with the available trials. This means that if we are fine-tuning with 45 trials, 36 are used for training and 9 are used for validating.
  3. The trained model is finetuned with the data from these trials, obtaining a subject-specific model.
  4. The fine-tuned model is tested with all the MI data for each subject.

In both of these approaches, fine-tuning is applied two times. First the entire architecture, except for the final softmax layer, remains frozen (its parameters are not updated during training). The softmax layer is trained with a very low learning rate, 1e-4, until early stopping is

Fig. 4.4. Cross-validation analysis performed over each subject while fine-tuning the pre-trained model with ME data (Perez-Velasco *et al.*, 2022).

triggered or it has reached the number of iterations predetermined. Then, the entire architecture is unfrozen and trained with this low learning rate until early stopping is again activated or the iterations are completed. The initial fine-tuning phase aims to preserve the knowledge gained during pre-training by only adjusting the weights in the softmax classification layer. The subsequent fine-tuning phase further adapts the feature extraction process, especially when the target dataset significantly differs from the pre-training datasets. This procedure is adapted from the fine-tuning guidelines described in François Chollet (2021).

In chapter 4, we explored the methodology for processing EEG data, covering the signal preprocessing techniques, feature extraction methods, and classification algorithms that form the core of our analysis. In the next chapter, chapter 5, we will shift our focus to the results obtained from these methods, presenting the performance metrics, analyzing the outcomes, and discussing the implications of these findings in the context of the study's objectives. This analysis will help us understand the effectiveness of the approaches used and guide future research directions

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 8, 384, 8]	520
BatchNorm2d-2	[-1, 8, 384, 8]	16
ELU-3	[-1, 8, 384, 8]	0
Conv2d-4	[-1, 8, 384, 8]	264
BatchNorm2d-5	[-1, 8, 384, 8]	16
ELU-6	[-1, 8, 384, 8]	0
Conv2d-7	[-1, 8, 384, 8]	136
BatchNorm2d-8	[-1, 8, 384, 8]	16
ELU-9	[-1, 8, 384, 8]	0
AvgPool2d-10	[-1, 24, 192, 8]	0
Conv2d-11	[-1, 48, 192, 1]	384
BatchNorm2d-12	[-1, 48, 192, 1]	96
ELU-13	[-1, 48, 192, 1]	0
AvgPool2d-14	[-1, 48, 96, 1]	0
Conv2d-15	[-1, 8, 96, 1]	6,144
BatchNorm2d-16	[-1, 8, 96, 1]	16
ELU-17	[-1, 8, 96, 1]	0
Conv2d-18	[-1, 8, 96, 1]	3,072
BatchNorm2d-19	[-1, 8, 96, 1]	16
ELU-20	[-1, 8, 96, 1]	0
Conv2d-21	[-1, 8, 96, 1]	1,536
BatchNorm2d-22	[-1, 8, 96, 1]	16
ELU-23	[-1, 8, 96, 1]	0
AvgPool2d-24	[-1, 24, 48, 1]	0
Conv2d-25	[-1, 12, 48, 1]	2,304
BatchNorm2d-26	[-1, 12, 48, 1]	24
ELU-27	[-1, 12, 48, 1]	0
AvgPool2d-28	[-1, 12, 24, 1]	0
Conv2d-29	[-1, 6, 24, 1]	288
BatchNorm2d-30	[-1, 6, 24, 1]	12
ELU-31	[-1, 6, 24, 1]	0
AvgPool2d-32	[-1, 6, 12, 1]	0
Flatten-33	[-1, 72]	0
Linear-34	[-1, 2]	146
Softmax-35	[-1, 2]	0
Total params:	15,022	
Trainable params:	15,022	
Non-trainable params:	0	
Input size (MB):	0.01	
Forward/backward pass size (MB):	2.30	
Params size (MB):	0.06	
Estimated Total Size (MB):	2.37	

Table 4.3. Overview of EEGInception Architecture

Layer (type)	Output Shape	Param #
Conv3d-1	[-1, 24, 2, 384, 5]	1,560
BatchNorm3d-2	[-1, 24, 2, 384, 5]	48
ELU-3	[-1, 24, 2, 384, 5]	0
Conv3d-4	[-1, 24, 2, 384, 5]	792
BatchNorm3d-5	[-1, 24, 2, 384, 5]	48
ELU-6	[-1, 24, 2, 384, 5]	0
Conv3d-7	[-1, 24, 2, 384, 5]	408
BatchNorm3d-8	[-1, 24, 2, 384, 5]	48
ELU-9	[-1, 24, 2, 384, 5]	0
AvgPool3d-10	[-1, 72, 2, 192, 5]	0
Conv3d-11	[-1, 72, 2, 192, 1]	360
BatchNorm3d-12	[-1, 72, 2, 192, 1]	144
ELU-13	[-1, 72, 2, 192, 1]	0
Conv3d-14	[-1, 24, 2, 192, 5]	27,672
BatchNorm3d-15	[-1, 24, 2, 192, 5]	48
ELU-16	[-1, 24, 2, 192, 5]	0
Conv3d-17	[-1, 24, 2, 192, 5]	13,848
BatchNorm3d-18	[-1, 24, 2, 192, 5]	48
ELU-19	[-1, 24, 2, 192, 5]	0
Conv3d-20	[-1, 24, 2, 192, 5]	6,936
BatchNorm3d-21	[-1, 24, 2, 192, 5]	48
ELU-22	[-1, 24, 2, 192, 5]	0
AvgPool3d-23	[-1, 72, 2, 96, 5]	0
Conv3d-24	[-1, 72, 2, 96, 1]	360
BatchNorm3d-25	[-1, 72, 2, 96, 1]	144
ELU-26	[-1, 72, 2, 96, 1]	0
Conv3d-27	[-1, 36, 2, 96, 5]	2,592
BatchNorm3d-28	[-1, 36, 2, 96, 5]	72
ELU-29	[-1, 36, 2, 96, 5]	0
Conv3d-30	[-1, 36, 2, 96, 5]	41,508
BatchNorm3d-31	[-1, 36, 2, 96, 5]	72
ELU-32	[-1, 36, 2, 96, 5]	0
AvgPool3d-33	[-1, 36, 2, 48, 5]	0
Conv3d-34	[-1, 36, 2, 48, 1]	6,480
BatchNorm3d-35	[-1, 36, 2, 48, 1]	72
ELU-36	[-1, 36, 2, 48, 1]	0
Conv3d-37	[-1, 36, 2, 48, 5]	1,296
BatchNorm3d-38	[-1, 36, 2, 48, 5]	72
ELU-39	[-1, 36, 2, 48, 5]	0
Conv3d-40	[-1, 36, 2, 48, 5]	10,404
BatchNorm3d-41	[-1, 36, 2, 48, 5]	72
ELU-42	[-1, 36, 2, 48, 5]	0
AvgPool3d-43	[-1, 36, 2, 24, 5]	0
Conv3d-44	[-1, 36, 2, 24, 1]	6,480
BatchNorm3d-45	[-1, 36, 2, 24, 1]	72
ELU-46	[-1, 36, 2, 24, 1]	0
Conv3d-47	[-1, 18, 2, 24, 5]	648
BatchNorm3d-48	[-1, 18, 2, 24, 5]	36
ELU-49	[-1, 18, 2, 24, 5]	0
Conv3d-50	[-1, 18, 2, 24, 5]	2,610
BatchNorm3d-51	[-1, 18, 2, 24, 5]	36
ELU-52	[-1, 18, 2, 24, 5]	0
AvgPool3d-53	[-1, 18, 2, 12, 5]	0
Conv3d-54	[-1, 18, 2, 12, 1]	1,620
BatchNorm3d-55	[-1, 18, 2, 12, 1]	36
ELU-56	[-1, 18, 2, 12, 1]	0
Conv3d-57	[-1, 18, 2, 12, 5]	1,296
BatchNorm3d-58	[-1, 18, 2, 12, 5]	36
ELU-59	[-1, 18, 2, 12, 5]	0
AvgPool3d-60	[-1, 18, 2, 6, 5]	0
Conv3d-61	[-1, 18, 1, 6, 1]	3,240
BatchNorm3d-62	[-1, 18, 1, 6, 1]	36
ELU-63	[-1, 18, 1, 6, 1]	0
Conv3d-64	[-1, 18, 1, 6, 1]	3,240
BatchNorm3d-65	[-1, 18, 1, 6, 1]	36
ELU-66	[-1, 18, 1, 6, 1]	0
Conv3d-67	[-1, 18, 1, 6, 1]	360
BatchNorm3d-68	[-1, 18, 1, 6, 1]	36
ELU-69	[-1, 18, 1, 6, 1]	0
Conv3d-70	[-1, 18, 1, 1, 1]	1,944
BatchNorm3d-71	[-1, 18, 1, 1, 1]	36
ELU-72	[-1, 18, 1, 1, 1]	0
Conv3d-73	[-1, 36, 1, 1, 1]	216
BatchNorm3d-74	[-1, 36, 1, 1, 1]	72
ELU-75	[-1, 36, 1, 1, 1]	0
Conv3d-76	[-1, 36, 1, 1, 1]	1,296
BatchNorm3d-77	[-1, 36, 1, 1, 1]	72
ELU-78	[-1, 36, 1, 1, 1]	0
Conv3d-79	[-1, 36, 1, 1, 1]	1,296
BatchNorm3d-80	[-1, 36, 1, 1, 1]	72
ELU-81	[-1, 36, 1, 1, 1]	0
Conv3d-82	[-1, 36, 1, 1, 1]	1,296
BatchNorm3d-83	[-1, 36, 1, 1, 1]	72
ELU-84	[-1, 36, 1, 1, 1]	0
Conv3d-85	[-1, 36, 1, 1, 1]	1,296
BatchNorm3d-86	[-1, 36, 1, 1, 1]	72
ELU-87	[-1, 36, 1, 1, 1]	0
Flatten-88	[-1, 36]	0
Linear-89	[-1, 2]	74
Softmax-90	[-1, 2]	0
Total params:	142,784	
Trainable params:	142,784	
Non-trainable params:	0	
Input size (MB):	0.01	
Forward/backward pass size (MB):	15.01	
Params size (MB):	0.54	
Estimated Total Size (MB):	15.56	

Table 4.4. Overview of the EEGSym Architecture for an input of 3 seconds and 8 EEG channels

5. RESULTS

This chapter presents the findings from the experiments conducted in this study. The results are structured into two main sections: the first section details the outcomes of the characterisation process, while the second section focuses on the binary classification results, showcasing the performance metrics of our cross-task transfer learning model in different settings.

5.1. Results of the Characterization of Motor Execution and Motor Imagery

In this section we present the results of the characterization performed to our datasets, encompassing both our spectral (RP) and temporal (ERD/ERS) features. A thorough statistical analysis is conducted to compare these features, providing insights into their distinct and shared characteristics.

5.1.1. Relative Bandpower

We begin this section by showing the topographic plots of the computed RP for the ME and MI datasets. For each dataset, RP was computed in our 7 bands of interest Delta (1-4Hz), Theta (4-8Hz), Alhpa (8-12Hz), Mu (12-15Hz), Beta1 (15-20Hz), Beta2 (20-30Hz), Gamma (30-50Hz). RP results for each respective pre-processed dataset were averaged separately for each task, left or right, between the 109 subjects. Figure 5.1 shows the topographic plots of the RP results of our ME dataset and Figure 5.2 shows the topographic plots of the RP results of our MI dataset.

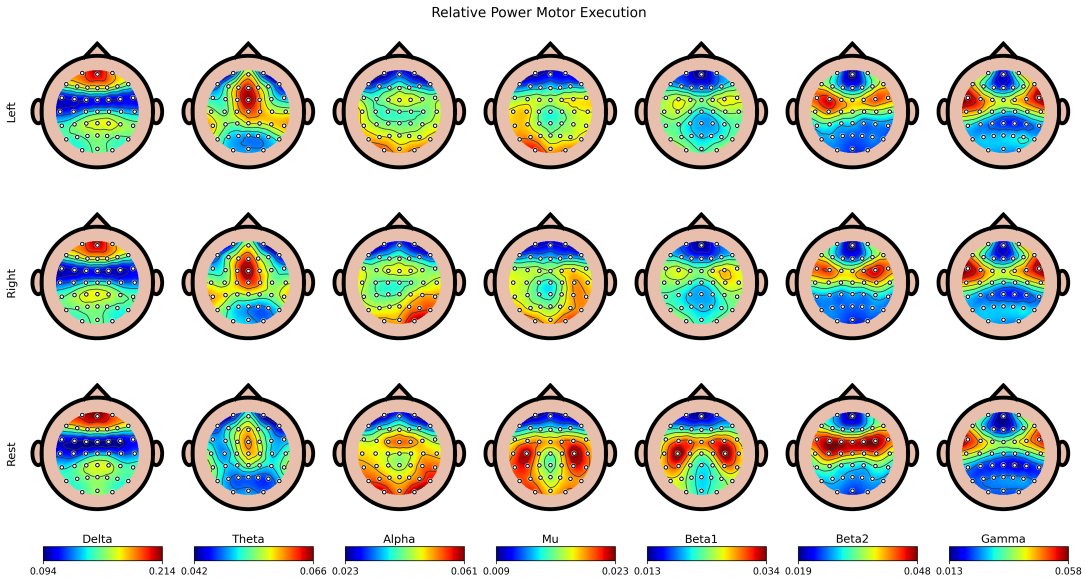


Fig. 5.1. Relative Power for the Execution of Movement of Left and Right Tasks

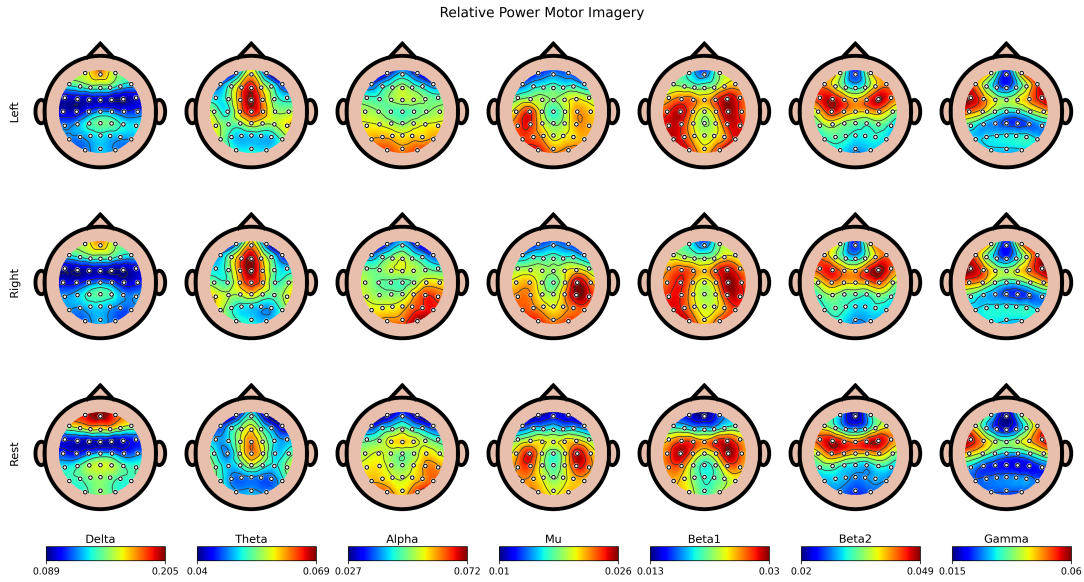
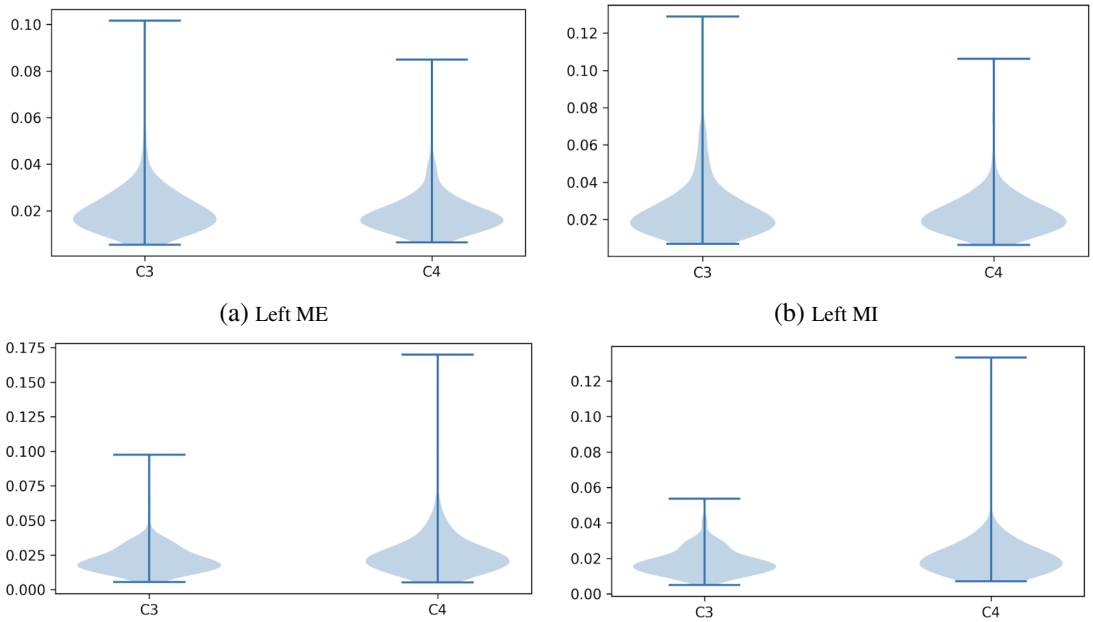


Fig. 5.2. Relative Power for the Imagination of Movement of Left and Right Tasks



(a) Left ME (b) Left MI

(c) Right ME (d) Right MI

Fig. 5.3. Violinplots of the RP distribution in electrodes C3 and C4 between the different tasks (left and right) and paradigms (ME and MI).

During the execution of motor tasks, neurons become less synchronized, which is associated with increased cortical excitability and enhanced information processing capacity (Pfurtscheller, 2001). If the task to be performed is related to the right hand, the primary effect observed in EEG is a desynchronization in the contralateral (left) sensorimotor cortex. This desyncronization is perceived as a decrease in power in the specific frequency bands (mu and beta). The right sensorimotor cortex (ipsilateral to the movement) is less engaged in the motor task. As a

result, it maintains more synchronized activity, often seen as relatively higher power in the mu and beta bands compared to the contralateral side.

With this in mind, we can observe how on both Figure 5.1 and Figure 5.2, this pattern is clearly present particulary on Mu and Beta 1 bands. Both for ME and MI we can observe a clear lateralization of the RP and centralization above the sensorimotor cortex, while the reduction in RP is greater on the contralateral side of the task being performed. This lateralization can be further observed in the violinplots represented in Figure 5.3.

In order to establish the significance of the results obtained, a comparison between ME and MI results was performed with Wilcoxon signed rank test (Wilcoxon, 1945). We compared each task, band and channel subject by subject across all 109 available to obtain the results. We selected this method due to its non-parametric nature meaning it does not assume that the data follows a normal distribution. This is particularly useful in our context because EEG data, which is often non-normally distributed. Additionally, this test is designed for paired data, making it ideal for our study where we are comparing ME and MI measurements from the same subjects. It effectively handles the within-subject variability by considering the differences between paired observations, thus providing a more accurate comparison of ME and MI.

After conducting the Wilcoxon signed-rank test, we applied Bonferroni correction to address the issue of multiple comparisons (Bonferroni, 1936). When performing multiple statistical tests, the likelihood of obtaining false-positive results increases. The Bonferroni correction mitigates this risk by adjusting the significance threshold based on the number of comparisons made. Data is considered statistically significant if a p-value inferior to a 0.05 threshold is obtained.

The results of this statistical analysis can be observed in both topographic plots in order to obtain a better spatial visualization, but also in matrix form to better observe the relevant channels. Figure 5.4 and Figure 5.6 show the results of the statistical analysis performed over our bands of interest between ME and MI paradigms for the Left task. On the other hand, Figure 5.5 and Figure 5.7 show the corresponding results for the Right task. On both Figure 5.4 and Figure 5.5, we can observe how particularly in our main bands of interest (mu and beta1), it can be seen how over the sensorimotor region, these are no statistical significant differences to be found, providing strong evidence that the neuron activation during the MI paradigms behaves similarly than those in ME. Furthermore, taking a closer look through Figure 5.6 and Figure 5.7, most EEG channels find statistical significant differences in the behaviour between both paradigms, except for those found within the primary motor cortex and include most frontal cortex electrodes (i.e. FC1, FC2, FCZ) and lateralized central cortex electrodes (C3 and C1 for the left task; C2, C3 and C4 for the right task).

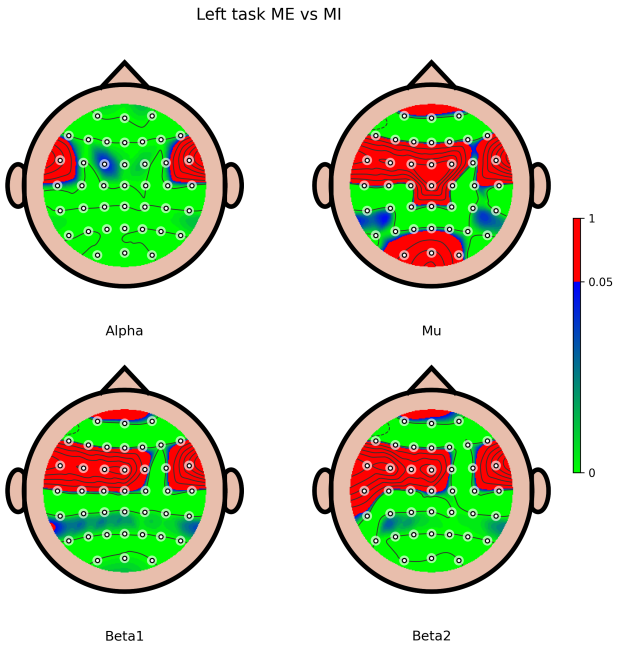


Fig. 5.4. Statistical differences corrected with Bonferroni between ME and MI over Alpha, Mu, Beta1 and Beta2 bands for the left task. Not significant differences are marked in red (p-value > 0.05)

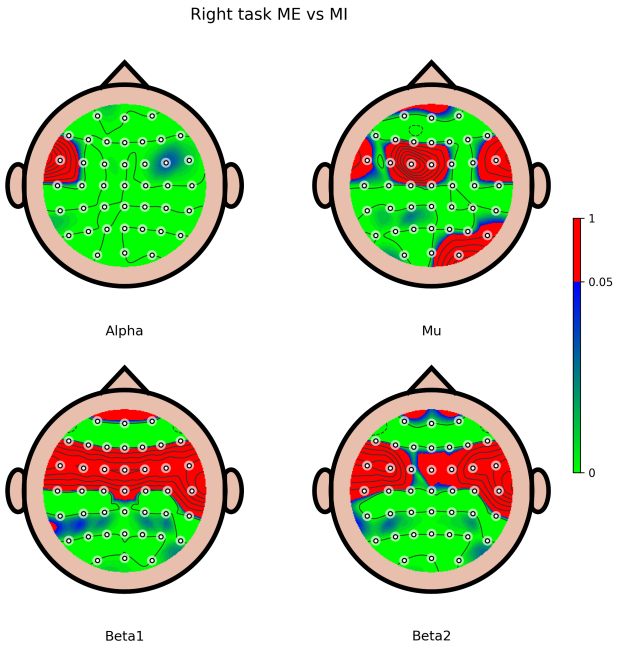


Fig. 5.5. Statistical differences corrected with Bonferroni between ME and MI over Alpha, Mu, Beta1 and Beta2 bands for the right task. Not significant differences are marked in red (p-value > 0.05)

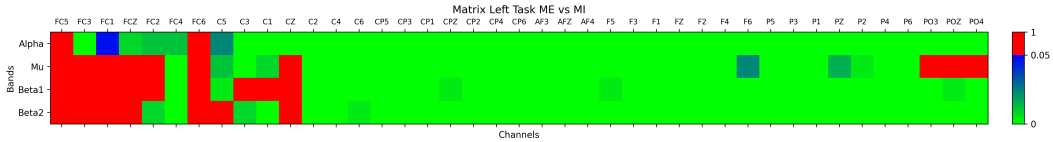


Fig. 5.6. Matrix containing the statistical analysis per band and per channel between ME and MI for the left task. Not significant differences are marked in red (p-value > 0.05)

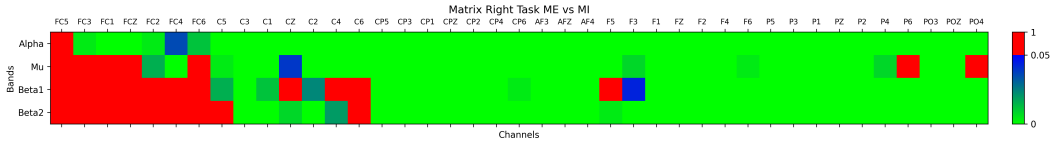


Fig. 5.7. Matrix containing the statistical analysis per band and per channel between ME and MI for the right task. Not significant differences are marked in red (p-value > 0.05)

5.1.2. ERD/ERS

We begin this section with a topographic view of the ERD/ERS of our pre-processed datasets, visible across 200ms temporal windows since the onset. We performed the computation of ERD/ERS averaged with our 109 subjects across all 7 target bands, but we only display the most relevant bands: Alpha, Mu, Beta1 and Beta2. For this feature, we expect to observe a strong event related desyncrhonization on the contralateral hemisphere of the scalp. This desynchronization should last the whole duration of the task, which lasts for 4000ms after the onset.

