MSc in Computer Science and Engineering
Scuola di Ingegneria Industriale e dell'Informazione
Dipartimento di Elettronica, Informazione e Bioingegneria

POLITECNICO
MILANO 1863

DESIGN OF AN EMBEDDED PROTOTYPING PLATFORM FOR BRAIN-COMPUTER INTERFACES

Supervisor: Prof. William FORNACIARI
Co-supervisor: Ph.D. Luca ASCARI

Master Graduation Thesis by:
Andrea BELLOTTI

Academic Year 2016-2017

A mia nonna, che ha lottato con me, e più di me,
per essere presente al raggiungimento di questo obiettivo.

Ringraziamenti

Guinto al termine di questa tesi vorrei ringraziare coloro che hanno contribuito al raggiungimento di questo grande traguardo.

I primi ringraziamenti vanno sicuramente al Professor William Fornaciari e a Luca Ascari per la disponibilità e cortesia, per i preziosi suggerimenti e per le competenze trasmesse durante lo svolgimento della tesi.

La mia riconoscenza va a tutte le persone che lavorano in Camlin Italy, l'azienda presso cui ho sviluppato questa tesi, per la disponibilità, i preziosi consigli e per avermi sempre fatto sentire parte integrante del gruppo.

Ringrazio la mia famiglia per i sacrifici e per l'appoggio incondizionato e l'incoraggiamento che mi hanno fornito durante tutto il mio percorso universitario.

Il ringraziamento più grande però va alla mia fidanzata Marta, perchè mi ha sostenuto psicologicamente e incoraggiato nei momenti di sconforto, sopportando insieme a me il lungo periodo di difficoltà senza farmelo pesare.

Infine ringrazio tutta la "famiglia" di Milano, con cui ho condiviso gioie e dolori durante gli anni trascorsi al Politecnico, per esserci sempre stata in caso di bisogno.

Andrea

Abstract

During the last years the development of Brain-Computer Interface (BCI) systems has experienced a rapid growth, due to the availability of more accurate and sophisticated signal processing algorithms and to an increasing understanding of brain signals. Thus, the range of BCI applications has been significantly widening, paving the way for a completely novel and more natural user experience in controlling remote devices.

Nowadays, the reduced portability of BCI systems limits their adoption in everyday applications, exacerbating the fine-tuning, testing and use in real-life scenarios. The ever-increasing computational power and flexibility of embedded systems make them a viable solution to alleviate this issue. This thesis proposes an infrastructure based on the NVIDIA Jetson TX2 embedded system designed to achieve this goal. In particular, it has been designed to be portable, modular, and compatible with the most popular BCI frameworks and commercial acquisition devices. Moreover, it enables for a rapid design, implementation and testing of BCI systems, notably reducing deployment time with respect to state-of-the-art architectures. The effectiveness of the proposed system has been evaluated in implementing a BCI system able to solve a two classes motor imagery problem (left vs right).

Estratto

Negli ultimi anni lo sviluppo di Brain-Computer Interface (BCI) ha avuto una rapida crescita, grazie alla disponibilità di algoritmi di elaborazione dei segnali più accurati e sofisticati e ad una crescente comprensione dei segnali cerebrali. Quindi, gli utilizzi delle BCI si sono notevolmente ampliati, aprendo la strada a modi completamente nuovi e più naturali per il controllo di dispositivi remoti.

Attualmente, la ridotta portabilità delle BCI limita la loro adozione nelle applicazioni di tutti i giorni, esacerbandone la messa a punto, la verifica e l'uso in contesti reali. La sempre crescente capacità computazionale e la flessibilità dei sistemi embedded li rende una possibile soluzione per alleviare questo problema. Questa tesi propone un'infrastruttura basato sul sistema embedded NVIDIA Jetson TX2 progettata per raggiungere questo obiettivo. In particolare, essa è stata progettata per essere portabile, modulare e compatibile con i più diffusi framework per BCI e dispositivi commerciali per l'acquisizione dei segnali. Inoltre, essa permette una rapida progettazione, implementazione e validazione delle BCI, riducendo notevolmente i tempi per arrivare al loro impiego rispetto allo stato dell'arte. L'efficacia del sistema proposto è stata testata implementando una BCI capace di risolvere un problema binario di motor imagery (destra contro sinistra).

Contents

List of Figures

List of Tables

Acronyms

BCI	Brain-Computer Interface
EEG	Electroencephalography
EOG	Electrooculography
EMG	Electromyography
ECG	Electrocardiography
CAR	Common Average Reference
MEG	Magnetoencephalography
ECoG	Electrocorticography
NIRS	Near-Infrared Spectroscopy

fMRI	Functional Magnetic Resonance Imaging
FPGA	Field Programmable Gate Array
SoC	System on Chip
GPU	General Purpose Unit
ICA	Independent Component Analysis
ERP	Event-Related Potentials
ErrP	Error-Related Potentials
SMR	Sensorimotor Rhythms
ERD	Event-Related Desynchronization
ERS	Event-Related Desynchronization
CSP	Common Spatial Patterns
RCSP	Regularized Common Spatial Patterns
LTCCSP	Local Temporal Correlation Common Spatial Patterns
CSSP	Common Spatio-Spectral Patterns
CSSSP	Common Sparse Spatio-Spectral Patterns
SpecCSP	Spectrally-weighted Common Spatial Patterns
FBCSP	Filter Bank Common Spatial Patterns
SVD	Singular Value Decomposition
SVM	Support Vector Machine
DAL	Dual-Augmented Lagragian
FIR	Finite Impulse Response
DWT	Discrete Wavelet Transform
LDA	Linear Discriminant Analysis

LSL	Lab Streaming Layer
TOBI	Tools For Brain-Computer Interaction
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
NTP	Network Time Protocol
RTT	Round-Trip Time
CFA	Clock Filter Algorithm
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
DBN	Deep Belief Network
RBM	Restricted Boltzmann Machine
SAE	Stacked Auto-Encoder
CAE	Convolutional Auto-Encoder
ADAM	Adaptive Moment Estimation
ADMM	Alternating Direction Method of Multipliers
IPC	Inter-Process Communication
MQTT	Message Queue Telemetry Transport
BLE	Bluetooth Low Energy
VR	Virtual Reality
SSVEP	Steady State Visual Evoked Potentials
PSD	Power Spectral Density

Chapter 1

Introduction

1.1 Brain-Computer Interfaces

A Brain-Computer Interface (BCI) is a system that maps electrophysiological brain signals to the corresponding mental state, with the purpose of controlling a program or a device. Classical BCIs rely uniquely on measurements of brain signals while hybrid or multi-modal BCIs can include also other physiological measures such as muscle or eye movement signals [12]. BCI applications can be parted in five main categories:

- *Replacement* of functions that were lost (for example, wheelchair control).
- *Restoration* of functions that were lost (for example, stimulation of muscles in a paralyzed person).
- *Improvement* of functions that were reduced (for example, stroke rehabilitation).
- *Enhancement* of functions (for example, anticipated detection and actuation of emergency brake).

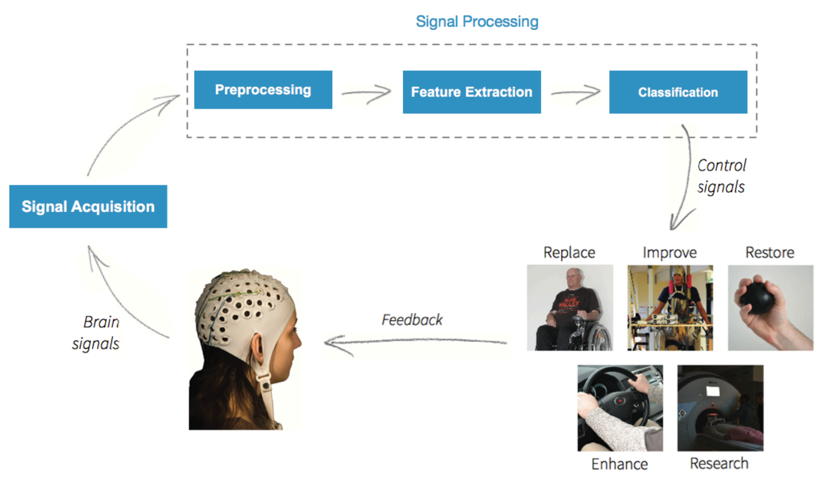


Figure 1.1: General architecture of a BCI. Adapted from:[1]

- *Research* to study brain functions

As we can see from Figure 1.1, a BCI system is made up of two main components: signal acquisition and signal processing. The signal processing part in turn is composed of three subparts, namely preprocessing, feature extraction and classification. The output of a BCI is generally a stream of commands whose meaning is highly application dependent.

1.1.1 Signal acquisition

The signal acquisition system monitors and aquires brain signals, usually in form of electrical or metabolic activity. Four main methods are used to measure electrical activity:

- Electroencephalography (EEG), which measures the voltage fluctuations on the scalp;
- Magnetoencephalography (MEG) which measures magnetic fields produced by electrical currents in the brain;
- Electrocorticography (ECoG) which measures voltage fluctuations *on* the cortex;
- Micro-electrode arrays which measure voltage fluctuations *in* the cortex.

All these measurements are the macroscopic result of ionic currents of populations of neurons. On the other hand, metabolic activity is monitored by means of Near-Infrared Spectroscopy (NIRS), which indirectly assesses neuronal activity by measuring changes in oxygenated and deoxygenated hemoglobin in tissues using near-infrared light, and Functional Magnetic Resonance Imaging (fMRI), which measures changes in blood flow that are effect of brain activity.

Previous methods differ in three important aspects: the spatial and temporal resolution of the measurements and the size of the area of the brain that can be measured. The temporal resolution for micro-electrode arrays, ECoG, EEG and MEG is in the millisecond range, whereas for NIRS and fMRI it is in the order of seconds. The best spatial resolution is achieved by micro-electrode arrays monitor the activity of up to a single neuron, then there are ECoG and fMRI which can reach a spatial resolution in the order of millimeters and then there is EEG in the centimeter range. EEG, fMRI and NIRS provide the best coverage capability, being able to measure the whole brain, while MEG can cover large parts of the brain and ECoG only small cortical areas. The worst method in terms of coverage is the micro-electrode arrays that covers only a few hundred neurons.

Signal acquisition is carried out by the hardware that actually measures the brain signals and the software that conveys the acquired data to the processing unit. This software component is responsible for synchronization and storage of the acquired signals.

1.1.2 Signal processing

The signal processing is the part of the BCI that predicts a mental state given some brain signals. As we have already mentioned it is divided in three steps: preprocessing, feature extraction and classification. The feature extraction and classification steps can be performed separately, extracting reasonable neuroscientific features in one step and feeding them to a classifier in another, or jointly, performing both automatic feature extraction and classification in a single step. The latter case corresponds to the signal processing performed by Deep Learning models such as Convolutional Neural Networks (CNN), while the former corresponds to the classical procedure in which two different machine learning models are used to perform feature extraction and classification respectively.

In the preprocessing step, which is common to both approaches, raw signals are usually filtered and cleaned to increase the signal-to-noise ratio. The cleaning procedure consists in the application of an artifact removal algorithm able to identify and remove the artifactual components of the signal preserving the integrity of the original brain signal. A subsampling is then usually applied on preprocessed signals, in order to reduce the computational complexity of subsequent stages.

Feature extraction turns the output signals of the previous steps into a different representation that maximizes their discriminability based on the mental states to which they are associated. This step is the core of the model construction and is highly application dependent since it must extract features relevant to the brain patterns that the adopted BCI paradigm aims to recognize. BCIs are not a mind-reading devices, they can only detect very specific patterns of brain activity that the subjects voluntarily or involuntarily produce. The objective of the feature extraction step is to obtain features that maximize the detectability of these patterns.

When the last step of the signal processing is performed separately, the computed feature vector is fed into a classifier to recognize the corresponding mental state. The classifier is a mathematical model usually trained in a supervised fashion, i.e., given some feature vectors and the corresponding labels it learns the pattern that best explains the association.

1.2 Thesis objective

BCIs are being investigated in an increasing number of everyday tasks, for instance they have been experimented as driving assistant systems for preventing accidents by detecting driver's cognitive states like drowsiness and mental fatigue [13] [14] or behavior intentions like emergency braking [15].

So far the majority of the experiments remain at a stage of laboratory demonstration with the employment of expensive laboratory-grade signal acquisition systems and bulky personal computers. These experiments had, and are still having, a great impact in the improvement of the current state of the art accuracy, but to effectively adopt BCIs in the everyday life we need portable and modular systems. Some studies regarding the design of portable BCI devices do exist, but most of them are focused on specific low level implementations of BCI based on FPGA [16] or neuromorphic prototypes [17].

The aim of this thesis is to design an easy-to-use and flexible system for BCI prototyping. The requirements for the target system are the ability to acquire simultaneous signals coming from heterogeneous devices, to process them with some general BCI frameworks and to aggregate in real-time the predictions of the models. The issues that need to be addressed for building such system are the synchronization of the signals in a context of mobility, the integration of the BCI frameworks in an embedded device and the real-time performance of such frameworks.

Another requirement of the target infrastructure is that it must be designed as a distributed system, thus allowing also the distribution of the various components on different machines to provide even more flexibility.

1.3 Thesis contribution

The work described in this thesis has been conducted in collaboration with the Research and Development laboratories of Camlin Italy situated in Parma. Camlin Italy is part of the CAMLIN Group, an international company operating in over 20 countries developing, producing and selling advanced products for a wide range of sectors, including power, rail and health; moreover it is also involved in several R&D projects in a variety of scientific sectors. In particular, Camlin Italy is the competence center for Data Science and Artificial Intelligence and is specialized in the development of applications based on Machine Learning algorithms.

This thesis proposes a portable and modular infrastructure based on the NVIDIA Jetson TX2 embedded system that aims to allow the use of BCIs in everyday life. The board has been chosen mainly to address the need of computational power and portability of the BCI models, especially considering the employment of Deep Learning models that require a GPU. The proposed platform takes care of signal acquisition and realignment, model building and real-time command or feedback delivery. It has been designed with flexibility in mind, allowing the users to choose acquisition devices and BCI frameworks to use for model training.

The main contributions regarding the design of the described platform are the integration of Lab Streaming Layer (LSL), the adopted transmission and synchronization framework, the validation of its synchronization capabilities in a real environment where multiple wireless devices stream data at high frequencies, the characterization of different BCI framework and the analysis of their performances in real-time. In addition, given that the NVIDIA Jetson provides an output video port, we also implemented a component (based on the python library PsychoPy [18]) which deals with the presentation of the visual stimuli or the real-time feedbacks used in the training sessions of specific BCI paradigms. The integration of this component enhances the infrastructure compactness as it provides also the ability to acquire internally the training sets.

To test the effectiveness of the platform we implemented a real-time BCI system with the future objective of presenting a realistic feedback by means of the movements of a toy robot. The real-time capabilities of the system have been verified (included the interaction with the robot), but to effectively use it to present a feedback we need to improve the generalization capabilities of the BCI models.

1.4 Structure of thesis

The thesis is structured as follows.

- In Chapter 2 we present an overview of the state-of-the-art concerning EEG-based BCIs. In particular, in Section 2.1 we describe the patterns that can be detected in EEG signals and some of the methods most widely used for discrimination. In Section 2.3, we present a brief description of the most valuable studies involved in the development of portable EEG-based BCIs.

- In Chapter 3 we describe the proposed infrastructure and its experimental evaluation. We start introducing the most relevant design choices we made, then, in Section 3.1, we describe core components and technologies integrated in the infrastructure. Then, in Section 3.2 we present the experimental validation of synchronization and processing capabilities of the infrastructure.

- In Chapter 4 we present the BCI system implemented to validate the effectiveness of the proposed platform. In Section 4.1 we describe the experimental paradigm designed for data acquisition, the methods used to train the BCI model and the operation of the feedback sessions. Then we present the results in Section 4.2.

- In Chapter 5 we draw the conclusions and propose some possible future improvements.

At the end of Chapter 3 and 4 we present two sections that summarize the original work done in the respective chapter.

Chapter 2

State of the art

2.1 Electroencephalography

Electroencephalography (EEG) is a monitoring technique measuring electrical activity along the scalp; in particular, it monitors voltage fluctuations produced by neural activity. It is one of the most common and used method in literature because it is easy to use, portable, non-invasive and inexpensive; moreover, it provides an excellent temporal resolution, that is an essential requirement for real-time BCIs.

For these reasons, EEG was chosen in this thesis as acquisition mechanism. However, its high sensitivity to artifacts induces a low signal-to-noise ratio, that is one of the main drawbacks of EEG; thus, the next section will show that artifact removal is an essential step in EEG signal analysis.

Several types of electrodes can be integrated in EEG recording systems; however, two main categories can be identified: wet and dry electrodes. The former employs saline-based gel to enhance the contact between the electrode and the scalp, in order to improve signal transmission; they are vary common in clinical applications and neuroscience research. On the other hand, dry electrodes do not require conductive pastes, and therefore are more suitable for commercial scenarios. Usually wet electrodes produce more accurate data than dry ones, as they allow for a better contact between electrodes and scalp [19]. On the contrary, dry electrodes possess very interesting characteristics (e.g., usability) and are significantly less expensive, representing the best compromise between cost and accuracy in several application scenarios. Active electrodes are a third viable solution. They are made up of active electronics circuitry designed to suppress interference, carrying out impedance transformation directly on the electrode (by means of an operational amplifier that reduces the output impedance) and reducing the signal path length between electrode and first amplifier stage. The most commonly used electrode type consists of sintered (coated) Ag/AgCl disks, which quickly establish and then maintain consistent and stable electrochemical potentials against biological tissues, together with low dc offset variability. Moreover, Ag/AgCl electrodes are free from potential allergenic compounds and have excellent long-term electrical stability [4].

For multichannel recordings with a large number of electrodes, electrode caps are often used. Traditionally, the International 10-20 system defined by the International Federation of Societies for Electroencephalography and Clinical Neurophysiology has been used to describe the locations of EEG scalp electrodes, relative to anatomic landmarks on the human head.

Two recording procedures can be adopted, namely bipolar and referential. In the bipolar mode, each active electrode has its own reference, whereas in the referential mode each active electrode is referenced with respect to the same electrode, i.e., the potential difference between each electrode and the reference is measured.

Several different reference electrode placements are proposed in scientific literature; the most commonly used are: vertex (Cz), linked-ears, linked-mastoids, ipsilateral ear, and contralateral ear. There are also re-referencing techniques, that perform spatial filtering to highlight relevant spatial patterns, blurred by the original reference mode; the most widely used are Common Average Reference (CAR) and Laplacian [4].

EEG acquisition systems can be classified in two main classes with respect to their connection to the processing system. Wired systems take advantage of a wired connection to send their acquisitions to a stationary amplifier. On the contrary, wireless systems acquire EEG signals from the electrodes and amplify them in a device attached to the cap, before forwarding them to a computational device via a wireless connection. Wireless connection avoids noise introduced by cable movements and, at the same time, allows for completely free movements; wired systems, on the other hand, can provide higher transmission frequencies and a better synchronization.

As mentioned earlier, one of the drawbacks of EEG is its low signal-to-noise ratio due to the high sensitivity to artifacts.

Indeed, EEG signals recorded from the scalp are a superimposition of neural and artifactual activity produced by multiple brain or extra-brain processes, due to volume conduction effects. To isolate the significant part of the signal is a very hard task, because it requires the removal of effects produced by several noises sources.

The greatest part of artifacts belongs to two main groups: physiological and system artifacts. The most common physiological artifacts arise from eye movements, eye blinks, muscle noise and heart signals. System artifacts are generated by 50/60 Hz power supply interference, impedance fluctuation, cable defects, electrical noises and unbalanced impedances of the electrodes.

Often the preprocessing stage is able to reduce these artifacts, restoring the informative information. Moreover, some artifacts are characterized by signal properties that make their correction simple. For instance, power line noise can be easily mitigated by applying a notch filter around the power line frequency; others noises, especially the endogenous ones, require more complex procedures.

