Computers in Biology and Medicine 182 (2024) 109097

Contents lists available at ScienceDirect

Computers in Biology and Medicine

journal homepage: www.elsevier.com/locate/compbiomed

SpeechBrain-MOABB: An open-source Python library for benchmarking deep neural networks applied to EEG signals

Davide Borra [a,1,*], Francesco Paissan [b], Mirco Ravanelli [c,d]

[a] Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi" (DEI), University of Bologna, Cesena, Forlì-Cesena, Italy
[b] Fondazione Bruno Kessler, Povo, Trento, Italy
[c] Department of Computer Science and Software Engineering, Concordia University, Montreal, Quebec, Canada
[d] Mila - Quebec AI Institute, Montreal, Quebec, Canada

ARTICLE INFO

Keywords:
Electroencephalography
Neural decoding
Deep learning
Benchmarking toolkit

ABSTRACT

Deep learning has revolutionized EEG decoding, showcasing its ability to outperform traditional machine learning models. However, unlike other fields, EEG decoding lacks comprehensive open-source libraries dedicated to neural networks. Existing tools (MOABB and braindecode) prevent the creation of robust and complete decoding pipelines, as they lack support for hyperparameter search across the entire pipeline, and are sensitive to fluctuations in results due to network random initialization. Furthermore, the absence of a standardized experimental protocol exacerbates the reproducibility crisis in the field. To address these limitations, we introduce SpeechBrain-MOABB, a novel open-source toolkit carefully designed to facilitate the development of a comprehensive EEG decoding pipeline based on deep learning. SpeechBrain-MOABB incorporates a complete experimental protocol that standardizes critical phases, such as hyperparameter search and model evaluation. It natively supports multi-step hyperparameter search for finding the optimal hyperparameters in a high-dimensional space defined by the entire pipeline, and multi-seed training and evaluation for obtaining performance estimates robust to the variability caused by random initialization. SpeechBrain-MOABB outperforms other libraries, including MOABB and braindecode, with accuracy improvements of 14.9% and 25.2% (on average), respectively. By enabling easy-to-use and easy-to-share decoding pipelines, our toolkit can be exploited by neuroscientists for decoding EEG with neural networks in a replicable and trustworthy way.

1. Introduction

Machine learning approaches provide powerful tools for analyzing and decoding the brain activity, both invasively recorded, e.g., using single-neuron recordings [1–4], and non-invasively recorded, e.g., using electroencephalographic (EEG) signals [5–7]. Most successful applications regard the design of accurate neural decoders for Brain–Computer Interfaces (BCIs), for properly guiding actuators (e.g., a neuroprosthesis) using directly the user's brain activity [8,9]. The most common BCIs rely on the detection of motor imagery, P300, and steady-state visual evoked potential (SSVEP) neural responses [10].

In the past years, deep learning solutions were proposed for EEG decoding [7,11–13], such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and hybrid convolutional-recurrent neural networks. Differently from traditional machine learning pipelines, that separate the extraction of handcrafted features [5] from their classification [6], deep neural networks solve the addressed classification problem in an end-to-end fashion, from the input neural time series to the classes of interest. In addition, deep neural networks automatically learn the most meaningful neurophysiological features contained in the input EEG for discriminating the contrasted brain states [14–31], without relying on signal characteristics defined a priori (e.g., power-based features at specific EEG bands [6]). CNNs are the most used deep neural networks in the literature, as pointed out in recent reviews [7,11–13]. Indeed, CNNs significantly outperformed other networks [7,11–13,23,30,32,33] – e.g., RNNs based on multi-layer long short-term memories (LSTMs) or gated recurrent units (GRUs), and hybrid convolutional-recurrent networks – when decoding the EEG for BCI purposes. Remarkably, CNNs also won international scientific competitions on EEG decoding [32,33], and proved to significantly outperform traditional machine learning widely across BCI recording paradigms, for example in motor imagery [14,15], P300 [15,19,32,34], and SSVEP [24,35–37] decoding. Recently, transformer-based neural networks also proved to accurately decode EEG, similar to CNNs [38].

* Corresponding author.
E-mail address: davide.borra2@unibo.it (D. Borra).
[1] Work conducted while on an internship at Mila - Quebec AI Institute

https://doi.org/10.1016/j.compbiomed.2024.109097
Received 10 May 2024; Received in revised form 20 August 2024; Accepted 30 August 2024
Available online 11 September 2024

The widespread adoption of complex deep learning methods is contributing significantly to the so-called 'replicability crisis' in artificial intelligence [39], a concern particularly evident in EEG decoding [7]. Within EEG decoding, several factors exacerbate this issue:

i. Lack of common data. Similar to other applications of deep learning, datasets play a crucial role in EEG decoding. A unique challenge in EEG decoding lies in the high cost of collecting data compared to other signals (e.g., speech waveforms). Consequently, studies often rely on relatively small datasets, mostly with 10^3–10^4 examples (from approx. 10–20 participants) [7]. This leads to both inadequate performance and statistically biased, noisy, and highly variable performance evaluations, also due to the high inter- and intra-subject variability in EEG signals [40]. Among the main EEG decoding studies, only approx. half of them rely on public datasets (see Roy et al. [7] for a review). Recently, the 'mother of all BCI benchmarks' (MOABB) initiative [41] has taken remarkable steps in addressing this issue by standardizing an extensive list of publicly available EEG datasets under a single Python library.
ii. Lack of open-source libraries. The lack of open-source libraries and toolkits in the neuroscience community working on EEG decoding hinders replicability, reproducibility, and transparency in research. Although initiatives like MOABB [41] contributed to benchmarking traditional machine learning in a trustworthy way, its support for deep learning techniques remains limited. Indeed, in the literature of EEG decoding via deep neural networks, less than 15% of studies also release their code publicly (see Roy et al. [7] for a review), for example exploiting the braindecode Python library [14]. However, in these cases, the lack of common programming standards and protocols makes shared source code often challenging for other users to use effectively, e.g., source code covering only part of the decoding pipeline, such as only the neural network (for example in [15,32,34,38]). Thus, establishing a widely adopted community-driven open-source library is crucial for the neuroscience community.
iii. Lack of standardization. As emerges from the literature [7], there is an absence of a standardized experimental protocol in EEG decoding, affecting also the existing toolkits for EEG decoding (MOABB and braindecode). Several critical aspects, if not addressed, may result in imprecise and biased evaluations of deep learning models.

Concerning the standardization of experimental protocols (i.e., the last point), two crucial aspects have to be carefully considered for improving replicability.

The first aspect regards *hyperparameter search*. Deep learning-based decoding pipelines introduce many hyperparameters, e.g., for designing data pre-processing (e.g., cut-off frequencies for band-pass filtering), data augmentation (e.g., the time shift used for randomly shifting the EEG in time), network architecture (e.g., the number of convolutional kernels), and network training (e.g., the learning rate). Therefore, a deep learning-based pipeline is characterized in general by a high-dimensional hyperparameter space (i.e., defined by more than 10–20 hyperparameters [42]). Many state-of-the-art pipelines do not search for optimal hyperparameters, leading to sub-optimal performance and lacking in robustness (see Roy et al. [7] for a review). Other EEG decoding studies [14,19,38,43,44] accompany their proposed decoding pipelines with post-hoc hyperparameter evaluations (i.e., sensitivity analyses on hyperparameters), analyzing the effect on the test set performance when using different hyperparameter values respect to baseline ones. Only a few studies consider pipelines including automatic hyperparameter search by properly using a separate validation set (different from the test set) [21,23,45–47]. However, here simplifications of the search space occur, searching for few architectural and training hyperparameters only (6 on average, across prior studies [21,23,45–47]), limiting the overall effectiveness and ignoring searching important hyperparameters on data pre-processing and augmentation. Additionally, relevant details that influence the final performance, such as the number of hyperparameter search iterations, and the dataset used for optimization (e.g., validation or test) are often omitted.

The second aspect regards the *performance variability due to random initialization*. The literature neglects the influence of performance fluctuations in deep neural networks due to random initialization [14–27,32,35–38,43–48]. Indeed, as the networks are initialized randomly in their parameters, the parameters learned after the training process depend on the initialization point. Consequently, the final decoding performance is affected by the random initialization. These fluctuations should be carefully considered in EEG decoding, especially when the datasets are small [49]. Current practices involve single training and evaluation with a fixed random seed for initializing networks [14–27,32,35–38,43–48], potentially biasing results. Indeed, practitioners might achieve artificially inflated performance by selectively reporting the best results from multiple changes in random seeds, rather than considering the average performance across the multiple random seeds considered. This way, researchers performing experiments on deep neural networks could report results biased towards a desirable direction (e.g., significant improvements compared to a state-of-the-art algorithm) just by changing the random seed of the random initialization until the desired effect is achieved. Of course, this needs to be avoided.

In this study, we introduce SpeechBrain-MOABB, a novel user-friendly Python library that aims to mitigate the replicability issues of deep learning approaches applied to EEG, by addressing the critical limitations presented above. Here we present our toolkit and describe the use case scenarios in order to ease its use by neuroscientists. Moreover, we provide a detailed guide for performing a complete decoding experiment and we also compare the so-obtained decoding performance with other existing EEG decoding toolkits (MOABB and braindecode). The main novel contribution of our toolkit is the enhanced standardization of the experimental protocol, addressing the challenges posed by searching the optimal hyperparameters in high-dimensional search spaces and by the fluctuations of performance estimates due to random initialization. Specifically, our toolkit includes a fair and robust experimental protocol for EEG decoding, promoting standardization by:

i. Suggesting a straightforward yet effective approach to conduct hyperparameter search in multiple steps, for addressing the search on high-dimensional search spaces, tuning hyperparameters from data pre-processing, augmentation to network architecture and training (i.e., the hyperparameters characterizing the entire decoding pipeline).
ii. Incorporating native support for multi-seed training and evaluation, accounting for the statistical fluctuations in deep neural network performance resulting from random parameter initialization with different random seeds.

Moreover, as additional important contributions, SpeechBrain-MOABB simplifies the design of complex decoding pipelines, by enabling easy integration of novel algorithms (e.g., a new deep learning model or a new data augmentation strategy) into state-of-the-art EEG decoding pipelines, and facilitates their evaluation on MOABB-supported datasets, spanning across different BCI recording paradigms (e.g., motor imagery, P300, and SSVEP). Our library serves as a comprehensive benchmarking tool for neuroscientists interested in evaluating deep learning solutions with EEG datasets in a replicable and trustworthy way.

2. Background

In this section we provide an overview on EEG decoding (that is, single-trial EEG decoding), and on the existing decoding toolkits available in the state-of-the-art, presenting their strengths and limitations.