For the statistical analysis, we performed Wilcoxon signed-rank test, correcting the False Discovery Rate with the Bonferroni approach for all channels on each temporal window and task.

Across all bands, it can be observed how the right task has a strong and more distinct contralateral ERD. On the other hand, contralateral dominance is not prominently observed during left hand motor imagery.

In all figures, we can observe how we obtain a gradually stronger desyncrhonization after the onset. However, an interesting behaviour is noticed when comparing ME and MI patterns across the temporal windows, as for the ME paradigm, the ERD is observed across the complete duration of the task. On the contrary, the MI paradigm presents an ERD that starts to gradually disipate at around 2500 seconds after the onset, being followed by the respective ERS.

When analysing each band individually, Beta1 and Beta2 visible in Figure 5.20, Figure 5.21, Figure 5.22, Figure 5.23 and Figure 5.26, Figure 5.27, Figure 5.28, Figure 5.29 respectively, show weaker ERDs that start to fade briefly after the onset, barely holding on through the task. This ERD is followed by the characteristic ERS known as the beta rebound. Furthermore,

Alpha patterns visible in Figure 5.8, Figure 5.9, Figure 5.10, Figure 5.11 manages to hold the ERD for a longer period, meanwhile even though Mu patterns also fade slightly in Figure 5.14, Figure 5.15, Figure 5.16, Figure 5.17, they are capable to hold the ERD during the whole length of the mental task. Finally, even though all bands show the ERD pattern above the sensorimotor region of the contralateral hemisphere, Beta1 and Mu appear to provide the more focalized patterns, with the Mu ERD being the strongest.

These findings are corroborated by the statistical analysis: Figure 5.18, Figure 5.19 Mu provides the least significant differences between the ERD/ERS patterns in the ME and MI paradigm across most channels and across all temporal windows. The remaining bands found in Figure 5.12, Figure 5.13, Figure 5.24, Figure 5.25 and Figure 5.30, Figure 5.31, provide similar results for the most instant response after the onset, but then start showing statistically significant differences in the patterns, specifically above the sensorimotor cortex characteristic to the ERD.

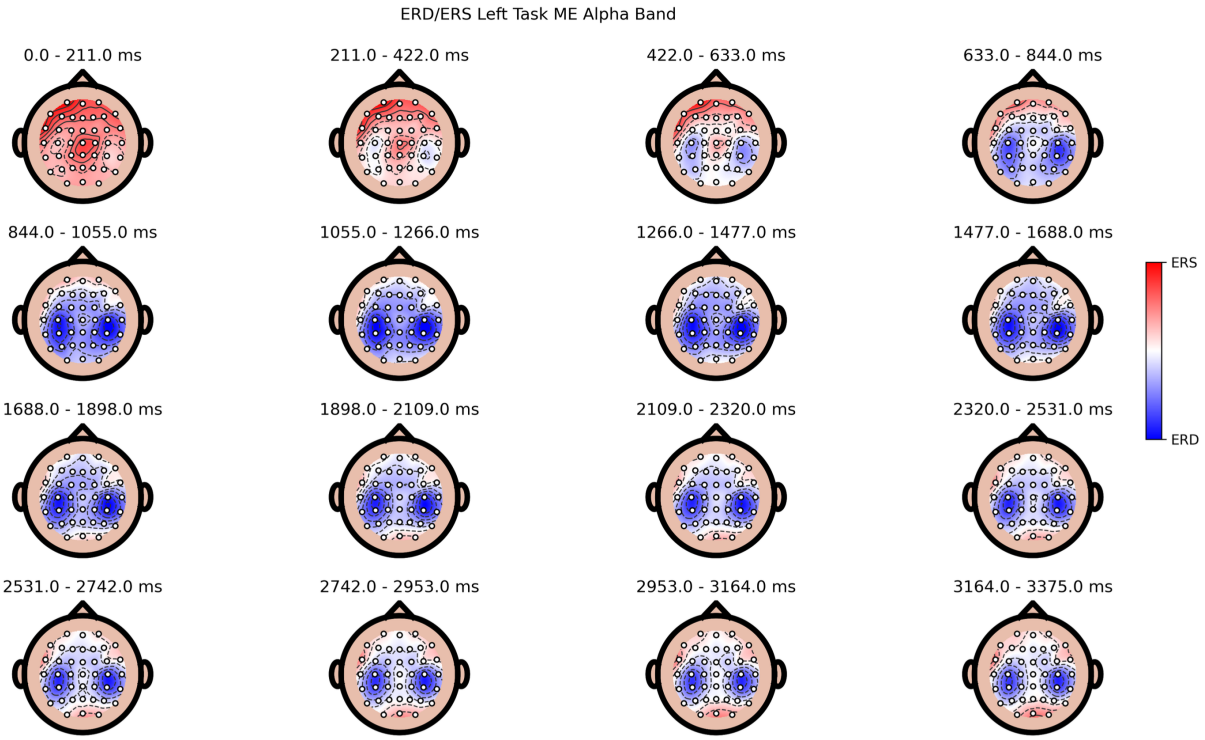


Fig. 5.8. ERD/ERS topoplots over 200ms temporal windows after the onset in the Alpha band for the left ME task.

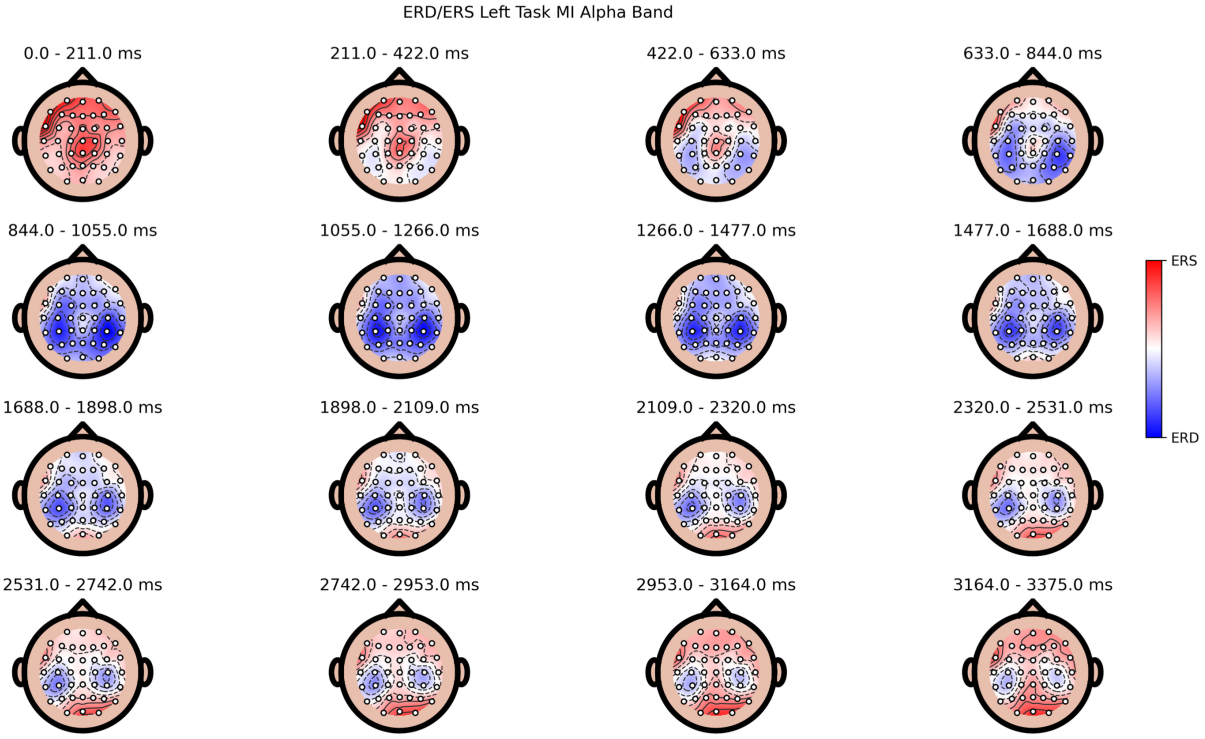


Fig. 5.9. ERD/ERS topoplots over 200ms temporal windows after the onset in Alpha band for the left MI task.

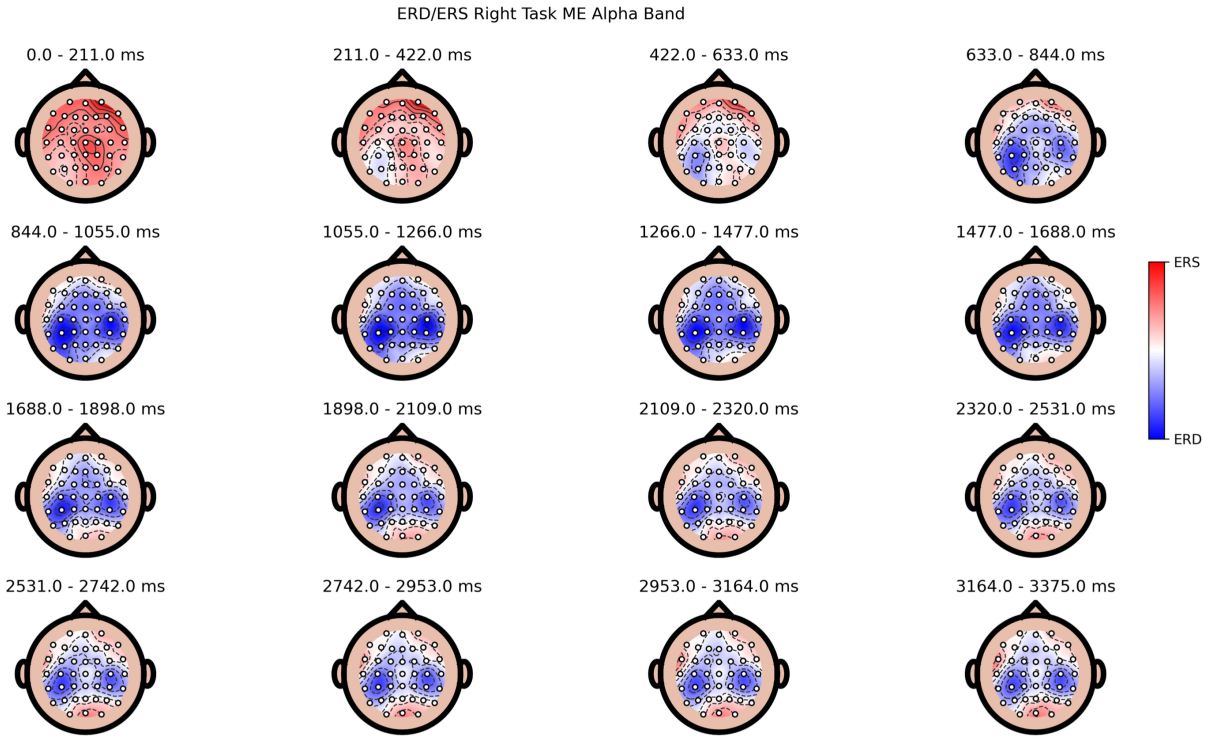


Fig. 5.10. ERD/ERS topoplots over 200ms temporal windows after the onset in Alpha band for the right ME task.

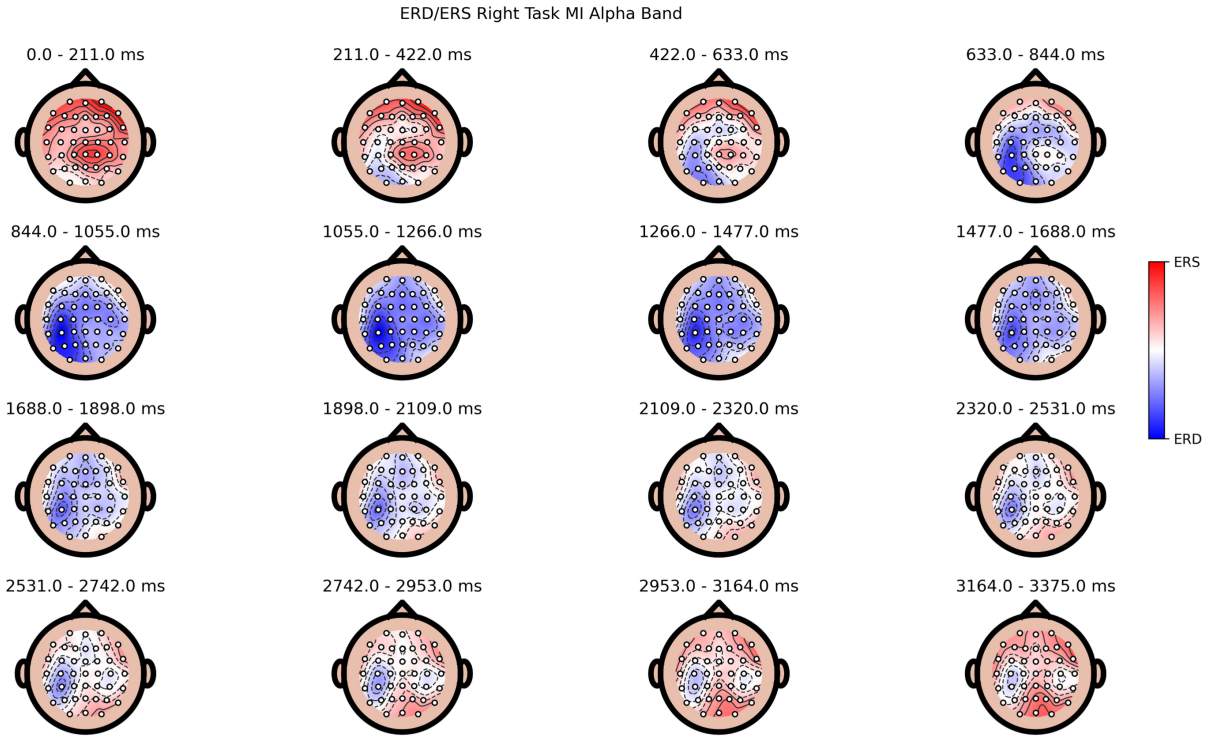


Fig. 5.11. ERD/ERS topoplots over 200ms temporal windows after the onset in Alpha band for the right MI task.

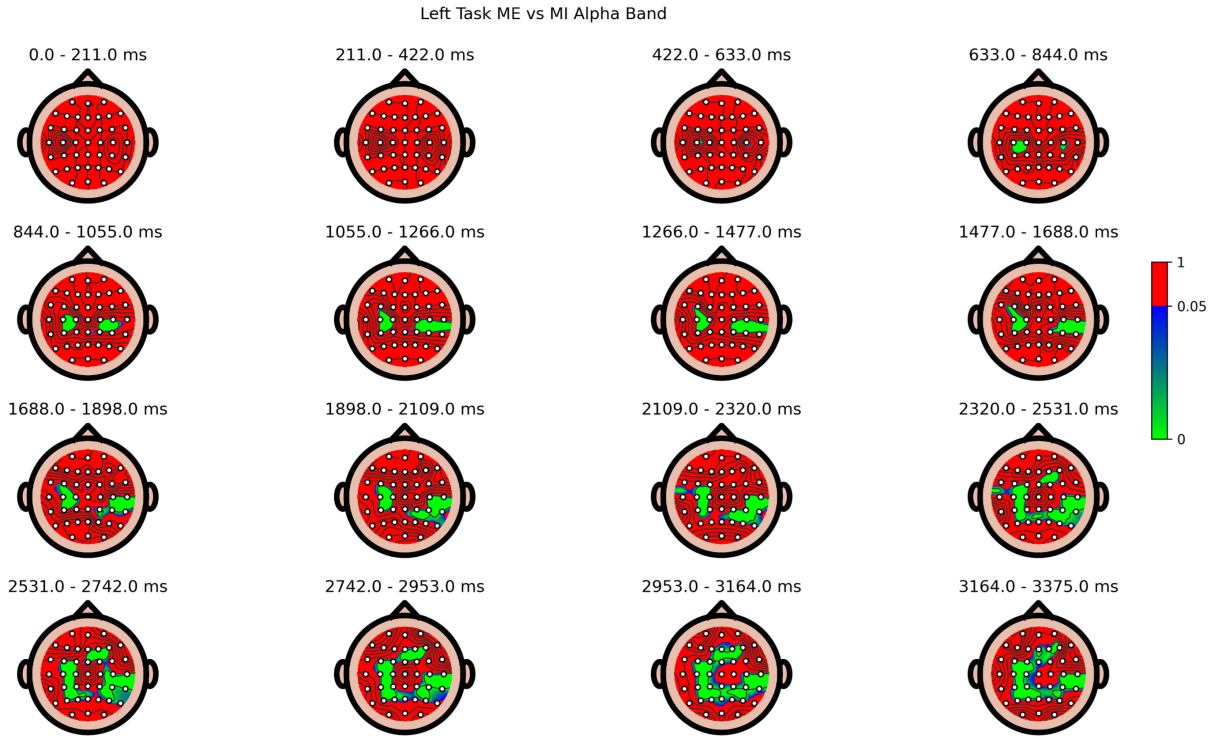


Fig. 5.12. Statistical differences between ME and MI over the Alpha band for the left task. Not significant differences are marked in red (p-value > 0.05)

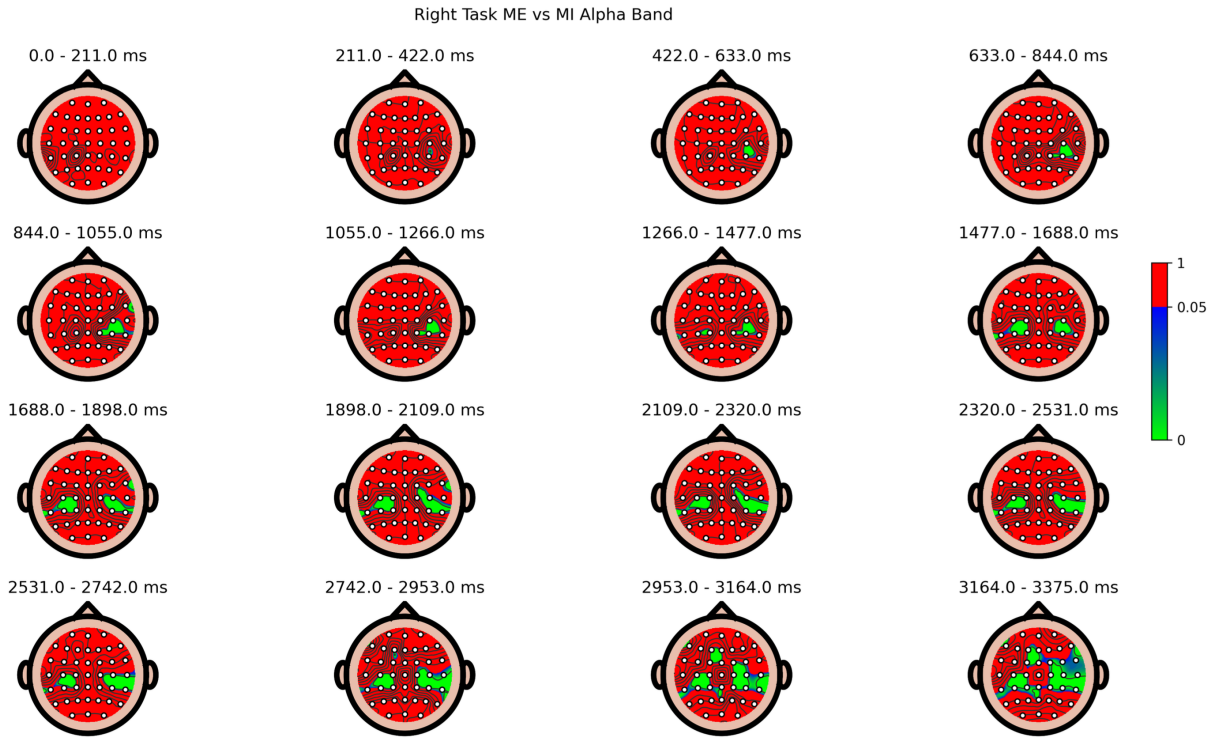


Fig. 5.13. Statistical differences between ME and MI over the Alpha band for the right task. Not significant differences are marked in red (p-value > 0.05)

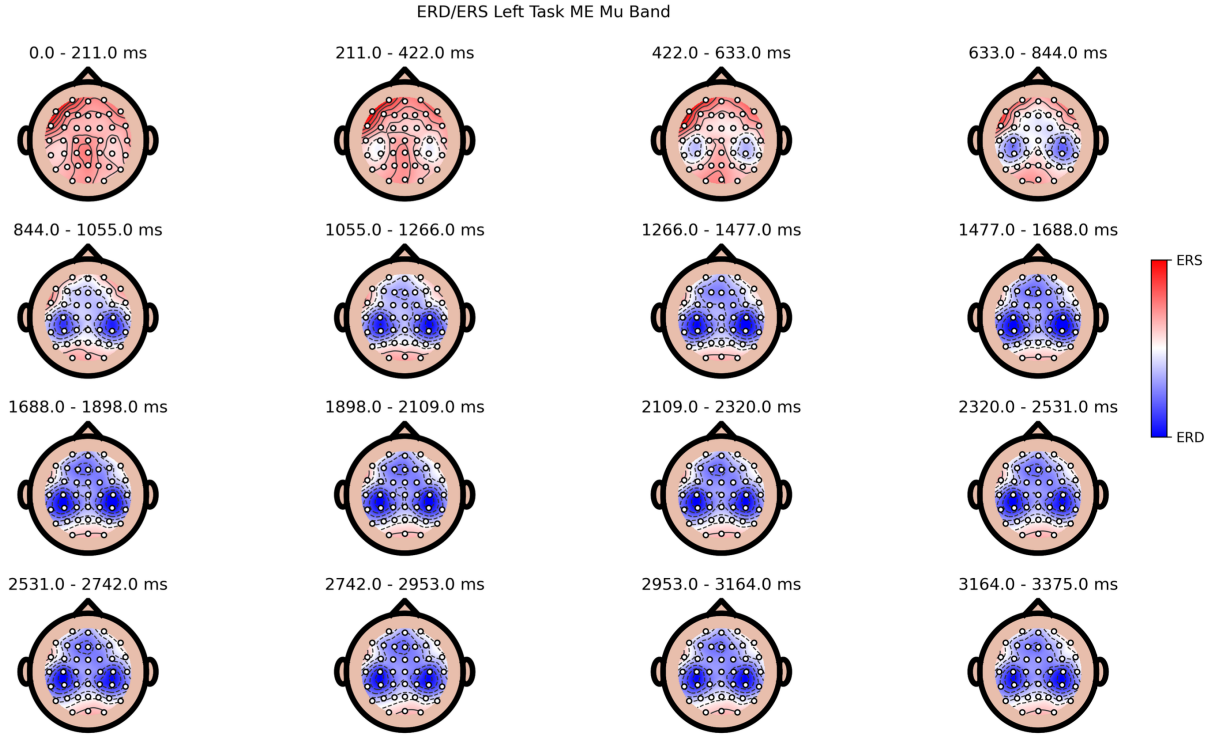


Fig. 5.14. ERD/ERS topoplots over 200ms temporal windows after the onset in the Mu band for the left ME task.