In the following section we present the state of the art of artifact removal; moreover, we introduce the physiological mechanisms that generate EEG signals and finally we delve into the approaches for pattern detection in EEG signals.

2.1.1 EEG artifacts

Because of volume conduction, EEG signals collected from the scalp are supervisions of neural and artifactual activity form multiple brain or extra-brain processes occurring within a large volume. Recently, Independent Component Analysis (ICA) has been shown very promising results (applied with other machine learning algorithms) for artifact removal as it allows to reverse the superposition by separating EEG signals into statistically independent components [20, 21, 22].

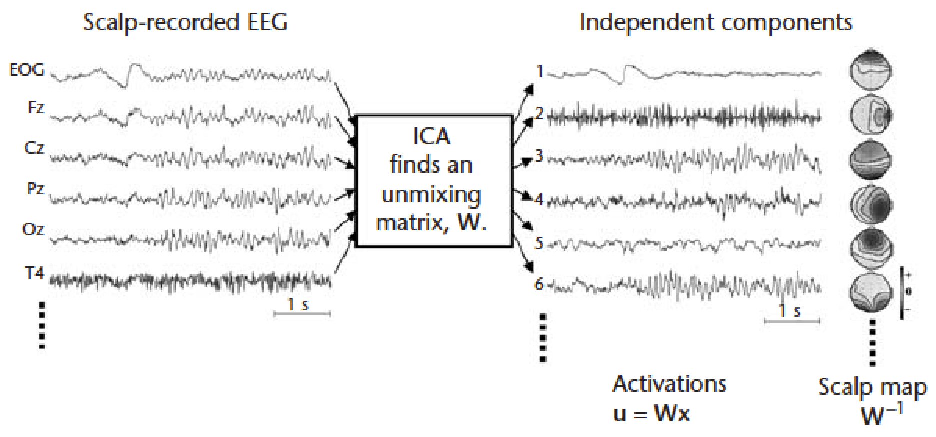


Figure 2.1: Schematic overview of ICA applied to EEG data. Source:[2]

Independent Component Analysis

ICA assumes that the analyzed signal is composed by several statistically independent components; it is based on the so-called *blind source separation*. Blind source separation tries to part statistically independent components that compose the observed signals, with no a-priori knowledge about source statistics or mixing process. If $\mathbf{x} = \{x_1(t), x_2(t), ..., x_N(t)\}$ is a signal composed by N components, ICA recovers its N sources, $\mathbf{s} = \{s_1(t), s_2(t), ..., s_N(t)\}$ whose linear mix has produced the measured signals $\mathbf{x}$. In particular:

$$\mathbf{x} = \mathbf{A}\mathbf{s}, \tag{2.1}$$

where $\mathbf{A}$ is an unknown matrix. More specifically, ICA computes a version $\mathbf{u} = \mathbf{W}\mathbf{x}$ identical to the actual sources $\mathbf{s}$ but for scaling and a permutation, being $\mathbf{W}$ the square matrix that linearly inverts the mixing process. The rows of $\mathbf{u}$, the so-called *component activations*, are the time courses of the respective independent components; the columns of the inverse of matrix $\mathbf{W}^{-1}$ represent the projection strengths of the respective components onto each scalp sensor; these may be interpolated to show the topography (*scalp map*) associated with each component.

ICA assumes that the source signals are independent and that their distributions are not Gaussian. Algorithms to compute $\mathbf{W}$ take generally advantage of an optimization process, thus, to choose their cost function is a very important step. There are many available cost functions, but in general they are related to the statistical independence of the estimated components. The two broadest definitions of independence for ICA are:

- Minimization of mutual information.
- Maximization of non-Gaussianity.

Thus, there are several algorithmic approaches to solve the ICA problem: InfoMax, FastICA [23] and JADE [24] are just some examples of the widely used ones in the scientific community.

ICA aims at obtaining statistically independent outputs, with no constraints on the matrix **W**, which represents the contributions of every source on the original signal. There are some caveats in this approach; for instance, ICA can not identify the actual number of source signals and an uniquely correct ordering of sources based on their importance. Thus, the processing of several time windows of EEG signals should be carefully carried out, because the order and resultant independent components are generally arbitrary. For this reason, ICA has been only successfully used for artifact removal in conjunction with machine learning algorithms like Support Vector Machine (SVM), trained to recognize the artifactual components.

Figure 2.2 demonstrates eye movement and temporal muscles artifact removal from a 5-seconds recording of EEG. The components representing the artifacts are identified from the topographies and are eliminated by zeroing out the corresponding rows of the activation matrix **u** and projecting the remaining components to the scalp electrodes.

$$\widehat{\mathbf{x}} = \mathbf{W}^{-1}\mathbf{u} \tag{2.2}$$

2.1.2 Brain rhythms

Before starting to delve into the approaches used to extract information in EEG-based BCIs, we need to understand the neurophysiological mechanisms that produce EEG signals.

EEG records rhythmic activity that reflects the neural oscillations in the central nervous system. These oscillations are fluctuations in the excitability of populations of neurons. At the level of neural ensembles, synchronized activity of large numbers of neurons gives rise to the macroscopic oscillations that are measured by EEG devices [2, 25, 26]. Oscillations are described

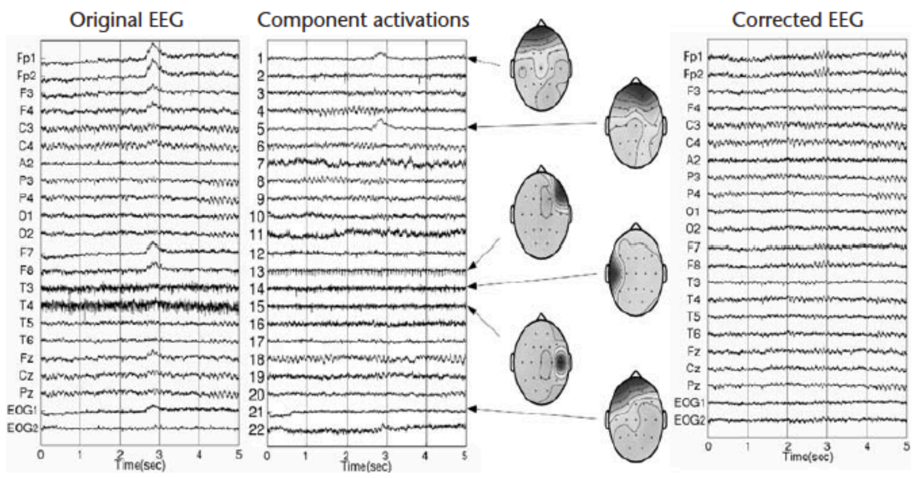


Figure 2.2: Demonstration of EEG artifact removal by ICA. Source:[2]

by three pieces of information: frequency, power and phase. Frequency is the speed of the oscillation, power is the amount of energy in a frequency band and is the squared amplitude of the oscillation, and phase is the position along the sine wave at any given time point. Power and phase are independent of each other, meaning that neural dynamics measured through power are different from those measured through phase. As we will explain in more details later, oscillations are also characterized by spatial localization and temporal dynamic, which are among the most used characteristics in the feature extraction step, especially in relation to the events that induced them. Brain rhythms are grouped into five main frequency bands: delta (0.5-4 Hz), theta (4-8 Hz), alpha (8-12 Hz), beta (12-30 Hz) and gamma (30-150 Hz). This grouping results from neurobiological mechanisms of brain oscillations [25] [27] [28], however there are no precise boundaries defining the bands and, above all, there is not a unique interpretation of the different frequency bands as the knowledge of the mechanisms that underline brain activity is still blurred. However some empirical interpretations have been proposed in some studies, for instance, delta waves have been associated with deep sleep and have been detected also in the waking state, theta waves sometimes appear in case of drowsiness or deep meditation, alpha have been recognized in cases of relaxed awareness without attention or concentration, and beta waves have been associated with active thinking, attention and problem solving [29].

These oscillations are the foundation of the methods described in the following sections.

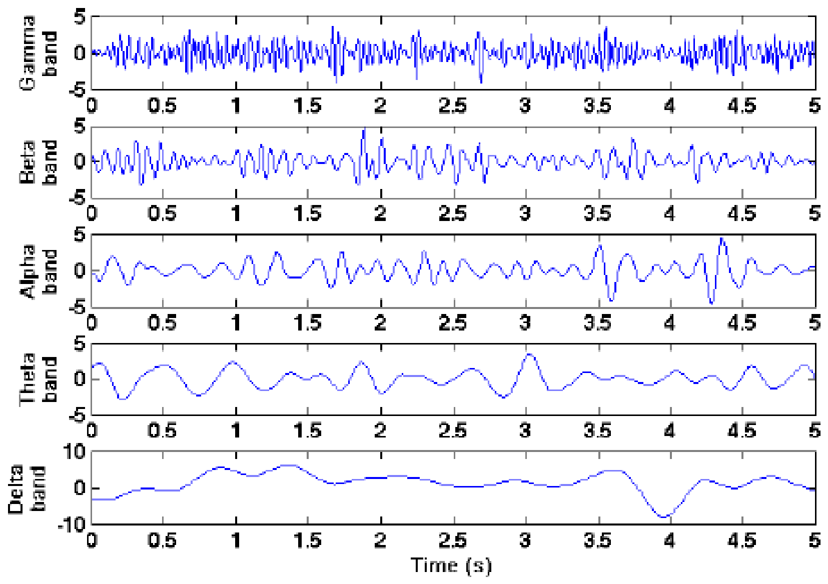


Figure 2.3: The five main frequency bands. Source:[3]

2.2 EEG-based Brain-Computer Interfaces

It is important to realize that a BCI system is not a mind-reading device, it can only detect very specific patterns of brain activity that the subject voluntarily or involuntarily produces.

Two of the most used patterns that have been identified in electrical signals generated from brain activity are event-related potentials (ERPs) and event-related desynchronization/synchronization (ERD/ERS). They are both time-locked to the inducing event, but while ERPs are also phase-locked, ERDs/ERSs are non-phase-locked [2]. In a finger movement process, for example, pre-movement negativity prominent prior to movement onset, and post-movement beta oscillations occurring immediately after movement offset are respectively phase-locked (evoked) and non-phase-locked processes [30]. ERPs and ERD/ERS are among the most used patterns in the BCI field because they can be modulated by the user's voluntary intent in case of ERD/ERS or by user's attention shift in case of ERPs. In fact, the design of an EEG-based BCI paradigm is largely about how to instruct the users to express their voluntary intents so that the corresponding patterns can be detected.

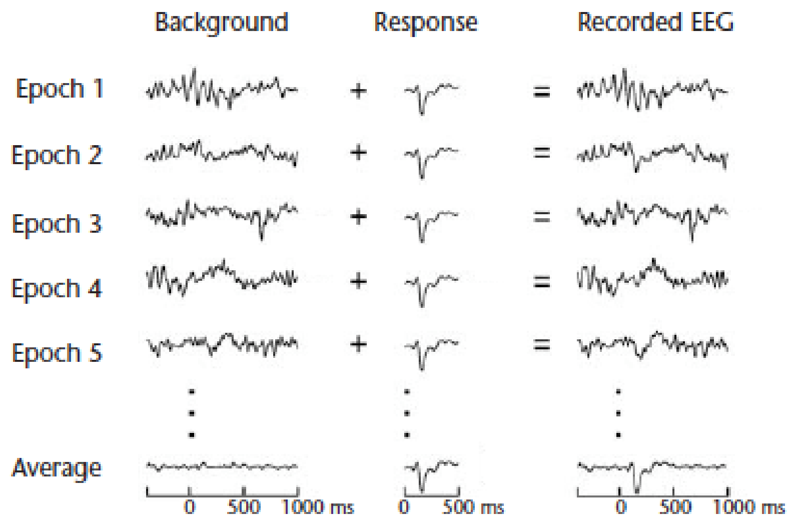


Figure 2.4: ERP computation. Source: [4]

2.2.1 ERP

An Event Related Potential (ERP) is the measured time-locked response that is direct result of a specific sensory, cognitive or motor event [31]. The extraction of the ERP waveform is done by repeatedly presenting an event of interest, such as a visual stimulus on a computer screen, and collecting the small fraction of EEG measurements following this event.

Computationally, the ERP is detected by extracting EEG epochs time-locked to the stimulus presentation and calculating the average over the epochs. The reason why averaging allows to separate ERPs from the background brain activity relies on some assumptions:

- The ERPs are time-locked to the event and have the same shape and latency across all the trials.
- The background brain activity can be approximated by a zero-mean Gaussian random process which is uncorrelated between trials and non time-locked to the event.

ERP waveforms in response to sensory (or cognitive) events usually consist of a number of peaks and deflections that can be qualitatively characterized by amplitude, latency and scalp distribution. The amplitude provides and index of the extent to which the event is perceived as natural whereas the latency reveals the activation timing.

ERPs have been successfully used in BCIs for anticipating emergency breaks [32] [15] [33] and lane change intentions [34]. They have been widely investigated also to increase accuracy and information transfer rate. For instance they have been used to correct user's erroneous decisions [35] [36] or to re-calibrate a BCI system in a reinforcement learning fashion [37] allowing it to 'learn from its mistakes'. All these experiments focus on the analysis of Error-related Potentials (ErrP), which are a specific subset of ERPs elicited by the perception of erroneous events in form of expectation mismatch.

2.2.2 ERD/ERS

Sensory and cognitive processing and motor behavior result not only in event-related potentials (ERPs), but also in changes in the sensorimotor rhythms (SMR), which are μ-band rhythms (a subset of the α-band rhythms recorded above the sensory-motor cortical area), in some circumstance with a β-band accompaniment, that can be measured over the sensorimotor cortex.

The μ and β rhythms are commonly considered as EEG indicators of motor cortex and adjacent somatosensory cortex functions [38]. In fact, during the real or imagery movement of a limb, a prominent attenuation of ongoing μ rhythm can be observed over the rolandic area on the contralateral hemisphere [39]. In particular, during motor execution, the initially contralateral ERD develops a bilateral distribution [40], whereas during motor imagery this ERD remains mostly limited to the contralateral hemisphere. This means that the suppression of μ and central β rhythms is more pronounced at the contralateral hemisphere when subjects imagine one-sided limb movements than when they actually perform such movements.

This SMR attenuation is termed event-related desynchronization (ERD), whereas the increase in SMR amplitude is termed event-related synchronization (ERS) [41] [42]. Both phenomena are time-locket but not phase-locked to the event and they are highly frequency band specific. An important feature of these patterns is that they are somehow related to the body map on the sensorimotor cortex. For example, the left hand and right hand produce the most prominent ERD/ERS pattern in the corresponding hand area in the contralateral sensorimotor cortex [43]. In this regard, Figure 2.6 shows the homunculus, the famous pictorial representation of the neurological "map" of the areas and proportions of the brain dedicated to processing motor functions, or sensory functions, for different parts of the body. It is an approx-

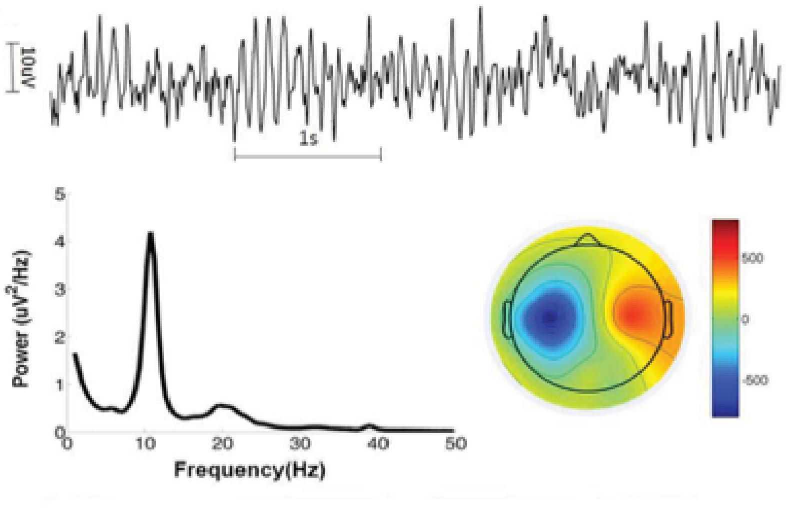


Figure 2.5: ERD in the sensorimotor area associated to the right upper limb. Source: [5]

imate and outdated representation, and in fact despite hundreds of studies it hasn't been discovered a unique layout able to discriminate sensory-motor functions with reliable generalization, but it still gives a coarse idea of the phenomenon.

Most of the BCI systems aimed at classifying single-trial EEGs during motor imagery are based on SMR, and more specifically on characteristic ERD/ERS spatial distributions corresponding to different motor imagery states, such as left-hand, right-hand, or foot movement imagination.

2.2.3 Common Spatial Patterns

Common Spatial Patterns (CSP) has proved to be one of the most effective techniques for detecting ERD/ERS [43] [44], and in fact it has been successfully adopted in some of the most important BCI competitions [45] [46]. It is a data-driven subject-specific spatial filter technique used in BCI systems as feature extraction method.

Given two distributions in a high-dimensional space, it determines spatial filters that maximize variance for one class and that at the same time minimize variance for the other class, assuming that the variance is an estimator of the contained information. In fact, after having bandpass filtered the EEG signals in the frequency domain of interest, high or low signal variance re-

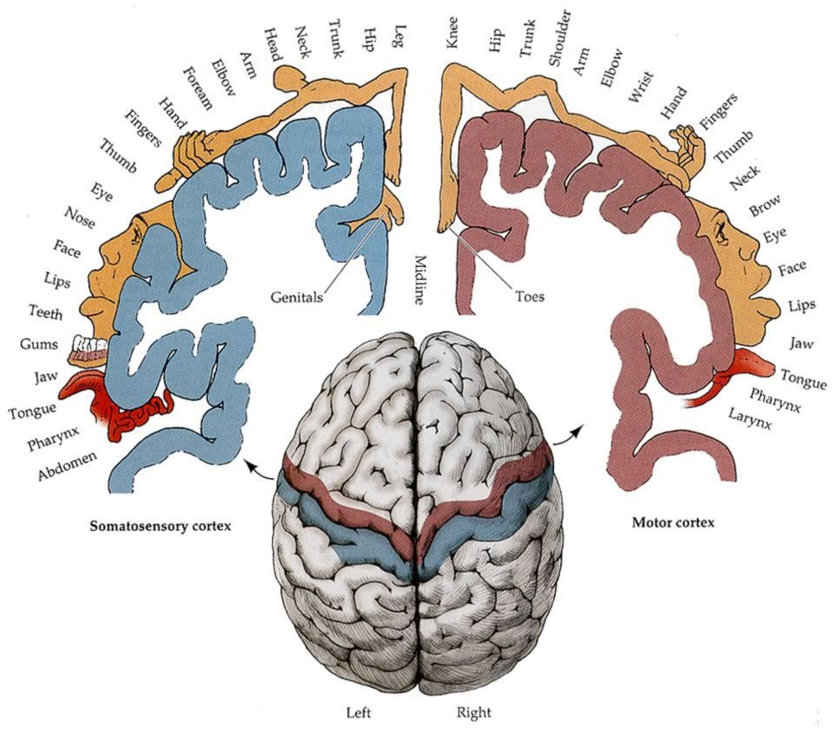


Figure 2.6: Homunculus. Source: [6]

flects respectively strong or attenuated rhythmic activity. As a consequence, if the EEG signals are bandpass filtered in the sensorimotor frequency band (7-30Hz) before applying the CSP algorithm, then the resultant spatial filters are those maximizing the separability of the two classes based on ERD/ERS.