2.1. Single-trial EEG decoding

Let us assume that EEG signals are recorded from many participants across different recording sessions (performed on the same day or different days) in a cue-based BCI recording paradigm. For each participant and each session, an EEG dataset is formed by trials obtained by epoching the continuous EEG recording with respect to an event of interest (e.g., the presentation onset of a stimulus in a P300 oddball paradigm, exploited in P300-based BCIs). Each trial is associated with a specific cognitive response (e.g., 'target' response: brain response to target stimuli in oddball paradigms, containing the P300 response) or motor state (e.g., 'left-hand imagined movement' response: brain response during the imagination of the left-hand movement), among few states under investigation. The EEG trial can be arranged as a 2-D representation with C scalp sites and T time samples along columns and rows, respectively ($\in \mathbb{R}^{T\times C}$).

The single-trial EEG (i.e., time-locked EEG activity) can be classified into the associated brain state, by training a classifier that learns to associate the correct label to the input single-trial EEG. The classifier can be implemented by using a deep neural network and it is parametrized in its trainable parameters (to fit during training on a training set) and in its hyperparameters (to optimize during hyperparameter search on a validation set). For brevity, in this study we refer to single-trial EEG decoding simply as 'EEG decoding'.

2.2. Existing EEG decoding libraries: Strengths and pitfalls

In contrast to fields like computer vision, natural language processing, and speech processing, EEG decoding lacks dedicated open-source libraries for using deep learning approaches in a reproducible and robust way. For example, for processing speech waveforms with deep neural networks, SpeechBrain [50] is widely used. SpeechBrain is a Python toolkit built on PyTorch [51], crafted for speech processing applications like speech recognition, enhancement, speaker recognition, and various related tasks. Renowned for its flexibility and user-friendly design, SpeechBrain stands as one of the most widely used open-source tools in the realm of speech processing. Despite its original focus, its adaptable and convenient structure allows it to be applied to diverse sequence processing tasks, including EEG decoding. This work proposes SpeechBrain-MOABB by extending SpeechBrain to accommodate EEG decoding, providing the neuroscience community with an opportunity to leverage its inherent flexibility and ease of use within this domain.

The most popular EEG decoding libraries, namely MOABB [41] and braindecode [14], are described in the following.

i. MOABB [41]. *MOABB* represents a reference tool for neuroscientists, for benchmarking traditional machine learning on public EEG datasets. Notably, MOABB supports a variety of EEG datasets, spanning from motor imagery to P300 and SSVEP datasets, and evaluation schemes, including leave-one-session-out and leave-one-participant-out (see Section 3.2.2 for data iterators). It achieves high replicability of machine learning pipelines by encapsulating EEG datasets, state-of-the-art classifiers, and evaluation schemes under the same library. It also provides an easy and standardized way to share the classifier with its main hyperparameters into a single YAML file (see the `pipelines/` folder in MOABB). MOABB supports hyperparameter search mainly of classifier hyperparameters (e.g., the kernel in support vector machine). However, MOABB primarily supports traditional machine learning algorithms, whereas SpeechBrain-MOABB places a strong emphasis on deep learning.
ii. Braindecode [14]. Among EEG decoding libraries, it is worth mentioning *braindecode*, a Python library based on PyTorch [51] providing several state-of-the-art deep neural networks proposed for EEG, together with relevant functions and classes for decoding EEG. This library, however, is not specifically designed to ensure high reproducibility and reliability. Indeed, braindecode does not support hyperparameter search within a high-dimensional search space that encompasses the entire decoding pipeline, including also aspects like data handling (data preprocessing, and data augmentation). Additionally, braindecode does not address the fluctuations of decoding performance due to the network random initialization (i.e., lacks in robustness to random seed variability). Even though MOABB offers classes and functions to integrate neural networks defined with braindecode (MOABB+braindecode combination), the previous limitations continue affecting decoding reproducibility and reliability of deep neural networks applied to EEG.

Despite the important contributions in the field of EEG decoding introduced by MOABB and braindecode, the neuroscience community is still awaiting for a comprehensive benchmarking tool for designing and fairly comparing deep learning-based decoding pipelines applied to EEG signals, recorded in various BCI recording paradigms (e.g., motor imagery, P300, SSVEP paradigms).

SpeechBrain-MOABB aims at filling this gap, providing its novel contribution to the field by overcoming the limitations of the previously presented EEG decoding libraries, namely:

i. the lack of replicability of deep learning-based pipelines;
ii. the absence of hyperparameter search support on high-dimensional search spaces, considering hyperparameters from data pre-processing, augmentation to network architecture and training;
iii. the high fluctuations of decoding performance estimates due to random initialization.

Crucially, these important limitations concur to the 'replicability crisis' in artificial intelligence within the neuroscience community.

SpeechBrain-MOABB addresses the previous limitations respectively by (i) adopting a standardized experimental protocol for performing decoding experiments, (ii) supporting multi-step hyperparameter search for dealing with high-dimensional search spaces, and (iii) introducing multi-seed training and evaluation for providing performance estimates robust to the variability caused by random initialization. We summarize in Table 1 the key features of the existing toolkits (MOABB and braindecode), comparing them with SpeechBrain-MOABB features.

In the following we present our toolkit, delineating its design principles and describing in detail its key components, conceived for overcoming the previous limitations currently affecting the open-source EEG decoding libraries.

3. SpeechBrain-MOABB: Architecture and key features

In this section, we initially outline the design principles of SpeechBrain-MOABB and then delve into the key elements and scripts involved in the management of data pre-processing, data iteration, data augmentation, neural network definition and training, and hyperparameter search. Finally, we illustrate the command-line interface (CLI) for using the proposed toolkit. A high-level overview of SpeechBrain-MOABB is depicted in Fig. 1, showcasing its main parts and relevant scripts. Our toolkit is publicly released under the highly permissive Apache 2.0 license, providing easy access and usability at https://github.com/speechbrain/benchmarks/tree/main/benchmarks/MOABB.

3.1. Design principles and strengths

SpeechBrain-MOABB is developed with the following goals:

Table 1
Comparison between SpeechBrain-MOABB and the existing EEG decoding libraries. Here, 'ML' stands for machine learning, and 'DL' stands for deep learning. Note that with 'complete hyperparameter search' we indicate a complete search spanning from data pre-processing, augmentation, to network definition and training.

Python library	Mainly used for	Easy to use	Easy replicability of the decoding pipeline	Support for a complete hyperparameter search	Robustness to random seed variability
MOABB [41]	ML	✓	✓	✗	✗
Braindecode [14]	DL	✓	✗	✗	✗
SpeechBrain-MOABB	DL	✓	✓	✓	✓

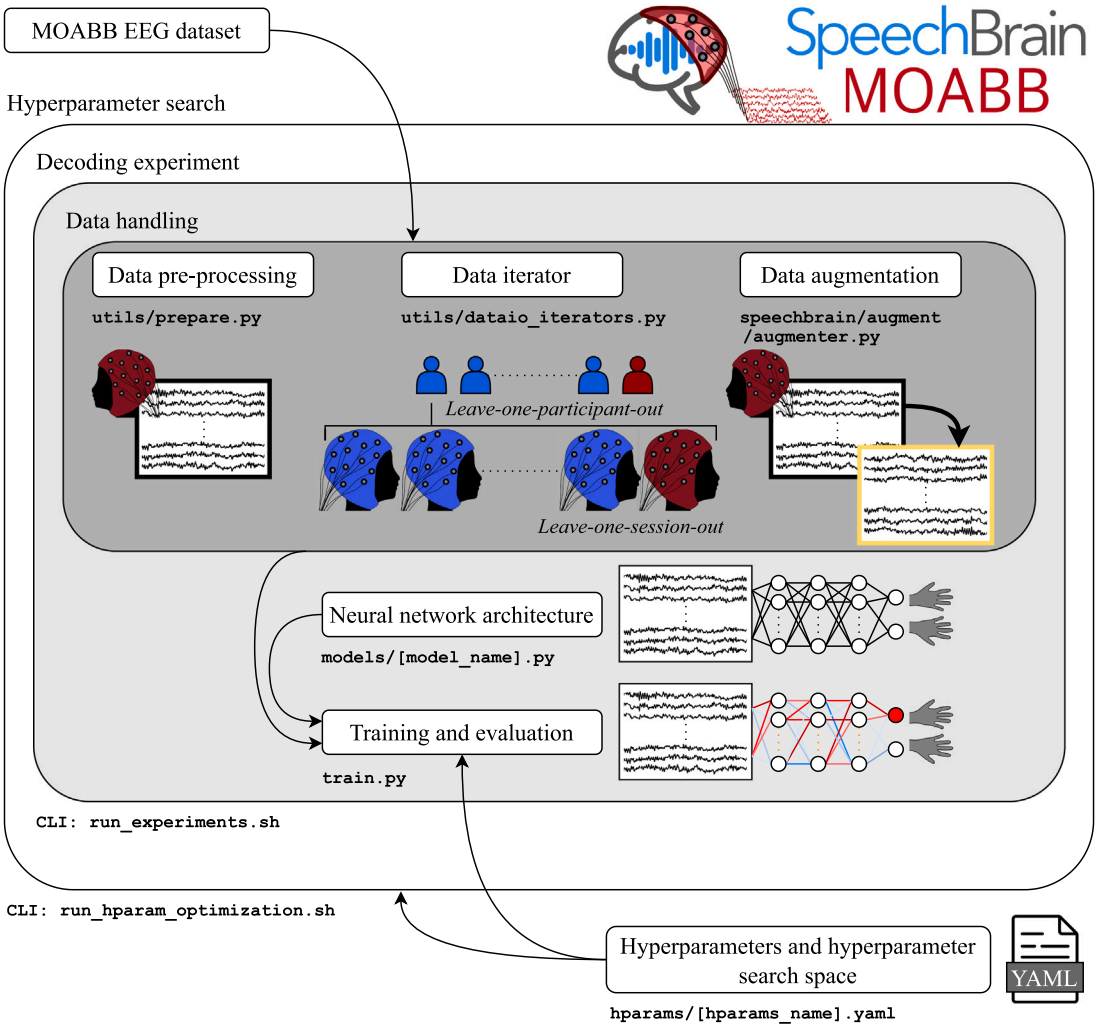

Fig. 1. SpeechBrain-MOABB: high-level scheme. The core elements are the data pre-processing, iterator, and augmentation (data handling), together with the neural network architecture, and the network training and evaluation loop. The basic decoding experiment runs a single cross-validation fold. To run the entire experimental setup, the script is wrapped in a command-line interface (CLI) that supports different training/evaluation strategies and also multi-seed random initialization (`run_experiments.sh`). Finally, this is augmented to run the multi-step hyperparameter search, using another CLI (`run_hparam_optimization.sh`). All the hyperparameters characterizing the entire decoding pipeline, together with the hyperparameter search space, are conveniently defined in a single hyperparameter file (YAML file). In the figure, to ease the comprehension of the fundamental processing steps, each step is accompanied by an illustration of its processing. The names of the relevant scripts are reported too (bold font).