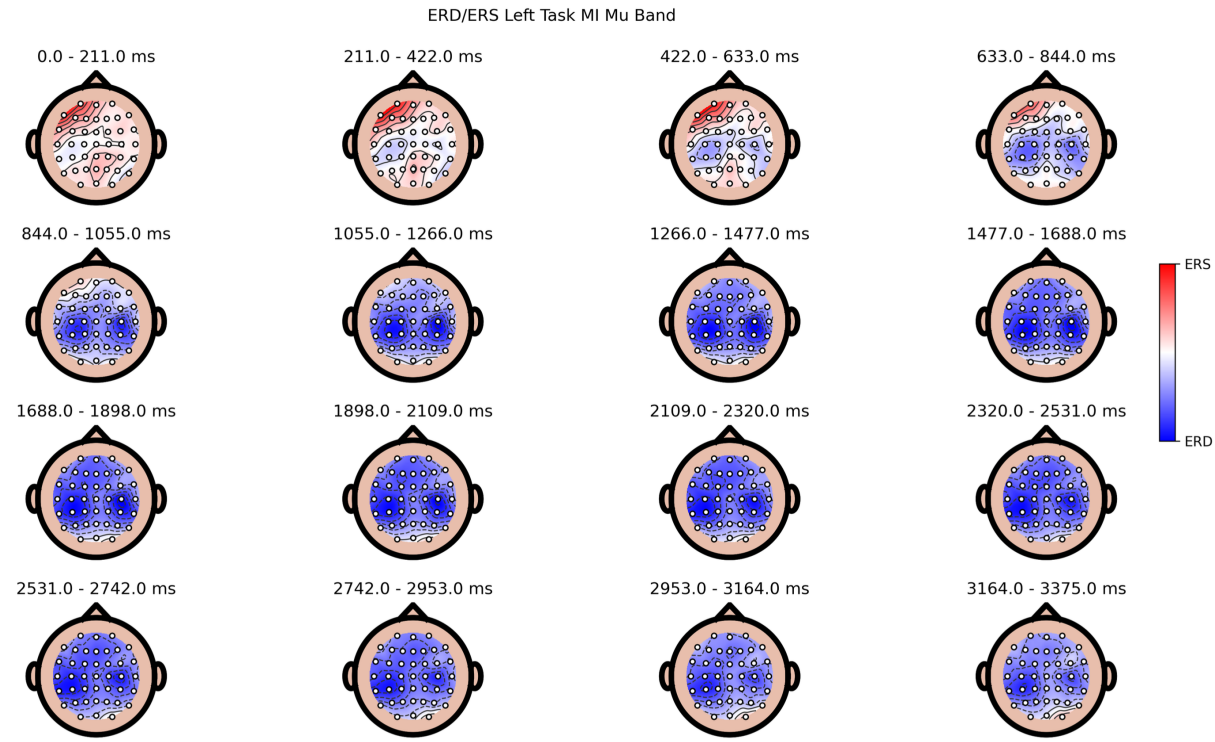


Fig. 5.15. ERD/ERS topoplots over 200ms temporal windows after the onset in the Mu band for the left MI task.

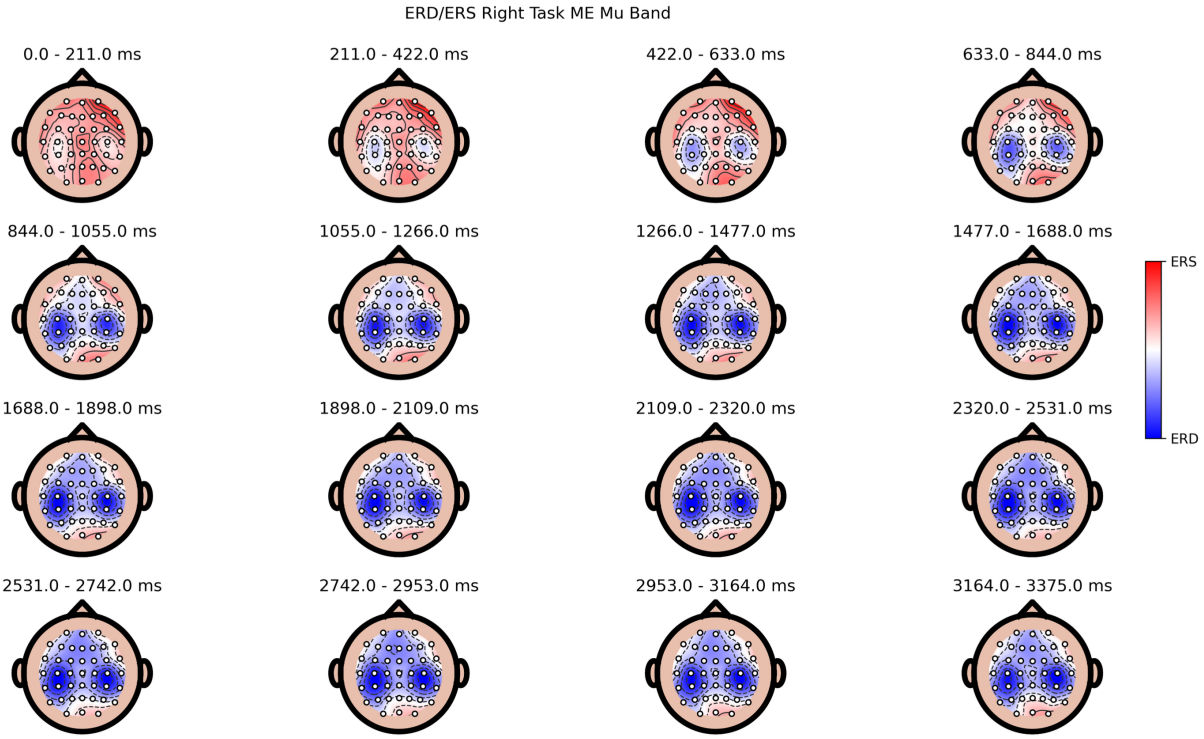


Fig. 5.16. ERD/ERS topoplots over 200ms temporal windows after the onset in the Mu band for the right ME task.

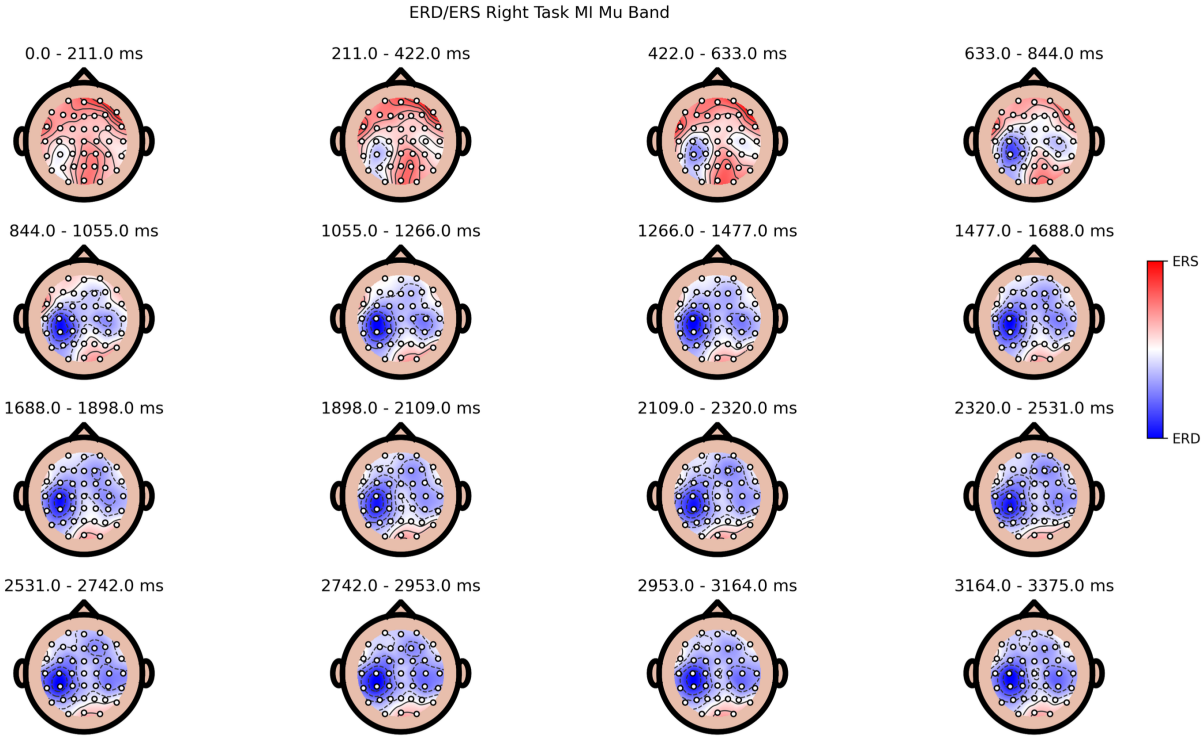


Fig. 5.17. ERD/ERS topoplots over 200ms temporal windows after the onset in the Mu band for the right MI task.

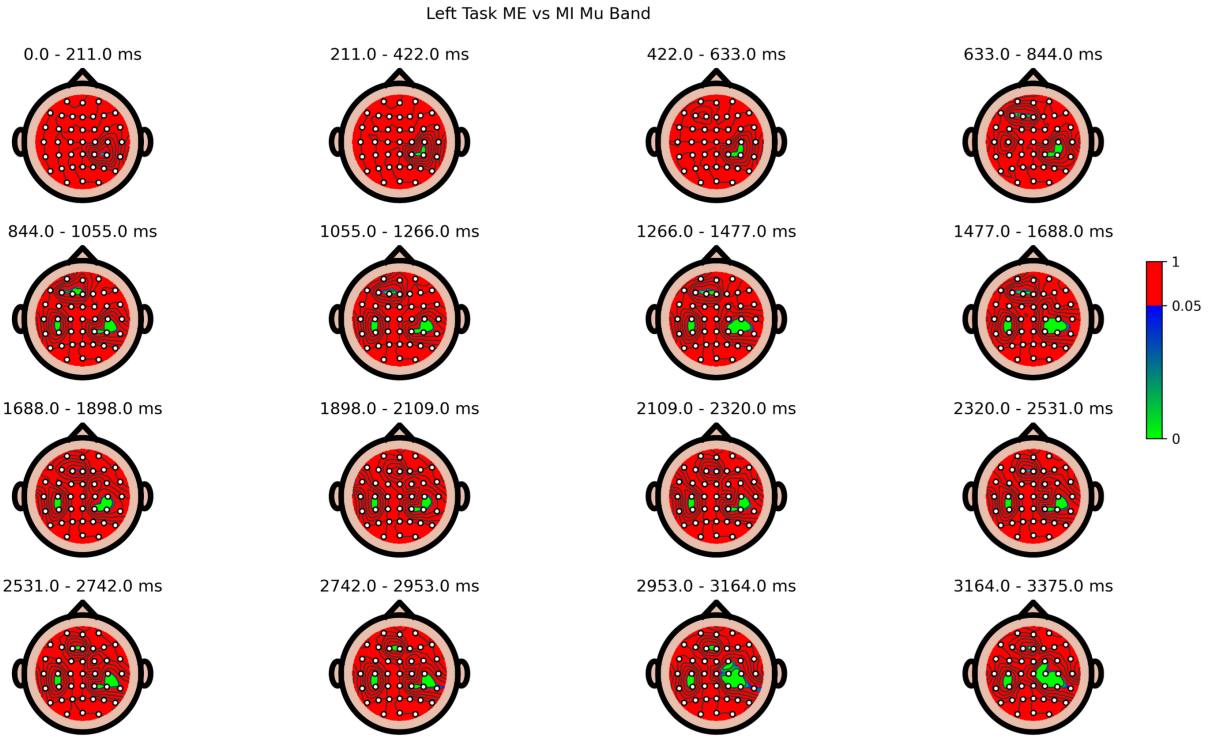


Fig. 5.18. Statistical differences between ME and MI over the Mu band for the left task. Not significant differences are marked in red (p-value > 0.05)

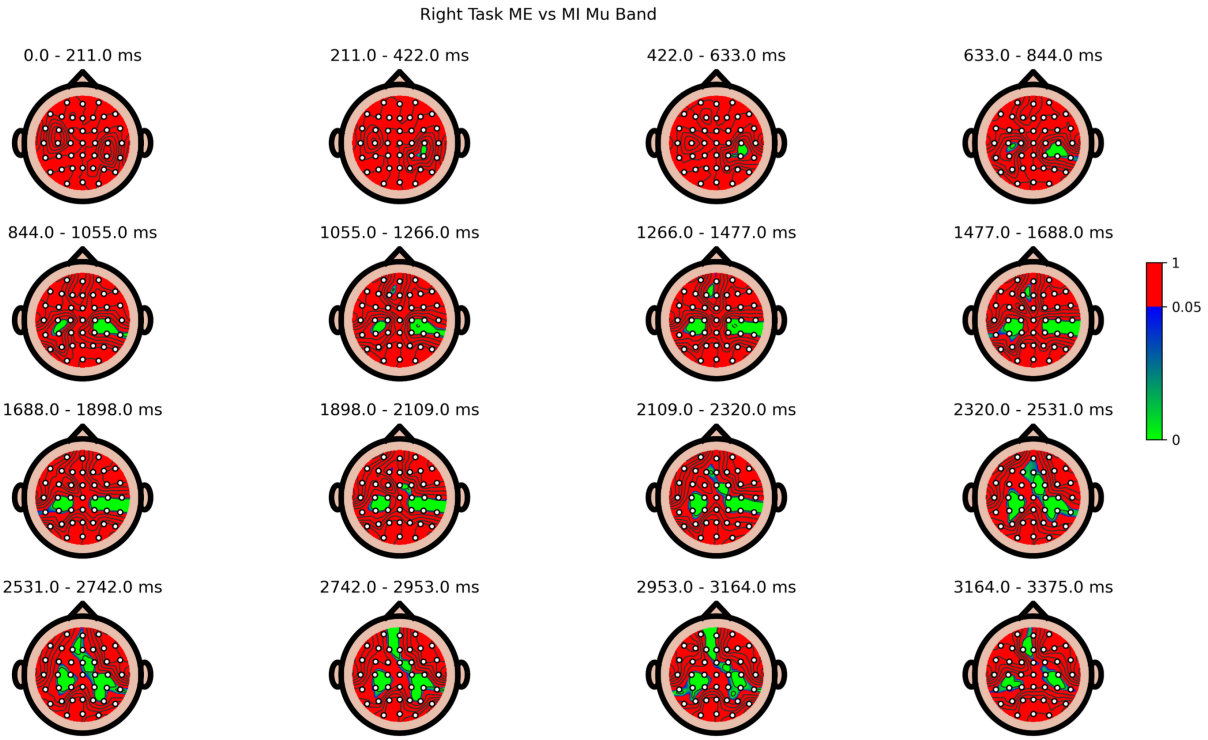


Fig. 5.19. Statistical differences between ME and MI over the Mu band for the right task. Not significant differences are marked in red (p-value > 0.05)

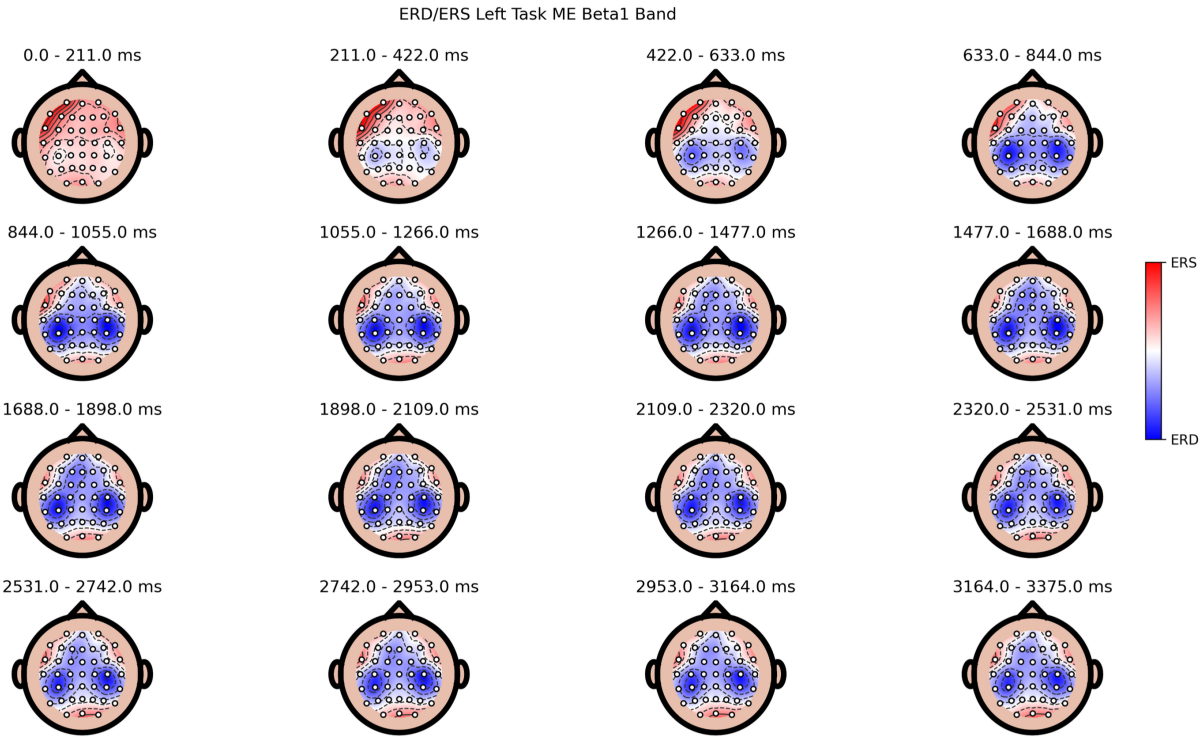


Fig. 5.20. ERD/ERS topoplots over 200ms temporal windows after the onset in the Beta1 band for the left ME task.

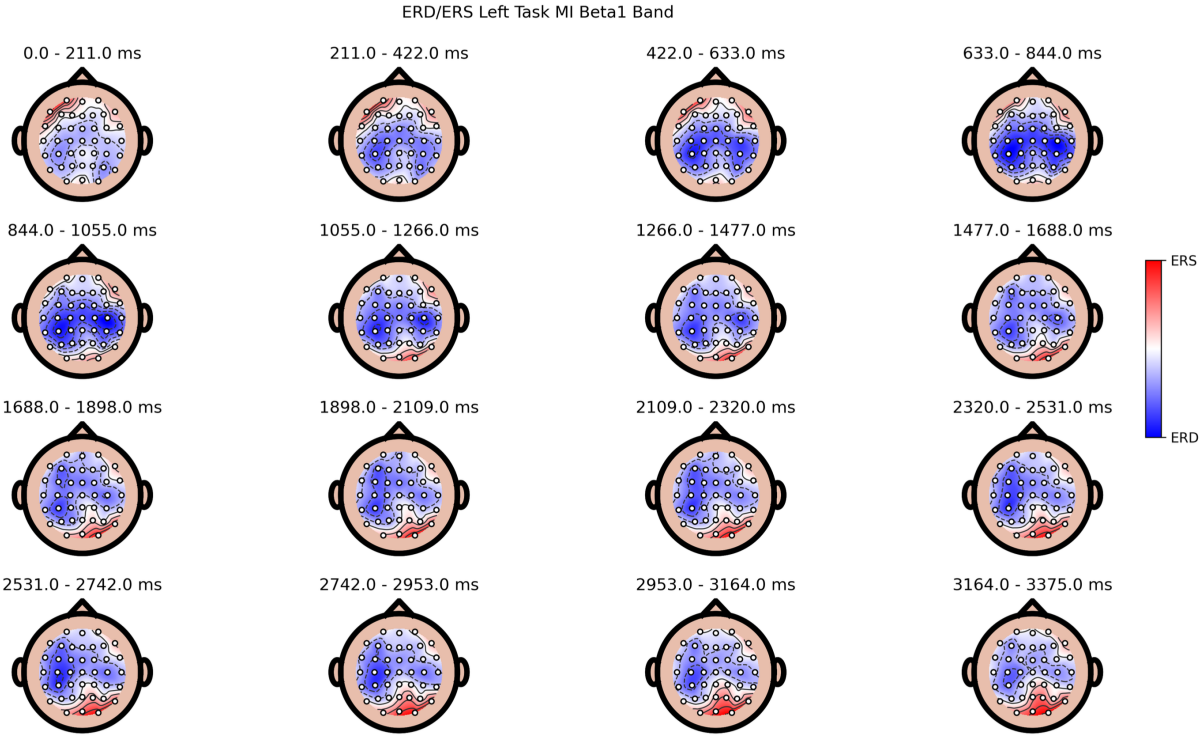


Fig. 5.21. ERD/ERS topoplots over 200ms temporal windows after the onset in the Beta1 band for the left MI task.

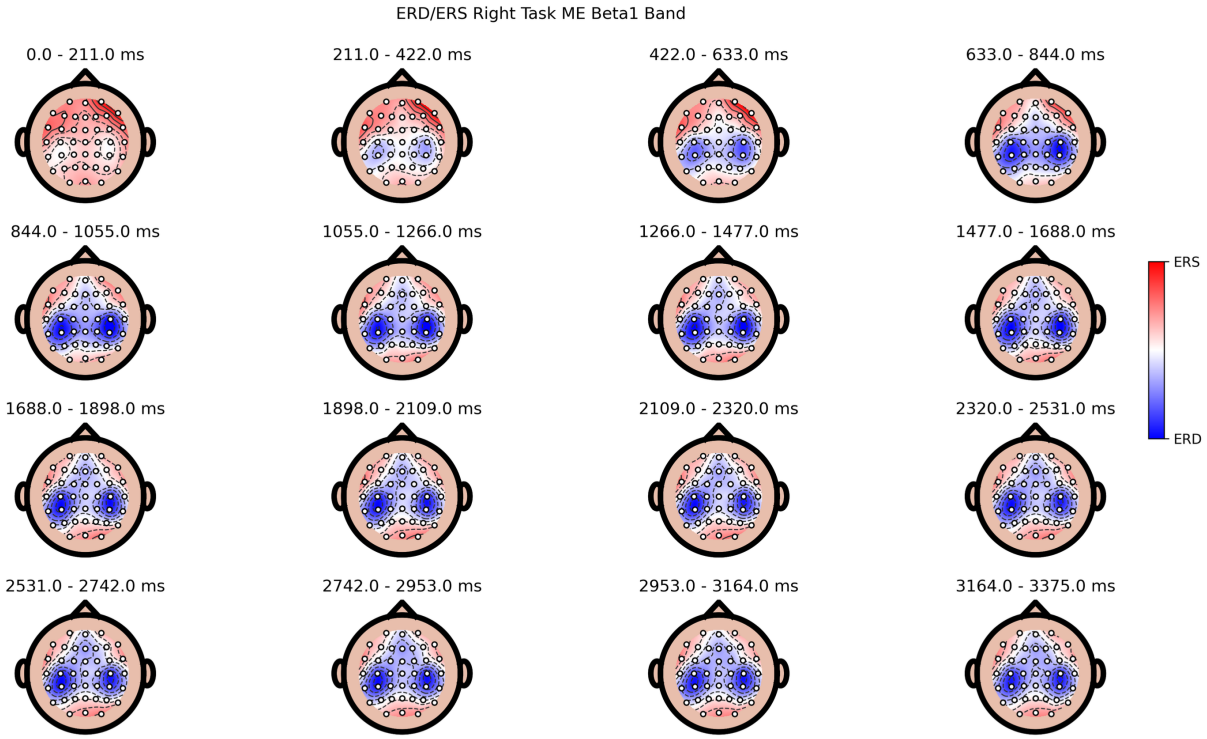


Fig. 5.22. ERD/ERS topoplots over 200ms temporal windows after the onset in the Beta1 band for the right ME task.