CSP computation

Let's denote the EEG data of a single trial as the $N \times T$ matrix $\mathbf{E}$, where N is the number of channels and T is the number of samples of the trial. For each of these matrices, the normalized spatial covariance can be calculated as

$$\mathbf{C} = \frac{\mathbf{EE}'}{trace(\mathbf{EE}')} \tag{2.3}$$

where $'$ denoted the transpose operator and $trace(x)$ is the sum of the diagonal elements of x. The spatial covariances $\mathbf{C_1}$ and $\mathbf{C_2}$ are calculated by averaging over the trials of the corresponding class, and the composite spatial covariance matrix is computed as

$$\mathbf{X} = \mathbf{C_1} + \mathbf{C_2}. \tag{2.4}$$

As $\mathbf{X}$ is a symmetrical matrix, it can be factored into its eigenvectors by Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{C_1} + \mathbf{C_2} = \mathbf{U}\Sigma\mathbf{U}' \tag{2.5}$$

where $\mathbf{U}$ is the matrix of eigenvectors and Σ is the diagonal matrix of eigenvalues. Note that the eigenvalues are assumed to be sorted in descending order. The whitening transformation

$$\mathbf{P} = \sqrt{\Sigma^{-1}}\mathbf{U}' \tag{2.6}$$

equalizes the variances in the space spanned by $\mathbf{U}$, i.e., all eigenvalues of $\mathbf{PXP'}$ are equal to one. If $\mathbf{C_1}$ and $\mathbf{C_2}$ are transformed as

$$\mathbf{S_1} = \mathbf{PC_1P}' \quad \text{and} \quad \mathbf{S_2} = \mathbf{PC_2P}' \tag{2.7}$$

then $\mathbf{S_1}$ and $\mathbf{S_2}$ share common eigenvectors, i.e.,

$$\text{if} \quad \mathbf{S_1} = \mathbf{B}\Sigma_1\mathbf{B}' \quad \text{then} \quad \mathbf{S_2} = \mathbf{B}\Sigma_2\mathbf{B}' \quad \text{and} \quad \Sigma_1 + \Sigma_2 = \mathbf{I} \tag{2.8}$$

where $\mathbf{I}$ is the identity matrix. Since the sum of the two corresponding eigenvalues is always one, the eigenvector with largest eigenvalue for $\mathbf{S_1}$ has the smallest eigenvalue for $\mathbf{S_2}$ and vice versa. This property makes the eigenvectors $\mathbf{B}$ useful for classification of the two distributions. The projection of whitened EEG onto the first and last eigenvectors in $\mathbf{B}$ (i.e., the eigenvectors corresponding to the largest Σ_1 and Σ_2) will give feature vectors that are optimal for discriminating two populations of EEG in the least squares sense.

With the projection matrix $\mathbf{W} = \mathbf{B'P}$, the decomposition (mapping) of a trial $\mathbf{E}$ is given as

$$\mathbf{Z} = \mathbf{WE}. \tag{2.9}$$

The columns of $\mathbf{W^{-1}}$ are the common spatial patterns and can be seen as time-invariant EEG source distribution vectors, i.e., the weights associated to the channels. The first and the last columns of $\mathbf{W^{-1}}$ are the spatial patterns that maximize the difference of variance.

The feature used for classification are obtained by filtering the EEG according to (2.9) and taking the variance of a subset of the m signals most suitable for discrimination. The signals Z_p $(p = 1 \ldots 2m)$ that maximize the difference of variance of the two tasks are the ones that are associated with the largest eigenvalues Σ_1 and Σ_2. These signals are the m first and last rows of $\mathbf{Z}$ by construction of $\mathbf{W}$. The features f_p associated to one trial are computed as log-variance of these signals, where the log transformation is used to approximate normal distribution of the data.

$$f_p = \log\left(\frac{var(\mathbf{Z_p})}{\sum_{i=1}^{2m} var(\mathbf{Z_i})}\right). \tag{2.10}$$

2.2.4 Improvements of classical CSP: xCSP

Since the first usage of CSP as feature extraction method to discriminate movement-related patterns in EEG [47] [44], many improved CSP-based methods have been put forward to enhance the classification accuracy.

By construction, the original version of CSP is sensitive to noise and prone to overfitting. To address this issue, in [48] Lotte and Guan proposed a list of regularized variants that are referred to as RCSP . These approaches are based on the addition of prior information to CSP and they differ in the location where the prior is inserted. One of the presented possibilities is at the covariance matrix estimation level and the other is at the objective function level. In the former case the regularization term is added to shrink the covariance matrix toward a prior on how the covariance matrix for mental state considered should be, while in the latter it is added in order to penalize solutions (i.e., regularizing spatial filters) that do not satisfy the given prior. Other studies approached the regularization aspect introducing the transfer-learning strategy into the classical CSP [49] [50] [51]. Samek et al., instead, proposed a method called stationary CSP (sCSP) which regularizes the CSP solution towards stationary subspaces; that is, the CSP is extended to be invariant to non-stationarities in the data [52]. It reaches the goal by reducing variations of the extracted features assuming that the variations are not task-related like eye movements or electrode artifacts. Yong et al. proposed a modified version of the RCSP [48] whereby the classical covariance estimates are replaced by the robust covariance estimates obtained using the Minimum Covariance Determinant (MCD) estimator [53]. The same goal of improving the robustness of the covariance matrices estimation was tackled by Zhang et al. in [54] by introducing local temporal correlation (LTC) information in the estimation (LTCCSP).

To address the problem of selecting the subject-specific frequency band for the CSP algorithm, Lemm et al. [55] proposed the Common Spatio-Spectral Pattern (CSSP) algorithm in which the filter is constructed by the method of time-delay embedding. The idea was improved first by Dornhege et al. in [56] with the Common Sparse Spatio-Spectral Pattern (CSSSP), and then by Tomioka et al. in [57] with the Spectrally-weighted CSP (SpecCSP) which simultaneously optimizes the filter in the frequency domain and the spatial filter in an iterative procedure. A more lightweight approach to select the optimal frequency and time range was proposed by Ang et al. in [58] with the Filter Bank CSP (FBCSP) algorithm.

2.2.5 Deep learning

One of the challenges in mapping mental states to EEG data is finding representations that are invariant to inter- and intra-subject variations, as well as to inherent noise associated with EEG signals. Over the last decade it has been proven that Deep Learning techniques are effective in solving similar issues, in fact they have become the state of the art in domains affected by the same issues like computer vision, speech recognition and natural language processing.

Some studies have in fact applied recent advances in Deep Learning techniques to EEG data with promising results. The used approaches can be divided in two categories based on the adopted input representation: one category uses directly the raw EEG signals, while the other converts the EEG time series in 2D images by means of the Fast Fourier Transform (FFT) in order to project the problem into a domain where deep networks are known to perform well, namely computer vision.

An example of the latter is the work proposed by Tabar and Halici [59], where the combination of a Convolutional Neural Network (CNN) and a Stacked Autoencoder (SAE) is used for binary (left vs right) motor imagery classification. In this work the inputs of the deep network are 2D time-frequency representations of the EEG time series. A similar approach is used in [60] where the spectral power of each channel is used to construct topographical maps (2D images) that are used as input of a recurrent-convolutional network. Na Lu et al. [61] proposed an architecture named Frequential Deep Belief Network (FDBN) based on FFT and stacked Restricted Boltzmann Machines (RBMs). The statistically significant performance improvement achieved by the proposed solution over the state of the art (FBCSP) suggests that frequency domain input to DBN can lead to much better performance on motor imagery classification than the raw EEG time series.

Regarding the other category, Stober et al. [62] propose a deep network that takes as input the raw EEG signals. The core of the proposed architecture, named Hydra-net, is composed by Convolutional Auto-Encoders (CAEs) and the peculiarity is that the network is trained to encode both cross-trial constraints (identification of features that are stable across trials of the same class) and relative similarity constraints (identification of features that allow to distinguish between classes by demanding that two trials from the same class are more similar to each other than to trials from other classes). Also the works proposed by Schirrmeister et al. [63] and Lawhern et al. [64] consider raw EEG data as input of the respective CNNs. The two studies employ classical CNNs and work on the same dataset (dataset 2A from BCI competition IV) consisting of a four-classes (left, right, feet and rest) motor imagery task.

The following section briefly describes notions and peculiarities related to CNNs in order to better understand the motivations that led us to their adoption for discriminating “move” vs “rest” states in this work. A more detailed description of the model will be presented in chapter 4.

Convolutional Neural Networks

Convolutional Neural Networks (CNN) [65] [66] [67] are a type of feed-forward multi-layer neural networks with several alternations of convolution and pooling layers and a fully connected layer at the end. Feed-forward networks pass an input through one or more layers of neurons until it reaches the output layer. In each neuron the linear combination of its inputs is passed through a, typically non-linear, activation function and the result is forwarded to the neurons of the next layer. Common activation functions are sigmoid $f(x) = \frac{1}{1+e^{-x}}$, tanh $f(x) = \frac{2}{1+e^{-2x}} - 1$, rectified linear unit (ReLU) $f(x) = max(0,x)$ and exponential linear unit (ELU) $f(\alpha,x) = \begin{cases} \alpha(e^x - 1) & x < 0 \\ x & x \geq 0 \end{cases}$

CNNs can learn highly non-linear features (through convolutions and non-linear activation functions) and represent higher-level features as composi-

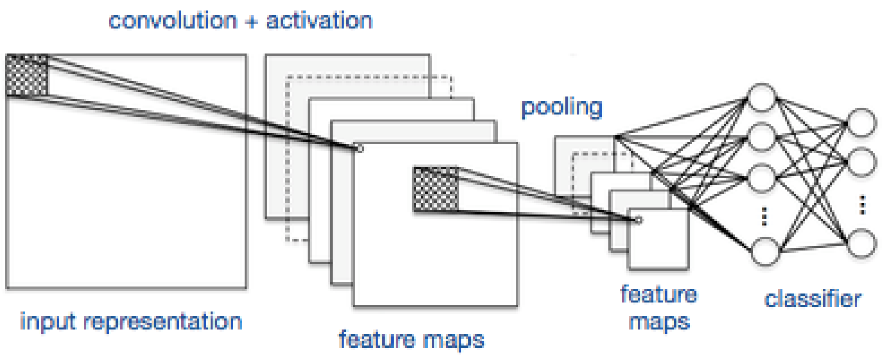


Figure 2.7: Convolutional Neural Network. Adapted from:[7]

tions of lower level features (through multiple layers of processing). In addition, the pooling layers create coarser intermediate feature representations that can make the model more translation-invariant.

Network training is usually performed through the combination of stochastic gradient descent and back-propagation to adjust weights and convolution filters by minimizing an objective function following the gradient. Many improvements on the basic stochastic gradient descent algorithm have been proposed, especially for avoiding to get stuck at local minima. Two commonly used variants are stochastic gradient descent with momentum [68] and ADAM (Adaptive Moment Estimation) [69].

Deep neural networks such as CNN are inherently affected by overfitting, i.e., the production of a model that explains too closely the available data and may therefore fail to explain additional data. It is a common problem in deep neural networks because it usually derives from the fact that the model contains many more parameters than can be justified by the available data. Some commonly used approaches used to address this issue are early stopping, i.e., stopping the training process before the learner's ability to generalize starts to decrease, regularization, i.e., introducing constraints into the objective function to force the model to be simpler, and dropout [70], i.e., randomly drop units (along with their connections) from the neural network during training so that to prevent co-adaptation. The most widely used regularization constraints are weight decay and sparsity constraint.

A CNN is a sequence of layers, each of which transforms one volume of feature maps to another through a differentiable function. The three main types of layers are: convolutional, activation, and pooling. In a convolutional layer, filters are slid over an input feature map with a certain stride, i.e., shifted by a certain number of pixels, and for every position the addition of the element wise multiplications between the two overlapping matrices forms a single element of the output map. Filters act as feature detectors, in fact different values of the matrix filter produce different feature maps for the same input. Activation layers, typically placed after convolution layers, have the purpose of introducing non-linearity and consist in the element-wise transformation of a feature map into another by means of one of the activation functions listed above. The function of pooling layers, instead, is to progressively reduce the spatial size of the input representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. A pooling layer reduces (downsamples) the dimensionality of each feature map while retaining the most important information. The most used pooling types are: *max*, *avg*, *sum*. In case of *max* pooling, the output is composed by the largest element of each (usually) non-overlapping subregions of the initial representation.

In essence, via the convolutional layers we aim to extract the useful features, via activation layers we introduce non-linearity to better explain the real world, and via the pooling layers, we aim to make the features somewhat invariant to scale, translation, distortion and rotation.

2.3 Portable infrastructures for BCI

As anticipated in the introduction, some studies regarding the design of portable BCI devices exist, but none of them is flexible and modular enough for our purpose. However two of them are worth mentioning because they employ promising technologies that could be further investigated in future.

Nurse et al. [17] proposed an inspiring work in which IBM's recently presented neuromorphic chip, named TrueNorth, is used to deploy Deep Learning models for offline self-paced motor execution decoding. Unfortunately the BCI has not been validated in real-time, which is the most critical context due to the high-frequency of the incoming brain signals.

Aravind and Babu [16], instead, used a FPGA to implement a basic BCI consisting of a Finite Impulse Response (FIR) filter, a Discrete Wavelet Transform (DWT) module for the feature extraction, and a Support Vector Machine (SVM) for classification.

Both systems consist of hard-coded modules implementing specific BCI components, and thus do not allow an easy prototyping for experimenting with different frameworks and methods. In addition they do not consider the possibility of acquiring signals from multiple devices for more complex BCI experiments.

As a consequence, these two platforms can be considered as interesting solutions mainly for final implementation, while the proposed infrastructure can be used both for continuous prototyping and implementation.

Chapter 3

Infrastructure

The main goal of this thesis is to design an *easy-to-use* infrastructure for *prototyping portable* BCIs able to operate in *real-time.*

As a consequence the aim of the proposed infrastructure is to support the designers in the implementation of BCIs on an embedded architecture where both the pipeline that makes up the BCI and the platform on which it runs have to be powerful enough to operate in real-time.

Thus, a custom infrastructure, based on an embedded system, is proposed to support designers in implementing BCI systems, guarantying the needed computational power for real-time operations. However, the careful and optimized implementation of the proposed infrastructure allows for real-time use even on an embedded system, that make it easily portable and usable in a wide range of everyday scenarios. Moreover, the architecture was designed to be easily adopted by an expert of the BCI field, with only a very simple training for platform use. An efficient way to achieve this goal is to build up the architecture from basic "bricks" that are already widespread in the neuroscientific community, and that, therefore, don't require specific or particular training effort by the typical user of the system

Each of the mentioned objectives sets some constraints in the architecture of the proposed system. For instance, portability constraint forced us to integrate the whole platform on an embedded system, acting also as computing core of the system. NVIDIA Jetson TX2 was chosen among all the available embedded platforms, because it is compatible with the portability requirement of the architecture and, at the same time, its hardware provides the computational power needed to carry out sophisticated processing methods like artifact removal based on ICA and Deep Learning even in a real-time scenario.

Moreover, flexibility and modularity are two mandatory requirements of the proposed architecture, so we decided to adopt a system that can handle complex models like Convolutional Neural Networks (CNN). In this context the NVIDIA Jetson TX2 is an ideal solution because its GPU, with the CUDA libraries, can considerably speed up training and parallel execution of the majority of machine learning algorithms and prototyping frameworks like PyTorch. Thus the integration of these frameworks in the platform does not require any modification to the architecture or hardware enhancement.

Moreover, the proposed platform can be easily turned into an heterogeneous system, able to combine CPU and GPU to improve overall performances. It is a notable advantage for real-time processing, allowing an efficient workload partitioning between CPU (delegated to handle mainly incoming streams handling and synchronization) and GPU (that can take care of intense parallelizable real-time processing tasks).

This scenario shows that the selected embedded system is appropriate for the proposed infrastructure because its hardware features enable for an efficient lead balancing.

Other embedded solutions have similar features to the NVIDIA Jetson and satisfy the computational power constraints of the proposed architecture; for example, FPGA-based systems (e.g., Xilinx Zynq) posses all these characteristics. Although these systems can deliver better performances of general purpose architectures, usually they are much more expensive than the NVIDIA Jetson and require at least a good working knowledge of HDL languages (e.g., Verilog, VDHL), in order to be efficiently programmed. Moreover, high level languages and scripting languages (e.g., C++ or Python), very common in the BCI community, can be only partially compatible with this kind of embedded solutions; thus, porting of existing BCI systems could be an hard task. All this considered, the NVIDIA platform appears to be the best compromise for the aims of the proposed infrastructure.

Portability and usability on the field have guided also the choice of the acquisition devices (EEG headsets). In particular, we decided to adopt Bluetooth systems with dry electrodes (more specifically, Enobio [71] and OpenBCI [72]) because they simplify the deployment and the use of the system. However, this choice introduced several issues due to Bluetooth channel latency and reduced signal quality, that have to be faced and solved in architecture design and testing.

3.1 Design

The greatest part of the system to deploy BCI systems described in literature are made up of three main components: signal acquisition, signal processing and actuation.

With the aim of utilizing standards and best practices of the field as much as possible, we decided to keep the same structure in our architecture. For each of the three components we looked for the most popular tools and we aimed at building up a system that could take advantage of these tools, and could be implemented efficiently maintaining portability and ease of use.

For this reason in a first design phase, an efficient implementation of the three blocks was investigated with respect to organization, communication and technologies to use. For instance, the communication between components involved in signal processing and actuation takes place within the NVIDIA Jetson TX2 and is a simple exchange of data (not even complex from the point of view of the the bandwidth), whereas the communication between signal acquisition and signal processing is much more complex. Moreover, in this phase, for each component we identified alternative technologies and selected the best ones according to architecture requirements. These choices have introduced several challenges, mainly related to the real-time behavior, that depends on three main aspects:

- amount of data to process (multiple multi-channel high-frequency streams);
- choice of using Bluetooth acquisition systems to achieve portability and flexibility for the usage on the field;
- complex processing pipeline (with potential demanding computations like artifact removal and Deep Learning) in conjunction with real-time actuation constraint.

The Bluetooth transmission issue has been analyzed and partially characterized in the infrastructure validation presented in Section 3.2. Nevertheless, as we demonstrate in Chapter 4, the infrastructure is able to operate in real-time and this means that latencies introduced by Bluetooth communication are efficiently circumvented. This, of course, holds for the specific paradigm considered, namely motor imagery/execution, because it doesn't require a time synchronization out of reach of Bluetooth and LSL since it involves only two simultaneous streams. In particular, the two streams are the EEG time series and the stream of markers associated with the stimuli presented by the stimulus presentation system.

Another aspect that we had to consider in the design phase is that, having to train offline the models used online by the component that deals with the signal processing, it has been necessary to integrate in the architecture a component that would allow the presentation of stimuli and feedback for labelling the acquired data.

3.1.1 Signal Acquisition

As mentioned earlier, we opted for wireless devices with dry electrodes to meet the requirements of portability and field usability. This choice introduced the issue of the latencies typical of the Bluetooth transmission channel. In general, however, a cabled system would bring better performances both in terms of achievable transmission frequency and introduced latencies, but at the same time it wouldn't allow the same portability and usability in a mobility scenario. Moreover, wireless transmission eliminates the noise introduced in the signals by movements of the wires.

As for the choice of the embedded system, also for the acquisition devices a relevant point is their price. In general, wireless devices are more expensive due to the more complex technologies adopted to guarantee reliability in the wireless data transmission. Moreover, the processes to obtain the compactness indispensable for the portability of the device increases its price. Despite being more expensive, we have used wireless devices because they are crucial to achieving the predetermined portability goal.