i. Ease of use. We prioritize accessibility for neuroscientists, including students and practitioners with varying levels of programming expertise. A clear distinction exists between the hyperparameters and the training script. Through CLIs orchestrating training, evaluation and hyperparameter search, and hyperparameters fully exposed in a single YAML-formatted hyperparameter file, users can set up the entire deep learning-based decoding pipeline simply by properly compiling one file (the hyperparameter file). This design allows neuroscientists to invest more time in experimenting and less in technical setup. To further ease the toolkit's use, we provide tutorials helping users at various educational levels, from beginners to experienced neuroscientists with deep learning expertise.

ii. Replicability. Besides encapsulating the entire decoding pipeline in a single hyperparameter file (thus, making it easily replicable across research groups), our toolkit incorporates a robust experimental protocol to mitigate the replicability crisis in EEG decoding, that natively supports multi-seed random initialization and multi-step hyperparameter search. This protocol comprehensively manages all experiment settings, ensuring replicable and trustworthy results.

iii. Sharing of entire decoding pipelines. We have structured SpeechBrain-MOABB to facilitate easy sharing of the decoding pipeline (from data pre-processing and data augmentation to network definition and network training) within the neuroscience community. This design encourages neuroscientists to

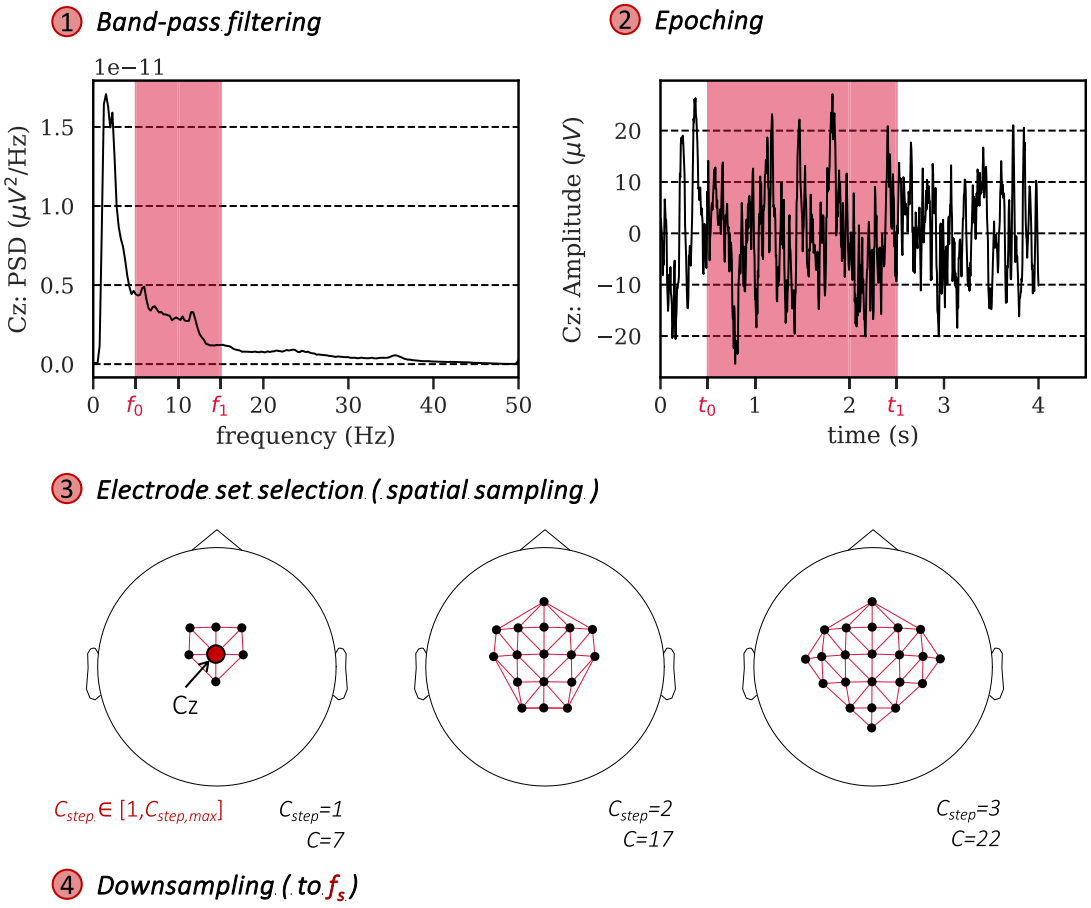

Fig. 2. Data pre-processing. Top-left and top-right panels show the power spectral density (PSD) and the amplitude over time of Cz electrode, respectively. In the lower panel are reported the sets of EEG electrodes generated by the performed spatial sampling procedure, as a function of the hyperparameter C_{step}, that defines the distance with respect to the seed electrode (Cz in the figures) for selecting the electrodes. The main hyperparameters defining the pre-processing are highlighted with red (f_0, f_1, t_0, t_1, C_{step}, and f_s). For these representations, BNCI2014-001 dataset [53] is used.

share their deep learning-based decoding approaches and finally obtain replicable results, fostering progress in the field.

iv. Research. Our primary aim is to promote research in EEG decoding, facilitating interdisciplinary collaboration between deep learning scientists and neuroscientists. By allowing deep learning experts to design novel EEG decoding pipelines without requiring in-depth knowledge of neural signal processing intricacies, SpeechBrain-MOABB encourages collaboration between speech and EEG researchers. This synergy has the potential to foster positive cross-contamination of ideas and methodologies. Indeed, insights taken from the speech recognition domain, even though appearing as a distant domain from EEG decoding, led to exciting results in recent EEG studies [17,18,21,23,26,52].

3.2. Data handling

3.2.1. Data pre-processing

SpeechBrain-MOABB is fully compatible with the MOABB [41] EEG data structure, allowing users to leverage MOABB datasets. The signals included in MOABB were collected during cue-based BCI recording paradigms (see Section 2.1). In SpeechBrain-MOABB the EEG pre-processing is performed with the script `utils/prepare.py`, by leveraging on MNE [54] and MOABB [41] Python libraries, and it is schematized in Fig. 2. Neural signals undergo a lightweight pre-processing in the frequency, temporal, and spatial domains to prepare them for decoding (see also Roy et al. [7]):

i. Band-pass filtering. Signals are band-pass filtered within the range $[f_0, f_1]\,Hz$, where f_0 and f_1 denote the low and high cut-off frequencies (3dB-frequencies), respectively. Here, a 4th order zero-phase infinite impulse response Butterworth filter is employed.

ii. Epoching. EEG trials are obtained by epoching continuous signals within the range $[t_0, t_1]\,s$, where t_0 and t_1 are the lower and upper limits of the EEG trial, respectively, having indicated with $t = 0$ s the onset of the reference event used for epoching. In motor imagery, P300, and SSVEP recording paradigms, the reference events correspond to the onset of movement imagery, standard/deviant stimulus, and SSVEP stimulus, respectively.

iii. Electrode set selection. A spatial sampling procedure is employed to select the set of channels from the complete channel set contained in each EEG trial. Beginning with a seed channel (Cz by default, but can be changed by the user), the selected set of channels includes adjacent channels within a distance of C_{step} channels from the seed channel. This spatial sampling procedure ensures that the set of considered channels ranges from a small subset (a few neighbors of the seed channel, $C_{step} = 1$) to the entire channel set ($C_{step} = C_{step,max}$), depending on the parameter C_{step}. $C_{step,max}$ depends on the number of EEG channels per EEG trial.

iv. Downsampling. For computational efficiency, signals are downsampled to f_s.

Of course, also other EEG pre-processing steps may be applied to enhance the task-related brain activity. For example, channel re-referencing and artifact removal via independent component analysis are widely used for canonical EEG analyses [55]. However, when moving from EEG analysis to EEG decoding via deep neural networks, simpler pre-processing pipelines are exploited [14–25,38,48], as the one used in our study. Our pre-processing pipeline takes inspiration from prior studies on deep learning-based EEG decoding [14–25,38,48] (see also Roy et al. [7] for a review). Here, the pre-processing is kept as light as possible, leaving the learning system able to automatically extract the most relevant features contained in the input EEG for accurately discriminate the target brain states (e.g., motor imagery conditions). Indeed, deep neural networks proved to be able to automatically extract the most relevant brain-related activity from such lightly pre-processed EEG [14–25,38], avoiding the use of a heavier pre-processing pipeline that include additional steps (e.g., artifact removal via independent component analysis). For example, in case a peculiar spatial filtering is relevant for enhancing specific task-related brain activity, a deep neural network is able of automatically learning such processing by exploiting spatial convolution. In this convolution, spatial filters are learned and convoluted with the input time series, enhancing the relevant spatial information for solving the EEG decoding problem (i.e., discriminating different brain states). Thus, this automatic EEG processing operated by deep neural networks is considered like a replacement for pre-processing steps such as re-referencing (e.g., common average referencing: computing the mean across channels and removing it from each channel) and artifact removal procedures (e.g., enhancing the contralateral brain activity over the sensorimotor cortex in a motor task via independent component analysis) [7].

The collection of the hyperparameters defining each pre-processing step (i.e., f_0, f_1, t_0, t_1, C_{step}, f_s) constitutes the data pre-processing hyperparameters that should be tuned together with the other hyperparameters of the decoding pipeline.

3.2.2. Data iterator

EEG datasets typically involve multiple participants, each recorded across multiple recording sessions. Consequently, neural decoders can be mainly trained as (see also Roy et al. [7]):

i. Participant-specific decoders, utilizing leave-one-session-out cross-validation. For each participant, one recording session is reserved as a test set (identified in red in Fig. 1), and the remaining sessions constitute the training and validation sets (identified in blue in Fig. 1). This results formally in a session-level cross-validation scheme, where the number of cross-validation folds is equal to the number of total sessions. A leave-one-session-out strategy can be used for evaluating the performance of participant-specific decoders, simulating a classic BCI recording paradigm in which many recording sessions are acquired for calibrating (i.e., training) decoders, and an online session is used for testing the decoder.
ii. Cross-participant and participant-agnostic decoders, employing leave-one-participant-out cross-validation. Here, one participant (i.e., all the recorded sessions for that participant) is held out as a test set (identified in red in Fig. 1), while the remaining participants contribute to the training and validation sets (identified in blue in Fig. 1). This corresponds to a participant-level cross-validation scheme, where the number of cross-validation folds is equal to the number of total participants. A leave-one-participant-out strategy can be used for evaluating the performance of participant-agnostic decoders, simulating the application, on a new user, of a decoder trained on other subjects, i.e., simulating a calibration-free BCI system.

Participant-specific decoders, as demonstrated in previous studies [19,23,25], outperform cross-participant decoders, primarily due to the high inter-participant variability in EEG signals characterizing cross-participant training strategies. SpeechBrain-MOABB accommodates both of these cross-validation approaches, providing the corresponding data iterators in the script `utils/dataio_iterators.py`. These iterators mimic the MOABB data iterator counterparts (namely, `CrossSessionEvaluation` and `CrossSubjectEvaluation` classes), while being compliant with SpeechBrain data handling.