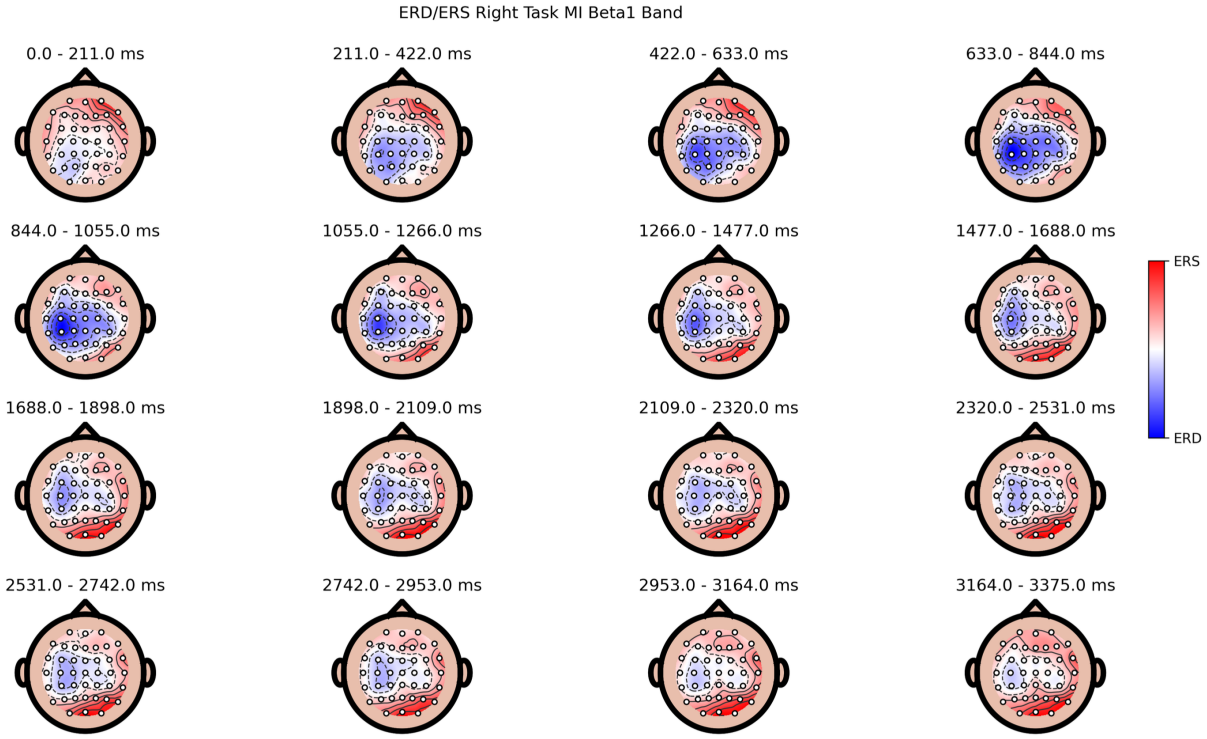


Fig. 5.23. ERD/ERS topoplots over 200ms temporal windows after the onset in the Beta1 band for the right MI task.

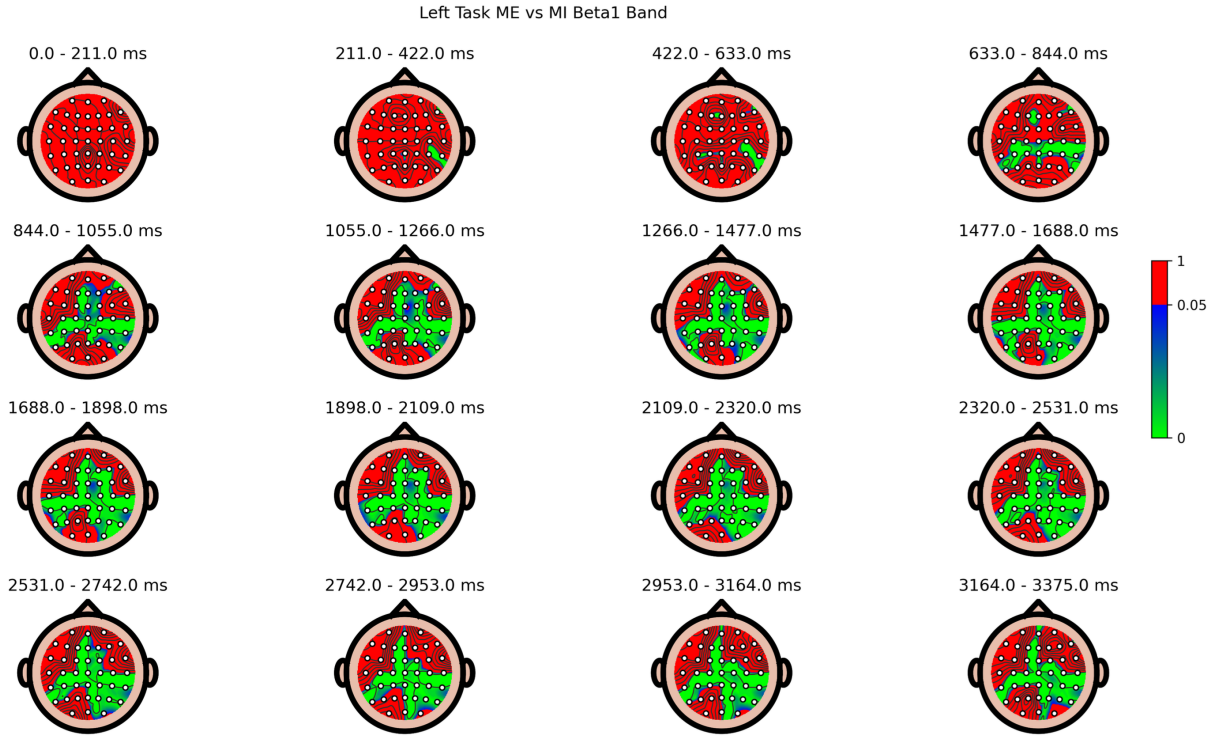


Fig. 5.24. Statistical differences between ME and MI over the Beta1 band for the left task. Not significant differences are marked in red (p-value > 0.05)

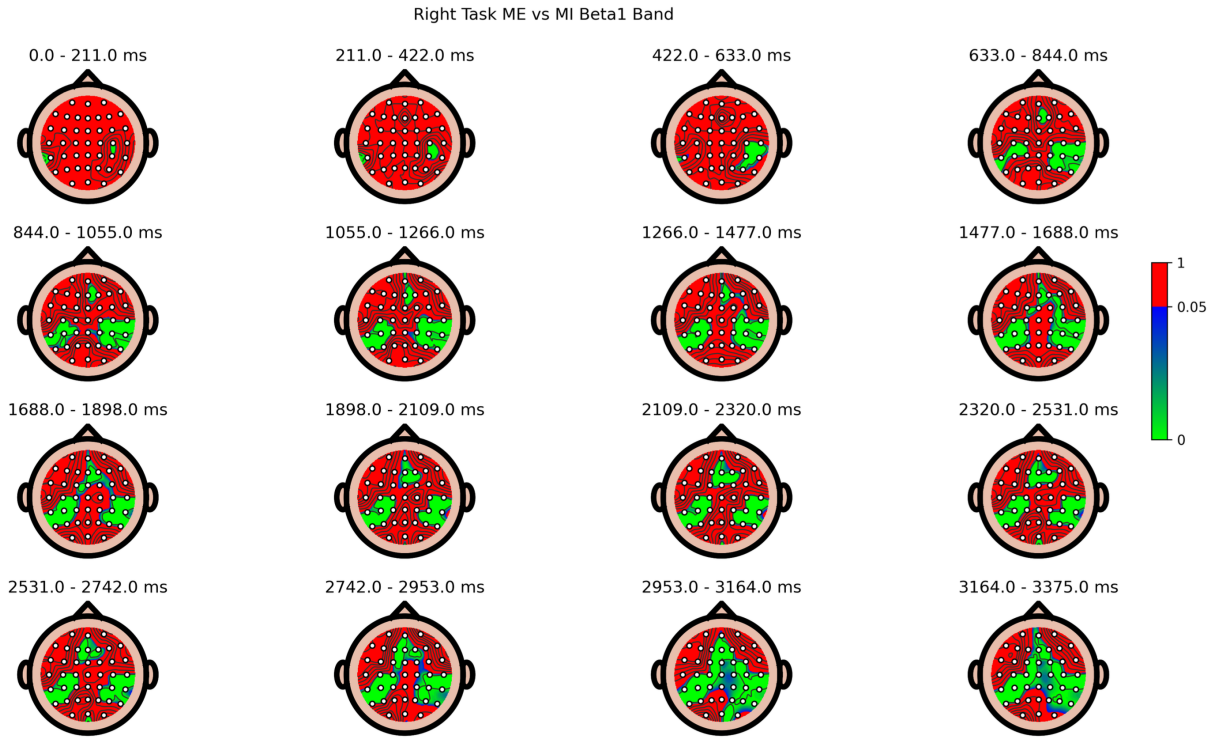


Fig. 5.25. Statistical differences between ME and MI over the Beta1 band for the right task. Not significant differences are marked in red (p-value > 0.05)

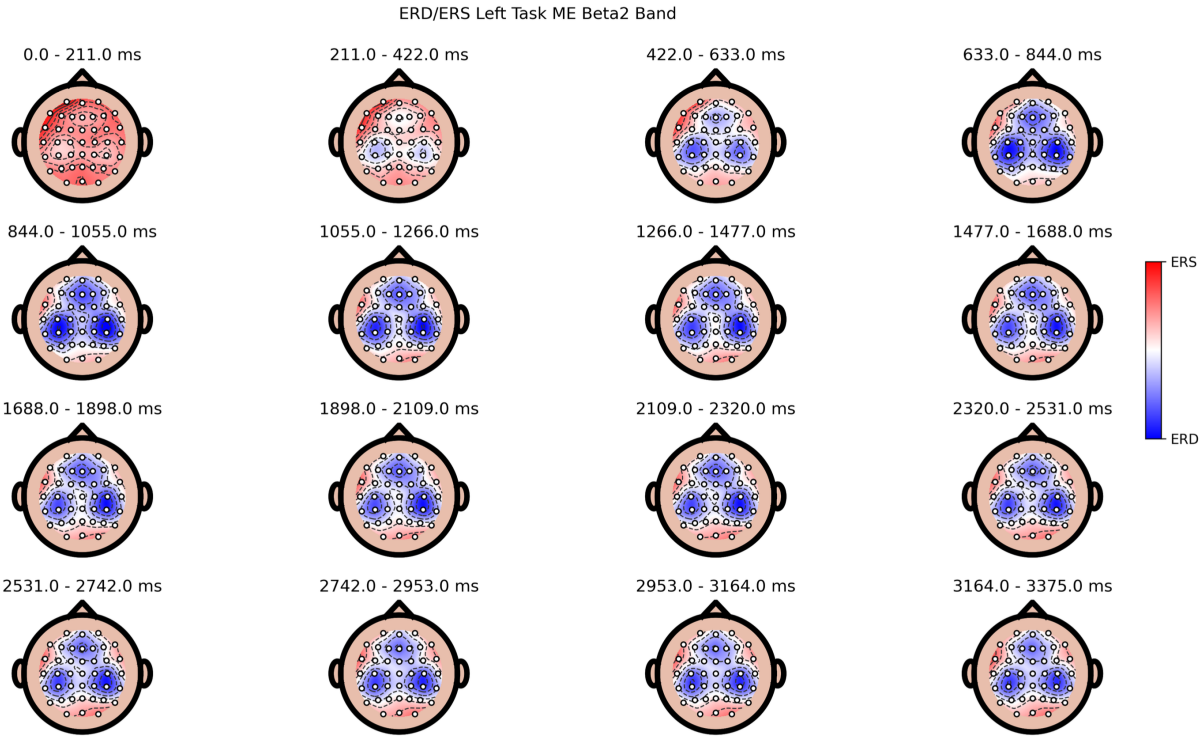


Fig. 5.26. ERD/ERS topoplots over 200ms temporal windows after the onset in the Beta2 band for the left ME task.

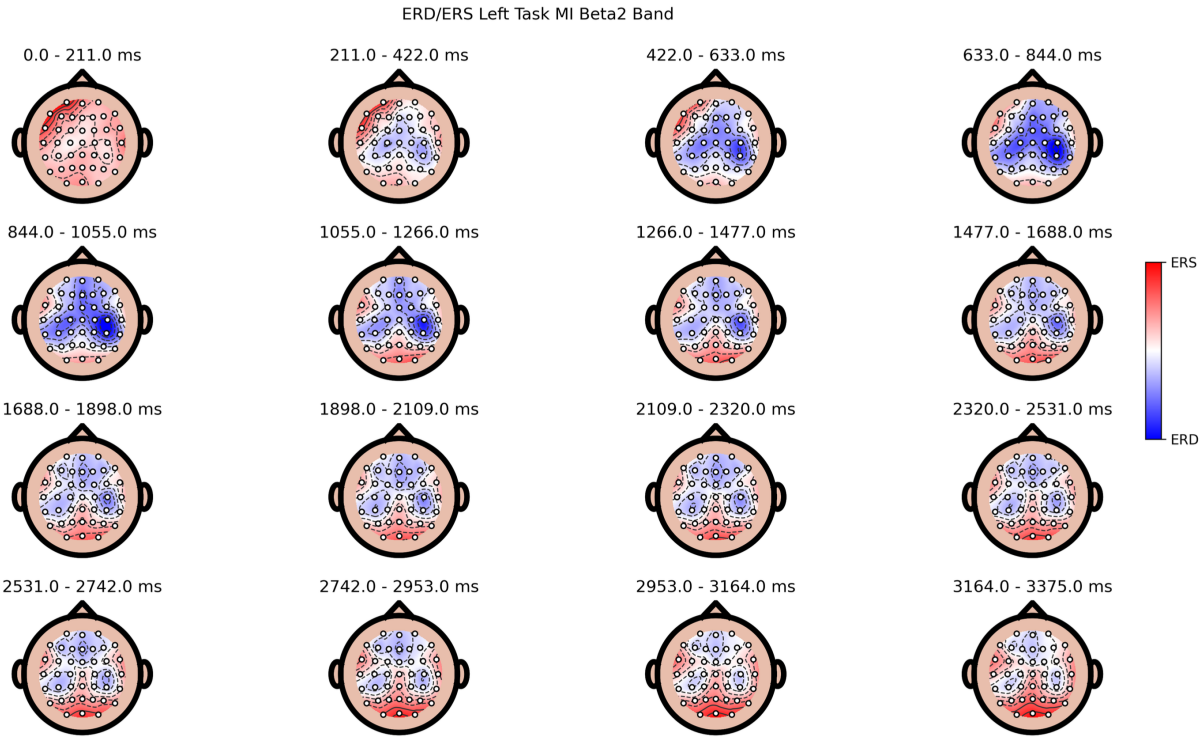


Fig. 5.27. ERD/ERS topoplots over 200ms temporal windows after the onset in the Beta2 band for the left MI task.

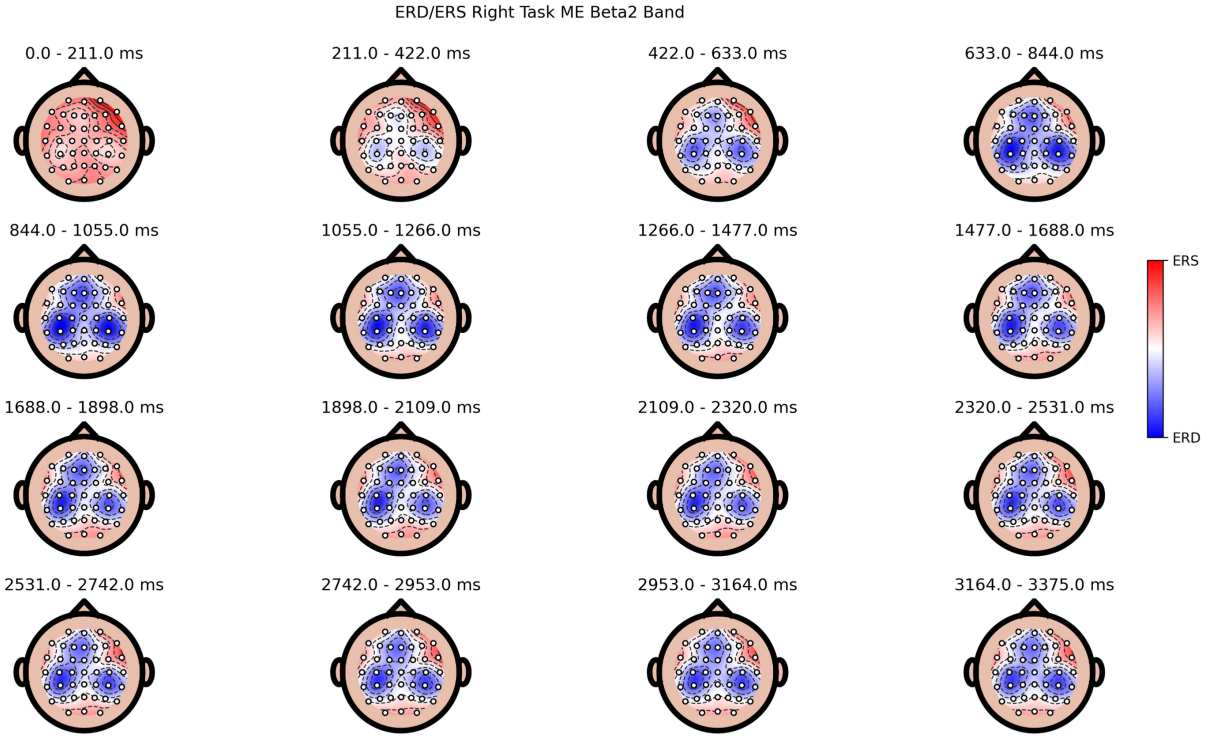


Fig. 5.28. ERD/ERS topoplots over 200ms temporal windows after the onset in the Beta2 band for the right ME task.

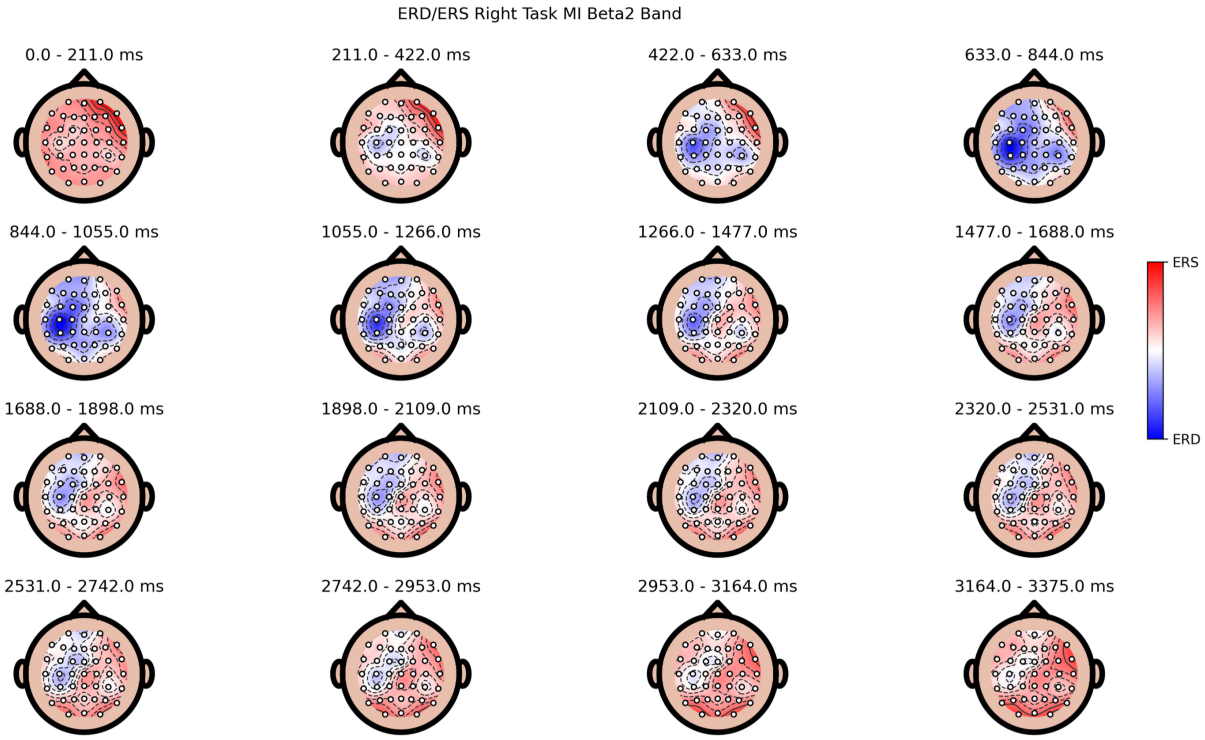


Fig. 5.29. ERD/ERS topoplots over 200ms temporal windows after the onset in the Beta2 band for the right MI task.