To the same extent, the choice of adopting dry electrodes has brought a slight reduction of the acquired signals quality, while has produced an increased usability, consistently reducing the preparation time and the discomfort due to the usage of the conductive gel. Also in this case the price is

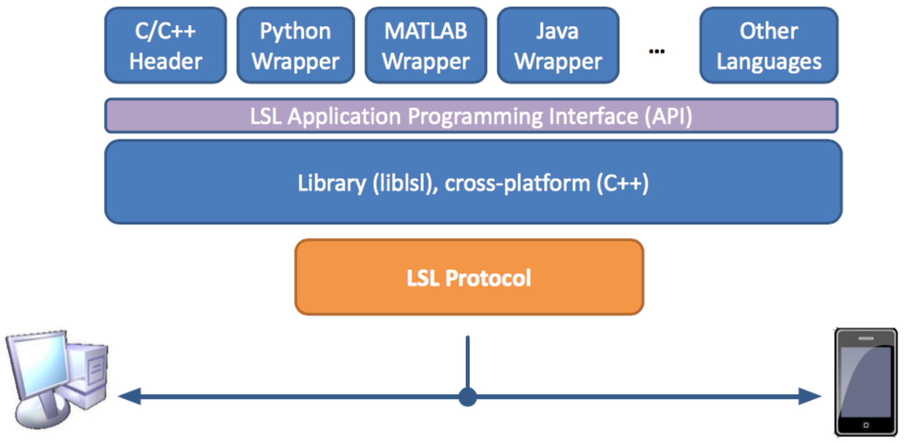


Figure 3.1: Lab Streaming Layer core components. Source:[8]

generally higher for the adopted typology, namely dry electrodes.

Enobio [71] has been used as main acquisition device; it is a research-grade commercial Bluetooth device with 20 channels to which we attached high-end dry electrodes produced by the same vendor. In addition, we have also roughly investigated the performance of OpenBCI [72] to evaluate results achieved by an open-hardware low-end device. The preliminary results confirmed that the Enobio produces considerably better signals, hence we've postponed further investigations for future developments.

Synchronization of signals produced by different devices is a fundamental issue in data acquisition. The proposed architecture aims to be as general as possible and therefore it has to allow also complex acquisition sessions involving multiple devices. To achieve this goal it is necessary to re-align all the involved streams over a unique time basis. We identified two main tools: Lab Streaming Layer (LSL) [8] and Tools for Brain-Computer Interaction (TOBI) [73] [74]. We adopted LSL because it has significant advantages over TOBI: firstly, unlike TOBI, it implements synchronization natively; secondly it is general-purpose and cross-platform with interfaces for the most used programming languages like C, C++, Python, Java, C# and Matlab; and thirdly it is highly employed in the BCI field so it guarantees modularity with respect to the BCI ecosystem.

Lab Streaming Layer

Lab Streaming Layer (LSL) is an open-source library for the unified collection of time series that handles both networking and time-synchronization of the data, providing tolerance to disconnections and intermittent network failures. Data is transmitted via TCP and the streams are made available to all the devices connected to the same network.

Networking is handled by two type of components: outlets and inlets. Stream outlets are responsible for making time series data available on the network. The data is pushed sample-by-sample or chunk-by-chunk into the outlet, and can consist of single- or multi-channel data, with regular or irregular sampling rate. Streams have XML meta-data attached containing information like ID, type of signal, name and sampling rate. Stream inlets connect to the outlets for receiving the time series data. The library guarantees that the samples are received in order relying on the TCP reliable message-oriented transmission. Streams are uniquely identified by the ID for recovery purposes, but can also be resolved by name or type. This architecture provides an abstraction mechanism to mask attached hardware and reduces the clients burden to exactly know connected hardware characteristics and address. A theoretically infinite number of clients (inlets) can connect and pull data from a server (outlet). In addition, data is buffered both at the sender and receiver side with configurable and arbitrarily large buffers to tolerate intermittent network failures.

Time synchronization is mandatory to analyze multi-modal biosignals, because streams typically come from different devices, whose clock are likely out of phase with respect to each other, so a realignment with respect to a unique timeline is needed in order to extract time-dependent information. This is a key concern in EEG experiments because EEG analysis is typically time-locked with respect to specific points in time, such as a key press or the display of an image on a computer screen. If the timing of these events cannot be realigned with the EEG data, this may cause the loss, reduction, or blurring of the measured quantities triggered by the events. For instance, Figure 3.2 shows the outcome of the synchronization procedure in an experiment involving an EEG headset streaming at 500Hz, an accelerometer with a sampling rate of 50Hz, and a program presenting visual stimuli at

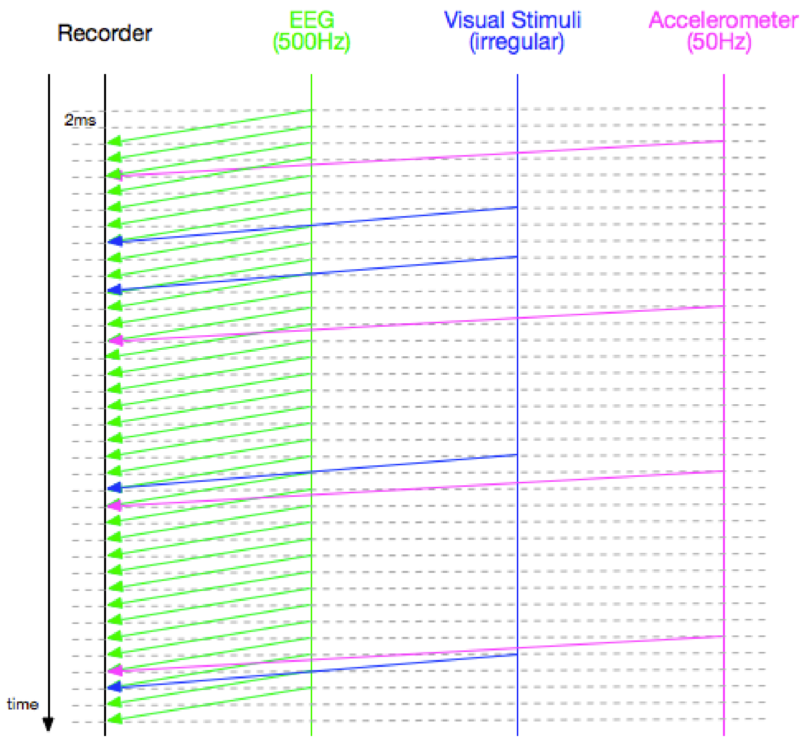


Figure 3.2: Multi-modal time synchronization

irregular sampling rate. After re-alignment all streams had been remapped onto the same time base.

The built-in time synchronization in LSL relies on two pieces of data that are collected with the actual sample data:

- A timestamp for each sample that is read from a local high-resolution clock of the computer.
- Out-of-band clock synchronization information that is transmitted along with each stream. This information consists of periodic measurements (every few seconds) of the momentary offset between sender's and receiver's clock.

The clock resolution is less than a microsecond on practically all consumer PCs and this allows LSL to achieve 50 ms synchronization accuracy with minimal effort and 1 ms accuracy with effort. The effort needed to achieve these performance corresponds to the proper design of the local network, the adoption of machines with enough computational power and, above all, the accurate prior estimation of the mean constant bias of each signal source involved in the acquisitions. This figure may be due to a constant drift in the device or to latencies in the transmission channel between the device and the machine starting the LSL stream (see red arrows in Figure 3.7). This constant bias needs to be subtracted to each timestamp after the LSL synchronization because it cannot be accounted for since it is not related to the transmission channel under its control. In general, to achieve high accuracy an estimation of all the latencies in the signal acquisition process is needed.

The clock offsets are periodically measured using an algorithm deriving from the Network Time Protocol (NTP). Each offset measurement involves a brief sequence of n round-trip UDP packet exchanges between the two involved computers. Each packet exchange yields an estimate of the Round-Trip Time (RTT) and an estimate of the clock offset at the time of the measurement with the RTT factored out. The final offset estimate for the given measurement is the one out of the n measured which is associated to the minimum RTT (to achieve a high chance of using only estimates that were acquired in nominal packet transport conditions and with minimal effect of transmission delays).

The clock offset θ is defined as

$$\theta = \frac{(t_1 - t_2) + (t_2 - t_3)}{2}, \tag{3.1}$$

and the round-trip time δ as

$$\delta = (t_3 - t_0) - (t_2 - t_1), \tag{3.2}$$

where

- t_0 is the master timestamp of the request packet transmission,
- t_1 is the slave timestamp of the request packet reception,
- t_2 is the slave timestamp of the response packet transmission,
- t_3 is the master timestamp of the response packet reception.

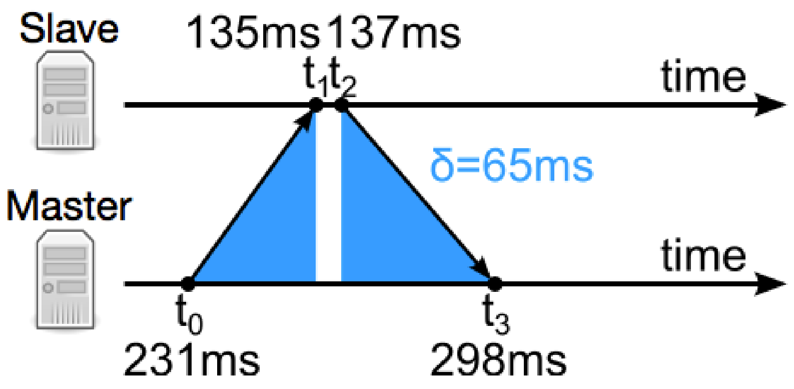


Figure 3.3: Round-trip delay time δ. Adapted from:[9]

These estimates are correct when both the incoming and outgoing routes between the master and the slave have symmetrical nominal delay. If the routes do not have a common nominal delay, a systematic bias equal to half the difference between the forward and backward travel times is present [9]. On a symmetric local area network, the travel time differences are the sum of random delays due to multi-threading, buffering or interruptions and the systematic difference in the latency of the two involved network stacks. Since the algorithm reduces the probability of the random effects on estimates, the residual inaccuracy is dominated by the systematic difference between the latencies of the two network stacks (e.g. Linux vs. Windows), which can be expected to be reasonably less than a millisecond on modern hardware and operating systems.

Given a history of clock offset measurements between a master (machine acquiring streams coming from different streaming devices) and a slave (streaming device), the timestamp of a sample collected at the slave side can be remapped into the time domain of the master in order to be comparable with the samples collected from the other slaves involved in the multi-modal acquisition. In our architecture (see Figure 3.7) the master is the *LabRecorder* and the slaves are the programs streaming data acquired by the EEG headsets and the accelerometers.

The synchronization performance is optimal when performed offline because the history of all clock offset measurements of the session is available; this means that an accurate statistical estimation of the correction value for

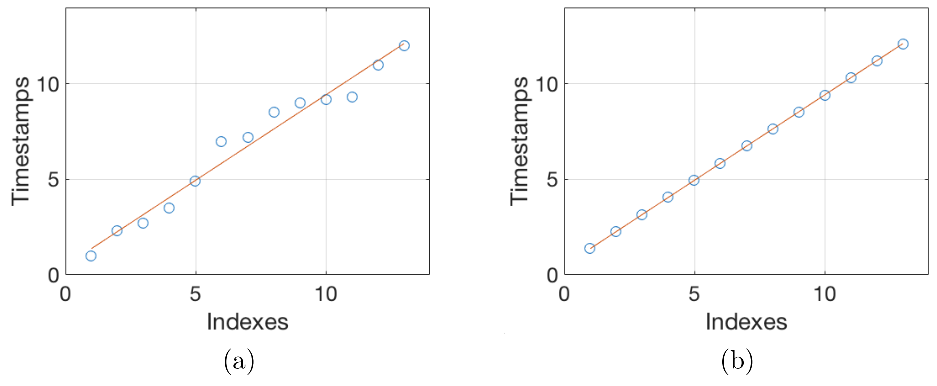


Figure 3.4: Illustration of the offline dejittering process: (a) Before. (b) After.

each time point is made considering also future measurements. If performed online only a subset of previous measurements is available and therefore the performance is suboptimal. The offset correction can be performed by a (robust) linear regression through the available clock offsets to estimate the coefficients a and b to compute the correction values necessary to remap timestamps. The linear regression attempts to model the relationship between correction values (dependent variable $\mathbf{Y}$) and clock offset measurements (explanatory variable $\mathbf{X}$) by fitting a linear equation to observed data

$$\mathbf{Y} = a + b\mathbf{X} \tag{3.3}$$

The correction values are added to the original timestamps to account for the clock offset progress between the two devices.

$$timestamps = timestamps + \mathbf{Y} \tag{3.4}$$

To apply the clock offset correction and the method to use is optional, however, by default it is performed through a robust linear regression minimizing the Huber loss function by the Alternating Direction Method of Multipliers (ADMM) algorithm [75].

In addition to the clock offset, also the jitter in the timestamps must be accounted to obtain a precise multi-modal alignment. Anyway, a jitter (e.g., Gaussian noise) will always be present in timestamps because the sampling is not at fixed rate but on a somewhat random schedule dictated by hardware, drivers and operating system.

Also the dejittering step is optional and can be either online at the time of data collection, or offline once all the data of the session have been collected. It consists in a trend-adjusted smoothing algorithm which performs a re-calculation of the timestamps of the streams that have regular sampling rate. When performed offline (see Figure 3.4), it simply calculates a linear fit between the index of each sample and its timestamp (thus assuming a constant but arbitrary effective sampling rate) and then re-calculates from it the timestamps of all samples based on their respective index.

Instead, to smooth the timestamps online, the Recursive Least Square (RLS) algorithm is used, which updates the regression coefficients sample by sample every time a chunk is pulled. Timestamps of a chunk are adjusted according to the following equation:

$$timestamps = \theta'\phi, \tag{3.5}$$

where

$$\phi = \begin{bmatrix} 1 & \dots & 1 & 1 \\ index_{n-p} & \dots & index_{n-1} & index_n \end{bmatrix},$$

$$t = \begin{bmatrix} timestamp_{n-p} & \dots & timestamp_{n-1} & timestamp_n \end{bmatrix},$$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}.$$

Elements of θ are the coefficients of the regression from sample index to timestamp, and more specifically the intercept (θ_1) and the slope (θ_2) of the linear mapping.

At each sample i of the chunk, the coefficients θ are update according to the following recursive formulas:

$$x_i = \begin{bmatrix} 1 \\ index_i \end{bmatrix},$$

$$u_i = \begin{bmatrix} timestamp_i \end{bmatrix},$$

$$e_i = u_i - \theta'_{n-1}x_i, \tag{3.6}$$

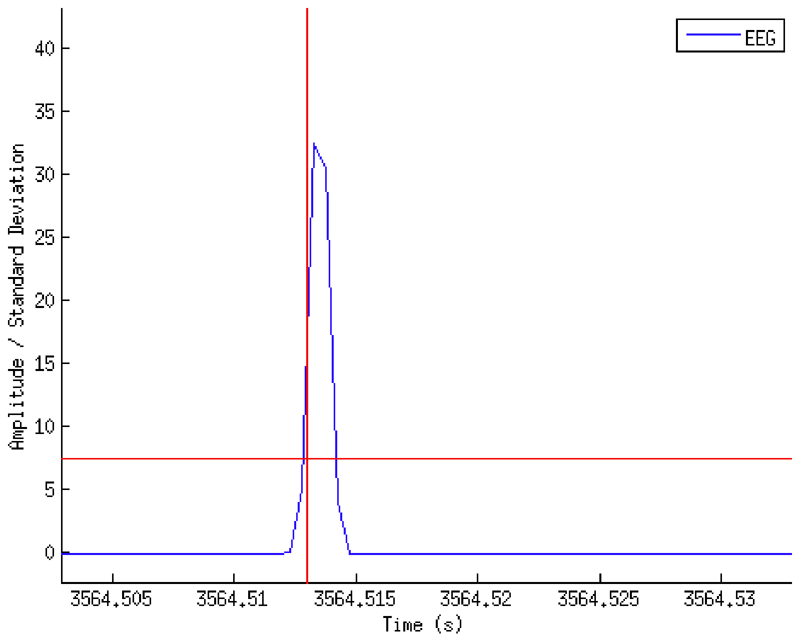


Figure 3.5: Software and hardware events. The vertical line is the software event, while the blue spike is the hardware event. The horizontal red line is the trigger of the software event based on the threshold. Source:[10]

$$k_i = x_i' P_{i-1}(\lambda + x_i' P_{i-1} x_i)^{-1}, \tag{3.7}$$

$$P_i = \lambda^{-1}(P_{i-1} - k_i x_i' P_{i-1}), \tag{3.8}$$

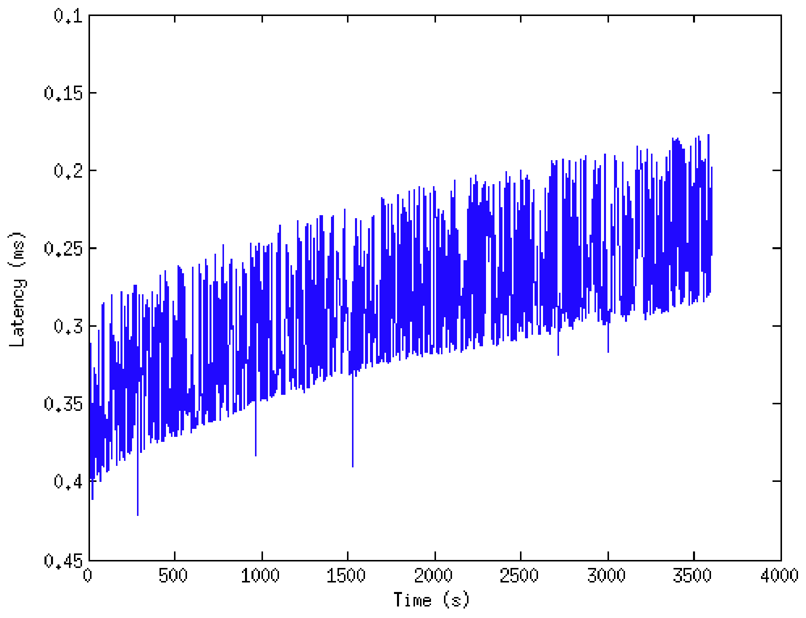
$$\theta_i = \theta_{i-1} + k_i e_i, \tag{3.9}$$

where λ is the tunable forgetting factor.

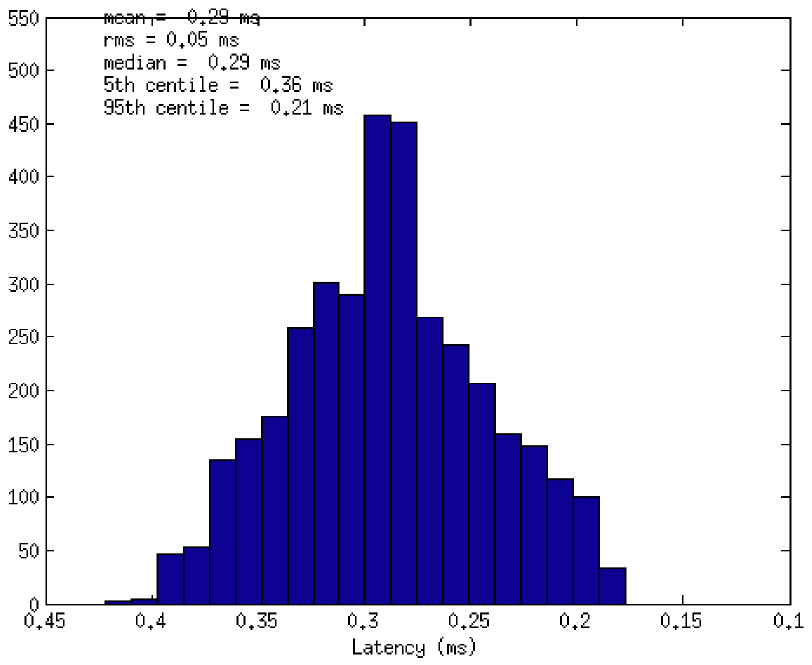
LSL validation

The authors of the library provide an experimental validation of the synchronization capabilities: events were streamed via LSL in a network every time an hardware event was triggered by a research grade data acquisition card (National Instrument PCI-6230) running at 10kHz.