3.2.3. Data augmentation

With SpeechBrain-MOABB, the EEG signals can be dynamically augmented on the fly during training, meaning that each training mini-batch undergoes a distinct augmentation. EEG augmentation adds synthetic EEG to the training set (see Lashgari et al. [56] for a review) and is achieved by:

i. Injecting noise with specific patterns (e.g., white Gaussian or pink noise) in signals, with a specific signal-to-noise-ratio (SNR). This augmenter introduces a spectral perturbation in the time series, by adding frequency components to the spectrum.
ii. Dropping a frequency band from signals. Similar to the previous augmenter, this one applies a perturbation to the spectrum of the time series. However, in contrast to the previous augmenter, here a range of frequencies is removed from the spectrum.
iii. Dropping chunks of the signals by zeroing portions of time series. Here, the augmented time series will encode less information in the time domain, simulating an EEG recording in which less potential components are evident in the neural response (e.g., removing the P300 component from the neural response).
iv. Dropping (i.e., zeroing) or swapping signals of few EEG channels. This augmenter forces the network to avoid relying on a specific EEG channel (channel dropping) or specific spatial combinations (channel swapping). For example, this augmenter simulates EEG data in which an electrode is not available or it is not reliable (e.g., due to electrode pop), so it is zeroed in the EEG multi-variate activity.
v. Applying a perturbation to the signal amplitude, by multiplying signal amplitude by a constant. Here, the augmenter increases the variability of the EEG signal amplitude, simulating the physiopathological modulations of the amplitude of potential components [57] (e.g., due to inter- and inter-subject variability or due to cognitive deficiencies). This is achieved by increasing or decreasing the amplitude of existing signals (A) by a factor δA. That is, the amplitude of augmented signals (A^*) is $A^* = (1 + \delta A)\, A$.
vi. Applying a time shift δt to the signals. Similar to the previous augmenter, this one simulates the physiopathological modulations of the latency of potential components [57] (e.g., due to inter- and inter-subject variability or due to cognitive deficiencies) by anticipating or delaying the signal.
vii. Applying CutCat [58]. CutCat generates new EEG trials by cutting existing trials into different segments (at least 2 segments) and by randomly concatenating together segments belonging to different EEG trials. In other words, just by exploiting the inter-trial EEG variability, this augmenter generates new trials by concatenating portions of different trials together.

The previous data augmenters generate new time series by working on the original time series in the frequency domain (points i. and ii.) or in the time domain (from point iii. to point vii.). To increase the variability in data augmentation, the hyperparameters of data augmenters (e.g., the SNR used while injecting noise, the amplitude perturbation δA, the time shift δt, the number of segments in CutCat, etc.) are randomly sampled within specified ranges during input mini-batch processing. The parameters defining ranges for

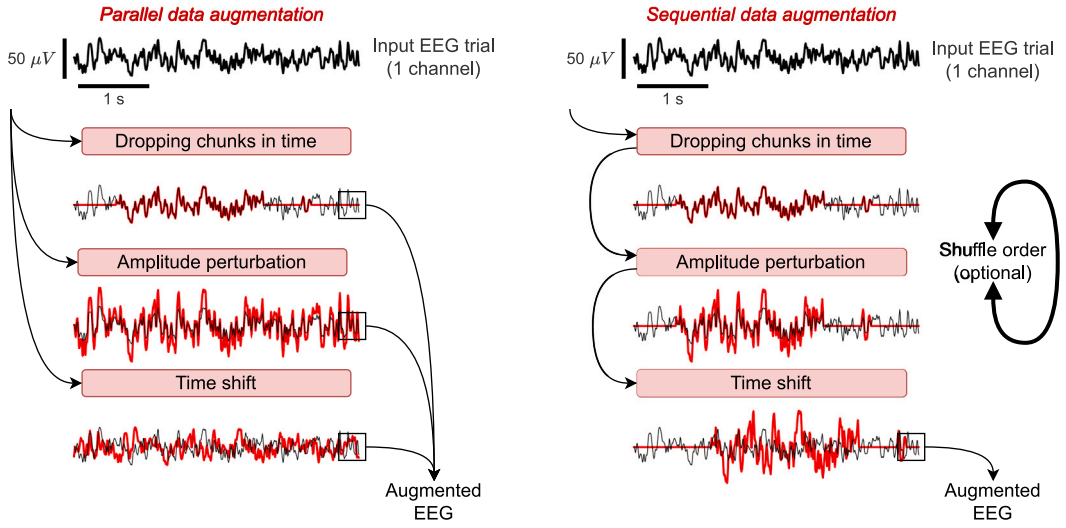

Fig. 3. Data augmentation: parallel and sequential augmentations. For these representations, a representative signal (Cz electrode) from BNCI2014-001 dataset [53] is used (4s-long EEG trial sampled at 125 Hz). A set of three augmenters (dropping randomly chunks of the signal, applying a random amplitude perturbation, and applying a random time shift) is used for illustrating how data augmentation works in SpeechBrain-MOABB. For each augmentation step, the original time series are reported (thin black lines) together with the augmented time series (red lines).

sampling augmenters' hyperparameters are hyperparameters them self, and should be tuned during hyperparameter search together with the other hyperparameters defining the decoding pipeline. Examples of these parameters are the low (SNR_{low}) and high (SNR_{high}) SNR values used while injecting noise for sampling the SNR in the interval $[SNR_{low}, SNR_{high}]$, the maximum absolute value of δA (δA_{max}) for sampling it in the interval $[-\delta A_{max}, \delta A_{max}]$, the maximum absolute value of δt (δt_{max}) for sampling it in the interval $[-\delta t_{max}, \delta t_{max}]$, and the maximum number of CutCat segments. The augmenters are defined in the scripts `speechbrain/augment/time_domain.py` and `speechbrain/augment/freq_domain.py`, respectively for augmenters that works in the temporal and frequency domains.

In SpeechBrain-MOABB the augmentation techniques can be applied either in parallel or sequentially, see Fig. 3 for a representative scheme, respectively in the left and right panels. In parallel augmentations, multiple versions of the original signal are generated, each augmented with a different augmentation technique (independent transformations). In sequential augmentations, augmenters are applied one after the other on the original signal, with shuffleable order (combined transformations). Sequential augmentations might introduce a higher variability in the generated synthetic EEG signals compared to the original ones. These might be preferred in applications in which different aspects of the EEG (e.g., amplitude and latency) change simultaneously, for example the reduced P300 amplitude and increased P300 latency in auditory and visuo-spatial tasks known reflecting deficiencies in cognitive, attentional, and working memory processes, when compared to healthy subjects [21,59]. Additionally, we encourage neuroscientists to consider sequential augmentations also from a computational prospective, as they are less computational demanding than parallel augmentations. This is an important aspect to consider, as data augmentation contributes to increasing the computational cost. Indeed, data augmentation for example increases the computational time of a single decoding experiment by 40.4%, on average across 9 MOABB datasets (see the experiments discussed in Section 4.4), with an increase ranging from 4.9% to 59.7%. Here, we have quantified the impact of data augmentation on the computational time by measuring the computational time variation (in percentage) between switching off and turning on data augmentation. To investigate more in detail data augmentations, let us consider the augmentations illustrated in Fig. 3 performed on a single EEG signal. Performing the augmentations in parallel requires a wall-clock time of 0.0029 ± 0.0006 s, while performing them in sequence requires 0.0026 ± 0.0004 s (mean ± standard deviation across 100 different runs of the random augmenters). Remarkably, the difference between these computational times is statistically significant ($p < 0.001$, pairwise Wilcoxon signed-rank test [60]). Moreover, as concerning the GPU memory required for data augmentation, let us denote with B the bytes occupied by the original input EEG trial on GPU. The GPU memory footprint for running N data augmenters in parallel is NB (for example $3B$ in Fig. 3-left panel, $N = 3$), as for each augmenter a new synthetic EEG trial is generated. On the other hand, the occupied GPU memory is always $2B$ in case of sequential augmentations, as only one new synthetic EEG trial is generated in total. Therefore, by using parallel augmentations the GPU memory footprint scales up linearly depending on the number of data augmenters, leading to higher GPU memory requirements than sequential augmentations when $N > 2$ augmenters are included. From these considerations, sequential augmentations are characterized by lower computational times and by a GPU memory footprint independent of the number of applied augmenters. Thus, we suggest neuroscientists to exploit sequential augmentations in case of limited computational resources available, for example limited computational time on high performing computing clusters or limited GPU memory. On-the-fly augmentation is applied to each mini-batch (i.e., no need for storing augmented data on disk), with a random selection of desired augmenters, either in parallel or sequence. The augmentation strategy is defined in the script `speechbrain/augment/augmenter.py`.

3.3. Neural network

The neural network architecture can be easily defined in a Pythonic way using PyTorch (see the code snippet in Fig. A.1 of Appendix). Models are stored in the `models/` folder as a separate Python file for each neural network. SpeechBrain-MOABB natively includes popular EEG models like EEGNet [15]. EEGNet is a prominent CNN for EEG decoding, known for its effectiveness across various tasks such as motor imagery, P300, and SSVEP decoding [17,19–24,32,33,48,61–66]. Notably, EEGNet-like architectures have consistently outperformed other CNNs, deep neural networks, and traditional machine learning methods in international scientific competitions [32,33]. The network's implementation is available in `models/EEGNet.py`. Furthermore, we also include ShallowConvNet [14] (in `models/ShallowConvNet.py`),

one of the first successful CNNs proposed for motor imagery decoding, and EEGConformer [38] (in `models/EEGConformer.py`), a recent transformer proposed for motor imagery and emotion decoding. Moreover, to facilitate the integration of SpeechBrain-MOABB with CNNs designed for braindecode [14], we also provide a dedicated class that enables the seamless integration and use of braindecode models within SpeechBrain-MOABB (in `models/BraindecodeNN.py`). See also the dedicated tutorial in the SpeechBrain-MOABB repository. Therefore, users have also the option to use braindecode models in SpeechBrain-MOABB, with the advantage of benefiting from the robust experimental protocol included in SpeechBrain-MOABB for ensuring a highly replicable and trustworthy EEG decoding (see Section 2.2 for a discussion on the strengths and pitfalls of the existing decoding tools).

Even though SpeechBrain-MOABB can be used with any network architecture (as it is architecture-agnostic), in our toolkit we primarily include models that reach state-of-the-art performance in the literature (see [7,11–13] for a review), to provide researchers the most relevant and promising available networks. These are CNN-based and transformer-based models. CNNs are included as they are considered a natural starting point for designing powerful decoders of multi-variate time series [67]. Moreover, CNNs are highly recommended over other deep neural networks (e.g., RNNs and hybrid convolutional-recurrent networks) for decoding the EEG recorded in the main BCI recording paradigms (e.g., motor imagery, P300, and SSVEP) [11]. Indeed, from the literature review presented in Section 1, CNNs are the most used deep neural networks for EEG decoding, providing the most accurate prediction compared to other deep neural networks [7,11–13,23,30,32,33]. Finally, we include transformer-based networks as they represent a recent advancement in the field of EEG decoding. Indeed, by including the attention mechanism in the network architecture [68], these networks accurately decode the EEG similar to CNNs [38].