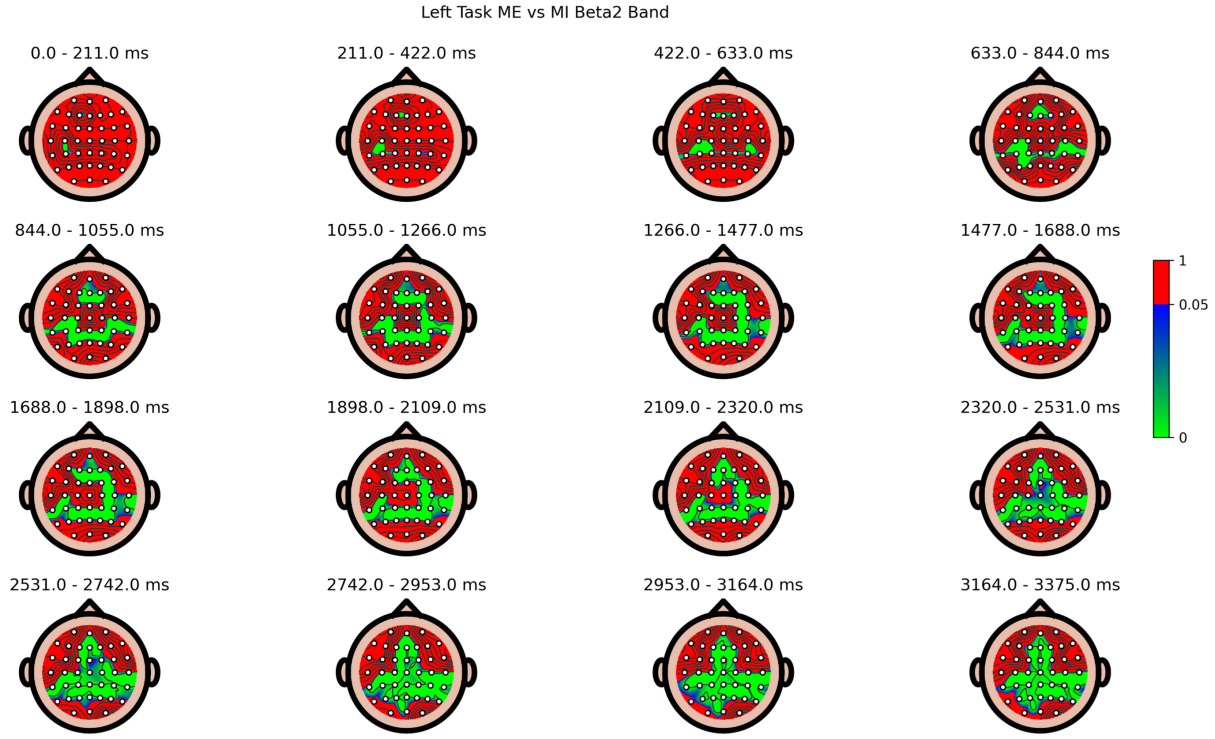


Fig. 5.30. Statistical differences between ME and MI over the Beta2 band for the left task. Not significant differences are marked in red (p-value > 0.05)

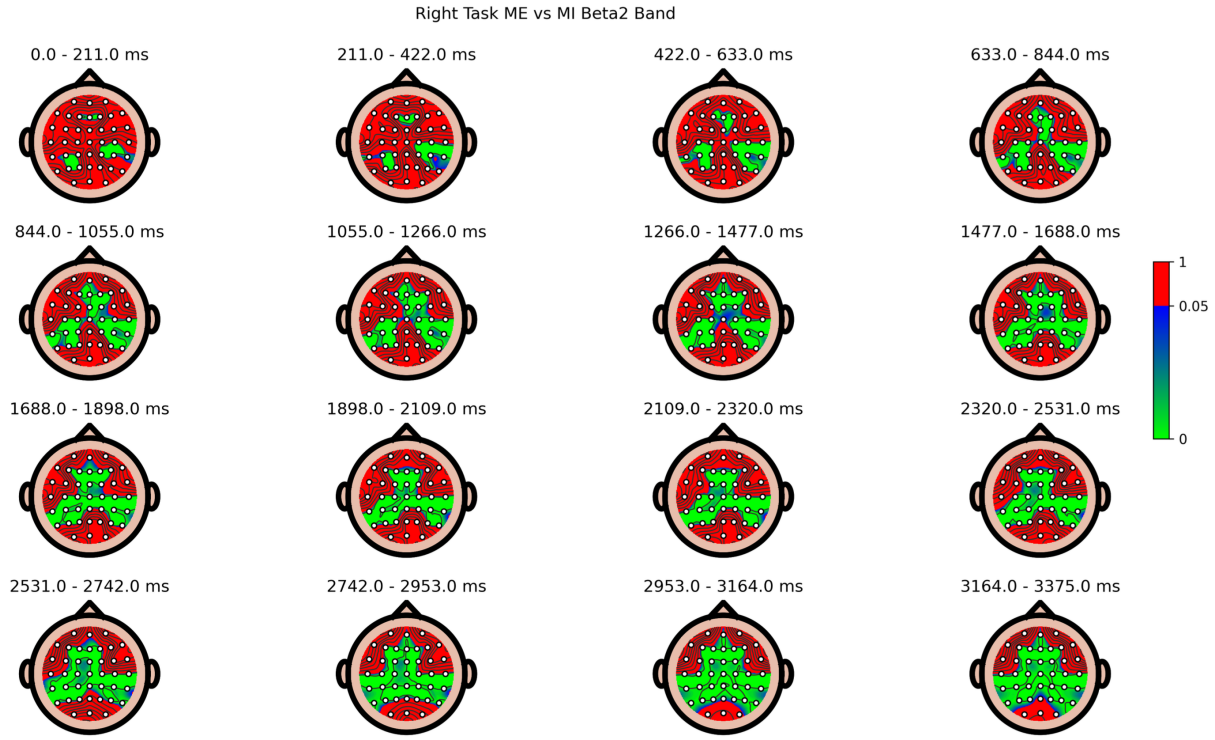


Fig. 5.31. Statistical differences between ME and MI over the Beta2 band for the right task. Not significant differences are marked in red (p-value > 0.05)

5.2. Classification experiment

In this section, we present the results of the binary classification algorithm applied to our dataset. To assess the performance of the model, we will mainly focus on the accuracy metric, which quantifies the proportion of correctly classified instances out of the total instances. Other metrics include the number of subjects that proved to achieve accuracies high enough for BCI control (>70%) or number of subjects that improved accuracy. This section provides the results for the transfer learning model, as well as for the 2 approaches towards fine-tuning.

All models were trained on a NVIDIA GeForce RTX 2080 Ti with PyTorch 2.2.1+cu121.

5.2.1. Transfer learning

During the LOSO training with our model, we can observe on Table 5.1 how our model presents really accuracies for the ME training and validation sets, reaching metrics of almost 90%. Furthermore, once our model is tested, we can observe highly satisfactory results for MI classification in a complete cross-subject pipeline without any calibration performed on test subjects. Our model can correctly predict over 80% of the MI trials, also allowing for BCI control in over 80% of the subjects.

Additionally, as it has been observed in the previous subsection that the ERD during the MI paradigm is shorter and does not persist throughout the entire task length as compared to ME, we tested our model both for inputs of 2 seconds and 3 seconds. The results concluded in similar results for both input times, but with slightly better metrics for the acquisition of signal during 3000ms.

Input time	Train Acc (ME)	Dev Acc (ME)	Test Acc (MI)	BCI control
2000 ms	89.01 ± 0.9	82.34 ± 1.1	80.31 ± 11.2	84/109
3000 ms	89.56 ± 0.7	83.14 ± 1.1	80.85 ± 10.5	88/109

Table 5.1. A table with two rows and two columns

5.2.2. Fine-tuning

In this subsection, we dive into the results of our model after our 2 fine-tuning techniques are applied separately on their respective pre-trained models. Test results are also provided for each process of fine-tuning. Table 5.2 shows the results of our fine tuning applied with MI data.

The performance metrics include accuracy (Acc), BCI control index (BCI), subjects that showed same accuracy or improved after fine-tuning (Subjs imp), and mean improvement of said subjects (mean imp).

As explained in subsubsection 4.3.6.1, two fine tuning methods were applied, one tuned with ME data, and another tuned with MI. In each of those methods, two fine tuning processes were consecutively applied. The first processes freezes the entire architecture except for the

final layer, while the second process unfreezes all layers. For both input times, the complete fine-tuning with both processes resulted in higher accuracies compared to only the first process, unfreezing of the softmax layer. The highest accuracy achieved was 81.94% with a 3000 ms input time and both MI tuning processes. In contrast, the highest accuracy with a 2000 ms input time was 80.87%, also with both MI fine-tuning processes.

The BCI control index was consistently higher when two fine-tuning processes were applied. The maximum BCI control index observed was 94 with a 3000 ms input time and two fine-tuning processes. For the 2000 ms input time, the highest BCI control index was 92, also with two fine-tuning processes.

The number of subjects showing improvement and the mean improvements were generally higher with the 3000 ms input time compared to the 2000 ms input time. Applying two fine-tuning processes consistently yielded better results. The highest number of subjects showing improvement was 79, observed with a 3000 ms input time and two fine-tuning processes. The maximum mean improvement was 3.35% and 3.24%, seen in the 2000 ms and 3000 ms input times respectively.

MI fine-tuning yielded higher results in terms of the metrics obtained compared to ME fine-tuning, particularly with longer input times and two fine-tuning processes. This suggests that MI fine-tuning is more effective for enhancing the performance of the classification model. Highest results for ME fine tuning were obtained while using only 39 of the 45 available trials, while the best results obtained for MI fine tuning are obtained while tuning with 26 of 45 trials available.

MI Fine tuning	N	Input time							
		3000 ms				2000 ms			
		Acc	BCI Control	Subjs imp	Mean imp	Acc	BCI Control	Subjs imp	Mean imp
1st process	0	80.59±10.4	88	-	-	80.11±11.0	84	-	-
	10	80.40±11.3	89	66	2.76	79.50±11.3	86	58	3.25
	14	81.37±10.7	91	70	3.24	79.64±11.1	87	68	2.28
	18	80.59±10.4	90	73	2.19	79.72±10.7	90	60	3.08
	22	80.27±11.0	88	62	2.70	79.12±10.9	87	58	2.84
	26	80.55±10.8	93	65	2.74	79.68±11.1	88	69	2.48
	30	80.74±11.1	90	77	2.10	80.08±11.2	86	68	2.99
Both processes	10	81.43±10.5	90	76	3.02	79.96±11.1	89	65	3.35
	14	81.17±10.4	92	69	3.04	80.33±11.2	89	70	3.34
	18	81.80±10.6	93	79	3.14	80.44±11.2	86	73	3.33
	22	81.42±10.7	91	80	2.55	81.14±10.5	91	77	3.17
	26	81.94±10.9	94	82	3.15	80.59±10.8	92	72	3.15
	30	81.70±10.7	91	77	3.24	80.79±10.9	91	75	3.29

Table 5.2. Performance metrics of dataset after MI fine tuning is applied with different number of trials

ME Fine tuning	N	Input time							
		3000 ms				2000 ms			
		Acc	BCI Control	Subjs imp	Mean imp	Acc	BCI Control	Subjs imp	Mean imp
1st process	0	80.59±10.4	88	-	-	80.11±10.97	84	-	-
	15	80.70±10.9	87	69	2.59	79.93±10.6	87	68	2.23
	21	80.82±10.3	91	72	2.42	79.62±10.7	86	56	2.98
	27	80.30±11.4	88	62	2.89	79.66±10.9	84	66	2.76
	33	79.92±10.8	85	61	2.56	79.88±10.9	90	64	2.90
	39	80.86±10.8	93	76	2.92	79.18±11.6	87	62	2.53
	45	80.28±10.6	88	65	2.74	79.71±10.9	90	58	3.18
Both processes	15	80.67±10.7	88	67	2.99	80.13±10.7	90	64	3.26
	21	80.87±11.1	89	69	3.02	79.99±10.7	91	68	2.81
	27	80.79±10.5	92	72	2.65	80.22±10.8	87	66	3.27
	33	80.63±11.1	86	63	3.23	80.30±11.6	87	70	3.24
	39	80.84±11.1	93	64	3.24	80.12±11.1	87	66	3.31
	45	80.62±10.6	88	66	3.18	80.27±10.7	85	64	3.35

Table 5.3. Performance metrics of dataset after ME fine tuning is applied with different number of trials

6. DISCUSSION

6.1. Interpretation of results

6.1.1. Characterization

The RP results, depicted in Figure 5.1 and Figure 5.2, show significant lateralization patterns during ME tasks, with decreased power in the contralateral hemisphere. During a goal-directed hand movement, a localized reduction in mu power is observed over the primary motor cortex of the opposite hemisphere. Concurrently, the ipsilateral hemisphere shows a broader increase in mu power over the central areas. This increase is interpreted as active inhibition of ipsilateral motor areas, which are associated with but not directly involved in the task, to prevent mirrored or associated movements (Duque *et al.*, 2007; Nam *et al.*, 2011). Such inhibition is thought to enhance the activity in specific cortical areas that control the precise goal-directed movement (Klimesch *et al.*, 2007; Suffczynski *et al.*, 2001).

This lateralization is indicative of focused cortical activity in the regions responsible for motor control. During MI tasks, although the RP changes are less pronounced, a similar pattern of lateralization is observed, suggesting that motor imagery engages the same cortical areas, albeit to a lesser degree.

Additionally, it is interesting to observe the behaviour in Beta1 RP in MI. Even though a reduction in power can be appreciated, it is not as significant as the reduction observed in ME in comparison with the behaviour found in Mu RP. This could be explained by the characteristic beta rebound (Jurkiewicz *et al.*, 2006). As seen on the ERD/ERS, beta1 ERD fades befores the end of the task during MI, followed up with the beta rebound ERS, implying an increase in power.

The interpretation of ERD is grounded in the understanding that a desynchronized EEG represents an activated state of cortical neurons. Specifically, a desynchronized occipital alpha rhythm is linked to visual information processing, while a desynchronized Rolandic mu rhythm is associated with motor behavior and sensorimotor activation (Pfurtscheller, 2001).

The stronger contralateral ERD in the right task, and the abscence of a strong contralateral dominance in the left hand during MI, aligns with the investigation of Pfurtscheller *et al.* (2006) who analyzed four different MI tasks, and Yi *et al.* (2013) who aimed to explore the differences in EEG patterns between simple limb motor imagery and compound limb motor imagery, and to examine the distinctiveness of various mental tasks. These studies revealed a similar ERD pattern and spatial distribution during left hand motor imagery, behaviour likely attributed to the participants' right-handedness.

The comparative analysis between ME and MI tasks reveals that while both tasks elicit ERD/ERS patterns in alpha, mu, and beta bands, the magnitude and spatial distribution of these patterns differ. ME consistently produces stronger and more widespread desynchronization compared to MI. This result may be expected given the physical execution of movements involves more extensive neural resources. The mu band in Figure 5.18, Figure 5.19, however, shows remarkable similarity between the two tasks, strengthening the idea that mu rhythm is a reliable marker for both executed and imagined movements. This similarity is critical for the application of our model.

Additionally, as it can be seen on figures present in subsection 5.1.2, ERD patterns seem to fade away during the MI paradigm. As sustaining attention on an imagined task can be cognitively demanding, the length of the task (4000ms) could imply that subjects may experience cognitive fatigue or attentional drift over time. The initial cognitive effort to initiate the imagery induces ERD, but the engagement fades as the task progress. Another explanation could be the fact that MI lacks actual sensory feedback, which means the neural circuits involved in processing this feedback are not engaged. The absence of this continuous reinforcement may result in a shorter ERD duration, as the brain does not receive the usual sensory inputs to sustain the desynchronization.

Our study results extend the work of Pfurtscheller *et al.* (2005), where they examined short-lasting beta bursts triggered after imagining hand, foot, or tongue movements in nine able-bodied participants. Using time–frequency maps to evaluate spatiotemporal ERD/ERS patterns, they found that after imagining foot movements, seven participants exhibited a significant short-lasting increase in beta power in the 23–29 Hz range, peaking at the vertex. In contrast, only two participants displayed a clear beta rebound at the vertex after imagining hand movements, though five showed significant beta rebound contralaterally. Additionally, in Pfurtscheller and Silva (1999), during continuous execution of movement in a few seconds, they also demonstrated a contralateral desynchronization of the mu rhythm and a localized synchronization of the alpha rhythm in the occipital region.

Our results reveal the presence of these patterns in our MI analysis. The induced beta oscillations, specially beta2, and the enhanced alpha activity are present around 2 seconds after onset, while the Rolandic mu rhythm remains in a desynchronized state with low amplitude. Moreover, in addition to Pfurtscheller *et al.* (2005), our averaged results across 109 subjects confirm the existence of a beta2 rebound starting from the vertex and spreading over time to the rest of the scalp in.

In summary, the results of our characterisation seem to align with previous studies regarding the behaviour of ME and MI. The inclusion of the mu band allowed us to better observe the similarities between ME and MI, specially above the sensorimotor areas. Furthermore, our temporal analysis with a MI task of longer duration (4000ms) compared to other studies such as Lubbe *et al.* (2021), allowed us to observe how the ERD pattern dissipates before the end of the task, while also showing the beta, mu and alpha acitivy characteristic of ME, providing valuable knowledge regarding the cognitive effort required for the imagination of movements.

6.1.2. Classification

ME and MI characterization results from the previous subsection, suggest that the neural representations of imagined movements closely mirror those of actual movements. This consistency is crucial as it implies that a model trained on ME data, which typically provides stronger and more reliable signals, can be effectively tested and utilized on MI data. This hypothesis is proved by the results from Table 5.1. Furthermore, since the ERD during MI . It was interesting to test our model with different input times.

First, the preference for longer input times (3000 ms) should be considered in the design of BCI protocols to maximize accuracy and reliability. However, the trade-off between longer input times and user comfort or system responsiveness must be balanced carefully, as the results obtained from 2000ms are also highly satisfactory. While 3000 ms input times provide better data for classification, they might lead to increased user fatigue or slower system responses.

Additionally, the overall small improvement obtained ($\approx$3%) through ME and MI fine tuning techniques performed on the pre-trained model proves the network has learned robust patterns from the ME data that are resistant to inter-subject variability, which is very positive and eliminates the need for calibration. The fact that the highest results for fine-tuning were achieved with fewer trials than the total available can be attributed to the inherent randomness in data. Even though the improvement in accuracy is relatively modest, it is possible that the selected subsets of trials provided a more favorable or informative representation for fine-tuning. On a more particular note, it can also be observed that the calibration has been very beneficial for certain specific subjects who previously could not achieve minimal control, possibly due to having very characteristic signal patterns. This calibration would be of great utility in the creation of subject specific models in certain cases.

Regarding both protocols, the superiority of MI fine-tuning over ME fine-tuning suggests that BCI systems should prioritize MI-based calibration processes. This is particularly important because MI tasks are less physically demanding than ME tasks, making them more suitable for users with physical impairments. Implementing MI fine-tuning can increase user experience by reducing the physical strain associated with prolonged ME tasks. However, due to the scarce number of trials available, this fine tuning process could be biased by the trials selected. For better generalization and interpretability, it would be appropriate to test both of the approaches in a larger dataset. Furthermore, in a key context to this study for neurorehabilitation contexts, the ME fine tuning approach cannot be performed as subjects would not be able to properly provide effective ME signals.

In Table 6.1 we can observe the results of our model (highlighted) when compared with current state of the art MI classification models.

Shuqfa *et al.* (2023) evaluated the performance of a novel Riemannian geometry decoding algorithm. The study applies various Riemannian geometry decoding algorithms using four adaptation strategies—baseline, rebias, supervised, and unsupervised—in both ME and MI scenarios with 64 electrodes to the same database used in our study, but removing 6 subjects. All

Architecture	Database	Learning	Transfer Learning	Test Acc	BCI Control
ShallowConvNet	Physionet	2 class MI	Cross-subject	82.0	92/109
DeepConvNet	Physionet	2 class MI	Cross-subject	82.8	95/109
EEGNet	Physionet	2 class MI	Cross-subject	81.6	92/109
EEG-Inception	Physionet	2 class MI	Cross-subject	82.7	92/109
EEGSym	Physionet	2 class MI	Cross-subject	84.5	99/109
RGA [a]	Physionet (103subj)	4 class MI	-	76.4	-
Mod ShallowConvNet	High Gamma	2 class ME	Cross-database, Cross-task	OpenBMI: 80.0 GIST: 72.7	-
CVNet	Dataset I	4 class ME	Cross-task	83	-
	Dataset I	4 class ME, MI	Mixed task, Cross-database	BNCI 2020: 69	-
EEG-Inception	**Physionet**	**2 class ME**	**Cross-Subject, Cross-task**	**81.94**	**94/109**

Table 6.1. Comparison of MI classification models

[a]RGA: Riemannian Geometry Algorithms

remaining models using the same Physionet database as our study, made use of only 8 electrodes, all 109 subjects and an input of 3000ms. Miao *et al.* (2023) trained the Modified ShallowConvNet model with ME inputs of 4000ms from 14 subjects with 20 channels, while testing on MI data from OpenBMI (54 subjects) for 3000ms, and on GIST (52 subjects) for 4000ms. Furthermore, D.-Y. Lee *et al.* (2022) used a proprietary database for training composed of 10 subjects performing upper limb ME for 4000ms with 27 channels, which was tested in those same subjects while performing MI, and in the public dataset BNCI Horizon 2020 containing 15 subjects. The data present in all the above studies were obtained from healthy subjects.

When observed against the other studies, it can be seen how our model presents competitive accuracies even when compared to intra-task MI classification models. Our model presents the most complete results, allowing for a cross task and cross subject binary classification trained in the largest dataset present, (109 subjects compared to only 14 or 10 in other studies), while allowing for BCI control in over 85% of subjects available. Additionally, we also tested our model in a XAI optimized 8 channel configuration through Shaply values from Pérez-Velasco *et al.* (2024), obtaining an accuracy of 79.7%. This significant reduction of channels used while being able to maintain accuracy could be of great use for clinical practice, allowing for easier use and quicker setup at the cost of a slight reduction in accuracy.

Our cross task transfer learning approach from ME to MI allows for a much easier calibration obtained from healthy individuals. Since this approach does not necessitate learning the imagination task, it reduces the time and effort required for participants to become proficient, thereby streamlining the setup phase. Additionally, this method improves robustness due to the inherent variability in how individuals perform the imagination paradigm. Since each person imagines movements differently, directly transferring ME patterns, which are more consistent and less subjective, leads to more reliable and stable results across diverse subjects. More im-

portantly, this approach presents a crucial advantage for neurorehabilitation due to the use of real ME patterns in training. Real ME patterns are closer to the actual neural processes involved in physical movement, making the training more effective and relevant for rehabilitative purposes. This can result in better outcomes for patients undergoing neurorehabilitation, as the system can leverage the more precise and accurate representations of motor activities. However, a potential disadvantage could be the possible decreased accuracy when the ME and MI tasks are not perfectly aligned, as the nuances of imagined movements may not always be fully captured by patterns derived from actual execution. This misalignment might necessitate additional tuning and adjustment, slightly offsetting the initial ease of calibration.

As a result of our experiments, we have succesfully created a plug and play style BCI, that requires no calibration runs, which can be further subject personalized through two different fine tuning techniques.

6.2. Implications for stroke rehabilitation

Stroke survivors are likely to show variability in the accuracy and consistency of MI performance. Initial attempts at MI may result in low classification accuracy in BCI systems due to weak or inconsistent neural signals. Over time, with targeted training and rehabilitation, some improvement in accuracy and consistency can be expected, although it may not reach the levels of healthy individuals.

In this study, the models are initialized using recordings from healthy subjects, easier to obtain and less expensive, while also representing the goal behaviour that the patients want to achieve. This model, trained with ME from healthy subjects, allows people with brain damage to retrain their activity patterns to fall within normal parameters through NFB training that leverages residual brain plasticity mechanisms in the affected areas.

The availability of a plug and play BCI that needs no calibration runs is key to improve user's motivation when using MI-based BCIs for rehabilitation. Furthermore, since success is also a key motivator factor, if subjects can't achieve satisfactory results, the model can be fine-tuned through a few calibration runs, improving accuracy and thereby regaining motivation. The ability of our model to achieve subject-specific calibration opens up the possibility for highly personalized rehabilitation programs. Each patient's neural responses to MI can be unique, and our system's adaptability ensures that the rehabilitation exercises are tailored to individual needs, maximizing their effectiveness, resulting in optimized treatment plans and increased engagement. This fine-tuning process is required only when initially adapting the system to the intended MI application, or whenever there is a significant increase in the number of recorded trials beyond the initial fine-tuning dataset.

The implications of these findings extend to various domains within neurorehabilitation. The novel MI classification can facilitate more effective rehabilitation protocols, potentially accelerating motor recovery in stroke patients. The system's high accuracy and low calibration requirement make it a viable option for home-based rehabilitation, reducing the need for

frequent hospital visits.