In the protocol, the hardware and software events were output by a computer A and acquired by a computer B. In particular, the hardware events were output by the NI-6230 in computer A and acquired by an analog input box attached to computer B. In addition, the hardware events were fed back into an input channel of the NI-6230 itself.



(a)



(b)

Figure 3.6: LSL validation results. Adapted from:[10] (a) Latency (software event time - hardware event time) over time. (b) Histogram and statistics of the latencies related to the whole validation session.

Once a certain threshold was reached, generated and streamed the software events via LSL over the network from computer A to computer B.

Figure 3.6 (a) shows the latency (software event time - hardware event time) over time, while Figure 3.6 (b) shows the histogram of the latencies and statistics of the whole experimental campaign. Figure 3.6 (b) shows that the jitter range is 0.36 to 0.21 ms, with a mean of 0.29 ms. This means that the uncorrectable synchronization between a continuous LSL stream and an irregular LSL stream is less than 1 ms in optimal conditions. Figure 3.6 (a) shows that some sporadic outliers can occur. They are due to delays introduced by OS in software events because of scheduling policies. In addition a drift can be clearly observed, caused by statistical errors in the synchronization or by variations in clock frequency due to temperature fluctuations.

It is worth noting that these results were achieved after the correction of the mean constant bias of the acquisition device. As previously anticipated, this figure needs to be estimated and subtracted to each timestamp after the LSL synchronization because it cannot be estimated by LSL since it is not related to the transmission channel. If it is not considered, it results in a constant drift that is summed to the uncorrectable one.

These positive results are confirmed by the experimental validation of LSL proposed by the vendor of the Enobio in [76] where a mean synchronization bias of 0 ms with a 2 ms jitter is achieved after compensation of the mean constant bias of the acquisition device.

LSL integration

Lab Streaming Layer is the core component of the proposed infrastructure, indeed everything relies on the realignment of streams originating from different devices connected to the same network; thus, LSL has been integrated in most of the components of the infrastructure.

First of all, it was compiled for the ARM architecture in order to be able to use the library both on the NVIDIA Jetson TX2 and on Android devices. Since the infrastructure aims at the acquisition of multi-modal biosignals in mobility, we also developed an LSL-capable Android application in order to acquire (via Bluetooth) and stream (over LSL) accelerometer and gyroscope data. We also refactored and ported onto the NVIDIA Jetson TX2 the LSL acquisition program (LabRecorder) which acts as master in the acquisition process by collecting and storing all the incoming streams along with their histories of measured clock offsets. More specifically, we improved the streams identification to allow multiple streams of the same kind during the same session. This is mandatory, for instance, when multiple accelerometers (thus, multiple streams of the same type) are involved. Moreover, we enhanced LabRecorder with a system to handle actions scripts to automatically instantiating and starting the communication among all programs, possibly running on different devices, involved in a session. This communication represents the Inter-Process Communication (IPC) mechanism adopted in the infrastructure for achieving synchronization among the different programs and is generally based on sockets, publish-subscribe messaging protocols like MQTT (Message Queue Telemetry Transport) or by LSL streams themselves. This automatic synchronization is particularly important in sessions characterized by a causal dependence between different softwares. For instance, it occurs at the beginning of the trials triggered by the program that presents the visual stimuli need to be communicated to the BCI models running on other devices in order to erase the accumulated classification probabilities. A simpler and more common example is when the program that deals with the aggregation of the models predictions needs to send commands to an actuator (e.g., a robot) in order to execute the desired actions.

3.1.2 Signal Processing

From the study of the literature, several frameworks have emerged for the design of BCIs. Some already implement most of the state-of-the-art algorithms and therefore we considered them for the choice of the components to integrate in the infrastructure. In particular, the frameworks most used by BCI experts turned out to be BCI2000 [77], OpenViBE [78] and BBCI [79], BCILAB [80], MNE [81], BrainDecode [63] and Wyrm [11].

Since this thesis aims at designing a system for prototyping BCIs that has to be easily usable by experts of the field, we decided to rely on these off-the-shelf libraries in such a way to facilitate the adaptation to the infrastructure, and at the same time to allow the easy porting of existent projects that already make use of these libraries. An elicitation process has been performed through a study of the different use cases to understand which of these frameworks are more suitable for our purposes, i.e., real-time capabilities and easy prototyping. The choice has fallen on three real-time capable frameworks because of their manifold characteristics, which are:

BCILAB, Wyrm and BrainDecode.

BCILAB is a very complete general-purpose library that has been developed by the same authors of LSL. It is written in MATLAB, and therefore it is affected by disadvantages like cost (being MATLAB a commercial software) and inability to be directly integrated on the NVIDIA Jetson TX2. For this reason its integration would require additional hardware, a solution that clashes with the portability requirement.

Wyrm is, instead, a very simple and intuitive low-level library written in Python, that therefore can be directly integrated on the NVIDIA Jetson TX2. Moreover it can take advantage of other Python libraries for signal processing (for instance, MNE [81]) and machine learning (for instance, scikit-learn [82]).

Brain Decode is a very recent library, also written in Python, that implements various deep architectures based on Convolutional Neural Networks (CNN) that are present in the literature. It has been chosen to investigate the automatic feature extraction capabilities of Deep Learning methods applied to EEG because it is one of the most promising as it takes directly in input the raw EEG signals instead of the usual image representation used in most of the other Deep Learning frameworks.

An important aspect regarding this choice is that, being both written in Python, Wyrm and BrainDecode can be used together.

The actual choice had fallen since the beginning on Wyrm because it has all the characteristics needed by the infrastructure. We still decided to test it against BCILAB (which is the state of the art in terms of BCI prototyping) for approaching the BCI subject and, at the same time, for making a comparison of the different results. It has emerged that methods implemented offline with BCILAB could not be used in real-time because of conflicts in signal acquisition performed by the framework. In particular, latencies introduced in data collection produced distortions that prevented the correct signal filtering and thus reliable classification results. In addition, it has emerged also that the performance achieved by CSP models trained with Wyrm are generally higher than those trained with BCILAB, and this is due to different implementations and default values in the algorithm. These two aspects justify even further the choice of adopting Wyrm as BCI framework.

	Wyrm	**BCILAB**
Generality	Limited - only a few specific methods are natively implemented	Extensive - many state-of-the art algorithms are implemented
Extensibility	High - can make use of any Python library	High - can be extended with MATLAB scripts
User-friendliness	Limited - the experiments have to be designed as Python scripts	High - the experiments can be run with a GUI
Online behavior	Solid - it is possible to effectively run online experiments and there is room for complex computations	Inadequate - insufficient performance for real-time use
Portability	High - written in Python then it can be ported onto most of the computer architectures	Limited - written in MATLAB then it cannot be ported on ARM architectures (in fact, it cannot be directly ported onto the NVIDIA Jetson)
Cost	Low - open-source and can run on any platform supporting Python	High - requires MATLAB, which is a costly commercial product

Table 3.1: BCI prototyping frameworks comparison

3.1.3 Actuation

As anticipated, the infrastructure has to offline train the models to use online, so we integrated a component that allows to present audio-visual stimuli to label acquired data. Moreover, in the literature there are examples of BCI protocols with feedback and others without feedback, therefore the included component must be able also to present feedbacks. For this reason we decided to rely on PsychoPy [18] which is one of the most widely used frameworks in the neuroscience community for this purpose. It is a free open-source library allowing to run a wide range of neuroscience, psychology and psychophysics experiments, written in Python, so it has all the aforementioned advantages over the corresponding MATLAB counterparts. Besides PsychoPy, some more advanced packages exist, such Unity and Blender, that are used to create complex and involving stimulation protocols, for instance by means of the Virtual Reality (VR).

To get a more realistic user interaction during the real-time experiments we implemented the communication with a LEGO MINDSTORMS EV3 robot in order to provide additional feedbacks using its movements.

3.1.4 Architecture

The proposed infrastructure is composed by three main parts, namely signal acquisition, signal processing and actuation. The signal processing is handled by Lab Streaming Layer LSL) [8], the signal processing by Wyrm [11] (along with Mushu [83] for interfacing with LSL) and the actuation by PsychoPy [18]. All these components have been integrated into an NVIDIA Jetson TX2 embedded system to achieve the portability goal. LSL is responsible for networking, synchronization and storage of the biosignals, Wyrm is responsible for preprocessing, feature extraction and classification, and PsychoPy is responsible for stimulus and feedback presentation.

In practice, the infrastructure is composed of three main softwares that interact with each other: *LabRecorder*, which receives the LSL streams and stores the desired ones, *StimulusPresentation*, which presents stimuli and feedback, and *Predictor* which continuously computes the predictions based on the received LSL streams and sends them back to *StimulusPresentation* in order to present the feedback.

More specifically, the biosignal source devices stream the data, and these data can be accessed by any program running on machines connected to the same local network. In this way, both *LabRecorder* and *Predictor* can access the same data simultaneously. While *LabRecorder* stores them, *Predictor* computes the prediction probabilities by means of a pre-trained model fed by the incoming signals and sends them back to *StimulusPresentation* which presents the feedback accordingly. The stimuli presented by *StimulusPresentation* are important because they force the users to think specific tasks and at the same time are used to label the data stored by *LabRecorder*. This process is fundamental for the offline analysis used to train the model used by *Predictor* to produce the predictions in real-time.

Figure 3.7 represents the employment of the infrastructure in a complex multi-modal BCI demo experiment. The biosignals are produced by two headsets (OpenBCI and Enobio). The Enobio transmits signals to the proprietary acquisition program, named NIC, via Bluetooth low energy

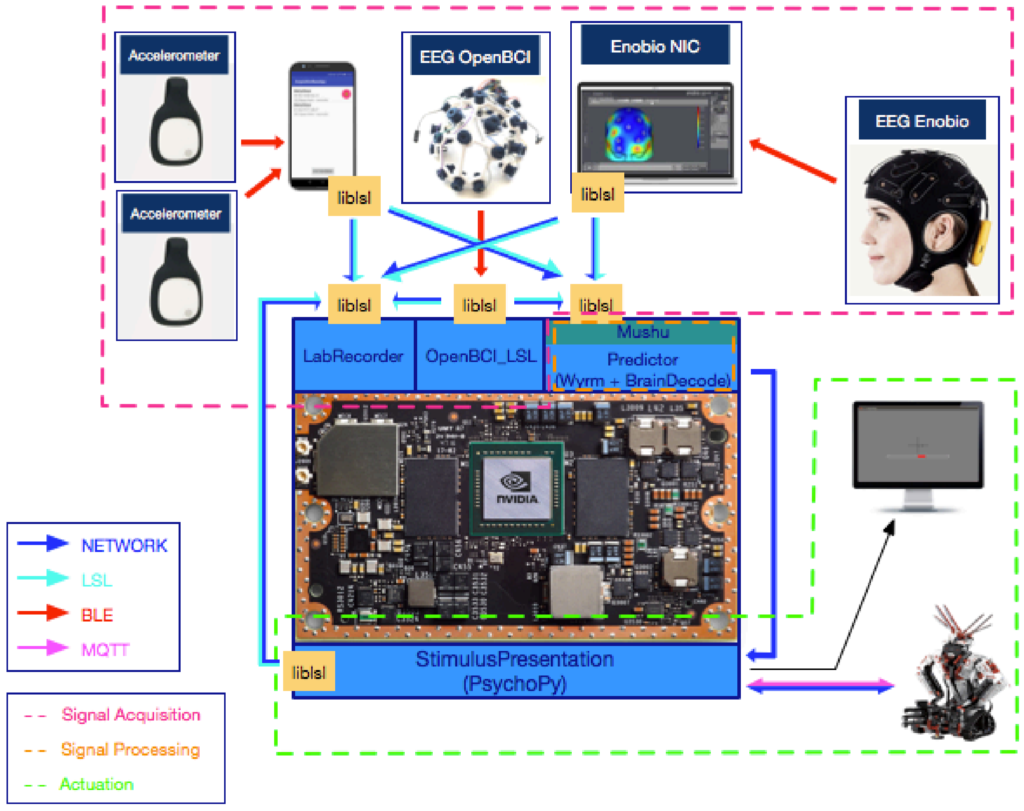


Figure 3.7: Infrastructure

(BLE). Data is then streamed over the network by means of LSL. The stream is acquired by *LabRecorder* that stores the time series, and at the same time it is acquired also by *Predictor* by means of the LSL interface implemented in Mushu. The same route is followed by the OpenBCI data, with the only difference that the data is reveived via BLE and sent directly by a program running on the NVIDIA Jetson TX2. In addition to the two EEG devices, in this demonstration we have introduced also two accelerometers connected via BLE. An ad-hoc Android application acquires the data and streams it via LSL. In this way, both the *LabRecorder* and the *Predictor* can acquire the four streams and act accordingly. More specifically, *LabRecorder* stores them and *Predictor* processes them to produce continuous probability predictions that are sent via socket to the *StimulusPresentation* that presents the corresponding feedback on a screen connected to the Jetson. At the same time specific movement instructions

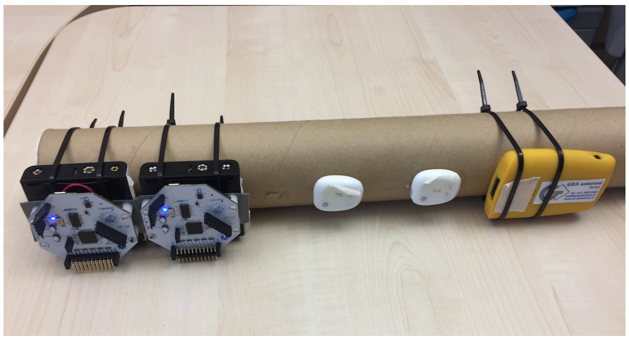


Figure 3.8: Experimental validation setting. Two OpenBCI, two accelerometers and one Enobio are secured to a tube that is rolled to produce simultaneous quasi-sinusoidal signals captured by the accelerometers integrated in the five devices.

deriving from the computed predictions are delivered to a robot via the publish-subscribe messaging protocols MQTT.

3.2 Validation

3.2.1 Synchronization

To validate the synchronization capabilities of the infrastructure in a real mobile context we designed an empirical experiment in which multiple Bluetooth low energy devices stream data via LSL.

This experiment is based on the assumption that LSL is able to synchronize incoming streams with 1 ms accuracy, as demonstrated in the LSL validation presented in Section 3.1.1. Based on this assumption we tried to estimate the relative misalignment introduced among pairs of devices by the Bluetooth transmission. The estimate of the misalignment is fundamental for BCI applications because it determines which analysis can be performed on the collected data and the maximum prediction frequency.

In particular, in the experiment we secured two OpenBCI, two accelerometers and one Enobio to a tube that we rolled to produce simultaneous quasi-sinusoidal signals captured by the accelerometers integrated in the five devices. One session was divided in windows (intervals) where the bar was rolled separated by intervals in order to be able to analyze the alignment between two signals without worrying about possible large misalignments that could lead to wrong overlaps breaking the correlation estimate. The windows were

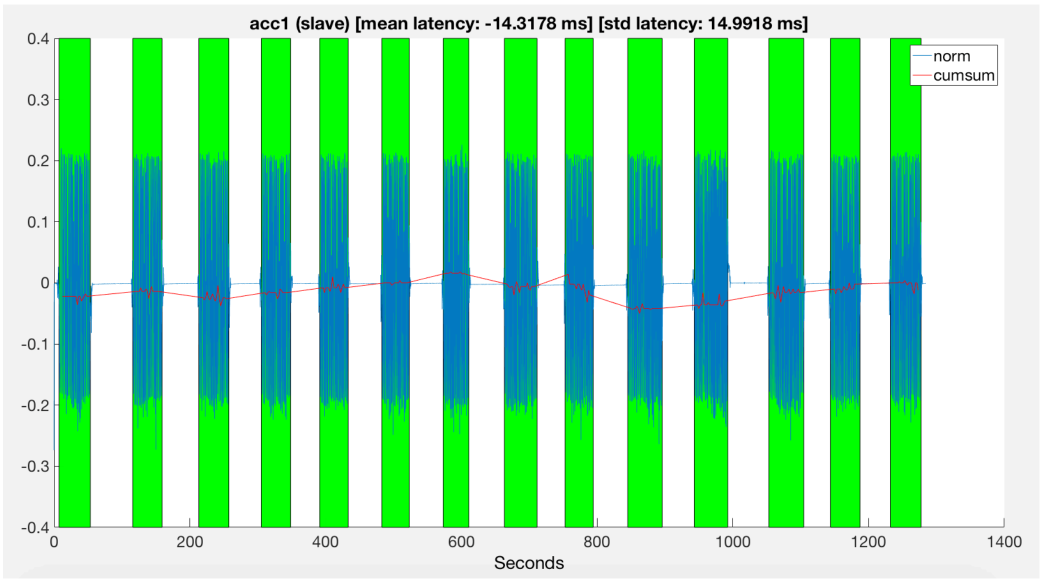
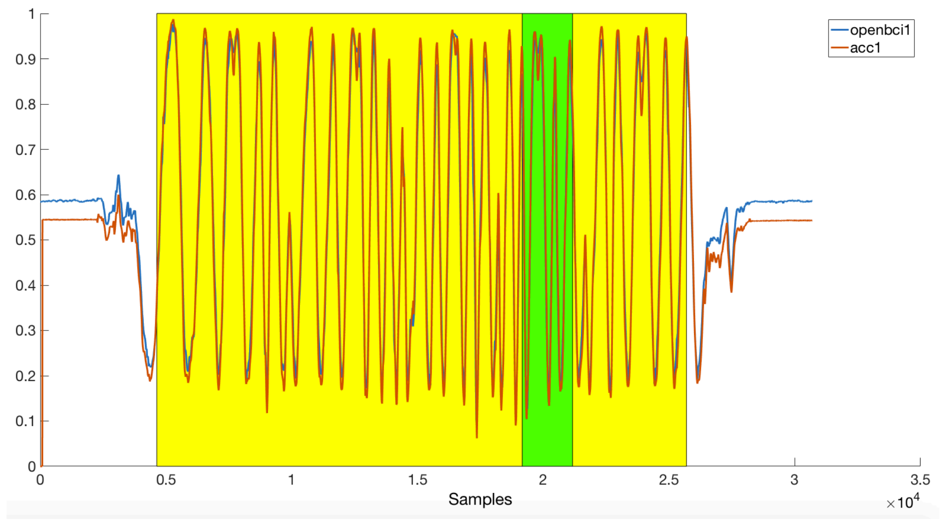
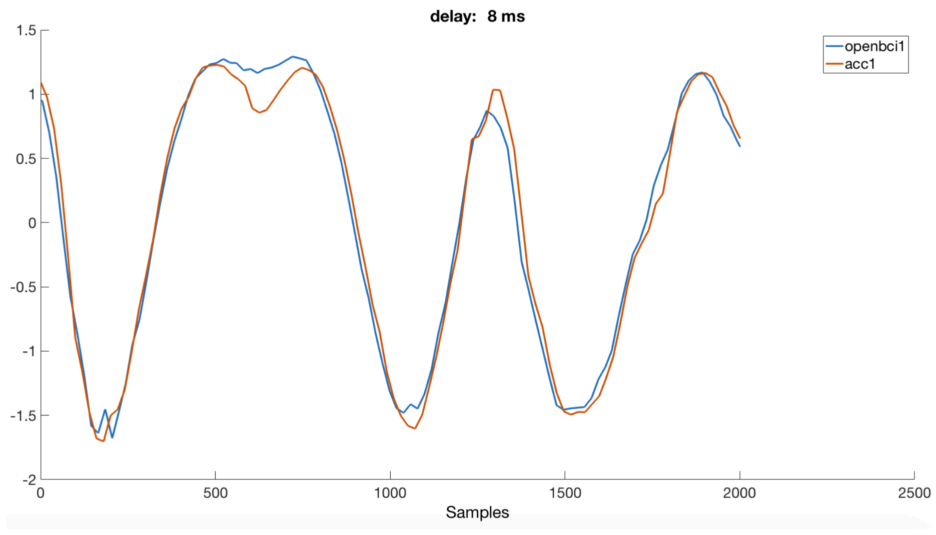


Figure 3.9: Experimental validation session. The bar is rolled back and forth various times within 14 intervals spaced apart. The red line represents the progress of the relative cumulative delay between one accelerometer (slave) and one OpenBCI (master).

identified by means of the rolling mean of the signals and a threshold. Each window was analyzed by computing the cross correlation between two signals in order to estimate the number of samples of misalignment, and therefore the temporal delay. The cumulative sum was then computed starting from the beginning of the session to produce the progress of the relative delay. Figure 3.9 shows the signal corresponding to the z axis of one accelerometer of one entire session divided in 14 windows. The red line represents the cumulated delay of the accelerometer with respect to one OpenBCI from the beginning of the session. As we can see, the mean relative delay is -14.3 ms (meaning that, on average, the accelerometer1 signals are received 14.3 ms later than the OpenBCI ones) and the standard deviation is 14.99 ms. As we have already mentioned, this delay is due to the Bluetooth transmission, and is two orders of magnitude higher than the uncorrectable one introduced by LSL. The fact that the standard deviation is in the same order of magnitude of the mean value does not allow us to improve alignment because even though we can reduce the worst case delay by subtracting the mean value to each timestamp, the order of magnitude of the nominal delays remains the same. Nevertheless, the nominal values are acceptable for a good deal of



(a)



(b)

Figure 3.10: One windows of the validation session session: (a) Zoom of one interval (in yellow) and identification one subwindow (green). (b) Trend of the two signals within the green subwindow highlighted in Figure (a) with the relative delay estimated by means of the cross correlation of the two signals.

paradigms, including motor imagery/execution, which is our target.