Of course, many network hyperparameters define the network architecture, for example the number of temporal convolutional kernels of the first convolutional layer of EEGNet [15]. The collection of these hyperparameters constitutes the network architecture hyperparameters to tune during hyperparameter search, and depends on the specific architecture adopted in the decoding pipeline.

3.4. Network training and evaluation

Neural networks are randomly initialized and trained using the categorical cross-entropy as loss function. Here, the user can choose among PyTorch optimizers, optionally applying learning rate schedulers (e.g., cyclic learning rate [69]). Networks can be trained up to the epoch with best validation performance (suitable for enabling early stopping) or up to the last training epoch set (suitable for performing hyperparameter search also of the number of training epochs). Polyak averaging [70] is implemented, allowing the averaging of the network parameters over past epochs instead of returning the parameters from the final training epoch (temporal averaging of network parameters). This helps addressing convergence problems due to the noisy training process (characterizing mini-batch stochastic gradient descent), thus, it provides a more stable optimization solution. All aspects defining network training, such as the optimizer, the learning rate, the learning rate scheduler, the number of epochs averaged in Polyak averaging, etc., define the network training hyperparameters whose optimal values are found during hyperparameter search.

Finally, once trained the networks, the performance measures on the validation and test sets are returned. The training and evaluation loops of one cross-validation fold are wrapped into a convenient script `models/train.py`.

SpeechBrain-MOABB addresses the challenge posed by the statistical fluctuations in deep neural network performance resulting from random parameter initialization by exploiting multi-seed training and evaluation. This procedure consists in training and evaluating multiple times the decoding pipeline, each time using a different random seed for initializing the trainable parameters of the neural network. This way, multiple performance estimates are obtained (on the validation and test sets), each by using a different random seed while initializing networks. The final decoding performance is then computed as the average performance across the adopted random seeds. Here, we recommend to run experiments with 10 different random seeds, for obtaining stable decoding performance, with a performance variability consistently less than 0.5%, on average across 9 MOABB datasets [53,71–77] (see SpeechBrain-MOABB documentation for additional details and results).

3.5. Hyperparameters

SpeechBrain-MOABB conveniently exposes all the main hyperparameters affecting the entire decoding pipeline in a single file containing all hyperparameters. To do so, SpeechBrain-MOABB uses an enhanced version of YAML called HyperPyYAML. The hyperparameter file handles all the main operations of the decoding pipeline, including hyperparameters for data pre-processing (e.g., the sampling frequency during downsampling), data augmentation (e.g., the ranges used in data augmenters), network architecture (e.g., the number of convolutional kernels) and network training (e.g., the learning rate). It goes beyond associating hyperparameters with standard data types, also allowing the linkage to complex Python objects, such as the definition of the target MOABB dataset to use (e.g., the widely used BNCI2014-001 dataset [53]) and of the neural network (e.g., EEGNet). In Fig. A.2 of Appendix we provide a snippet of a hyperparameter file, showing a representative example for defining data pre-processing with the BNCI2014-001 dataset [53] and for defining EEGNet as neural network.

Conveniently, in SpeechBrain-MOABB the YAML-file comments can be used to identify the hyperparameters to be considered in hyperparameter search and to define their search space (see Section 3.6 for further details about hyperparameter search).

For instance, to search for the optimal dropout rate the comment could be set as follows:

```
dropout: 0.25 # @orion_step1: --dropout 'uniform(0.0, 0.5)'.
```

The comment should specify the step of the multi-step search ('orion_step1' for the first step), the hyperparameter name ('dropout'), the distribution for sampling values during optimization ('uniform' for using a uniform distribution), and the range for searching the hyperparameter ('(0.0, 0.5)'), with the syntax indicated in the previous example. See later Section 3.6 for details about hyperparameter search. The user can include as many optimization steps as desired by simply using the tags 'orion_step2', 'orion_step3', etc. This approach allows users to conveniently specify in a single file not only the hyperparameters of the entire decoding pipeline (at specific values or Python objects, see Fig. A.2 of Appendix), but also to mark the ones to optimize during hyperparameter search, defining their search space and the optimization strategy (single-step or multi-step, see Section 3.6 for further details). This ensures both replicability and simplicity – even of advanced aspects like hyperparameter search settings – when addressing EEG decoding via deep learning-based pipelines. Moreover, this strategy ensures an easy share of decoding pipelines, as neuroscientists can simply share one single file with the scientific community, overall, to fully cover the entire experimental decoding setup.

Remarkably, SpeechBrain-MOABB comes with a set of optimal hyperparameters (characterizing from data pre-processing, augmentation, to network architecture and training) for decoding various EEG datasets with different models. Specifically, we provide the optimal hyperparameters when using EEGNet [15], ShallowConvNet [14], and EEGConformer [38] for decoding the following MOABB datasets, recorded for motor imagery, P300, and SSVEP BCIs: BNCI2014-001 [53], BNCI2014-004 [71], BNCI2015-001 [72], Lee2019-MI [73], Zhou2016 [74], BNCI2014-009 [75], EPFLP300 [76], BI2015a [77], and Lee2019-SSVEP [73]. These optimal hyperparameters are obtained while processing a total of 204 participants and 26 recording sessions. The

optimal hyperparameter files (one per dataset and neural network) are contained in the `hparams/` folder, and are organized depending on the nature of the BCI paradigm that was adopted for recording EEG signals (i.e, motor imagery, P300, and SSVEP paradigms). These files (containing the optimal hyperparameters for various BCI datasets) resulted from computations performed on high-performance computing clusters with an Intel Gold 6148 Skylake CPU (2.4 GHz), an NVIDIA V100 GPU (16 GB of memory) and 12 GB RAM.

3.6. Hyperparameter search

The goal of hyperparameter search is to discover good hyperparameters for a decoding pipeline, yielding the best performance as assessed on an independent validation set. SpeechBrain-MOABB relies on the Oríon library [78], an asynchronous framework for black-box function optimization. In each iteration of the optimization procedure, a different hyperparameter configuration is sampled and used to train and evaluate the decoding pipeline. The validation performance to optimize is computed as the average validation performance across cross-validation folds and across participants (in case of leave-one-session-out iteration scheme).

Common search algorithms exploit a pre-defined rule to sample hyperparameters, e.g., sampling each possible hyperparameter configuration (grid search) or randomly sampling a fixed number of hyperparameter configurations (random search [79]). On the contrary, sequential model-based search algorithms [80] maintain a record of past validation performance. They update a probabilistic model, such as a tree-structured parzen estimator (TPE), to inform the selection of the next hyperparameter configuration to try. SpeechBrain-MOABB supports all these search modalities, with sequential model-based search algorithm (TPE-based) being the default choice due to its proven superiority over grid search and random search [80,81]. In SpeechBrain-MOABB we offer configuration files for properly setting the search algorithm in the YAML files contained in the `hparams/orion/` folder (e.g., `hparams_tpe.yaml` for TPE).

As highlighted widely across Sections 3.2.1, 3.2.3, 3.3, and 3.4 while describing data pre-processing, data augmentation, network architecture and training, the definition of a complete EEG decoding pipeline requires to set many hyperparameter values. SpeechBrain-MOABB addresses the challenge posed by searching hyperparameters in high-dimensional search spaces (i.e., defined by more than 10–20 hyperparameters [42]) through its support for multi-step hyperparameter search. This approach divides the search into sequential steps, each operating on a low-dimensional sub-space sampled from the original search space. Performing sequential searches on low-dimensional sub-spaces of the entire search space instead of a single search on it, should prevent finding sub-optimal hyperparameter values due to the high-dimensionality of the entire search space [82]. Multi-step hyperparameter search has proven to strike a favorable balance between performance and computational complexity. Specifically, we recommend a 2-step process. During the first step, data pre-processing, network architecture, and network training hyperparameters are optimized, without performing data augmentation. In the second step, data augmentation is enabled and only its hyperparameters are optimized.

Despite the aforementioned measures, hyperparameter search can anyway be computationally expensive, especially for large datasets with numerous participants and sessions. Indeed, for example when using a leave-one-session-out iteration scheme, the validation performance is extracted by leveraging on all the trained neural networks across cross-validation folds (i.e., held-out sessions) and across participants. To mitigate this issue, SpeechBrain-MOABB supports hyperparameter search on a subset of data (i.e., a subset of participants). Surprisingly, optimizing hyperparameters only on 3–5 participants results in optimal hyperparameter values that perform well across other participants. This is the default choice in SpeechBrain-MOABB, which makes it feasible to find a good set of hyperparameters with manageable computational costs, for example with a wall-clock time of approx. 2.3 days (using a subset of 3–5 participants) vs. 13.4 days (using all participants), on average across 9 MOABB datasets [53,71–77] (see also Section 4.4). To further speed up hyperparameter search, SpeechBrain-MOABB conveniently caches datasets obtained with a specific data pre-processing applied. Indeed, data pre-processing can require a few minutes to be performed on large datasets for each search iteration (e.g., on Lee2019-MI [73] MOABB dataset), thus affecting the overall computational time required for hyperparameter search.

It is important to note that the key aspects of the proposed experimental protocol, such as the type of hyperparameter search algorithm, number of hyperparameter optimization steps of multi-step search, number of participants for hyperparameter search, and number of random seeds to use for network initialization during multi-seed training and evaluation (see Section 3.4), have been object of an extensive experimental validation. This validation was performed on 9 MOABB datasets [53,71–77] recorded during motor imagery-based, P300-based, and SSVEP-based BCI paradigms. For brevity, we present experimental results covering the main aspects of the protocol, while the full battery of experiments is available in the documentation of the referenced SpeechBrain-MOABB repository.