6.3. Limitations of the study

During the development of the project, various limitations have been identified that have influenced the research process and the results obtained. Most of the limitations within the study come from the availability of public MI datasets. While manually inspecting the dataset, we found some inconsistencies between the information that was provided and the actual information of the dataset. For example, sampling frequency is stated to be 160Hz for all recordings, however it was found that a selected portion of recordings were sampled at 128Hz. Furthermore, we were only able to extract 3400ms of the 4000ms epochs due to data acquisition inconsistency.

Even though this dataset is one of the largest available, containing 109 subjects, the total number of trials and epochs may still be insufficient to capture the interindividual variability in brain activity. This limitation particularly restricts our ability to adequately evaluate the fine-tuning strategy with MI data. Furthermore, the dataset description and experimental paradigm lacked detailed information regarding the BCI literacy or training levels of the subjects, which means some participants might have performed the tasks poorly, potentially affecting the model's results.

Despite applying a thorough pre-processing pipeline, manual inspection revealed that some artifacts persisted in individual trials. It could be possible that this punctual artifacts have an effect in the characterization results. The characterization itself was based on averages across all 109 subjects. It is possible that the results might have overlooked individual variability and present loss of specificity. In terms of the neural network, systematic differences between PyTorch and Tensorflow may cause the models to obtain slightly different results. The selection of EEGInception over EEGSym was performed on a small test with the final decision being made in favor of EEGInception in order to decrease computational costs. However, it is possible that EEGSym network from Perez-Velasco *et al.* (2022) could obtain better results if used with a larger quantity of data and applying data augmentation techniques.

Finally, although the hypothesis presented in this study may be of great use for neurorehabilitation, it has not yet been tested on the target subjects. Additionally, the implementation of explainable AI could have provided a more comprehensive perspective on the underlying processes in both ME and MI tasks.

7. CONCLUSION

In this final chapter, the main contributions of this work to the field of MI and ME characterization and binary classification are summarized. Furthermore, the most relevant conclusions drawn from the obtained results are highlighted, and potential future research directions are proposed to continue this investigation.

7.1. Contributions

- **Time and spectral characterization of ME and MI paradigms**:
  - Findings regarding beta bursts, contralateral desynchronization, and localized synchronization confirm and extend previous research.
  - Similar lateralization patterns observed during MI tasks, though less pronounced than ME, indicating engagement of the same cortical areas.
  - Beta1 RP during MI shows a reduction in power, but less significant compared to ME, possibly due to characteristic beta rebound. Presence of beta2 rebound starting from the vertex and spreading over time.
  - Temporal analysis with a longer duration MI task shows ERD patterns dissipate before the end of the task. Sustained attention on an imagined task is cognitively demanding, leading to cognitive fatigue or attentional drift over time resulting in a shorter effective execution of the task.
- **Novel transfer learning strategy**:
  - Developed a cross task and cross subject transfer learning strategy for MI classification using ME data.
  - Demonstrated the feasibility of this approach in enabling BCI control without the need for extensive calibration runs.
- **Implications for neurorehabilitation**:
  - Provided a robust method for MI classification, which is critical for neurorehabilitation applications.
  - Showed potential to simplify and enhance rehabilitation protocols, making them more accessible to stroke patients and individuals with motor impairments.

7.2. Conclusions

- The analysis of ME and MI tasks reveals significant lateralization patterns, with increased power in the contralateral hemisphere during goal-directed hand movements. This indicates focused cortical activity in motor control regions. During MI tasks, although the changes are less pronounced, a similar pattern of lateralization is observed, suggesting that motor imagery engages the same cortical areas as actual movement, albeit to a lesser degree.

- The mu rhythm shows remarkable similarity between ME and MI tasks, highlighting its reliability as a marker for both executed and imagined movements. This finding is critical for the application of our BCI model, as it underscores the potential for using mu rhythm in neurorehabilitation and BCI applications.

- The comparative analysis indicates that ME tasks produce stronger and more widespread desynchronization compared to MI tasks. The beta rebound observed in MI tasks, especially in the Beta1 range, further elucidates the neural mechanisms underlying motor imagery and its potential application in BCI systems. The temporal analysis with a longer MI task (4000ms) provides valuable insights into the cognitive effort required for imagining movements. The observed fading of ERD patterns before the end of the task highlights the challenges of sustaining attention during prolonged MI tasks.

- Our cross-task and cross-subject model demonstrates competitive accuracy (81.94%) even when compared to intra-task MI classification models. The model also proves effective even with only 2 seconds of input, or only 8 channels. The cross-task transfer learning approach from ME to MI allows for a plug and play device with easier calibration, reducing the time and effort required for participants to become proficient in the imagination task. This approach enhances the robustness of the model and is particularly advantageous for neurorehabilitation.

- The plug-and-play nature of our BCI with a large dataset, which requires no calibration runs, is crucial for improving user motivation in MI-based rehabilitation. The ability to achieve subject-specific calibration through a few calibration runs enhances the accuracy and effectiveness of the system, providing highly personalized rehabilitation programs. This adaptability ensures that the rehabilitation exercises are tailored to individual needs, maximizing their effectiveness and resulting in optimized treatment plans and increased patient engagement.

- The high accuracy and low calibration requirement of our system make it a viable option for home-based rehabilitation, reducing the need for frequent hospital visits. This has significant implications for stroke survivors, who can benefit from targeted training and rehabilitation in the comfort of their own homes.

7.3. Future Research

Future research should focus on several key areas to address these limitations and improve the applicability of BCIs in neurorehabilitation:

- Dataset expansion and diversity: Exploring additional MI datasets such as the ones available in Gwon *et al.*, 2023 and applying data augmentation techniques could improve the model's robustness and generalizability. Incorporating datasets from stroke patients would provide a more comprehensive understanding of the neural mechanisms underlying motor control and recovery.
- Explainable AI techniques: Implementing explainable AI techniques, such as Shapley values, could enhance the interpretability of neural network models, offering deeper insights into their decision-making processes and facilitating clinical adoption.
- Optimization of EEG setup: Testing how the EEG setup can be optimized by reducing the number of channels without compromising data quality could make the system more user-friendly and practical for clinical use.
- Advanced model training: Further optimization of fine-tuning techniques with larger datasets and exploring alternative neural network architectures could lead to improved performance and reliability of BCI systems.
- Target testing: Validate the model with stroke patients and test its effectiveness in complementing traditional neurorehabilitation therapies.

Bibliography

Alzubaidi, Laith *et al.* (Mar. 2021). "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions". In: *Journal of Big Data 2021 8:1* 8 (1), pp. 1–74. DOI: 10.1186/S40537-021-00444-8. URL: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00444-8.

Anaya, Manuel A. and Meret Branscheidt (July 2019). "Neurorehabilitation After Stroke". In: *Stroke* 50 (7). DOI: 10.1161/STROKEAHA.118.023878.

Ang, Kai Keng and Cuntai Guan (Apr. 2017). "EEG-Based Strategies to Detect Motor Imagery for Control and Rehabilitation". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25 (4), pp. 392–401. DOI: 10.1109/TNSRE.2016.2646763.

Anilkumar, AC and CS Nayak (July 2023). "EEG Normal Waveforms". In: *StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing*. URL: https://www.ncbi.nlm.nih.gov/books/NBK539805/.

Ávila-Tomás, Jose Francisco, Miguel Angel Mayer-Pujadas, and Victor Julio Quesada-Varela (Jan. 2021). "La inteligencia artificial y sus aplicaciones en medicina II: importancia actual y aplicaciones prácticas". In: *Atención Primaria* 53 (1), pp. 81–88. DOI: 10.1016/J.APRIM.2020.04.014.

Bai, Zhongfei, Kenneth N.K. Fong, Jack Jiaqi Zhang, Josephine Chan, and K. H. Ting (Apr. 2020). "Immediate and long-term effects of BCI-based rehabilitation of the upper extremity after stroke: A systematic review and meta-analysis". In: *Journal of NeuroEngineering and Rehabilitation* 17 (1), pp. 1–20. DOI: 10.1186/S12984-020-00686-2/FIGURES/6. URL: https://link.springer.com/articles/10.1186/s12984-020-00686-2%20https://link.springer.com/article/10.1186/s12984-020-00686-2.

Baldwin, Carryl L. and B. N. Penaranda (Jan. 2012). "Adaptive training using an artificial neural network and EEG metrics for within- and cross-task workload classification". In: *NeuroImage* 59 (1), pp. 48–56. DOI: 10.1016/J.NEUROIMAGE.2011.07.047.

Batula, Alyssa M., Jesse A. Mark, Youngmoo E. Kim, and Hasan Ayaz (2017). "Comparison of Brain Activation during Motor Imagery and Motor Movement Using fNIRS". In: *Computational Intelligence and Neuroscience* 2017. DOI: 10.1155/2017/5491296. URL: /pmc/articles/PMC5435907/%20/pmc/articles/PMC5435907/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5435907/.

Biga, Lindsay M. *et al.* (Sept. 2019). *Sensory and Motor Pathways*. OpenStax/Oregon State University. Chap. 14.5.

Binnie, Colin D., Raymond Cooper, F. Mauguiere, John W. Osselton, Pamela F. Prior, and B. M. Tedman (2003). *Clinical Neurophysiology: EEG, Paediatric Neurophysiology, Special Techniques and Applications, Vol. 2, 1er Ed. of the Series (Handbook of Clinical Neurophysiology)*. Elsevier, p. 1028. URL: https://www.amazon.com/Clinical-Neurophysiology-Paediatric-Techniques-Applications/dp/0444512578.

Blankertz, Benjamin, Ryota Tomioka, Steven Lemm, Motoaki Kawanabe, and Klaus-Robert Müller (2008). "Optimizing Spatial Filters for Robust EEG Single-Trial Analysis". In: *IEEE SIGNAL PROCESSING MAGAZINE* XX.

Bonferroni, C.E. (1936). *Teoria statistica delle classi e calcolo delle probabilità*. Pubblicazioni del R. Istituto superiore di scienze economiche e commerciali di Firenze. Seeber. URL: https://books.google.es/books?id=3CY-HQAACAAJ.

Branco, Mariana P. *et al.* (Mar. 2021). "Brain-Computer Interfaces for Communication: Preferences of Individuals With Locked-in Syndrome". In: *Neurorehabilitation and neural repair* 35 (3), pp. 267–279. DOI: 10.1177/1545968321989331. URL: https://pubmed.ncbi.nlm.nih.gov/33530868/.

Buch, Ethan, Cornelia Weber, Leonardo G. Cohen, Christoph Braun, and Michael A. Dimya (2008). "Think to Move: a Neuromagnetic Brain-Computer Interface (BCI) System for Chronic Stroke". In: *Stroke* 39 (3), p. 910. DOI: 10.1161/STROKEAHA.107.505313. URL: /pmc/articles/PMC5494966/%20/pmc/articles/PMC5494966/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5494966/.

Cantillo-Negrete, Jessica, Ruben I. Carino-Escobar, Emmanuel Ortega-Robles, and Oscar Arias-Carrión (Dec. 2023). "A comprehensive guide to BCI-based stroke neurorehabilitation interventions". In: *MethodsX* 11, p. 102452. DOI: 10.1016/j.mex.2023.102452.

Chang, Won Hyuk and Yun-Hee Kim (2013). "Robot-assisted Therapy in Stroke Rehabilitation". In: *Journal of Stroke* 15 (3), p. 174. DOI: 10.5853/JOS.2013.15.3.174. URL: /pmc/articles/PMC3859002/%20/pmc/articles/PMC3859002/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3859002/.

Chen, Shugeng, Xiaokang Shu, Hewei Wang, Li Ding, Jianghong Fu, and Jie Jia (Nov. 2021). "The Differences Between Motor Attempt and Motor Imagery in Brain-Computer Interface Accuracy and Event-Related Desynchronization of Patients With Hemiplegia". In: *Frontiers in Neurorobotics* 15. DOI: 10.3389/fnbot.2021.706630.

Choi, Rene Y., Aaron S. Coyner, Jayashree Kalpathy-Cramer, Michael F. Chiang, and J. Peter Campbell (2020). "Introduction to machine learning, neural networks, and deep learning". In: *Translational Vision Science and Technology* 9 (2). DOI: 10.1167/TVST.9.2.14.

Chollet, François (2021). *Deep Learning With Python.* Simon and Schuster. Chap. 5.

Chollet, Francois (July 2017). "Xception: Deep Learning with Depthwise Separable Convolutions". In: IEEE, pp. 1800–1807. DOI: 10.1109/CVPR.2017.195.

Cisnal, Ana *et al.* (2022). "An Overview of M3Rob, a Robotic Platform for Neuromotor and Cognitive Rehabilitation Using Augmented Reality". In: URL: www.medusabci.com.

Coogan, Christopher G. and Bin He (Feb. 2018). "Brain-computer interface control in a virtual reality environment and applications for the internet of things". In: *IEEE access : practical innovations, open solutions* 6, p. 10840. DOI: 10.1109/ACCESS.2018.2809453. URL: /pmc/articles/PMC6157750/%20/pmc/articles/PMC6157750/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6157750/.

Daly, Janis J, Roger Cheng, Kenneth Hrovat, Jean M Rogers, Krisanne Litinas, and Mark E Dohring (2008). "Development and Testing of Non-Invasive BCI + FES/Robot Sys-tem For Use in Motor Re-Learning After Stroke". In: *Proc. Int. Funct. Electr. Stimul. Soc.* URL: http://www.ifess2008.de/NR/rdonlyres/31C92C70-AAB8-4A0EAF21-828F81251C8D/27648/IFESS_2008_final.pdf..

Daly, Janis J. and Jonathan R. Wolpaw (Nov. 2008). "Brain-computer interfaces in neurological rehabilitation". In: *The Lancet. Neurology* 7 (11), pp. 1032–1043. DOI: 10.1016/S1474-4422(08)70223-0. URL: https://pubmed.ncbi.nlm.nih.gov/18835541/.

Dekleva, Brian M. *et al.* (Jan. 2024). "Motor cortex retains and reorients neural dynamics during motor imagery". In: *Nature Human Behaviour* 8 (4), pp. 729–742. DOI: 10.1038/s41562-023-01804-5.

Dose, Hauke, Jakob S. Møller, Helle K. Iversen, and Sadasivan Puthusserypady (Dec. 2018). "An end-to-end deep learning approach to MI-EEG signal classification for BCIs". In: *Expert Systems with Applications* 114, pp. 532–542. DOI: 10.1016/J.ESWA.2018.08.031.

Duque, Julie *et al.* (Feb. 2007). "Intermanual Differences in Movement-related Interhemispheric Inhibition". In: *Journal of Cognitive Neuroscience* 19 (2), pp. 204–213. DOI: 10.1162/jocn.2007.19.2.204.

Enzinger, Christian *et al.* (Sept. 2008). "Brain motor system function in a patient with complete spinal cord injury following extensive brain-computer interface training". In: *Experimental brain research* 190 (2), pp. 215–223. DOI: 10.1007/S00221-008-1465-Y. URL: https://pubmed.ncbi.nlm.nih.gov/18592230/.

Faller, Josef, Reinhold Scherer, Ursula Costa, Eloy Opisso, Josep Medina, and Gernot R. Müller-Putz (July 2014). "A co-adaptive brain-computer interface for end users with severe motor impairment". In: *PloS one* 9 (7). DOI: 10.1371/JOURNAL.PONE.0101168. URL: https://pubmed.ncbi.nlm.nih.gov/25014055/.

Feigin, Valery L. *et al.* (Dec. 2018). "GLOBAL, REGIONAL, AND COUNTRY-SPECIFIC LIFETIME RISK OF STROKE, 1990–2016". In: *The New England Journal of Medicine* 379 (25), p. 2429. DOI: 10.1056/NEJMOA1804492. URL: /pmc/articles/PMC6247346/%20/pmc/articles/PMC6247346/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6247346/.

Franc, Salomé Le *et al.* (July 2022). "Toward an Adapted Neurofeedback for Post-stroke Motor Rehabilitation: State of the Art and Perspectives". In: *Frontiers in Human Neuroscience* 16. DOI: 10.3389/fnhum.2022.917909.

Gandhi, Dorcas B.C., Albert Sterba, Himani Khatter, and Jeyaraj D. Pandian (2020). "Mirror therapy in stroke rehabilitation: Current perspectives". In: *Therapeutics and Clinical Risk Management* 16, pp. 75–85. DOI: 10.2147/TCRM.S206883. URL: https://www.tandfonline.com/action/journalInformation?journalCode=dtcr20.

Gerloff, Christian, Jacob Richard, Jordan Hadley, Andrew E. Schulman, Manabu Honda, and Mark Hallett (Aug. 1998). "Functional coupling and regional activation of human cortical motor areas during simple, internally paced and externally paced finger movements." In: *Brain* 121 (8), pp. 1513–1531. DOI: 10.1093/BRAIN/121.8.1513. URL: https://dx.doi.org/10.1093/brain/121.8.1513.

Goldberger, A. L. *et al.* (2000). "PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals". In: *Circulation* 101 (23). DOI: 10.1161/01.CIR.101.23.E215. URL: https://pubmed.ncbi.nlm.nih.gov/10851218/.

Gomez-Pilar, J., R. Corralejo, L. F. Nicolas-Alonso, D. Alvarez, and R. Hornero (Nov. 2014). "Assessment of neurofeedback training by means of motor imagery based-BCI for cognitive rehabilitation". In: *Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual International Conference* 2014, pp. 3630–3633. DOI: 10.1109/EMBC.2014.6944409. URL: https://pubmed.ncbi.nlm.nih.gov/25570777/.

Gunduz, Muhammed Enes, Bilal Bucak, and Zafer Keser (Oct. 2023). "Advances in Stroke Neurorehabilitation". In: *Journal of Clinical Medicine* 12 (21), p. 6734. DOI: 10.3390/jcm12216734.

Gwon, Daeun, Kyungho Won, Minseok Song, Chang S. Nam, Sung Chan Jun, and Minkyu Ahn (Mar. 2023). "Review of public motor imagery and execution datasets in brain-computer interfaces". In: *Frontiers in Human Neuroscience* 17, p. 1134869. DOI: 10.3389/FNHUM.2023.1134869/BIBTEX.

Hamzelou, Jessica (2022). *A locked-in man has communicated in sentences by thought alone | MIT Technology Review*. URL: https://www.technologyreview.com/2022/03/22/1047664/locked-in-patient-bci-communicate-in-sentences/.

Hardwick, Robert M., Svenja Caspers, Simon B. Eickhoff, and Stephan P. Swinnen (Nov. 2018). "Neural correlates of action: Comparing meta-analyses of imagery, observation, and execution". In: *Neuroscience and biobehavioral reviews* 94, pp. 31–44. DOI: 10.1016/J.NEUBIOREV.2018.08.003. URL: https://pubmed.ncbi.nlm.nih.gov/30098990/.

Iftikhar, Memoona, Shoab Ahmad Khan, and Ali Hassan (July 2018). "A Survey of Deep Learning and Traditional Approaches for EEG Signal Processing and Classification". In: *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON 2018*, pp. 395–400. DOI: 10.1109/IEMCON.2018.8614893.

Iman, Mohammadreza, Hamid Reza Arabnia, and Khaled Rasheed (Mar. 2023). "A Review of Deep Transfer Learning and Recent Advancements". In: *Technologies 2023, Vol. 11, Page 40* 11 (2), p. 40. DOI: 10.3390/TECHNOLOGIES11020040. URL: https://www.mdpi.com/2227-7080/11/2/40/htm%20https://www.mdpi.com/2227-7080/11/2/40.

Indolia, Sakshi, Anil Kumar Goswami, S. P. Mishra, and Pooja Asopa (Jan. 2018). "Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach". In: *Procedia Computer Science* 132, pp. 679–688. DOI: 10.1016/J.PROCS.2018.05.069.

Jamil, Nuraini, Abdelkader Nasreddine Belkacem, Sofia Ouhbi, and Christoph Guger (2021). "Cognitive and affective brain-computer interfaces for improving learning strategies and enhancing student capabilities: A systematic literature review". In: *IEEE Access* 9, pp. 134122–134147. DOI: 10.1109/ACCESS.2021.3115263.

Jiang, Xiao, Gui Bin Bian, and Zean Tian (Mar. 2019). "Removal of Artifacts from EEG Signals: A Review". In: *Sensors (Basel, Switzerland)* 19 (5). DOI: 10.3390/S19050987. URL: /pmc/articles/PMC6427454/%20/pmc/articles/PMC6427454/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6427454/.

Jung, Yoonsuh and Jianhua Hu (Apr. 2015). "A K-fold Averaging Cross-validation Procedure". In: *Journal of nonparametric statistics* 27 (2), p. 167. DOI: 10.1080/10485252.2015.1010532. URL: /pmc/articles/PMC5019184/%20/pmc/articles/PMC5019184/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5019184/.

Jurkiewicz, Michael T., William C. Gaetz, Andreea C. Bostan, and Douglas Cheyne (Sept. 2006). "Post-movement beta rebound is generated in motor cortex: evidence from neuromagnetic recordings". In: *NeuroImage* 32 (3), pp. 1281–1289. DOI: 10.1016/J.NEUROIMAGE.2006.06.005. URL: https://pubmed.ncbi.nlm.nih.gov/16863693/.

Kamath, Cannannore Nidhi, Syed Saqib Bukhari, and Andreas Dengel (Aug. 2018). "Comparative study between traditional machine learning and deep learning approaches for text classification". In: *Proceedings of the ACM Symposium on Document Engineering 2018, DocEng 2018* 18. DOI: 10.1145/3209280.3209526. URL: https://dl.acm.org/doi/10.1145/3209280.3209526.

Kawala-Sterniuk, Aleksandra *et al.* (Jan. 2021). "Summary of over Fifty Years with Brain-Computer Interfaces—A Review". In: *Brain Sciences* 11 (1), pp. 1–41. DOI: 10.3390/BRAINSCI11010043. URL: /pmc/articles/PMC7824107/%20/pmc/articles/PMC7824107/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7824107/.

Keirn, Z.A. and J.I. Aunon (1990). "A new mode of communication between man and his surroundings". In: *IEEE Transactions on Biomedical Engineering* 37 (12), pp. 1209–1214. DOI: 10.1109/10.64464.

Kelley, Roger E. and Aimee P. Borazanci (Oct. 2009). "Stroke rehabilitation". In: *Neurological research* 31 (8), pp. 832–840. DOI: 10.1179/016164109X12445505689689. URL: https://pubmed.ncbi.nlm.nih.gov/19723452/.

Kevric, Jasmin and Abdulhamit Subasi (Jan. 2017). "Comparison of signal decomposition methods in classification of EEG signals for motor-imagery BCI system". In: *Biomedical Signal Processing and Control* 31, pp. 398–406. DOI: 10.1016/J.BSPC.2016.09.007.

Khan, Muhammad Ahmed, Rig Das, Helle K. Iversen, and Sadasivan Puthusserypady (Aug. 2020). "Review on motor imagery based BCI systems for upper limb post-stroke neurorehabilitation: From designing to application". In: *Computers in Biology and Medicine* 123, p. 103843. DOI: 10.1016/J.COMPBIOMED.2020.103843.

Klimesch, Wolfgang, Paul Sauseng, and Simon Hanslmayr (Jan. 2007). "EEG alpha oscillations: The inhibition–timing hypothesis". In: *Brain Research Reviews* 53 (1), pp. 63–88. DOI: 10.1016/J.BRAINRESREV.2006.06.003.

Kraeutner, Sarah, Alicia Gionfriddo, Timothy Bardouille, and Shaun Boe (Nov. 2014). "Motor imagery-based brain activity parallels that of motor execution: Evidence from magnetic source imaging of cortical oscillations". In: *Brain Research* 1588, pp. 81–91. DOI: 10.1016/j.brainres.2014.09.001.