Figure 3.10 shows one of the windows of the session and outlines the procedure used to estimate the cumulative delay. In particular, the window was divided into non-overlapping subwindows and the cross correlation of the two signals was computed in a rolling fashion in each of these subwindows. In this way we were able to track the progress of the delay in each window and, as a consequence, also in the entire session.

In the figure the same subwindow is presented in two different ways: in the plot above it is highlighted in green within the respective window, while in the plot below it is enlarged to show in more details the two signals. The delay of 8 ms of the accelerometer with respect to the OpenBCI is estimated by means of the cross correlation between the two signals within the subwindow.

From the experimental validation have emerged two significant issues regarding the Bluetooth transmission. First of all, we discovered that the Enobio continuously loses the connection when other Bluetooth devices are active in the same area and therefore we haven't been able to evaluate its relative delay with respect to other devices. As a consequence, the experimental validation didn't take into account the Enobio as the problem is not due to the infrastructure but it is device specific. However, it is important to note that the Enobio, which is an expensive laboratory-grade device, can be safely used in classical BCI experiments in which the only source of information is the EEG stream since being the only active device it is not affected by the mentioned problem, rather, it provides much cleaner signals with respect to the OpenBCI.

The estimated delays of the other devices highlight a positive and a negative aspect: the positive is that the misalignment introduced by the Bluetooth communication is acceptable for the motor imagery paradigm, which is our target, because the interesting frequencies of the signals are compatible with the measured delays. The negative is that it is not true for all BCI paradigms, and thus to use the infrastructure in a mobility scenario for paradigms requiring a more accurate alignment we need to address the Bluetooth communication issue. For instance, the P300 paradigm requires sub-millisecond alignment because it is based on the detection of a positive deflection of the EEG with latency of around 300 ms with respect to the stimulus, and therefore needs a precise time base.

Figure 3.11 shows the progress of the relative delay of each device compared to one of the two OpenBCI. As we can see the other OpenBCI has a standard deviation of 61.8 ms, while the two accelerometers remain under 15 ms. We must take into account that the authors of LSL claim that it is possible to achieve 50 ms accuracy without effort, and it was confirmed by the order of magnitude shown in the figure despite the Bluetooth burden.

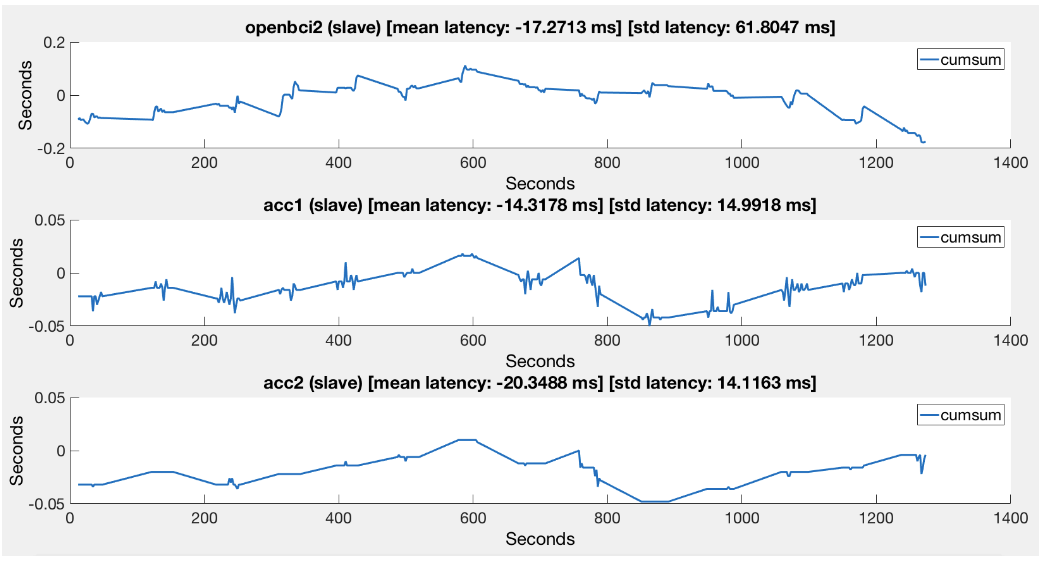


Figure 3.11: Experimental validation results. Each plot represents the progress of the relative delay accumulated during the entire session with respect to one of the two OpenBCI.

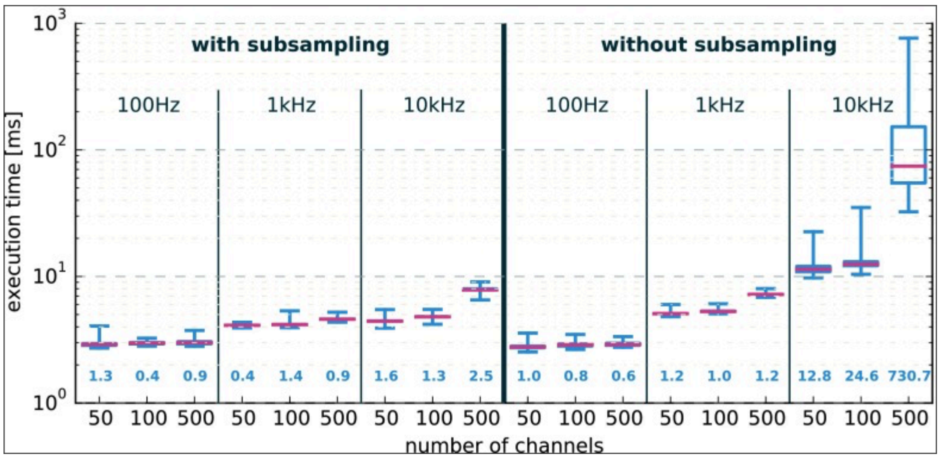


Figure 3.12: Execution time of a single pass of Wyrm loop with artificial data. Source:[11]

Another thing to consider is that the relative delay between the streams of the two accelerometers is under 1 ms, so it is possible to achieve the desired accuracy even with BLE devices. Nevertheless, these delays represent a problem for the analysis of complex high-frequency signals, because they make undetectable some brain mechanisms, and therefore this issue limits the infrastructure generality. In other words, some BCI paradigms like motor imagery/execution are feasible, because they don't require an extremely precise alignment among the streams, while others, like those based on Evoked Potentials (e.g., Steady State Visual Evoked Potentials (SSVEP)) are not, because, for instance, require a more accurate alignment between EEGs and markers to detect the specific brain patterns.

To this regard, we have identified some possible improvements, mainly concerning the acquisition devices. For instance, being the OpenBCI open-hardware, it would be possible to either modify the firmware in order to continuously estimate the Bluetooth transmission delay or directly integrate LSL into the Arduino module of the OpenBCI. An alternative, in case these modifications resulted to be too cumbersome, would be to switch to the WIFI transmission which has been recently implemented by means of an extension board. This change would also result in transmission benefits enabling higher streaming frequencies but at the cost of a dramatically faster battery drain.

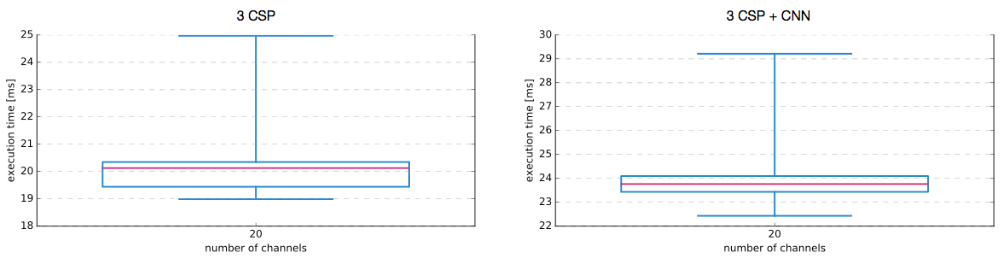
3.2.2 Processing

Wyrm is light and low-level, with only few and specific algorithms implemented. However, due to its simplicity (the online evaluation is basically just the Python loop sketched in the Algorithm 1) it can take advantage of other Python libraries like MNE [81] and scikit-learn [82] to perform specific tasks, or even interact with other BCI frameworks written in Python like BrainDecode.

Algorithm 1 Wyrm loop

```
while True do
    getNewData
    convertMushuData
    preprocess
    updateBuffer
    getLastWindow
    extractFeatures
    predict
    sendPrediction
end while
```

The performance estimated by the authors in terms of execution time



(a) (b)

Figure 3.13: Execution time of a single pass of Wyrm loop on the NVIDIA Jetson TX2 with real data (without subsampling). (a) Three CSP (left vs right, rest vs right and rest vs left) (b) The same three CSP plus one CNN (move vs rest) implemented in BrainDecode.

of a single pass of the loop is presented in Figure 3.12 and is obtained with synthetic data by varying sampling frequency and number of channels. The execution time is lower than 10 ms in all concrete configurations and therefore it is a promising result that paves the way to try more complex models, such as the Convolutional Neural Networks implemented in BrainDecode.

Indeed, we evaluated the execution time of the loop in a real context (i.e., with real EEG data streamed via LSL) with the computation of four models: three CSP (left vs right, rest vs right and rest vs left) for the 3-class soft-voting and one CNN implemented in BrainDecode. Figure 3.13 shows the estimates in two different configurations, namely with and without the CNN, where the figures have been computed running through the loop 1000 times. The obtained results are interesting because they show that in a real context with high-frequency incoming signals, the infrastructure is able to deliver commands, on average up to 40 Hz, and in the worst case up to 33 Hz. This is an important result because originally we were targeting the goal of reaching 20 Hz of command delivery frequency.

In addition, the comparison of the two plots shows that the introduction of the CNN produced an overhead of just 4 ms, so it demonstrates the applicability of Deep Learning methods for real-time EEG analysis.

3.3 Original work

The original work consists of three main contributions. The first concerns with the integration of the LSL ecosystem into the NVIDIA Jetson TX2, and in general into all the devices that require a library to stream or receive biosignal time series. As explained in Section 3.1.1, apart from adapting LSL to run on the Jetson, we developed an LSL-capable Android application to acquire and stream (over LSL) accelerometer and gyroscope data. This component is essential for the infrastructure because it allows multi-modal acquisitions involving motion detection, that is needed by self-paced motor execution BCI experiments, i.e., more natural experiments where the users perform the actions independently, without the presentation of stimuli imposing the action to perform.

The second contribution is the validation of the streaming and synchronization capabilities of the infrastructure in a complex acquisition with several devices streaming data via Bluetooth low energy (BLE). It was tested in an experimental setting taking advantage of the accelerometers integrated in the EEG headsets, thus without making use of expensive hardware. The validation demonstrates that in acquisitions involving multiple devices streaming via Bluetooth is hard to obtain synchronization accuracies lower than 100 ms, and in some cases it is even impossible to acquire data because of conflicts in transmission medium sharing. It is important to estimate the lower bound of synchronization capabilities because it determines which BCI paradigms can be analyzed (for instance P300 ERPs cannot be analyzed because they need a much finer alignment) and the frequency at which the model can be interrogated to compute predictions.

The third, and most important, contribution regards the implementation of the softwares constituting the BCI and the design of their orchestration in order to operate in real-time. The core of the infrastructure is the *Predictor* program which is based on Wyrm and Mushu. It continuously acquires data from LSL and computes the predictions according to a pre-trained model. These predictions are sent to the *StimulusPresentation* program which performs actions like presenting a feedback and/or sending commands to an actuator.

This infrastructure has been designed to be flexible and modular: it is possible to collect data from many different devices in a transparent way, provided that they can stream data via LSL, and also query different *Predictor* programs simultaneously and aggregate the outputs in the *StimulusPresentation* program in order to enhance the classification performance. As we have highlighted in this chapter, the real-time behavior is a crucial aspect for BCIs, and designing an infrastructure that can work online is a very hard task, mostly because the signals arrive at very high frequencies and therefore all the computation needs to be performed very fast. This aspect is even more relevant if we want to perform such computations on an embedded system to achieve portability. To succeed in this purpose we tried different BCI frameworks to test the respective online performance. The definitive choice has fallen on the most lightweight and minimal one to reduced the computational burden avoiding fancy computations. This choice affected the generality of the infrastructure because more complex frameworks like BCILAB provide a greater variety of methods for many BCI paradigms, while Wyrm has only a few and very specific methods natively implemented. Nevertheless, it is extensible and, at the same time, can make use of other python frameworks like MNE [81] and Scikit-learn [82] for specific tasks in the computation.

Chapter 4

Proof of concept

4.1 Experimental setup

The infrastructure described in the previous chapter was tested in prototyping a motor imagery/execution BCI system able to recognize in real-time brain patterns associated with two different intentions, namely left and right hand movement. In the experiment we have been able to use the Enobio acquisition device because its Bluetooth issues did not affect the motor imagery/execution paradigm. In this scenario EEG signals are the only source of information, and thus no conflicts in transmission medium sharing were present.

As first step we acquired the calibration datasets from four different subjects. These datasets are made up of different sessions in which subjects were instructed, by means of specific visual stimuli presented by *StimulusPresentation*, to imagine and execute the target actions. The resultant datasets are composed of a series of trials, marked according to the presented visual stimuli. These markers were sent via LSL by *StimulusPresentation* at the same time as the visual stimuli presentation.

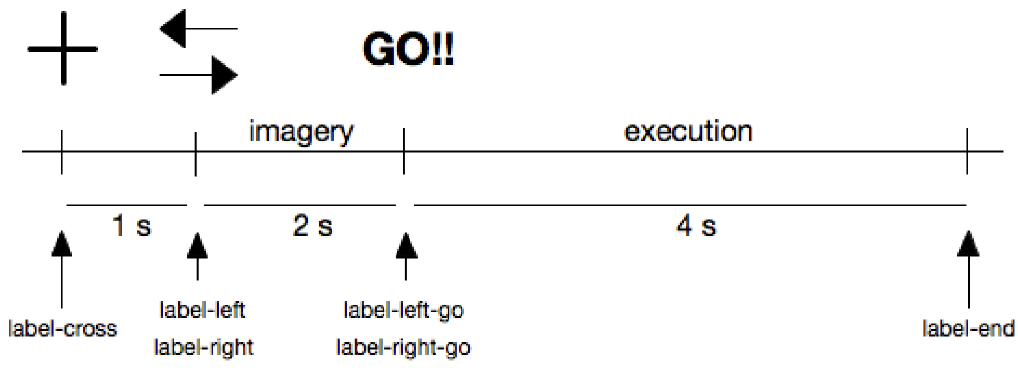


Figure 4.1: Acquisition protocol.

In the second phase, the calibration datasets were used to train subject-specific models needed to recognize the mental states hidden in the subsequent feedback sessions. Models were based on some of the state-of-the-art methods found in the literature, in particular we investigated the most used variants of the Common Spatial Pattern (CSP) and the Convolutional Neural Networks (CNN) implemented in BrainDecode.

The aim of the experiment was twofold: on the one hand we wanted to test the infrastructure capabilities in the phases necessary to build a BCI, in particular data acquisition and model training, and on the other we wanted to build a preliminary BCI having sufficient discrimination performance to be used as baseline for future developments. Indeed, we have been able to use the infrastructure to acquire structured datasets and to use them to train the BCI models with the framework of choice, achieving an offline classification accuracy of 75% for three of the four subjects both in motor imagery and motor execution, which is a promising result that can be considered as a reliable baseline for future improvements.

4.1.1 Dataset acquisition

To record the calibration datasets we designed a specific stimulation protocol that allowed to discriminate the two intentions both in motor imagery and in motor execution in order to extract useful insights about the brain patterns involved.

The protocol is depicted in Figure 4.1 and is composed of the following steps:

- Each trial begins with the presentation of a small cross at the center of the screen. The cross is kept on the screen for the whole trial and its purpose is twofold: it indicates the beginning of the trial and instructs the subject to focus on a fixed point of the screen in order to avoid ocular artifacts.

- After 1 second from the presentation of the cross, the trial cue is displayed to indicate the beginning of the motor imagery interval. It lasts 2 seconds, and during this period the subjects are instructed to imagine the respective hand movement (or absence of movement in case of rest trials). The cue is presented close to the cross in order to avoid ocular saccades, and corresponds to an arrow pointing either left or right or another rotated cross overlaid with the fixation one (rest trial).

- After the 2 seconds a little green circle appears in the center of the fixation cross to indicate the beginning of the motor execution interval for 4 seconds. At the same time, a feedback bar identical to the one presented in the feedback sessions, starts to move at constant speed in the direction indicated by the arrow. The speed is computed so that the bar reaches the end exactly after 4 seconds.

- Between the end of one trial and the beginning of the next one, an interval of random duration (from 1.5 second to 2 seconds) is introduced to allow the subjects to recover from the tiring activity. We used a random duration interval to avoid expectation effects.

The presentation of the feedback bar in the calibration session avoids evoking different brain mechanisms that could lead to inconsistencies with respect to feedback sessions.

A calibration dataset corresponds to one session. It is divided in 3 runs of 42 randomly distributed trials, for a total of 126 trials evenly distributed among the classes of interest. This means that a calibration dataset contains 42 trials for each class (left, right and rest). The rest trials are recorded for future developments involving also the corresponding absence of movement intention. Before the first run of each session we recorded a sequence of induced artifacts like blinks, horizontal and vertical eye movements, jaw clenches and swallowings to use in the offline analysis for artifact rejection (by means of a specific pipeline for bad channel identification, bad epochs and artifact rejection not described in this work).