3.7. Command-line interfaces (CLIs)

SpeechBrain-MOABB includes two scripts providing CLIs to users for easily benchmarking neural networks. These CLIs are of central importance for users, as they interact directly with SpeechBrain-MOABB by using them. In Table 2 we summarize the main properties and use case scenarios of the CLIs, to ease their use by neuroscientists, and in the following we describe them.

i. `run_experiments.sh`: This command executes a complete EEG decoding experiment with assigned hyperparameters, supporting multi-seed training and evaluation. It is suitable for neuroscientists who desire to easily replicate existing state-of-the-art pipelines, shared by another research group (e.g., replicate a benchmark). The script performs multiple calls at the `train.py` script depending on the selected data iterator (leave-one-session-out or leave-one-participant-out), iterating over cross-validation folds. The number of participants, sessions, and the data iterator should be specified (options 'nsbj', 'nsess', and 'train_mode', respectively). This CLI supports multi-seed training and evaluation by setting the number of runs to perform (option 'nruns'), each by using a different seed for initializing the trainable parameters of the neural network. For example, assuming that a decoding pipeline is defined in the hyperparameter file `hparams/sample_hparams.yaml`, the user can simply use this CLI as reported in Fig. 4a. Additionally, we also provide a detailed dedicated tutorial in the SpeechBrain-MOABB repository.

ii. `run_hparam_optimization.sh`: This command performs multi-step hyperparameter search, supporting multi-seed training and evaluation. It is ideal for neuroscientists looking to adapt a state-of-the-art pipeline from one decoding problem to another (e.g., transposing a network proposed for P300 decoding to SSVEP decoding) or for researchers aiming to design and benchmark a novel decoding pipeline testing new algorithms (e.g., a new neural network or a new data augmentation strategy). This CLI introduces multi-step hyperparameter search by performing multiple sequential hyperparameter searches. Each search iteration is conducted by calling the command `run_experiments.sh` with a different hyperparameter configuration. Therefore, most options are shared with the previous CLI. Additionally, the user should specify the options related to hyperparameter search, such as the configuration file

Table 2
SpeechBrain-MOABB command-line interfaces: main properties and use case scenarios for neuroscientists.

Command-line interface	Main properties	Neuroscientist use case
`run_experiments.sh`	– Performs a complete EEG decoding experiment on all subjects and sessions – Exploits a fixed hyperparameter configuration (no hyperparameter search) – Supports multi-seed training and evaluation	– Replicate existing state-of-the-art pipelines, shared by another research group (e.g., replicate a benchmark) – Post-hoc hyperparameter evaluations, that is, manually test variant hyperparameter configurations by changing one hyperparameter at a time of a baseline configuration (e.g., using a number of convolutional kernels in the network different from the one used by another research group)
`run_hparam_optimization.sh`	– Performs a complete EEG decoding experiment on all subjects and sessions – Performs multi-step hyperparameter search (the optimal hyperparameters are searched within the search space) – Supports multi-seed training and evaluation during both hyperparameter search and the final training and evaluation (using the optimal hyperparameters found)	– Adapt a state-of-the-art pipeline from one decoding problem to another (e.g., adapting a network proposed for P300 decoding to SSVEP decoding) – Design and benchmark a novel decoding pipeline testing new algorithms (e.g., a new neural network)

(a)

Command-line interface 1: pipeline with known hyperparameters supporting multi-seed training and evaluation

```
./run_experiments.sh --hparams 'hparams/sample_hparams.yaml' \
--data_folder '/path/to/data' \ # data folder
--cached_data_folder '/path/to/cache' \ # cache folder
--output_folder '/path/to/results' \ # results folder
--nsbj 9 --nsess 2 --nruns 10 --train_mode 'leave-one-session-out' \
--device 'cpu' # by default run on GPU
```

(b)

Command-line interface 2: pipeline integrating multi-step hyperparameter search and multi-seed training and evaluation

```
./run_hparam_optimization.sh --hparams 'hparams/sample_hparams.yaml' \
--data_folder '/path/to/data' \ # data folder
--cached_data_folder '/path/to/cache' \ # cache folder
--output_folder '/path/to/results' \ # results folder
--nsbj 9 --nsess 2 --nruns 1 --train_mode 'leave-one-session-out' \
--device 'cpu' # by default run on GPU \
--config_file 'hparams/orion/hparams_tpe.yaml'
--nsbj_hpsearch 3 --nsess_hpsearch 2 \
--eval_metric 'acc' \
--exp_max_trials 50 \
--nruns_eval 10
```

Fig. 4. SpeechBrain-MOABB command-line interfaces (CLIs). Panel a – CLI for experiments involving a pipeline with known hyperparameters, supporting multi-seed training and evaluation. Panel b – CLI for experiments involving a pipeline integrating multi-step hyperparameter search, and supporting multi-seed training and evaluation.

defining the search algorithm (option 'config_file'), the number of participants and sessions to use (options 'nsbj_hpsearch' and 'nsess_hpsearch', respectively), the evaluation metric to optimize (option 'eval_metric', e.g., set to 'acc' for accuracy), and the number of iterations to perform in each step of the multi-step search (option 'exp_max_trials'). Once the search is concluded, this CLI additionally performs a final training and evaluation using the optimal hyperparameters found. Multi-seed random initialization is supported. The user can specify how many times train and evaluate the pipeline with a different random seed at each iteration of the hyperparameter search ('nruns' option) and how many times re-train and re-evaluate the pipeline using the optimal hyperparameters found during the search in the final training and evaluation stage ('nruns_eval' option). For example, assuming that a pipeline is defined in the hyperparameter file `hparams/sample_hparams.yaml`, containing also the search space for the hyperparameters to tune, the user can use this CLI as reported in Fig. 4b. Additionally, we also provide a detailed dedicated tutorial in the SpeechBrain-MOABB repository.

Of course, the second CLI – involving multi-step hyperparameter search – requires more computational time to be executed, compared to the first CLI. For example, the first CLI requires approx. 18 h to complete an experiment on the BNCI2014-001 dataset [53] (10 random seeds exploited in multi-seed training and evaluation). On the other hand, the second CLI requires approx. 70 h to complete an experiment on the same dataset (2-step hyperparameter search performed for 100 iterations, with 10 random seeds in multi-seed training and evaluation).

In the following section, we show how conduct experiments with SpeechBrain-MOABB on a representative EEG dataset, and we report and discuss the main results obtained with the proposed toolkit.

4. Experimental results and discussion

Conducting a comprehensive EEG experiment with SpeechBrain-MOABB requires a few steps, including (i) dataset and data handling definitions, (ii) network architecture and training definitions, (iii) hyperparameter search space definition, (iv) running hyperparameter search followed by final training and evaluation, and (v) analyzing the performance. This section will go through each of these steps for illustrating to neuroscientists how to perform them on one representative EEG dataset. Additionally, a comparative analysis with the results obtained using pipelines based on MOABB and braindecode will be presented, to show the effective decoding potentialities of our toolkit compared with the state-of-the-art libraries. Finally, as an additional contribution, we report benchmark results obtained with SpeechBrain-MOABB on 9 datasets with a total of 204 participants and 26 recording sessions, using different deep neural networks (EEGNet, ShallowConvNet, and EEGConformer).

4.1. Experiment definition

4.1.1. Dataset and data handling

The initial step involves selecting a dataset from the available options in MOABB. Here, we focus on decoding motor imagery from the EEG using the BNCI2014-001 dataset [53], commonly known as 'BCI IV2a'. Widely recognized in the literature as a primary benchmark for assessing the efficacy of new neural networks in EEG applications, this dataset has been extensively employed in previous studies [7]. The BNCI2014-001 dataset comprises 22-channel EEG recordings from 9 healthy participants conducted over 2 recording sessions. Signals are sampled at 250 Hz and band-pass filtered between 0.5 and 100 Hz. Electrodes are placed according to the 10–10 international system at Fz, FC3, FC1, FCz, FC2, FC4, C5, C3, C1, Cz, C2, C4, C6, CP3, CP1, CPz, CP2, CP4, P1, Pz, P2, POz. The task involves the imagination of four distinct movements for 4 seconds: left hand, right hand, feet, and tongue. For each participant and each session, 288 trials were recorded, ensuring a balanced distribution across classes. See also Section 4.4 for further details about the main properties of this dataset.

The dataset is processed by applying the parametrized pre-processing outlined in Section 3.2.1. Signals are downsampled to 125 Hz (i.e., this hyperparameter is not tuned in these experiments). A leave-one-session-out data iterator is employed, extracting training and validation sets using a 80%–20% ratio within each cross-validation fold, see Section 3.2.2. Finally, the dataset is augmented by applying four data augmentation techniques among the ones presented in Section 3.2.3: injection of white Gaussian noise and application of amplitude perturbation, time shift, and CutCat.

4.1.2. Network architecture and training

The next step involves defining a model, as discussed in Section 3.3. SpeechBrain-MOABB pipelines are crafted to be model-agnostic, enabling users to easily integrate their own model. Here we showcase the toolkit's capabilities by employing EEGNet [15], a widely recognized model already pre-implemented in SpeechBrain-MOABB (please refer to Lawhern et al. [15] for further details about this neural network). After implementation, it is essential to incorporate the model as an entry in the hyperparameter file, as described in Section 3.5. Moreover, it is necessary to specify network training, as defined in Section 3.4. Specifically, in these experiments we adopt Adam [83] as optimizer, with cyclic learning rate annealing using a triangular function [69]. Finally, the target evaluation metrics need to be specified. Here, the accuracy serves as the primary performance metric, as the BNCI2014-001 dataset [53] is class-balanced.

4.1.3. Hyperparameter search space

The hyperparameter file not only specifies hyperparameters for data pre-processing, data augmentation, network architecture, and network training, but also marks the hyperparameters to tune during hyperparameter search, defining their search space and optimization strategy (e.g., 2-step search). This is accomplished using the notation style detailed in Section 3.5. In the experiments reported here, we search for the optimal values – in terms of validation accuracy – of the hyperparameters listed in Table 3 (column 'Hyperparameter'). The adopted search space is outlined in Table 3 (columns 'Search space' and 'Options'). It is crucial for users to define sufficiently large ranges to enable hyperparameter search algorithms to sample a diverse set of configurations. This step necessitates users to provide reasonable ranges based on their prior knowledge of the problem (e.g., range of cut-off frequencies for filtering EEG signals). In this case, we selected the ranges for these hyperparameters by considering the values used in prior studies (column 'Related works' of Table 3).

4.1.4. Hyperparameter search

After compiling the hyperparameter file – comprising the specification of the relevant hyperparameters and of the hyperparameters to tune with their search space – users can run a comprehensive experiment using the `run_hparam_optimization.sh` script (SpeechBrain-MOABB CLI described in point (ii) of Section 3.7). This script orchestrates all necessary steps to implement the proposed experimental protocol, including a multi-step hyperparameter search and multi-seed final training and evaluation. Specifically, here we conduct a 2-step search (with the final step focusing on data augmentation hyperparameters), performing 50 iterations for each search step (100 iterations in total). The hyperparameter optimization is performed via sequential model-based search using TPE as surrogate model.