Kübler, A. *et al.* (May 2005). "Patients with ALS can use sensorimotor rhythms to operate a brain-computer interface". In: *Neurology* 64 (10), pp. 1775–1777. DOI: 10.1212/01.WNL.0000158616.43002.6D. URL: https://pubmed.ncbi.nlm.nih.gov/15911809/.

Ladda, Aija Marie, Florent Lebon, and Martin Lotze (June 2021). "Using motor imagery practice for improving motor performance – A review". In: *Brain and Cognition* 150, p. 105705. DOI: 10.1016/j.bandc.2021.105705.

Laver, Kate E., Belinda Lange, Stacey George, Judith E. Deutsch, Gustavo Saposnik, and Maria Crotty (Apr. 2018). "Virtual Reality for Stroke Rehabilitation". In: *Stroke* 49 (4). DOI: 10.1161/STROKEAHA.117.020275. URL: https://www.ahajournals.org/doi/10.1161/STROKEAHA.117.020275.

Lawhern, Vernon J, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung, and Brent J Lance (Oct. 2018). "EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces". In: *Journal of Neural Engineering* 15 (5), p. 056013. DOI: 10.1088/1741-2552/aace8c.

Lee, Do Yeun, Ji Hoon Jeong, Kyung Hwan Shim, and Seong Whan Lee (May 2020). "Decoding Movement Imagination and Execution from Eeg Signals Using Bci-Transfer Learning Method Based on Relation Network". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* 2020-May, pp. 1354–1358. DOI: 10.1109/ICASSP40776.2020.9052997.

Lee, Min Ho *et al.* (May 2019). "EEG dataset and OpenBMI toolbox for three BCI paradigms: an investigation into BCI illiteracy". In: *GigaScience* 8 (5), pp. 1–16. DOI: 10.1093/GIGASCIENCE/GIZ002. URL: https://dx.doi.org/10.1093/gigascience/giz002.

Lee, Tih Shih *et al.* (Nov. 2013). "A brain-computer interface based cognitive training system for healthy elderly: a randomized control pilot study for usability and preliminary efficacy". In: *PloS one* 8 (11). DOI: 10.1371/JOURNAL.PONE.0079419. URL: https://pubmed.ncbi.nlm.nih.gov/24260218/.

Lee, Do-Yeun, Ji-Hoon Jeong, Byeong-Hoo Lee, and Seong-Whan Lee (2022). "Motor Imagery Classification Using Inter-Task Transfer Learning via a Channel-Wise Variational Autoencoder-Based Convolutional Neural Network". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 30, pp. 226–237. DOI: 10.1109/TNSRE.2022.3143836.

Lotte, F, M Congedo, A Lécuyer, F Lamarche, and B Arnaldi (June 2007). "A review of classification algorithms for EEG-based brain–computer interfaces". In: *Journal of Neural Engineering* 4 (2), R1–R13. DOI: 10.1088/1741-2560/4/2/R01.

Lubbe, Rob H J Van der, Jagna Sobierajewicz, Marijtje L A Jongsma, Willem B Verwey, and Anna Przekoracka-Krawczyk (2021). "Frontal brain areas are more involved during motor imagery than during motor execution/preparation of a response sequence". In: *International Journal of Psychophysiology* 164, pp. 71–86. DOI: https://doi.org/10.1016/j.ijpsycho.2021.02.020. URL: https://www.sciencedirect.com/science/article/pii/S0167876021000787.

Mahla, Ranjeet Singh (2016). "Stem Cells Applications in Regenerative Medicine and Disease Therapeutics". In: *International Journal of Cell Biology* 2016, pp. 1–24. DOI: 10.1155/2016/6940283.

Marcos-Martínez, Diego, Víctor Martínez-Cagigal, Eduardo Santamaría-Vázquez, Sergio Pérez-Velasco, and Roberto Hornero (Nov. 2021). "Neurofeedback Training Based on Motor Imagery Strategies Increases EEG Complexity in Elderly Population". In: *Entropy* 23 (12), p. 1574. DOI: 10.3390/e23121574.

Miao, Minmin, Zhong Yang, Hong Zeng, Wenbin Zhang, Baoguo Xu, and Wenjun Hu (Dec. 2023). "Explainable cross-task adaptive transfer learning for motor imagery EEG classification". In: *Journal of Neural Engineering* 20 (6), p. 066021. DOI: 10.1088/1741-2552/ad0c61.

Miao, Minmin *et al.* (Apr. 2019). "Deep learning for electroencephalogram (EEG) classification tasks: a review". In: *Journal of Neural Engineering* 16 (3), p. 031001. DOI: 10.1088/1741-2552/AB0AB5. URL: https://iopscience.iop.org/article/10.1088/1741-2552/ab0ab5%20https://iopscience.iop.org/article/10.1088/1741-2552/ab0ab5/meta.

Miller, Kai J., Gerwin Schalk, Eberhard E. Fetz, Marcel den Nijs, Jeffrey G. Ojemann, and Rajesh P. N. Rao (Mar. 2010). "Cortical activity during motor execution, motor imagery, and imagery-based online feedback". In: *Proceedings of the National Academy of Sciences* 107 (9), pp. 4430–4435. DOI: 10.1073/pnas.0913697107.

Murphy, Stephen JX and David J. Werring (Sept. 2020). "Stroke: causes and clinical features". In: *Medicine (Abingdon, England : UK Ed.)* 48 (9), p. 561. DOI: 10.1016/J.MPMED.2020.06.002. URL: /pmc/articles/PMC7409792/%20/pmc/articles/PMC7409792/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7409792/.

Nam, Chang S., Yongwoong Jeon, Young-Joo Kim, Insuk Lee, and Kyungkyu Park (Mar. 2011). "Movement imagery-related lateralization of event-related (de)synchronization (ERD/ERS): Motor-imagery duration effects". In: *Clinical Neurophysiology* 122 (3), pp. 567–577. DOI: 10.1016/j.clinph.2010.08.002.

Neuper, Christa, Reinhold Scherer, Selina Wriessnegger, and Gert Pfurtscheller (Feb. 2009). "Motor imagery and action observation: modulation of sensorimotor brain rhythms during mental control of a brain-computer interface". In: *Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology* 120 (2), pp. 239–247. DOI: 10.1016/J.CLINPH.2008.11.015. URL: https://pubmed.ncbi.nlm.nih.gov/19121977/.

Nicolas-Alonso, Luis F., Rebeca Corralejo, Javier Gomez-Pilar, Daniel Álvarez, and Roberto Hornero (July 2015). "Adaptive semi-supervised classification to reduce intersession non-stationarity in multiclass motor imagery-based brain–computer interfaces". In: *Neurocomputing* 159 (1), pp. 186–196. DOI: 10.1016/J.NEUCOM.2015.02.005.

NIH, NHLBI (2024). *Stroke - Symptoms*. URL: https://www.nhlbi.nih.gov/health/stroke/symptoms.

Novac, Ovidiu Constantin *et al.* (Nov. 2022). "Analysis of the Application Efficiency of TensorFlow and PyTorch in Convolutional Neural Network". In: *Sensors (Basel, Switzerland)* 22 (22). DOI: 10.3390/S22228872. URL: /pmc/articles/PMC9699128/%20/pmc/articles/PMC9699128/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9699128/.

Nunez, Paul L. and Ramesh Srinivasan (Jan. 2006). *Electric Fields of the Brain*. Oxford University Press. DOI: 10.1093/acprof:oso/9780195050387.001.0001.

Onishi, Akinari and Kiyohisa Natsume (Apr. 2014). "Overlapped partitioning for ensemble classifiers of P300-based brain-computer interfaces". In: *PLoS ONE* 9 (4). DOI: 10.1371/JOURNAL.PONE.0093045.

Padfield, Natasha, Jaime Zabalza, Huimin Zhao, Valentin Masero, and Jinchang Ren (Mar. 2019). "EEG-Based Brain-Computer Interfaces Using Motor-Imagery: Techniques and Challenges". In: *Sensors* 19 (6), p. 1423. DOI: 10.3390/s19061423.

Paszke, Adam *et al.* (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*.

Perez-Velasco, Sergio, Eduardo Santamaria-Vazquez, Victor Martinez-Cagigal, Diego Marcos-Martinez, and Roberto Hornero (2022). "EEGSym: Overcoming Inter-Subject Variability in Motor Imagery Based BCIs With Deep Learning". In: *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society* 30, pp. 1766–1775. DOI: 10.1109/TNSRE.2022.3186442. URL: https://pubmed.ncbi.nlm.nih.gov/35759578/.

Pérez-Velasco, Sergio, Diego Marcos-Martínez, Eduardo Santamaría-Vázquez, Víctor Martínez-Cagigal, Selene Moreno-Calderón, and Roberto Hornero (Apr. 2024). "Unraveling motor imagery brain patterns using explainable artificial intelligence based on Shapley values". In: *Computer Methods and Programs in Biomedicine* 246. DOI: 10.1016/j.cmpb.2024.108048.

Perna, Robert and Lindsey Harik (2020). "The role of rehabilitation psychology in stroke care described through case examples". In: *NeuroRehabilitation* 46 (2), pp. 195–204. DOI: 10.3233/NRE-192970.

Pfurtscheller, G. (May 2001). "Functional brain imaging based on ERD/ERS". In: *Vision Research* 41 (10-11), pp. 1257–1260. DOI: 10.1016/S0042-6989(00)00235-2.

Pfurtscheller, G., C. Brunner, A. Schlögl, and F. H. Lopes da Silva (May 2006). "Mu rhythm (de)synchronization and EEG single-trial classification of different motor imagery tasks". In: *NeuroImage* 31 (1), pp. 153–159. DOI: 10.1016/J.NEUROIMAGE.2005.12.003.

Pfurtscheller, G., C. Guger, G. Müller, G. Krausz, and C. Neuper (Oct. 2000). "Brain oscillations control hand orthosis in a tetraplegic". In: *Neuroscience Letters* 292 (3), pp. 211–214. DOI: 10.1016/S0304-3940(00)01471-3. URL: https://pubmed.ncbi.nlm.nih.gov/11018314/.

Pfurtscheller, G., C. Neuper, C. Brunner, and F. Lopes da Silva (Apr. 2005). "Beta rebound after different types of motor imagery in man". In: *Neuroscience Letters* 378 (3), pp. 156–159. DOI: 10.1016/j.neulet.2004.12.034.

Pfurtscheller, G. and F.H. Lopes da Silva (Nov. 1999). "Event-related EEG/MEG synchronization and desynchronization: basic principles". In: *Clinical Neurophysiology* 110 (11), pp. 1842–1857. DOI: 10.1016/S1388-2457(99)00141-8.

Pfurtscheller, G., A. Stancák, and C. Neuper (Apr. 1996). "Post-movement beta synchronization. A correlate of an idling motor area?" In: *Electroencephalography and Clinical Neurophysiology* 98 (4), pp. 281–293. DOI: 10.1016/0013-4694(95)00258-8.

Purves, David, George J. Augustine, David Fitzpatrick, *et al.* (2001). *Neuroscience*. Ed. by David Purves, George J. Augustine, David Fitzpatrick, *et al.* 2nd. Available from: https://www.ncbi.nlm.nih.gov/books/NBK10962/. Sunderland, MA: Sinauer Associates. Chap. The Primary Motor Cortex: Upper Motor Neurons That Initiate Complex Voluntary Movements. URL: https://www.ncbi.nlm.nih.gov/books/NBK10962/.

Raj, Bharath (2019). *A Simple Guide to the Versions of the Inception Network | by Bharath Raj | Towards Data Science*. URL: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202.

Ramadan, Rabie A. and Athanasios V. Vasilakos (Feb. 2017). "Brain computer interface: control signals review". In: *Neurocomputing* 223, pp. 26–44. DOI: 10.1016/J.NEUCOM.2016.10.024.

Ridderinkhof, K. Richard and Marcel Brass (Feb. 2015). "How Kinesthetic Motor Imagery works: A predictive-processing theory of visualization in sports and motor expertise". In: *Journal of Physiology-Paris* 109 (1-3), pp. 53–63. DOI: 10.1016/J.JPHYSPARIS.2015.02.003.

Saha, Simanto and Mathias Baumert (Jan. 2020). "Intra- and Inter-subject Variability in EEG-Based Sensorimotor Brain Computer Interface: A Review". In: *Frontiers in Computational Neuroscience* 13, p. 506286. DOI: 10.3389/FNCOM.2019.00087/BIBTEX. URL: www.frontiersin.org.

Santamaria-Vazquez, Eduardo, Victor Martinez-Cagigal, Fernando Vaquerizo-Villar, and Roberto Hornero (Dec. 2020). "EEG-Inception: A Novel Deep Convolutional Neural Network for Assistive ERP-Based Brain-Computer Interfaces". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 28 (12), pp. 2773–2782. DOI: 10.1109/TNSRE.2020.3048106.

Santamaría-Vázquez, Eduardo, Víctor Martínez-Cagigal, Javier Gomez-Pilar, and Roberto Hornero (2020). "Deep Learning Architecture Based on the Combination of Convolutional and Recurrent Layers for ERP-Based Brain-Computer Interfaces". In: *IFMBE Proceedings* 76, pp. 1844–1852. DOI: 10.1007/978-3-030-31635-8_224/TABLES/4. URL: https://link.springer.com/chapter/10.1007/978-3-030-31635-8_224.

Santamaría-Vázquez, Eduardo *et al.* (Mar. 2023). "MEDUSA©: A novel Python-based software ecosystem to accelerate brain-computer interface and cognitive neuroscience research". In: *Computer Methods and Programs in Biomedicine* 230, p. 107357. DOI: 10.1016/J.CMPB.2023.107357.

Schalk, G., D.J. McFarland, T. Hinterberger, N. Birbaumer, and J.R. Wolpaw (2004). "BCI2000: a general-purpose brain-computer interface (BCI) system". In: *IEEE Transactions on Biomedical Engineering* 51.6, pp. 1034–1043. DOI: 10.1109/TBME.2004.827072.

Schalk, Gerwin, Dennis J McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R Wolpaw (2022). *"EEG Motor Movement/Imagery Dataset"*. PhysioNet. DOI: https://doi.org/10.13026/C28G6P.

Schirrmeister, Robin Tibor *et al.* (Nov. 2017). "Deep learning with convolutional neural networks for EEG decoding and visualization". In: *Human Brain Mapping* 38 (11), pp. 5391–5420. DOI: 10.1002/hbm.23730.

Schlaug, Gottfried, Vijay Renga, and Dinesh Nair (Dec. 2008). "Transcranial Direct Current Stimulation in Stroke Recovery". In: *Archives of Neurology* 65 (12), pp. 1571–1576. DOI: 10.1001/ARCHNEUR.65.12.1571. URL: https://jamanetwork.com/journals/jamaneurology/fullarticle/1107499.

Sheng, Rongjun, Changchun Chen, Huan Chen, and Peipei Yu (2023). "Repetitive transcranial magnetic stimulation for stroke rehabilitation: insights into the molecular and cellular mechanisms of neuroinflammation". In: *Frontiers in Immunology* 14, p. 1197422. DOI: 10.3389/FIMMU.2023.1197422. URL: /pmc/articles/PMC10239808/%20/pmc/articles/PMC10239808/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10239808/.

Shih, Jerry J., Dean J. Krusienski, and Jonathan R. Wolpaw (2012). "Brain-Computer Interfaces in Medicine". In: *Mayo Clinic Proceedings* 87 (3), p. 268. DOI: 10.1016/J.MAYOCP.2011.12.008. URL: /pmc/articles/PMC3497935/%20/pmc/articles/PMC3497935/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3497935/.

Shuqfa, Zaid, Abdelkader Nasreddine Belkacem, and Abderrahmane Lakas (May 2023). "Decoding Multi-Class Motor Imagery and Motor Execution Tasks Using Riemannian Geometry Algorithms on Large EEG Datasets". In: *Sensors* 23 (11), p. 5051. DOI: 10.3390/s23115051.

Sibilano, Elena *et al.* (2024). "Brain–Computer Interfaces". In: *Neuromethods* 206, pp. 203–240. DOI: 10.1007/978-1-0716-3545-2_10.

Simon, Colin, David A. E. Bolton, Niamh C. Kennedy, Surjo R. Soekadar, and Kathy L. Ruddy (July 2021). "Challenges and Opportunities for the Future of Brain-Computer Interface in Neurorehabilitation". In: *Frontiers in Neuroscience* 15. DOI: 10.3389/fnins.2021.699428.

Sinha, Anita M., Veena A. Nair, and Vivek Prabhakaran (Sept. 2021). "Brain-Computer Interface Training With Functional Electrical Stimulation: Facilitating Changes in Interhemispheric Functional Connectivity and Motor Outcomes Post-stroke". In: *Frontiers in Neuroscience* 15. DOI: 10.3389/FNINS.2021.670953. URL: /pmc/articles/PMC8503522/%20/pmc/articles/PMC8503522/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8503522/.

Suffczynski, Piotr, Stiliyan Kalitzin, Gert Pfurtscheller, and F. H. Lopes Da Silva (Dec. 2001). "Computational model of thalamo-cortical networks: dynamical control of alpha rhythms in relation to focal attention". In: *International Journal of Psychophysiology* 43 (1), pp. 25–40. DOI: 10.1016/S0167-8760(01)00177-5.

Szegedy, Christian *et al.* (June 2015). "Going deeper with convolutions". In: IEEE, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

Unnithan, Ajaya Kumar A., Joe M. Das, and Parth Mehta (May 2023). "Hemorrhagic Stroke". In: *StatPearls*. URL: https://www.ncbi.nlm.nih.gov/books/NBK559173/%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4221651.

Vidal, J. J. (June 1973). "Toward direct brain-computer communication." In: *Annual review of biophysics and bioengineering* 2 (Volume 2, 1973), pp. 157–180. DOI: 10.1146/ANNUREV.BB.02.060173.001105/CITE/REFWORKS.

Walczak, Steven and Narciso Cerpa (2003). "Artificial Neural Networks". In: *Encyclopedia of Physical Science and Technology*, pp. 631–645. DOI: 10.1016/B0-12-227410-5/00837-1. URL: https://linkinghub.elsevier.com/retrieve/pii/B0122274105008371.

Wan, Zitong, Rui Yang, Mengjie Huang, Nianyin Zeng, and Xiaohui Liu (Jan. 2021). "A review on transfer learning in EEG signal analysis". In: *Neurocomputing* 421, pp. 1–14. DOI: 10.1016/J.NEUCOM.2020.09.017.

Wang, Xianheng, Veronica Liesaputra, Zhaobin Liu, Yi Wang, and Zhiyi Huang (Jan. 2024). "An in-depth survey on Deep Learning-based Motor Imagery Electroencephalogram (EEG) classification". In: *Artificial Intelligence in Medicine* 147, p. 102738. DOI: 10.1016/j.artmed.2023.102738.

Weiss, Karl, Taghi M. Khoshgoftaar, and DingDing Wang (Dec. 2016). "A survey of transfer learning". In: *Journal of Big Data* 3 (1), p. 9. DOI: 10.1186/s40537-016-0043-6.

Wilcoxon, Frank (1945). "Individual Comparisons by Ranking Methods". In: *Biometrics Bulletin* 1 (6), pp. 80–83. URL: http://www.jstor.org/about/terms.html..

Wolpaw, Jonathan and Elizabeth Winter Wolpaw (Jan. 2012). *Brain–Computer InterfacesPrinciples and Practice*. Oxford University Press. DOI: 10.1093/acprof:oso/9780195388855.001.0001.

Wolpaw, Jonathan R. and Dennis J. McFarland (Dec. 2004). "Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans". In: *Proceedings of the National Academy of Sciences of the United States of America* 101 (51), pp. 17849–17854. DOI: 10.1073/PNAS.0403504101. URL: https://pubmed.ncbi.nlm.nih.gov/15585584/.

Wolpaw, Jonathan R., Dennis J. McFarland, Gregory W. Neat, and Catherine A. Forneris (1991). "An EEG-based brain-computer interface for cursor control". In: *Electroencephalography and clinical neurophysiology* 78 (3), pp. 252–259. DOI: 10.1016/0013-4694(91)90040-B. URL: https://pubmed.ncbi.nlm.nih.gov/1707798/.

Yadav, Drishti, Shilpee Yadav, and Karan Veer (Dec. 2020). "A comprehensive assessment of Brain Computer Interfaces: Recent trends and challenges". In: *Journal of Neuroscience Methods* 346, p. 108918. DOI: 10.1016/J.JNEUMETH.2020.108918.

Yang, Yu Jin, Eun Jeong Jeon, June Sic Kim, and Chun Kee Chung (Feb. 2021). "Characterization of kinesthetic motor imagery compared with visual motor imageries". In: *Scientific Reports 2021 11:1* 11 (1), pp. 1–11. DOI: 10.1038/s41598-021-82241-0. URL: https://www.nature.com/articles/s41598-021-82241-0.

Yi, Weibo, Shuang Qiu, Hongzhi Qi, Lixin Zhang, Baikun Wan, and Dong Ming (Oct. 2013). "EEG feature comparison and classification of simple and compound limb motor imagery". In: *Journal of NeuroEngineering and Rehabilitation* 10 (1), pp. 1–12. DOI: 10.1186/1743-0003-10-106/FIGURES/10. URL: https://link.springer.com/articles/10.1186/1743-0003-10-106%20https://link.springer.com/article/10.1186/1743-0003-10-106.

Yu, Qiu-Hua, Amy S.N. Fu, Adeline Kho, Jie Li, Xiao-Hua Sun, and Chetwyn C.H. Chan (June 2016). "Imagery perspective among young athletes: Differentiation between external and internal visual imagery". In: *Journal of Sport and Health Science* 5 (2), pp. 211–218. DOI: 10.1016/j.jshs.2014.12.008.

Yuan, Han and Bin He (May 2014). "Brain–Computer Interfaces Using Sensorimotor Rhythms: Current State and Future Perspectives". In: *IEEE Transactions on Biomedical Engineering* 61 (5), pp. 1425–1435. DOI: 10.1109/TBME.2014.2312397.

Zhang, Kaishuo, Neethu Robinson, Seong Whan Lee, and Cuntai Guan (Apr. 2021). "Adaptive transfer learning for EEG motor imagery classification with deep Convolutional Neural Network". In: *Neural Networks* 136, pp. 1–10. DOI: 10.1016/J.NEUNET.2020.12.013.

Zheng, Minmin and Yiwen Lin (Mar. 2024). "A deep transfer learning network with two classifiers based on sample selection for motor imagery brain-computer interface". In: *Biomedical Signal Processing and Control* 89, p. 105786. DOI: 10.1016/J.BSPC.2023.105786.

Zich, Catharina, Stefan Debener, Cornelia Kranczioch, Martin G. Bleichner, Ingmar Gutberlet, and Maarten De Vos (July 2015). "Real-time EEG feedback during simultaneous EEG–fMRI identifies the cortical signature of motor imagery". In: *NeuroImage* 114, pp. 438–447. DOI: 10.1016/j.neuroimage.2015.04.020.

Zippo, Antonio (Apr. 2011). “Neuronal Ensemble Modeling and Analysis with Variable Order Markov Models Contents”. PhD thesis.