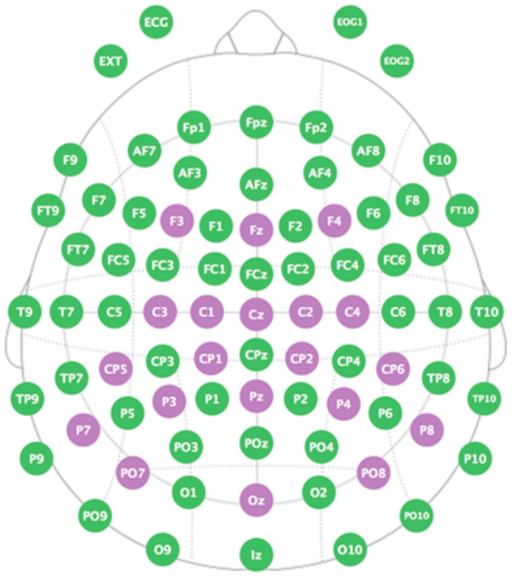


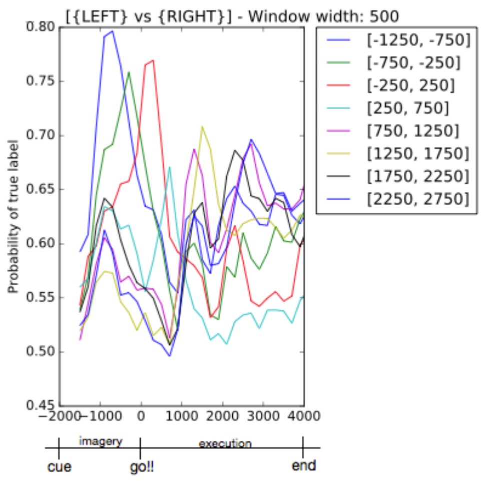
Figure 4.2: Adopted electrodes montage (10-20 system). Reference and Ground are placed on the same ear lobe.

As we have already mentioned, we used the Enobio acquisition device to acquire the EEG signals arranging the 20 dry electrodes according to the montage shown in Figure 4.2, where Ground and Reference were placed on the same ear lobe.

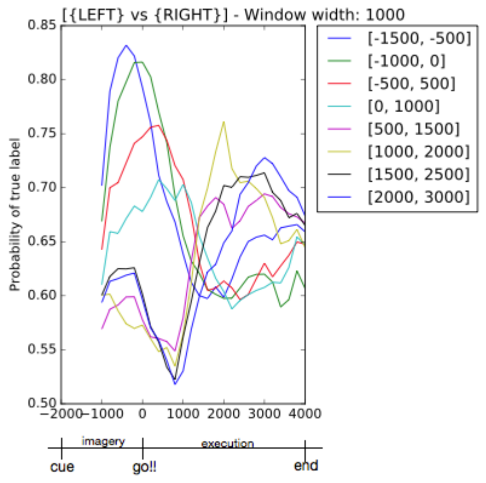
To mark the trials, the *StimulusPresentation* pushes on an LSL stream the marker associated to the cue in the exact moment it is displayed. In this way, during offline analysis it is possible to recover the epochs related to the different classes. Alignment of the markers with the EEG signals is performed by the storage program (*LabRecorder*).

4.1.2 Model training

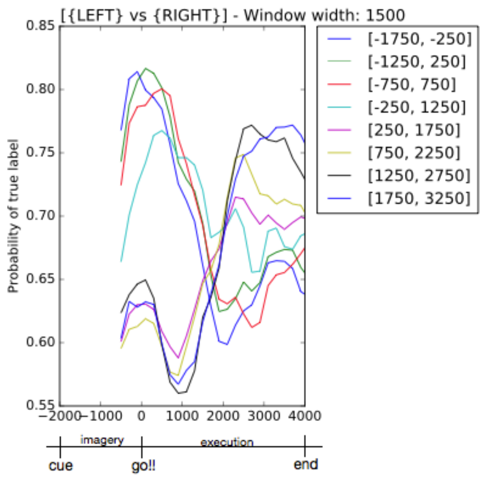
Based on the neuroscientific literature and past BCI competitions results, we decided to adopt the Common Spatial Pattern (CSP) as feature extraction method. It provides state-of-the-art performance and requires relatively low computational power, thus is well suited for real-time. We experimented with the FBCSP, SpecCSP and RCSP variants for the automatic tuning of the hyperparameters and regularization, but we did not obtained improvements despite the increase in complexity and computational time. For this reason



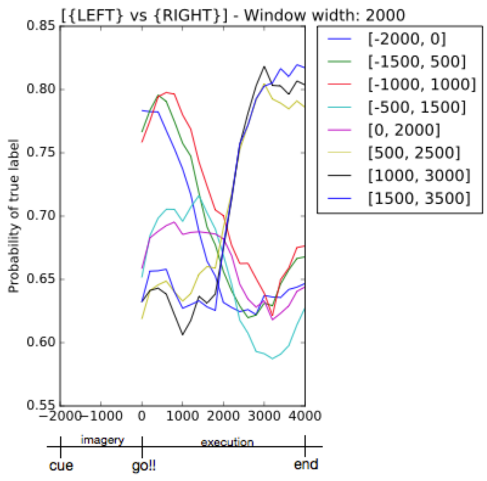
(a)



(b)



(c)



(d)

Figure 4.3: Grid search for width and position of the best window computed with Wyrm (subject = S3, number of patterns = 3, classifier = LDA). (a) Width = 500, (b) Width = 1000, (c) Width = 1500, (d) Width = 2000.

we focused on the classical CSP and we manually performed the tuning of the hyperparameters by means of a grid search. The selected hyperparameters are:

- Subject-specific frequency bands of the pass-band FIR filter used in the preprocessing phase.

- Width of the window for the computation of the spatial filter. This parameter affects both performances and responsiveness of the system.
- Position of the window within the epochs. This depends on subject-specific brain mechanisms and concentration ability.
- Number of patterns.

We performed the grid search experimenting with two different classifiers, namely Linear Discriminant Analysis (LDA) and Support Vector Machines (SVM), and we obtained the best results with the former.

Figure 4.3 shows results for the best subject data (S3) by keeping number of patterns (3) and classifier (LDA) fixed. Before computing the spatial filter data were subsampled from 500 Hz to 100 Hz and band-pass filtered between 4 and 30 Hz (subject-specific frequency band identified by means of a visual inspection of the Power Spectral Density (PSD) computed on the previous sessions data).

Each of the four plots shows the performance progress (i.e., the evaluated probability of the true label) along the trial produced by models with windows of same size. Each line represents the average progress of the corresponding model computed with a 10-fold cross validation. For instance, the blue line of Figure 4.3 (a) represents the performance of the model trained using the widow [-1250, -750] evaluated in all time instants of the trial. In this way, we can evaluate the performance of all trained model as if they were interrogated in real-time for the whole trial duration. This method gives an approximation of the performance a model trained in a specific window would have in all the time instances of the trial, thus spanning from motor imagery to motor execution. This is particularly evident because some models perform well mostly in the second part of the trial (motor execution) while others perform well mostly in the first part of the trial (motor imagery).

To evaluate the quality of a model we use also two other metrics, that are *responsiveness* and *robustness*. *Responsiveness* can be characterized as the time needed by the model to reach the maximum performance after the beginning of the task, whereas *Robustness* corresponds to the duration of the interval in which the model manages to maintain maximum performance.

Considering the three metrics together, the best motor imagery model for subject S3 is the one computed extracting the window [-1500, -500] (Figure 4.3 (b)) because it achieves the best overall performance and, at the

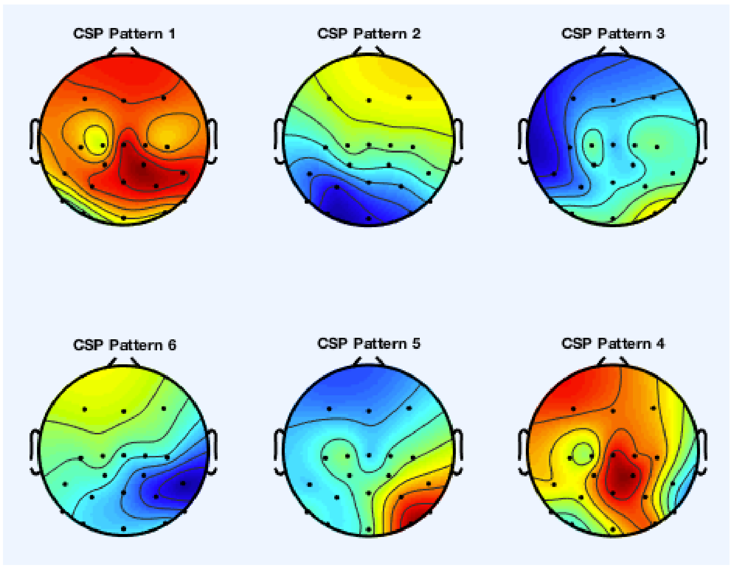


Figure 4.4: CSP patterns of the best model.

same time, it is the one reaching the best performance earlier. In addition, it also *robust* because the best performances are maintained for almost 1 s. To the same extent, the best motor execution model for subject S3 is the one computed extracting the window [1000, 3000] (Figure 4.3 (d)).

Figure 4.4 shows the six scalp maps (2D interpolations of the channel weights given the channel positions) corresponding to the 3 spatial patterns of the best motor imagery model of subject S3. As we explained in Section 2.2.3, the first and last m columns of the inverse of the projection matrix $\mathbf{W}$ represent the patterns corresponding to the m most discriminative spatial filters that are used to extract the features for classification.

The analysis of brain signals is, in general, a challenging task for a number of reasons. The two most critical are *variability* across different subjects of the brain mechanisms underlying the various intentions, and their *non-stationarity* across different sessions.

This aspect is evident as we have not been able to reach satisfactory results with one subject and even with the other three subjects we have experienced difficulties in training models able go generalize across different sessions. This aspect is one of the main challenges of the BCI field and is generally due to slight differences in the electrodes disposition in different sessions, but can also be due to changes in the environmental conditions that

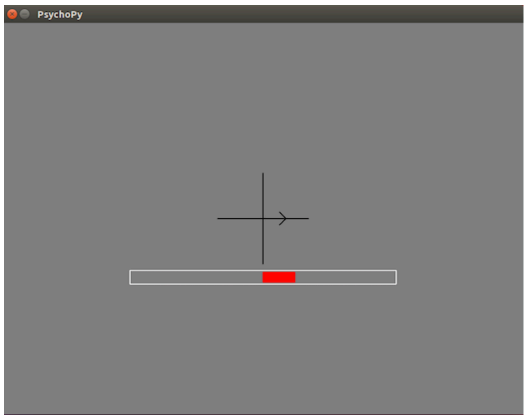


Figure 4.5: Snapshot of the feedback session.

affect users concentration.

However, the solution of these issues was not among the objectives of this thesis, rather we wanted to build an infrastructure allowing an agile training and implementation of BCIs on a single portable platform, and the fact that subjects need ad-hoc models continuously trained to cope with non-stationarity further reinforces the NVIDIA Jetson choice.

4.1.3 Feedback sessions

To further evaluate real-time performance of the infrastructure and quality of the trained models, we run a series of feedback sessions in which the feedback bar movement was controlled by the continuous estimated probabilities output by the *Predictor* according to the aggregation formula

$$p = (p' * 0.02) + (p * 0.98). \tag{4.1}$$

We decided to aggregate the probabilities for two reasons: firstly, we wanted to incorporate in the instantaneous feedback value also past predictions to create a robust averaged aggregated value, secondly we wanted to smooth the movement of the bar not to distract the subjects in case of spikes of the instantaneous predicted probability. To do so, we assign to the last produced prediction a small "weight" of 0.02 giving greater importance to accumulated past predictions. A snapshot of one of these feedback sessions is shown in Figure 4.5.

In the feedback sessions we also started experimenting with the control of the robot by means of commands issued based on the progress of the

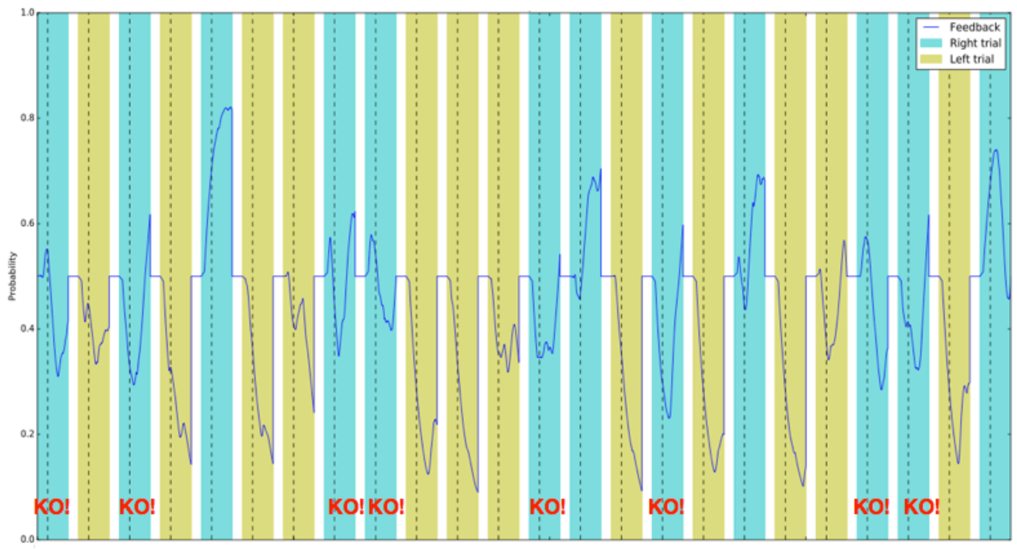


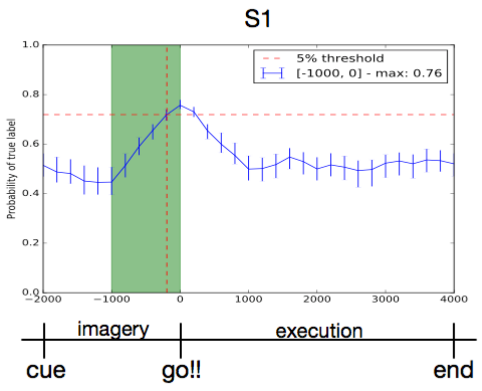
Figure 4.6: Reproduction of a feedback session snapshot. The progress of the aggregated probability of "right" class used as feedback is reproduced in retrospect from one stored dataset

aggregated probability. In this way we tested the full pipeline and the real-time communication for the coming developments. To manage a simplified context, we opted to send a command in a single-trial fashion (i.e., once for each trial) once the accumulated prediction probability of the trial reached a threshold. If no threshold was reached, no command was issued. The accumulated probability was reset to 0.5 (even probability for left and right) at the beginning of each trial, and the beginning of the aggregation started only once the buffer of length equal to the model window was full. This paradigm was set up just for preliminary evaluation purposes, but it could be easily extended to a non-cued scenario where the commends are continuously sent based on more complex decisions.

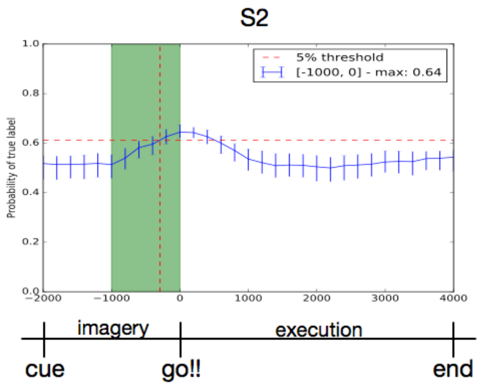
Figure 4.6 shows the reproduction in retrospect of the aggregated probability presented as feedback to subject S3 in one of the few successful feedback sessions. In general, we have been able to achieve a minimum model generality across different sessions only with subject S3.

In the figure, Cyan intervals correspond to "right" trials, while yellow intervals correspond to "left". The presented feedback corresponds to the aggregated probability of "right" class.

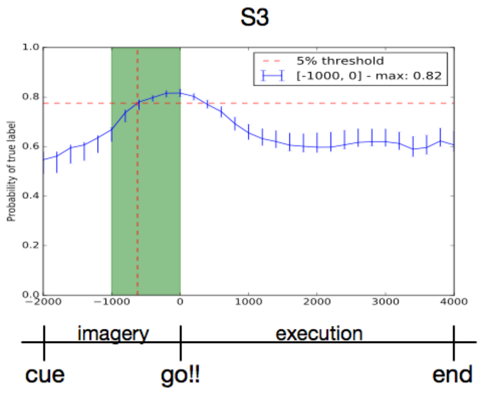
As we can see in the figure, the accuracy of the model in the presented



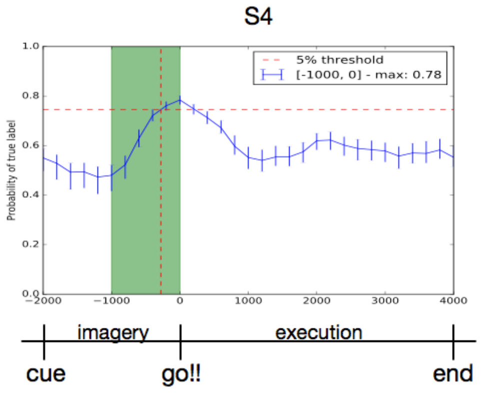
(a)



(b)



(c)



(d)

Figure 4.7: Model performance (best window highlighted in green) with identification of the command-delivery threshold set 5% lower than the maximum probability. Upper and lower error bars correspond to the misclassification error of "right" and "left" respectively. (a) S1, (b) S2, (c) S3, (d) S4.

snapshot is 66,6% (16 correctly classified trials out of 24, since in "left" trials the "right" probability should decrease under 0.5, whereas it shouls increase above 0.5 in "right" trials). However, considering that the model was trained using motor imagery data, with a low command-delivery thresholds like 0.55 (right) and 0.45 (left) the real-time performance can be increased to more than 70% by giving higher importance to the progress of the aggregated probability in the motor imagery interval (before the vertical dashed lines). A similar reasoning can be used to choose an greater command-delivery threshold if a motor execution model is used.

The threshold is small compared to the classification probabilities produced by the model because it is used with the aggregated probability, which takes longer to reach high values due to the weight of 0.02 assigned to new probabilities. An alternative way to trigger the command delivery could be to use a threshold associated directly to the raw prediction probability at each time point, and an empirical procedure to choose this threshold is depicted in Figure 4.7, where it set to 95% of the maximum reached probability, where the percentage can be decided based on the application level of criticality.

In general, the command-delivery threshold is strictly model dependent and very difficult to tune. In addition, we experienced difficulties in achieving model generalization across different sessions and therefore we leave the work of improving feedback session behavior to future developments.

4.2 Results

As we have seen in Figures 4.3 and 4.7, obtained results are promising because we achieved an offline accuracy of 75% for three of the four subjects both in motor imagery and motor execution. In particular the performances of the best models trained with Wyrm (CSP for feature extraction and LDA for classification) are summarized in Table 4.1.

	S1 [%]	**S2** [%]	**S3** [%]	**S4** [%]
CSP	**76.2**	**64.1**	**83.1**	**78.4**

Table 4.1: Wyrm results (left vs right, see Figure 4.7)

As we mentioned, we decided to adopt the classical CSP for feature extraction and LDA as classifier after an evaluation of the performance of some alternatives. In this regard, we compared the results of the classical CSP against those of FBCSP, RCSP and SpecCSP, evaluating also the differences using LDA and SVM as classifiers. This evaluation was carried out with BCILAB because the three CSP variants are not implemented in Whyrm. The best results adopting LDA and SVM as classifier (by varying CSP algorithm) are summarized in Tables 4.2 and 4.3 respectively.

	S1 [%]	**S2** [%]	**S3** [%]	**S4** [%]	**Computation time** [s]
CSP	**66.1**	63.5	**80.5**	**66.3**	5
FBCSP	61.0	**64.1**	78.4	65.5	10
RCSP	62.3	58.0	69.0	62.8	137
SpecCSP	64.2	61.8	77.1	62.3	185

Table 4.2: xCSP comparison with BCILAB (window width = 1 s, classifier = LDA)

	S1 [%]	**S2** [%]	**S3** [%]	**S4** [%]	**Computation time** [s]
CSP	64.2	**64.0**	**78.3**	**65.8**	128
FBCSP	58.6	57.5	72.5	59.7	250
RCSP	61.2	56.8	69.8	59.0	971
SpecCSP	**64.4**	59.7	75.0	65.2	185

Table 4.3: xCSP comparison with BCILAB (window width = 1 s, classifier = SVM)

As we can see from the comparison of Tables 4.2 and 4.1, the performance of signal processing model composed of CSP and LDA is substantially different between BCILAB and Wyrm implementations (the maximum difference is 12.1% relative to subject S4). This can be explained by the different libraries used to implement the covariance estimates of CSP and by the different default values assigned to the LDA algorithm by the two frameworks. In any case Wyrm obtains the best results with all four subjects, and this further reinforces the choice of its adoption in the architecture design.