The last column of Table 3 reports the results obtained with SpeechBrain-MOABB after hyperparameter search (bold values in the table). Specifically, it contains the optimal hyperparameter values for decoding motor imagery, for the definition of data pre-processing, augmentation, network architecture and network training. Interestingly, among the identified optimal hyperparameters, the optimal EEG pre-processing includes band-pass filtering between 0.13–46 Hz, utilizing all time samples (from 0 to 4 s), and focusing on a subset of 16 neighbor channels (corresponding to $C_{step} = 2$) around Cz (17 channels used in total, refer to Fig. 2 for the channel subset corresponding to $C_{step} = 2$ for the same dataset). The optimal decoding performance is achieved by using fewer channels concentrated over the sensorimotor cortex, rather than all available channels, and the entire epoch window with a spectral content covering all EEG bands up to low-γ. This approach contrasts with the one adopted in Lawhern et al. [15] in which manually selected hyperparameter values (sub-optimal hyperparameter configuration) for preparing EEG signals were used with EEGNet, including band-pass filtering between 4–40 Hz, time samples up to 2.5 s, and using all 22 channels. Notably, our optimal data pre-processing matches with the recent literature about motor correlates, reporting that also EEG frequencies other than α (8–12 Hz) and β (12–30 Hz) [93], such as δ (up to 4 Hz) and low-γ (30–50 Hz), encode movement-related information both in motor execution and motor imagery [17,23,94–96]. Thus, our results support the current theories and results on the involvement of slow and fast oscillations in movement imagery. Moreover, the decoding pipeline used in SpeechBrain-MOABB exploited a subset of 17 electrodes over the sensorimotor cortex – as this set resulted the more optimal one during hyperparameter search – vs. all available electrodes (22 electrodes) used in Lawhern et al. [15]. Therefore, SpeechBrain-MOABB addressed EEG decoding in a more parsimonious way in terms of EEG setup required, that could be useful in prospective for helping reducing user preparation times in EEG-based BCIs.

Once identified the optimal set of hyperparameters on the validation set, the final training and evaluation is performed by re-training and re-evaluating the decoding pipeline with the optimal hyperparameters

Table 3
Overview of hyperparameters along with their search spaces, probability distributions, related works exploited for setting the search spaces, and optimal values achieved with hyperparameter search. The search space for each hyperparameter is defined using square brackets for intervals and curly brackets for sets of accepted values. Uniform probability distributions, sampling floating-point values, integer values (using the 'int' option), or a selection of values (using the 'choice' option) are considered. Note that, kernel and pool sizes are indicated as tuples (with round brackets), as they are defined for 2-D operators.

	Hyperparameter	Search space	Options	Related works	Optimal value
Data pre-proc.	Low cut-off frequency (f_0, Hz)	[0.1, 5]	uniform	[14,15,17,62,63]	**0.13**
	High cut-off frequency (f_1, Hz)	[20, 50]	uniform	[14,15,17,62,63]	**46.0**
	Trial upper limit (t_1, s)	[1, 4]	uniform	[14,15,17,62,63]	**4.0**
	Spatial sampling distance (C_{step})	[1, $C_{step,max}$ = 3]	uniform, int	[84–86]	**2 (C = 17)**
Data augm.	Max. no. of CutCat segments	[2, 6]	uniform, int	[58,87]	**3**
	Max. absolute amplitude perturbation (δA_{max})	[0, 0.5]	uniform	[56,88,89]	**0.017**
	Max. absolute time shift (δt_{max}, s)	[0, 0.25]	uniform	[56,88]	**0.010**
	White Gaussian noise low-end SNR (SNR_{low}, dB)	[0, 15]	uniform	[56,86,88–90]	**15.0**
	White Gaussian noise high-end SNR (SNR_{high}, dB)	SNR_{low}+[5, 20]	uniform	[56,86,88–90]	**34.1**
EEGNet arch.	No. of temporal conv. kernels (K_0)	[4, 64]	uniform, int	[15,17,19–24,32,33,48,61–66]	**61**
	Temporal conv. kernel size	([24, 62], 1)	uniform, int	[15,17,19–24,32,33,48,61–66]	**(51, 1)**
	Spatial conv. depth multiplier (D_1)	[1, 4]	uniform, int	[15,17,19–24,32,33,48,61–66]	**4**
	No. of temporal separable conv. kernels	[1, $2K_0D_1$]	uniform, int	[15,17,19–24,32,33,48,61–66]	**428**
	Temporal separable conv. kernel size	([3, 24], 1)	uniform, int	[15,17,19–24,32,33,48,61–66]	**(15, 1)**
	Temporal pooling size (after temporal separable conv.)	([1, 8], 1)	uniform, int	[15,17,19–24,32,33,48,61–66]	**(7, 1)**
	Dropout probability	[0, 0.5]	uniform	[15,17,19–24,32,33,48,61–66]	**0.008**
Training	Learning rate (max. value during annealing)	{0.01, 0.005, 0.001, 0.0005, 0.0001}	uniform, choice	[15,17–24,32,33,48,61–66]	**0.0001**
	Mini-batch size	{16, 32, 64}	uniform, choice	[15,17–24,32,33,48,61–66]	**16**
	No. of epochs	[250, 1000]	uniform	[15,17–24,32,33,48,61–66]	**862**
	No. of averaged models (Polyak averaging)	[1, 15]	uniform	[70,91,92]	**10**

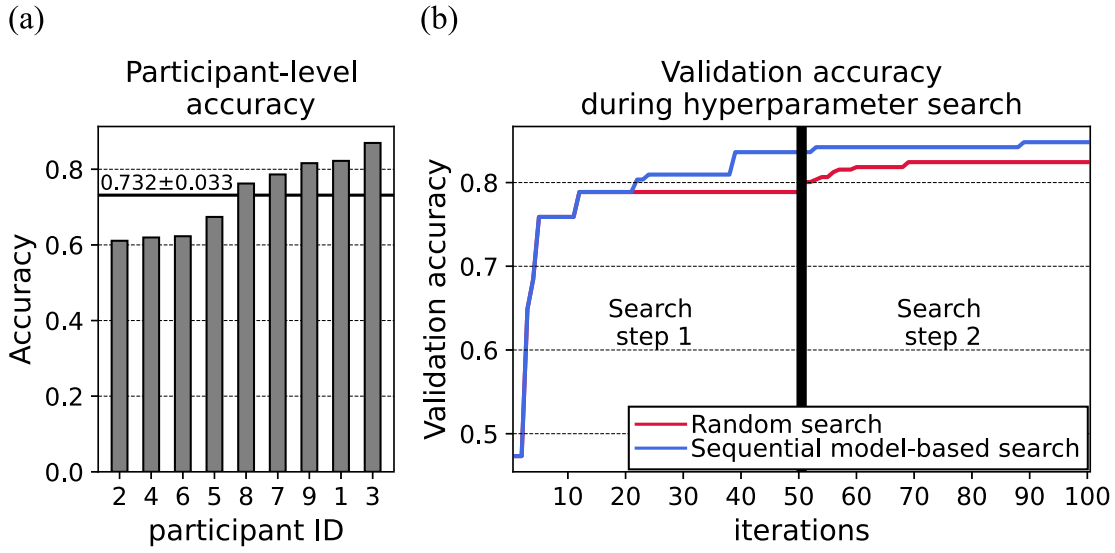

Fig. 5. Decoding accuracy scored with a decoding pipeline based on EEGNet, as obtained with SpeechBrain-MOABB. Panel a – Participant-level accuracy values. Here, the test accuracy is averaged across cross-validation folds (i.e., held-out sessions, 2 in total) and across the used random seeds during multi-seed training and evaluations (10 in total). The horizontal black line denotes the mean accuracy across participants. Panel b – Validation accuracy dynamics during hyperparameter search. For each iteration, the best validation performance reached up to that iteration is reported. The validation performance is displayed separately for a search conducted by using sequential model-based search (blue) or random search (red) as hyperparameter search algorithm.

10 times, each time using a different random seed for the network initialization (i.e., 10-seed final training and evaluation). For each participant, the performance on the test set is computed for each cross-validation fold. Then, the participant-level performance metric is obtained by averaging metrics across the used random seeds, and across folds.

4.2. Performance analysis

After the completion of the experimental decoding protocol, users can analyze the test set performance achieved by their decoding pipelines. The performance achieved with EEGNet using SpeechBrain-MOABB is presented in Fig. 5a, separately for each participant (ranked from the lowest to the highest). Here, the average accuracy across cross-validation folds and random seeds used during multi-seed training and evaluation is reported. EEGNet achieves a decoding accuracy of 73.2 ± 3.3% (mean ± standard error of the mean across participants), well above the chance level (25%) for each participant. Additionally, users can also analyze the dynamics of the validation set performance during the hyperparameter search (i.e., as a function of the iterations of hyperparameter search), which is shown in Fig. 5b (blue line). As can be observed, the validation accuracy reaches higher values during the second step (from iteration no. 51 to 100), in which data augmentation is enabled and optimized in its hyperparameters, compared to the first step (from iteration no. 1 to 50), in which data augmentation is disabled. This is observed also by using a hyperparameter search algorithm different from sequential model-based search (random search, red line), suggesting a positive effect on the decoding performance operated by data augmentation.

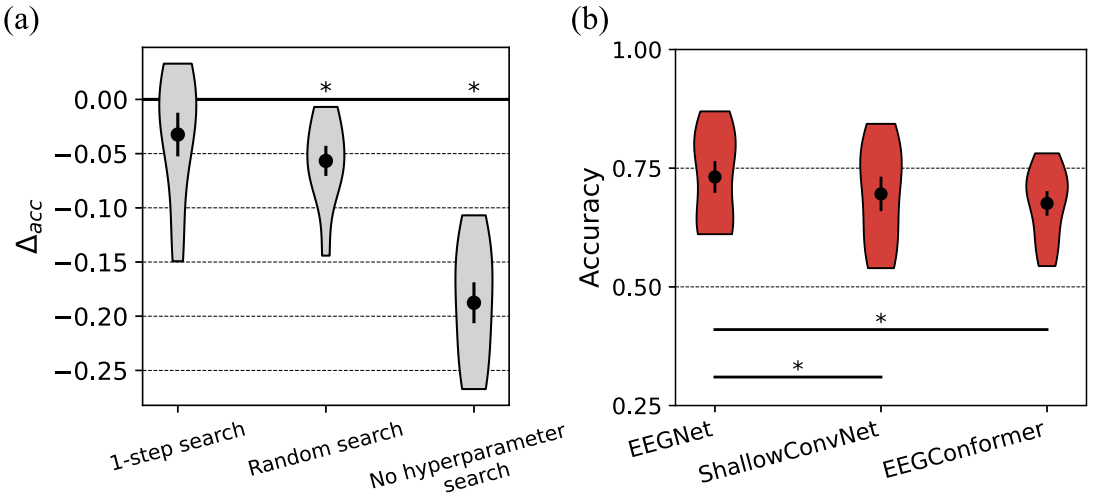

Fig. 6. Effect of different SpeechBrain-MOABB protocol's key components and of different neural networks on the test accuracy. Panel a – Ablation tests on key components of the protocol (grey distributions). We consider three variant protocols, defined by changing one component at a time as follows: (i) searching hyperparameters in the entire search space at once (i.e., '1-step search', without using a 2-step search); (ii) utilizing 'random search' instead of sequential model-based search; (iii) exploiting fixed hyperparameters, without optimizing them ('no hyperparameter search'). Here, we report the accuracy difference between each variant protocol and the proposed protocol (see Fig. 5a), adopting an ablation procedure similar to prior studies [14,19,25,38]. Statistically significant ($*$ $p < 0.05$) differences are marked. Panel b – Performance comparison using decoding pipelines based on different neural network architectures (red distributions) that reach state-of-the-art performance in the literature [7,11–13,32,33,38]. The results from the performed statistical analysis are reported too ($*$ $p < 0.05$), by marking only the comparisons that resulted significant among all possible combinations of decoding pipelines. Distributions are shown as violin plots; black dots indicate mean values and error bars represent standard errors of the mean, computed across participants.