Annex

EEGInception

```
class EEGInception(nn.Module):
    def __init__(self, input_time=in_time*1000, fs=128, n_cha=64,
                 filters_per_branch=8, scales_time=(500, 250, 125),
                 dropout_rate=0.25, activation='ELU',
                 n_classes=2, learning_rate=0.001):

        """Torch implementation of EEG Inception.

            This model was initially designed for MI decodification of either
            left/right hand.
            Hyperparameters and architectural choices are explained in the
            original article.

            Parameters
            ----------
            input_time : int
                EEG epoch time in milliseconds.
            fs : int
                Sample rate of the EEG.
            ncha :
                Number of input channels.
            filters_per_branch : int
                Number of filters in each Inception branch. The number should be
                multiplies of 8.
            scales_time : list
                Temporal scale of the temporal convolutions on first
                Inception module. This parameter determines the kernel
                sizes of the filters.
            dropout_rate : float
                Dropout rate
            activation : str
                Activation
            n_classes : int
                Number of output classes
            learning_rate : float
                Learning rate
```

```
    Returns
    -------
    model : nn.Model
        Torch model already compiled and ready to work
    """

super(EEGInception, self).__init__()

# Parameters
self.input_time = input_time
self.fs = fs
self.n_cha = n_cha
self.filters_per_branch = filters_per_branch
self.scales_time = scales_time
self.dropout_rate = dropout_rate
self.Activation = getattr(nn, activation)()
self.n_classes = n_classes
self.learning_rate = learning_rate

# Useful variables
self.input_samples = int(input_time * fs / 1000)
self.scales_samples = [int(s * fs / 1000) for s in scales_time]

# BLOCK 1: SINGLE CHANNEL ANALYSIS

for i in range(len(self.scales_samples)):
    conv_layer = nn.Conv2d(in_channels=1,
                           out_channels=self.filters_per_branch,
                           kernel_size=(self.scales_samples[i], 1),
                           stride=1,
                           padding='same')
    nn.init.kaiming_normal_(conv_layer.weight)
    nn.init.zeros_(conv_layer.bias)
    batch = nn.BatchNorm2d(self.filters_per_branch, momentum=0.01)
    setattr(self, f'Conv1_{i}', conv_layer)
    setattr(self, f'Batch1_{i}', batch)
self.Batch = nn.BatchNorm2d(self.filters_per_branch, momentum=0.01)
self.Pool = nn.AvgPool2d((2, 1))

# BLOCK 2: SPATIAL FILTERING
# 2 in out channels is for Depth multiplier
self.Depth = nn.Conv2d(in_channels=len(self.scales_samples)*
                       self.filters_per_branch,
                       out_channels=2*len(self.scales_samples)*
```

```
                        self.filters_per_branch,
                        kernel_size=(1, self.n_cha),
                        groups=len(self.scales_samples)*
                        self.filters_per_branch,
                        padding='valid', bias=False,
                        stride=(1, 1))
self.Depth.weight = nn.Parameter(torch.clamp(self.Depth.weight,
min=-1., max=1.))
self.Batch_depth=nn.BatchNorm2d(2*len(self.scales_samples)*
                                self.filters_per_branch,
                                momentum=0.01)

# BLOCK 3: MULTI CHANNEL ANALYSIS
for i in range(len(self.scales_samples)):
    conv_layer = nn.Conv2d(in_channels=self.Depth.out_channels,
                           out_channels=self.filters_per_branch,
                           kernel_size=(int(self.scales_samples[i]
                           / 4), 1),
                           stride=1,
                           padding='same', bias=False)
    nn.init.kaiming_normal_(conv_layer.weight)
    setattr(self, f'Conv3_{i}', conv_layer)
    batch = nn.BatchNorm2d(conv_layer.out_channels, momentum=0.01)
    setattr(self, f'Batch3_{i}', batch)

self.Batch3 = nn.BatchNorm2d(self.Conv3_0.out_channels, momentum=0.01)

# BLOCK 4 OUTPUT BLOCK
self.Conv4_1=nn.Conv2d(in_channels=len(self.scales_samples)*
                       self.filters_per_branch,
                       out_channels=int(self.filters_per_branch
                       *len(self.scales_samples) / 2),
                       kernel_size=(8, 1), stride=1,
                       padding='same', bias=False)
self.Conv4_1.weight = nn.init.kaiming_normal_(self.Conv4_1.weight)
self.Batch4_1 = nn.BatchNorm2d(self.Conv4_1.out_channels, momentum=0.01)
self.Conv4_2=nn.Conv2d(in_channels=int(len(self.scales_samples)*
                       self.filters_per_branch/2),
                       out_channels=int(self.filters_per_branch *
                       len(self.scales_samples) / 4),
                       kernel_size=(4, 1), stride=1,
                       padding='same', bias=False)
self.Conv4_2.weight = nn.init.kaiming_normal_(self.Conv4_2.weight)
self.Batch4_2 = nn.BatchNorm2d(self.Conv4_2.out_channels, momentum=0.01)
self.Flatten = nn.Flatten()
```

```
        self.Dense = nn.Linear(in_features=int(6*self.input_samples/32),
        out_features=self.n_classes)
        self.soft = nn.Softmax(dim=1)

    def forward(self, x):
        # Block 1 forward
        b1_units = list()
        for i in range(len(self.scales_samples)):
            conv = getattr(self, f'Conv1_{i}')
            batch = getattr(self, f'Batch1_{i}')
            unit = F.dropout2d(self.Activation(batch(conv(x))),
            self.dropout_rate)
            b1_units.append(unit)
        # Concatenate + Average Pooling
        b1_out = torch.cat(b1_units, dim=1)
        b1_out = self.Pool(b1_out)
        #print('Block1: ', b1_out.shape)

        # BLOCK 2 forward
        b2_unit=F.dropout2d(self.Activation(self.Batch_depth(self.Depth(b1_out)
        b2_out = self.Pool(b2_unit)
        #print('Block2: ', b2_out.shape)
        # BLOCK 3 forward
        b3_units = list()
        for i in range(len(self.scales_samples)):
            conv = getattr(self, f'Conv3_{i}')
            batch = getattr(self, f'Batch3_{i}')
            unit = F.dropout2d(self.Activation(batch(conv(b2_out))),
            self.dropout_rate)
            b3_units.append(unit)
        # Concatenate + Average Pooling
        b3_out = torch.cat(b3_units, dim=1)
        b3_out = self.Pool(b3_out)
        #print('Block3: ', b3_out.shape)

        # Block 4 output block
        b4_u1=F.dropout2d(self.Activation(self.Batch4_1(self.Conv4_1(b3_out))),
                          self.dropout_rate)
        b4_u1 = self.Pool(b4_u1)
        #print('Block4_1: ', b4_u1.shape)
        b4_u2=F.dropout2d(self.Activation(self.Batch4_2(self.Conv4_2(b4_u1))),
                          self.dropout_rate)
        b4_u2 = self.Pool(b4_u2)
       # print('Block4_2: ', b4_u2.shape)
        # output layer
```

```
        output_layer = self.Flatten(b4_u2)
        #print('Flatten: ', output_layer.shape)
        output_layer = self.Dense(output_layer)
        #print('Dense: ', output_layer.shape)
        output_layer = self.soft(output_layer)
        return output_layer
```

EEGSym (8 channels symmetric hardcoded)

```
class EEGSym(nn.Module):
def __init__(self, input_time=3000, fs=128, ncha=8,
             filters_per_branch=24, scales_time=(500, 250, 125),
             dropout_rate=0.25, activation='ELU', n_classes=2,
             learning_rate=0.001, ch_lateral=3,
             spatial_resnet_repetitions=1, residual=True,
             symmetric=True):
    """Torch implementation of EEG Inception.

    This model was initially designed for MI decodification of either
    left/right hand.
    Hyperparameters and architectural choices are explained in the
    original article.

    Parameters
    ----------
    input_time : int
        EEG epoch time in milliseconds.
    fs : int
        Sample rate of the EEG.
    ncha :
        Number of input channels.
    filters_per_branch : int
        Number of filters in each Inception branch. The number should be
        multiplies of 8.
    scales_time : list
        Temporal scale of the temporal convolutions on first
        Inception module. This parameter determines the kernel
        sizes of the filters.
    dropout_rate : float
        Dropout rate
    activation : str
        Activation
    n_classes : int
        Number of output classes
```

```
learning_rate : float
    Learning rate
ch_lateral : int
    Number of channels that are attributed to one hemisphere
    of the head.
spatial_resnet_repetitions: int
    Number of repetitions of the operations of spatial analysis
    at each step of the spatiotemporal analysis. In the original
    publication this value was set to 1 and not tested its variations.
residual : Bool
    If the residual operations are present in EEGSym architecture.
symmetric : Bool
    If the architecture considers the parameter ch_lateral to create
    two symmetric inputs of the electrodes.

Returns
-------
model : nn.Model
    Torch model already compiled and ready to work
"""
super(EEGSym, self).__init__()

# Parameters
self.input_time = input_time
self.fs = fs
self.n_cha = ncha
self.filters_per_branch = filters_per_branch
self.scales_time = scales_time
self.dropout_rate = dropout_rate
self.Activation = getattr(nn, activation)()
self.n_classes = n_classes
self.learning_rate = learning_rate
self.ch_lateral= ch_lateral
self.spatial_resnet_repetitions = spatial_resnet_repetitions
self.residual = residual
self.symmetric = symmetric

# Useful variables
self.input_samples = int(input_time * fs / 1000)
self.scales_samples = [int(s * fs / 1000) for s in scales_time]

# ====================== TEMPOSPATIAL ANALYSIS ====================== #
# ========================== Inception (x2) ========================== #
# Inception block 1
unit_conv_t = nn.ModuleList()
```

```
unit_batchconv_t = nn.ModuleList()
for i in range(len(self.scales_samples)):
    conv_layer=nn.Conv3d(in_channels=1,
                         out_channels=self.filters_per_branch,
                         kernel_size=(1, self.scales_samples[i], 1),
                         stride=(1, 1, 1), padding='same',
                         bias=True)
    # print(conv_layer.shape)
    conv_layer.weight = nn.init.kaiming_normal_(conv_layer.weight)
    conv_layer.bias = nn.init.zeros_(conv_layer.bias)
    unit_conv_t.append(conv_layer)
    unit_batchconv_t.append(nn.BatchNorm3d(self.filters_per_branch,
                            momentum=0.01))

self.Inception1_1 = unit_conv_t
self.Batch_Inception = unit_batchconv_t
self.Pool_Inception = nn.AvgPool3d((1, 2, 1))

#Grouped (Depthwise) Convolution
# TODO change kernel size based on ncha

self.Depth = nn.Conv3d(in_channels=len(self.scales_samples)*
                       self.filters_per_branch,
                       out_channels=len(self.scales_samples)*
                       self.filters_per_branch, kernel_size=(1, 1, 5),
                       groups=len(self.scales_samples)*
                       self.filters_per_branch, padding='valid',
                       bias=False)
self.batch_depth_inc = nn.BatchNorm3d(self.Depth.out_channels, momentum=0.01)

# Inception block 2
unit_conv_t_2 = nn.ModuleList()
unit_batchconv_t_2 = nn.ModuleList()
for i in range(len(self.scales_samples)):
    conv_layer = nn.Conv3d(in_channels=self.Depth.out_channels,
                           out_channels=self.filters_per_branch,
                           kernel_size=(1, int(self.scales_samples[i]
                           / 4), 1),
                           stride=(1, 1, 1),
                           padding='same',
                           bias=True)
    # print(conv_layer.shape)
    conv_layer.weight = nn.init.kaiming_normal_(conv_layer.weight)
    conv_layer.bias = nn.init.zeros_(conv_layer.bias)
    unit_conv_t_2.append(conv_layer)
```

```
    batch = nn.BatchNorm3d(filters_per_branch, momentum=0.01)
    unit_batchconv_t_2.append(batch)

self.Inception1_2 = unit_conv_t_2
self.Batch_Inception_2 = unit_batchconv_t_2

self.Depth_2 = nn.Conv3d(in_channels=len(self.scales_samples)*
                         self.filters_per_branch,
                         out_channels=len(self.scales_samples)*
                         self.filters_per_branch, kernel_size=(1, 1, 5),
                         groups=len(self.scales_samples)*
                         self.filters_per_branch, padding='valid',
                         bias=False)
self.batch_depth_inc_2 = nn.BatchNorm3d(self.Depth_2.out_channels,
                                        momentum=0.01)


# ============================== Residual (x3) ========================== #
scales_samples = [16, 8, 4]
channels_in = [int(self.Depth.out_channels / x) for x in (1, 2, 2, 4)]
res_units = nn.ModuleList()
batch_res = nn.ModuleList()
for i in range(3):
    res_unit_1 = nn.Conv3d(in_channels=channels_in[i],
                           out_channels=int(channels_in[i+1]),
                           kernel_size=(1, 1, 1), stride=(1, 1, 1),
                           padding='valid', bias=False)
    res_unit_1.weight = nn.init.kaiming_normal_(res_unit_1.weight)

    batch1 = nn.BatchNorm3d(res_unit_1.out_channels, momentum=0.01)

    res_unit_2 = nn.Conv3d(in_channels=channels_in[i],
                           out_channels=int(channels_in[i+1]),
                           kernel_size=(1, scales_samples[i], 1),
                           stride=(1, 1, 1),
                           padding='same', bias=True)
    res_unit_2.weight = nn.init.kaiming_normal_(res_unit_2.weight)
    res_unit_2.bias = nn.init.zeros_(res_unit_2.bias)
    batch2 = nn.BatchNorm3d(res_unit_2.out_channels, momentum=0.01)

    res_unit_3 = nn.Conv3d(in_channels=res_unit_2.out_channels,
                           out_channels=res_unit_2.out_channels,
                           kernel_size=(1, 1, 5), stride=(1, 1, 1),
                           padding='valid', bias=False)
    # res_unit_3.weight = nn.init.kaiming_normal_(res_unit_3.weight)
```

```
    batch3 = nn.BatchNorm3d(res_unit_3.out_channels, momentum=0.01)

    temp_list = nn.ModuleList()
    temp_list.append(res_unit_1)
    temp_list.append(res_unit_2)
    temp_list.append(res_unit_3)
    res_units.append(temp_list)
    temp_lista_2 = nn.ModuleList()
    temp_lista_2.append(batch1)
    temp_lista_2.append(batch2)
    temp_lista_2.append(batch3)
    batch_res.append(temp_lista_2)

self.res_units = res_units
self.batch_res = batch_res


# ======================== TEMPORAL REDUCTION ======================== #
conv_unit_1 = nn.Conv3d(in_channels=res_unit_3.out_channels,
                        out_channels=res_unit_3.out_channels,
                        kernel_size=(1, int(self.scales_samples[2] / 4), 1),
                        stride=(1, 1, 1), padding='same', bias=False)
nn.init.kaiming_normal_(conv_unit_1.weight)
self.temp_red = conv_unit_1

self.batch_res3 = nn.BatchNorm3d(conv_unit_1.out_channels, momentum=0.01)

# ======================== CHANNEL MERGING ========================== #
ch_merg = nn.ModuleList()
batch_ch_merg = nn.ModuleList()
for i in range(2):
    conv = nn.Conv3d(in_channels=res_unit_3.out_channels,
                         out_channels=int(len(self.scales_samples)*
                         self.filters_per_branch/4),
                         kernel_size=(2, 1, 5), stride=(1, 1, 1),
                         padding='valid', bias=False)

    conv.weight = nn.init.kaiming_normal_(conv.weight)
    batch = nn.BatchNorm3d(conv.out_channels, momentum=0.01)
    ch_merg.append(conv)
    batch_ch_merg.append(batch)

self.ch_merg = ch_merg
self.batch_ch_merg = batch_ch_merg
```

```
self.Depth_CM = nn.Conv3d(in_channels=int(len(self.scales_samples)*
                          self.filters_per_branch/4),
                          out_channels=int(len(self.scales_samples)*
                          self.filters_per_branch/4),
                          kernel_size=(2, 1, 5),
                          groups=int(len(self.scales_samples)*
                          self.filters_per_branch/8),
                          padding='valid', bias=False)
self.batch_depth_cm = nn.BatchNorm3d(self.Depth_CM.out_channels,
                                     momentum=0.01)

# ======================== TEMPORAL MERGING ========================== #
self.temp_merg = nn.Conv3d(in_channels=self.Depth_CM.out_channels,
                           out_channels=int(len(self.scales_samples)*
                           self.filters_per_branch/4),
                           kernel_size=(1, self.input_samples // 64, 1),
                           stride=(1, 1, 1), padding='valid', bias=False)
self.temp_merg.weight = nn.init.kaiming_normal_(self.temp_merg.weight)

self.batch_tempmerg = nn.BatchNorm3d(self.temp_merg.out_channels,
                                     momentum=0.01)

self.depth_temp = nn.Conv3d(in_channels=self.temp_merg.out_channels,
                          out_channels=int(len(self.scales_samples)*
                          self.filters_per_branch/4)*2,
                          kernel_size=(1, self.input_samples // 64, 1),
                          groups=int(len(self.scales_samples)*
                          self.filters_per_branch/4),
                          padding='valid', bias=False)

self.batch_depthtemp = nn.BatchNorm3d(self.depth_temp.out_channels,
                                      momentum=0.01)
# ================================ OUTPUT ================================== #
output_conv = nn.ModuleList()
batch_output_conv = nn.ModuleList()
for i in range(4):
    conv = nn.Conv3d(in_channels=self.depth_temp.out_channels,
                     out_channels=int(self.filters_per_branch*
                     len(self.scales_samples)/2),
                     kernel_size=(1, 1, 1), stride=(1, 1, 1),
                     padding='valid', bias=False)

    conv.weight = nn.init.kaiming_normal_(conv.weight)
    batch = nn.BatchNorm3d(conv.out_channels, momentum=0.01)
    output_conv.append(conv)
```

```
            batch_output_conv.append(batch)

        self.out_conv = output_conv
        self.batch_output_conv = batch_output_conv

        self.flat = nn.Flatten()
        self.dense = nn.Linear(in_features=int(self.filters_per_branch*
                               len(self.scales_samples)/2),
                               out_features=n_classes)
        self.soft = nn.Softmax(dim=1)


    def forward(self, x, symmetric=True):
        # ============================== INPUT ============================== #
        ch_lateral = self.ch_lateral
        input_data = x.unsqueeze(1)
        if symmetric == True:
            superposition = False
            if ch_lateral < self.n_cha // 2:
                superposition = True
            ncha = self.n_cha-ch_lateral
            #idx = torch.zeros(self.n_cha)
            #left_idx = (idx[:ch_lateral] == 1)
            left_idx = list(range(ch_lateral))
            ch_left = input_data[:, :, :, :, left_idx]
            #ch_left = torch.gather(input, index=left_idx, dim=-2)
            right_idx = list(np.array(left_idx)+int(ncha))
            ch_right = input_data[:, :, :, :,  right_idx]
            #ch_right = torch.gather(input, index=right_idx, dim=-2)

            if superposition:
                central_idx = list(np.array(range(ncha-ch_lateral))+ch_lateral)
                #ch_central = torch.gather(input, index=central_idx, dim=-2)
                ch_central = input_data[:, :, :, :, central_idx]
                left_init = torch.cat((ch_left, ch_central), dim=-1)
                right_init = torch.cat((ch_right, ch_central), dim=-1)
            else:
                left_init = ch_left
                right_init = ch_right
            input_data = torch.cat((left_init, right_init), dim=2)
            division = 2
        else:
            division = 1
        # print(input.shape)
        # ====================== TEMPOSPATIAL ANALYSIS ====================== #
```

```
# ========================== Inception (x2) ========================== #
# INCEPTION BLOCK 1
b1 = list()
for i in range(len(self.Inception1_1)):
    conv = self.Inception1_1[i]
    batch = self.Batch_Inception[i]
    unit = F.dropout3d(self.Activation(batch(conv(input_data))),
    self.dropout_rate)
    #unit = self.Activation(batch(conv(input)))
    #unit = F.dropout3d(self.Activation(batch(conv(input))),
    self.dropout_rate)
    b1.append(unit)
    # print(unit.shape)
temp_b1_out = torch.cat(b1, dim=1)
# print(temp_b1_out.shape)
temp_b1_out = temp_b1_out + input_data
temp_b1_out = self.Pool_Inception(temp_b1_out)
# print(temp_b1_out.shape)
# Grouped conv
temp_b1_out_c = F.dropout3d(self.Activation(self.batch_depth_inc(
self.Depth(temp_b1_out))), self.dropout_rate)
# print(temp_b1_out_c.shape)
b1_out = temp_b1_out_c + temp_b1_out
# print(b1_out.shape)


# INCEPTION BLOCK 2
b2 = list()
for i in range(len(self.Inception1_2)):
    conv = self.Inception1_2[i]
    batch = self.Batch_Inception_2[i]
    unit = F.dropout3d(self.Activation(batch(conv(b1_out))),
    self.dropout_rate)
    b2.append(unit)
    #print('Unit', unit.shape)
temp_b2_out = torch.cat(b2, dim=1)
# print(temp_b2_out.shape)
temp_b2_out = temp_b2_out + b1_out
temp_b2_out = self.Pool_Inception(temp_b2_out)
# print(temp_b2_out.shape)
# Grouped conv
temp_b2_out_c = F.dropout3d(self.Activation(self.batch_depth_inc_2(
self.Depth_2(temp_b2_out))), self.dropout_rate)
# print(temp_b2_out_c.shape)
b2_out = temp_b2_out_c + temp_b2_out
```

```
# print('End block2: ' , b2_out.shape)

# ============================ Residual (x3) ======================== #
temp_res = b2_out
# RESIDUAL
for i in range(3):
    conv1 = self.res_units[i][0]
    batch1 = self.batch_res[i][0]
    temp_res_1 = F.dropout3d(self.Activation(batch1(conv1(temp_res))),
    self.dropout_rate)
    conv2 = self.res_units[i][1]
    batch2 = self.batch_res[i][1]
    temp_res_2 = F.dropout3d(self.Activation(batch2(conv2(temp_res))),
    self.dropout_rate)
    temp_res_out = temp_res_1 + temp_res_2
    temp_res_out = self.Pool_Inception(temp_res_out)
    conv3 = self.res_units[i][2]
    batch3 = self.batch_res[i][2]
    #batch2 = self.res_units[i][1]
    temp_res_3 = F.dropout3d(self.Activation(batch3(conv3(temp_res_out))),
    self.dropout_rate)
    res_out = temp_res_3 + temp_res_out
    temp_res = res_out

#print(res_out[0].shape)

# ======================== TEMPORAL REDUCTION ======================== #
temp_rb4 = F.dropout(self.Activation(self.batch_res3(self.temp_red(res_out))),
self.dropout_rate)
rb4 = temp_rb4 + res_out
rb4_out = self.Pool_Inception(rb4)

# print(rb4_out.shape)

# ========================== CHANNEL MERGING ========================== #
ch_merg = rb4_out
for _ in range(2):
    conv = self.ch_merg[_]
    batch = self.batch_ch_merg[_]
    temp_cm = F.dropout3d(self.Activation(batch(conv(ch_merg))),
    self.dropout_rate)
    temp_cm = temp_cm + ch_merg
    ch_merg = temp_cm

CM3 = F.dropout3d(self.Activation(self.batch_depth_cm(self.Depth_CM(
```

```
temp_cm))), self.dropout_rate)
# print(CM3.shape)

# ======================== TEMPORAL MERGING ========================== #
temp_TM = F.dropout3d(self.Activation(self.batch_tempmerg(
self.temp_merg(CM3))), self.dropout_rate)
TM = temp_TM + CM3
TM_out = F.dropout3d(self.Activation(self.batch_depthtemp(
self.depth_temp(TM))), self.dropout_rate)
output = TM_out
# ============================== OUTPUT ============================== #
for _ in range(4):
    output_temp = output
    conv = self.out_conv[_]
    batch = self.batch_output_conv[_]
    output_temp = F.dropout3d(self.Activation(batch(conv(output_temp))),
    self.dropout_rate)

    #output_temp = F.dropout3d(self.Activation(batch(self.out_conv(
    output_temp))), self.dropout_rate)
    output = output_temp + output

flat = self.flat(output)
output_layer = self.soft(self.dense(flat))
return output_layer
```