To conclude the investigation of the frameworks, we also experimented with the CNNs implemented in Brain Decode. The work conducted in this thesis has arrived up to the analysis of "move" vs "rest" configuration, leaving more complex analysis for future developments. The results are summarized in Table 4.4, and have been obtained with a shallow CNN inspired by the FBCSP pipeline (see [63]).

	S1 [%]	**S2** [%]	**S3** [%]	**S4** [%]
CNN	**62.7**	**65.8**	**66.7**	**65.4**

Table 4.4: BrainDecode results [move vs rest].

4.3 Original work

Regarding the BCI system implemented to test the infrastructure, the original work consists of two main contributions.

The first refers to design and implementation of the protocol used during data acquisition to instruct the users to think at the specific actions we wanted to discriminate. The peculiarity of the designed protocol is that it comprehends in each trial both motor imagery and motor execution intervals, with the objective of collecting datasets allowing to discriminate the two intentions both in motor imagery and in motor execution in order to extract useful insights about the brain patterns in the two different paradigms. We also implemented the feedback delivery system whose peculiarity is the usage of aggregated prediction probabilities in order to incorporate in the instantaneous feedback also past predictions to make it smoother and more robust.

The second contribution consists in the entire analysis that has led us to the identification of a models effectively able to detect hand movement intentions and executions. To achieve this result we evaluated different model families available in literature and, once identified the most suitable to our purpose, namely the Common Spatial Pattern, we investigated some variants and the specific hyperparameters following a grid search approach. One remarkable characteristic of the implemented grid search is the evaluation procedure, which has been expressly designed to consider as metric not only the absolute best performance as in a single-trial fashion, but also the running performance along the whole trial length in order to consider also *robustness* and *reactivity* of the models. This choice was made because the progress of the prediction performance (enhanced with the breakdown of single class performances) along an evaluation interval was much more informative than the single best point, especially considering that our objective was to use these predictions continuously to present a feedback (see Figure 4.7). By using this method we have been able to train models able to achieve around

75% of accuracy in offline binary (“left” vs “right”) classification.

In addition we also started investigating Brain Decode capabilities in EEG analysis. In particular, we trained and run on the NVIDIA Jetson TX2 deep models (CNNs) achieving an average performance of 65.1% among four subjects in the “move” vs “rest” configuration, i.e., discrimination of left and right together against rest, and the results are summarized in Table 4.4.

Chapter 5

Conclusions and future work

The work of this thesis has been conducted in collaboration with Camlin Italy, which is the competence center in Data Science and Artificial Intelligence within CAMLIN, specialized in development and application of Machine Learning algorithms in sectors like power, health and rail.

This thesis aimed at building a portable and flexible infrastructure for prototyping BCIs to facilitate development and deployment, speeding up performance improvement and real-life adoption. To achieve this goal we integrated all the components necessary to prototype a BCI into the NVIDIA Jetson TX2 embedded system. In particular, we integrated LSL for signal synchronization, Wyrm for model building and real-time operation, and PsychoPy for stimulus and feedback presentation. During the development of the system we faced four main challenges:

- synchronization involving multiple devices streaming via Bluetooth is not sufficiently accurate for all BCI paradigms;

- issues in guaranteeing BCIs operating in real-time: not all the BCI frameworks are suitable for the purpose;
- brain signals are extremely sensitive to artifacts and non-stationary;
- patterns used by CSP method do not appear in all subjects' brain activity.

The main objective of the thesis has been achieved since we built a portable infrastructure with which we prototyped a BCI able to operate in real-time. The most interesting aspect of the proposed infrastructure is that it is able to produce prediction probabilities with complex models like CNNs up to a frequency of 30 Hz.

The work done is valuable because it proposes an infrastructure that is compatible with integration and deployment on the platform of state-of-the-art softwares like autoreject [84] and the MNE [81] implementation of ICA already used by the BCI team of company to address the artifact removal challenge. In particular, this work paves the way for tackling the artifact removal challenges also in a real-time context.

The proposed infrastructure aims to become the general solution for BCI research and development and therefore we need to complete the work related to the streams synchronization by characterizing the transmission delays introduced by Bluetooth communication channel and individual devices. The BCI team of the company implemented a system written in LabVIEW running on a National Instruments platform that will provide an absolute reference (characterization of the master device) to the module implemented in this thesis work to validate the synchronization. In this way we will be able to characterize each single device involved in the acquisitions.

Concerning the BCI experiment, we achieved promising binary classification accuracies around 75% with three out of the four subjects despite little effort has been done to remove potential artifacts. We also started investigating the real-time behavior of the models in feedback sessions for controlling the robot, but due to the experienced difficulties in achieving model generality across different sessions we leave it to future developments.

Considering the infrastructure so far developed, some possible future developments are:

- integration of LSL-style continuous delay estimation mechanisms in every acquisition device to mitigate Bluetooth transmission issues;
- enhancement of BCI models generalization across sessions to improve feedback reliability;
- robust discrimination of the third class (rest);
- integration of an online algorithm for artifact removal to improve accuracy of the model;
- deeper investigation of deep learning approaches and their adoption in BCIs for automatic feature extraction;
- enhancement of BCI systems with continuous learning approaches.

Bibliography

[1] "BNCI Horizon 2020: Roadmap," http://bnci-horizon-2020.eu/roadmap.

[2] M. Cohen, *Analyzing Neural Time Series Data: Theory and Practice*, MIT Press, 2014.

[3] S. N. A. M. Abo-Zahhad, Sabah M. Ahmed, "A new eeg acquisition protocol for biometric identification using eye blinking signals," 2015.

[4] S. Tong and N. Thakor, *Quantitative EEG Analysis Methods and Clinical Applications*, Artech House engineering in medicine & biology series. Artech House, 2009.

[5] S. Yin, Y. Liu, and M. Ding, "Amplitude of sensorimotor mu rhythm is correlated with bold from multiple brain regions: A simultaneous eeg-fmri study," *Frontiers in Human Neuroscience*, vol. 10, 2016.

[6] "Laughter Online University," http://www.laughteronlineuniversity.com.

[7] S. Raschka, *Python Machine Learning*, Packt Publishing, Birmingham, UK, 2015.

[8] "Lab Streaming Layer (LSL)," https://github.com/sccn/labstreaminglayer.

[9] "Network Time Protocol (NTP)," https://en.wikipedia.org/wiki/Network_Time_Protocol.

[10] "LSL Validation," `https://sccn.ucsd.edu/~mgrivich/LSL_Validation.html`.

[11] B. Venthur, S. Dähne, J. Höhne, H. Heller, and B. Blankertz, "Wyrm: A brain-computer interface toolbox in python," *Neuroinformatics*, vol. 13, no. 4, pp. 471–486, 2015.

[12] C. Brunner, N. Birbaumer, B. Blankertz, C. Guger, A. Kübler, D. Mattia, J. del R. Millán, F. Miralles, A. Nijholt, E. Opisso, N. Ramsey, P. Salomon, and G. R. Müller-Putz, "Bnci horizon 2020: towards a roadmap for the bci community," *Brain-Computer Interfaces*, vol. 2, no. 1, pp. 1–10, 2015.

[13] C. T. Lin, C. J. Chang, B. S. Lin, S. H. Hung, C. F. Chao, and I. J. Wang, "A real-time wireless brain-computer interface system for drowsiness detection," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 4, no. 4, pp. 214–222, Aug 2010.

[14] A. Maglione, G. Borghini, P. Aricò, F. Borgia, I. Graziani, A. Colosimo, W. Kong, G. Vecchiato, and F. Babiloni, "Evaluation of the workload and drowsiness during car driving by using high resolution eeg activity and neurophysiologic indices," in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2014, pp. 6238–6241.

[15] S. Haufe, J.-W. Kim, I.-H. Kim, A. Sonnleitner, M. Schrauf, G. Curio, and B. Blankertz, "Electrophysiology-based detection of emergency braking intention in real-world driving," *Journal of Neural Engineering*, vol. 11, no. 5, pp. 056011, 2014.

[16] M. Aravind and S. S. Babu, "Embedded implementation of brain computer interface using fpga," in *2016 International Conference on Emerging Technological Trends (ICETT)*, Oct 2016, pp. 1–5.

[17] E. Nurse, B. S. Mashford, A. J. Yepes, I. Kiral-Kornek, S. Harrer, and D. R. Freestone, "Decoding eeg and lfp signals using deep learning: Heading truenorth," in *Proceedings of the ACM International Conference on Computing Frontiers*, New York, NY, USA, 2016, CF '16, pp. 259–266, ACM.

[18] J. W. Peirce, “Psychopy—psychophysics software in python,” *Journal of Neuroscience Methods*, vol. 162, no. 1, pp. 8 – 13, 2007.

[19] A. Searle and L. Kirkup, “A direct comparison of wet, dry and insulating bioelectric recording electrodes,” *Physiological Measurement*, vol. 21, no. 2, pp. 271, 2000.

[20] F. Matsusaki, T. Ikuno, Y. Katayama, and K. Iramina, *Online artifact removal in EEG signals*, pp. 352–355, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[21] S. Halder, M. Bensch, J. Mellinger, M. Bogdan, A. Kubler, N. Birbaumer, and W. Rosenstiel, “Online artifact removal for brain-computer interfaces using support vector machines and blind source separation,” 2007.

[22] T. Radüntz, J. Scouten, O. Hochmuth, and B. Meffert, “Automated eeg artifact elimination by applying machine learning algorithms to ica-based features,” *Journal of Neural Engineering*, vol. 14, no. 4, pp. 046004, 2017.

[23] A. Hyvarinen, “Fast and robust fixed-point algorithms for independent component analysis,” *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 626–634, May 1999.

[24] J. F. Cardoso, “High-order contrasts for independent component analysis,” *Neural Computation*, vol. 11, no. 1, pp. 157–192, Jan 1999.

[25] G. Buzsaki, *Rhythms of the Brain*, Oxford University Press, 2006.

[26] M. L. V. Quyen, J. Foucher, J.-P. Lachaux, E. Rodriguez, A. Lutz, J. Martinerie, and F. J. Varela, “Comparison of hilbert transform and wavelet methods for the analysis of neuronal synchrony,” *Journal of Neuroscience Methods*, vol. 111, no. 2, pp. 83 – 98, 2001.

[27] K. Nancy, K. M. A., M. Paola, and W. M. A., “Are different rhythms good for different functions?,” *Frontiers in Human Neuroscience*, vol. 4, pp. 187, 2010.

[28] X.-J. Wang, “Neurophysiological and computational principles of cortical rhythms in cognition,” *Physiological Reviews*, vol. 90, no. 3, pp. 1195–1268, 2010.

[29] S. Sanei and J. Chambers, *EEG Signal Processing*, Wiley, 2008.

[30] G. Pfurtscheller, A. Stancák, and C. Neuper, "Post-movement beta synchronization. a correlate of an idling motor area?," *Electroencephalography and Clinical Neurophysiology*, vol. 98, no. 4, pp. 281 – 293, 1996.

[31] S. Luck, *An Introduction to the Event-related Potential Technique*, An Introduction to the Event-related Potential Technique. MIT Press, 2005.

[32] S. Haufe, M. S. Treder, M. F. Gugler, M. Sagebaum, G. Curio, and B. Blankertz, "Eeg potentials predict upcoming emergency brakings during simulated driving," *Journal of Neural Engineering*, vol. 8, no. 5, pp. 056001, 2011.

[33] I.-H. Kim, J.-W. Kim, S. Haufe, and S.-W. Lee, "Detection of braking intention in diverse situations during simulated driving based on eeg feature combination," *Journal of Neural Engineering*, vol. 12, no. 1, pp. 016001, 2015.

[34] H. Zhang, R. Chavarriaga, L. Gheorghe, and J. D. R. Millán, "Brain correlates of lane changing reaction time in simulated driving," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, Oct 2015, pp. 3158–3163.

[35] L. C. Parra, C. D. Spence, A. D. Gerson, and P. Sajda, "Response error correction-a demonstration of improved human-machine performance using real-time eeg monitoring," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 11, no. 2, pp. 173–177, June 2003.

[36] P. W. Ferrez and J. d. R. Millán, "Simultaneous real-time detection of motor imagery and error-related potentials for improved bci accuracy," in *Proceedings of the 4th International Brain-Computer Interface Workshop and Training Course*, 0 2008.

[37] A. Llera, M. van Gerven, V. Gómez, O. Jensen, and H. Kappen, "On the use of interaction error potentials for adaptive brain computer interfaces," *Neural Networks*, vol. 24, no. 10, pp. 1120 – 1127, 2011.

[38] W. A. MacKay, "Wheels of motion: Oscillatory potentials in the motor cortex," *Motor Cortex in Voluntary Movements*, 2004.

[39] G. Pfurtscheller and C. Neuper, "Motor imagery and direct brain-computer communication," *Proceedings of the IEEE*, vol. 89, pp. 1123–1134, 2001.

[40] C. Toro, G. Deuschl, R. Thatcher, S. Sato, C. Kufta, and M. Hallett, "Event-related desynchronization and movement-related cortical potentials on the ecog and eeg," *Electroencephalography and Clinical Neurophysiology/Evoked Potentials Section*, vol. 93, no. 5, pp. 380 – 389, 1994.

[41] G. Pfurtscheller and A. Aranibar, "Event-related cortical desynchronization detected by power measurements of scalp eeg," *Electroencephalography and Clinical Neurophysiology*, vol. 42, no. 6, pp. 817 – 826, 1977.

[42] G. Pfurtscheller and F. L. da Silva, "Event-related eeg/meg synchronization and desynchronization: basic principles," *Clinical Neurophysiology*, vol. 110, no. 11, pp. 1842 – 1857, 1999.

[43] G. Pfurtscheller, C. Neuper, D. Flotzinger, and M. Pregenzer, "Eeg-based discrimination between imagination of right and left hand movement," *Electroencephalography and Clinical Neurophysiology*, vol. 103, no. 6, pp. 642 – 651, 1997.

[44] H. Ramoser, J. Muller-Gerking, and G. Pfurtscheller, "Optimal spatial filtering of single trial eeg during imagined hand movement," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 4, pp. 441–446, Dec 2000.

[45] "BCI Competitions," `http://www.bbci.de/competition/`.

[46] D. S. Andreas Schwarz and G. R. Müller-Putz, "Brain-computer interface adaptation for an end user to compete in the cybathlon," 2016.

[47] J. Müller-Gerking, G. Pfurtscheller, and H. Flyvbjerg, "Designing optimal spatial filters for single-trial eeg classification in a movement task," *Clinical Neurophysiology*, vol. 110, no. 5, pp. 787 – 798, 1999.

[48] F. Lotte and C. Guan, "Regularizing common spatial patterns to improve bci designs: Unified theory and new algorithms," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 2, pp. 355–362, Feb 2011.

[49] W. Samek, F. C. Meinecke, and K. R. Müller, "Transferring subspaces between subjects in brain–computer interfacing," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 8, pp. 2289–2298, Aug 2013.

[50] H. Wang and X. Li, "Regularized filters for l1-norm-based common spatial patterns," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 24, no. 2, pp. 201–211, Feb 2016.

[51] M. Cheng, Z. Lu, and H. Wang, "Regularized common spatial patterns with subject-to-subject transfer of eeg signals," *Cognitive Neurodynamics*, vol. 11, no. 2, pp. 173–181, Apr 2017.

[52] W. Samek, C. Vidaurre, K.-R. Müller, and M. Kawanabe, "Stationary common spatial patterns for brain–computer interfacing," *Journal of Neural Engineering*, vol. 9, no. 2, pp. 026013, 2012.

[53] X. Yong, R. K. Ward, and G. E. Birch, "Robust common spatial patterns for eeg signal preprocessing," in *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2008, pp. 2087–2090.

[54] R. Zhang, P. Xu, T. Liu, Y. Zhang, L. Guo, P. Li, and D. Yao, "Local temporal correlation common spatial patterns for single trial eeg classification during motor imagery," *Computational and Mathematical Methods in Medicine*, 2013.

[55] S. Lemm, B. Blankertz, G. Curio, and K. R. Muller, "Spatio-spectral filters for improving the classification of single trial eeg," *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 9, pp. 1541–1548, Sept 2005.

[56] J. N. Sanes and J. P. Donoghue, "Oscillations in local field potentials of the primate motor cortex during voluntary movement," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 90, no. 10, pp. 4470–4474, 1993.

[57] R. Tomioka, G. Dornhege, G. Nolte, B. Blankertz, K. Aihara, and K.-R. Muller, "Spectrally weighted common spatial pattern algorithm for single trial eeg classification," 2006.

[58] K. K. Ang, Z. Y. Chin, H. Zhang, and C. Guan, "Filter bank common spatial pattern (fbcsp) in brain-computer interface," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, June 2008, pp. 2390–2397.

[59] Y. R. Tabar and U. Halici, "A novel deep learning approach for classification of eeg motor imagery signals," *Journal of Neural Engineering*, vol. 14, no. 1, pp. 016003, 2017.

[60] P. Bashivan, I. Rish, M. Yeasin, and N. Codella, "Learning representations from EEG with deep recurrent-convolutional neural networks," *CoRR*, vol. abs/1511.06448, 2015.

[61] N. Lu, T. Li, X. Ren, and H. Miao, "A deep learning scheme for motor imagery classification based on restricted boltzmann machines," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 6, pp. 566–576, June 2017.

[62] S. Stober, A. Sternin, A. M. Owen, and J. A. Grahn, "Deep feature learning for EEG recordings," *CoRR*, vol. abs/1511.04306, 2015.

[63] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball, "Deep learning with convolutional neural networks for eeg decoding and visualization," *Human Brain Mapping*, aug 2017.

[64] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, "Eegnet: A compact convolutional network for eeg-based brain-computer interfaces," *CoRR*, vol. abs/1611.08024, 2016.

[65] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, 2015.

[66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.

[67] J. Schmidhuber, "Deep learning in neural networks: An overview," *CoRR*, vol. abs/1404.7828, 2014.

[68] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, 1986.

[69] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[70] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[71] "Enobio," `http://www.neuroelectrics.com`.

[72] "OpenBCI," `http://openbci.com`.

[73] "Tools for Brain-Computer Interaction (TOBI)," `http://archiveweb.epfl.ch/www.tobi-project.org/`.

[74] "Tools4BCI," `http://tools4bci.sourceforge.net/index.html`.

[75] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[76] Neuroelectrics, "Event synchronization of eeg data using the lab streaming layer (lsl)," 2014.

[77] "BCI2000," `http://www.schalklab.org/research/bci2000`.

[78] "OpenViBE," `http://openvibe.inria.fr/`.

[79] "BBCI," `http://www.bbci.de/`.

[80] "BCILAB," `https://sccn.ucsd.edu/wiki/BCILAB`.

[81] A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, and M. S. Hämäläinen, "Mne software for processing meg and eeg data," *NeuroImage*, vol. 86, no. Supplement C, pp. 446 – 460, 2014.

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

[83] B. Venthur and B. Blankertz, “Mushu, a free- and open source bci signal acquisition, written in python,” in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2012, pp. 1786–1788.

[84] M. Jas, D. A. Engemann, Y. Bekhti, F. Raimondo, and A. Gramfort, “Autoreject: Automated artifact rejection for meg and eeg data,” *NeuroImage*, vol. 159, no. Supplement C, pp. 417 – 429, 2017.