For comparative analysis, in Fig. 6 we also include an ablation analysis on the key settings of the proposed decoding protocol defining hyperparameter search (see Section 3.6), and a comparative analysis of the decoding accuracy scored with different network architectures. As concerning the ablation analysis, in Fig. 6a we report the accuracy difference between a variant protocol defined by changing one setting of the proposed protocol at a time and the proposed protocol (characterized by the performance distribution shown in Fig. 5a). A similar procedure was exploited in prior EEG studies while conducting ablation analyses for evaluating the impact on the decoding performance of specific decoding choices [14,19,25,38]. We consider a variant protocol that (i) searches hyperparameters in the entire search space at once (1-step search, instead of 2-step search), (ii) searches hyperparameters utilizing random search (instead of sequential model-based search), and (iii) exploits fixed hyperparameters based on a prior study [15] (no hyperparameter search). To compare the results, we perform a pairwise Wilcoxon signed-rank tests [60], contrasting the performance scored by each variant protocol with the proposed protocol. P-values are corrected for multiple tests (a total of 3) using the Benjamini–Hochberg procedure [97]. Our results reveal that the hyperparameter search settings employed in our protocol provides higher accuracy than variant protocols (significantly different in 2 out of 3 ablation tests, $p < 0.05$). Indeed, searching hyperparameters without employing a multi-step search or without exploiting sequential model-based search worsened the performance by 3.2% and 5.7%, respectively, and switching off hyperparameter search (i.e., exploiting a fixed hyperparameter configuration) worsened the performance by 18.8%. It is worth highlighting that the superiority of a sequential model-based search was already evident from the results shown in Fig. 5b on the validation set. Indeed, utilizing random search, the decoding protocol achieved a lower validation accuracy than sequential model-based search within each hyperparameter search step (step 1 and step 2). As concerning the accuracy comparison with different network designs, we include in Fig. 6b the results obtained with the networks that reach state-of-the-art performance in the literature [7,11–13,32,33,38] (see also Sections 1 and 3.3). These are CNN-based (EEGNet and ShallowConvNet) and transformer-based (EEGConformer) networks. ShallowConvNet-based and EEGConformer-based pipelines are designed by adopting an equivalent decoding pipeline as the one illustrated above for EEGNet (from Section 4.1.1 to Section 4.1.4) and only changing the network used in the pipeline. To compare the different pipelines, we perform a pairwise Wilcoxon signed-rank tests [60] testing all possible combinations (i.e., EEGNet vs. ShallowConvNet, EEGNet vs. EEGConformer, and ShallowConvNet vs. EEGConformer). P-values are corrected for multiple tests (a total of 3) using the Benjamini–Hochberg procedure [97]. From Fig. 5b, our results reveal that EEGNet significantly outperforms both other models ($p < 0.05$), including the recent model based on a transformer. Moreover, in Section 4.4 we also report results obtained while extending this benchmark obtained with SpeechBrain-MOABB on the BNCI2014-001 dataset also on other 8 MOABB datasets – with a total of 204 participants and 26 recording sessions – using EEGNet, ShallowConvNet and EEGConformer. From our benchmark, EEGNet is consistently superior than the other models across the different datasets.

It is fair to note that a previous study on motor imagery decoding [38] reported that EEGConformer achieves superior performance than EEGNet and ShallowConvNet, contrasting our results. However, the experimental protocol applied by the authors considered only one fixed training-test split of the EEG dataset to estimate decoding performance, did not involve hyperparameter search (i.e., they used one fixed hyperparameter configuration, manually set), and performed network trainings and evaluations with only one random seed. On the other hand, in our robust experimental protocol we employed leave-one-session-out cross-validation (thus, we averaged performance across folds), we searched for the optimal hyperparameters characterizing the entire decoding pipeline, and we trained and evaluated networks 10 times with different random seeds to address the fluctuations of the performance estimates (thus, we averaged performance across the 10 different trained networks). As regarding the fluctuations in performance estimates, in order to quantify them, in Fig. 7 we report the accuracy variability obtained while training and evaluating networks using only one random seed (as done in Song et al. [38]), separately for each participant, cross-validation fold (i.e., held-out session), and neural network (EEGNet, ShallowConvNet, and EEGConformer). In this figure, we report the standard deviation of the accuracy across the 10

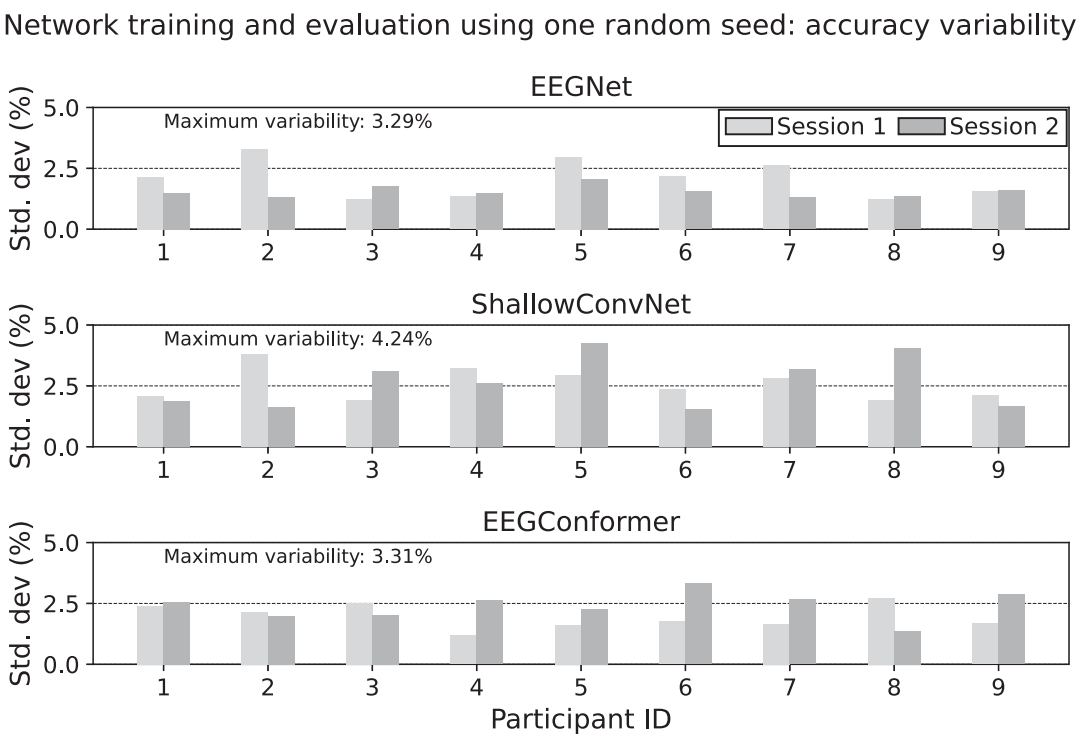

Fig. 7. Variability of the decoding accuracy due to the random initialization of network parameters. Here, the standard deviation of the accuracy across the 10 random seeds used during multi-seed training and evaluation is reported, separately for each participant, cross-validation fold (i.e., held-out session 1 and session 2, respectively reported in light and dark grey), and neural network adopted in the decoding pipelines (EEGNet, ShallowConvNet, and EEGConformer, reported in different panels). The variability is displayed as bar plots. The bar height represents the standard deviation of the decoding accuracy. When training and evaluating networks utilizing only one random seed, the maximum performance variability is approx. 4.2%.

runs performed during multi-seed training and evaluation, each using a different random seed. From these results, by exploiting only 1 random seed, the maximum variability in the decoding accuracy is approx. 4.2%, overall. When performing multi-seed training and evaluation, and considering the average accuracy across different random seeds used during initialization, the performance variability reduces due to the averaging procedure. Indeed, for example by considering the average of the performance across 10 random seeds, in preliminary experiments we found that the variability was consistently below 0.5% (see the SpeechBrain-MOABB documentation). Therefore, the eventual differences that can be found with prior literature – in this case compared to Song et al. [38] but also to other machine learning studies [98,99] – are primarily due to the different definition of the experimental protocol, here defined with the aim of providing a more trustworthy, transparent and replicable EEG decoding.

Crucially, users can replicate our results by running our decoding protocol from scratch (hyperparameter search, final training, and evaluation) using the SpeechBrain-MOABB CLI described in point (ii) of Section 3.7 (`run_hparam_optimization.sh` script), or by executing training and evaluation with the optimal hyperparameter files provided in the `hparams/` folder using the CLI described in point (i) of Section 3.7 (`run_experiments.sh` script). Alternatively, users have the option to leverage our saved models, conveniently provided in our repository.

4.3. Performance comparison with existing libraries

We now evaluate the performance of pipelines based on SpeechBrain-MOABB against machine learning and deep learning pipelines implemented with MOABB and braindecode on the same dataset (BNCI2014-001 dataset [53]).

The considered MOABB machine learning pipelines (6 in total) include popular approaches like common spatial pattern and linear discriminant analysis (*CSP+LDA*, and *regCSP+shLDA*) [100,101], Riemannian geometry classifiers such as Fisher geodesic minimum distance to the mean (*FgMDM*) [102], minimum distance to the mean (*MDM*) [103], and tangent space combined with logistic regression or support vector machine (*TS+LR*, and *TS+SVM*) [104]. We also exploit a pipeline in which the hyperparameters (SVM kernel and regularization term) were determined via hyperparameter search performed via grid search (TS+SVM).

As concerning deep learning pipelines, we combine MOABB with braindecode (MOABB+braindecode condition) for performing a comprehensive comparison with existing deep learning-based solutions. We include the following deep learning models (9 in total), spanning across diverse network architectures:

- Convolutional neural networks. *ShallowConvNet* [14] – the first successful CNNs proposed for decoding motor imagery – is included in the comparison, together with *EEGNet* [15]. Remarkably, EEGNet-like architectures consistently outperformed other deep neural networks (CNNs, RNNs and hybrid convolutional-recurrent networks) during international scientific competitions [32,33]. Additionally, we also use *EEGInception* [34] and *EEGITNet* [31], two recent CNNs that exploit inception modules for learning temporal features at multiple time scales (i.e., parallel temporal convolutions).
- Recurrent neural networks. As discussed in Sections 1 and 3.3, RNNs are rarely adopted for decoding the EEG [7,11–13]. This is primarily due to their low performance obtained in past benchmarks [7,11–13,23,30,32,33], for example near to the chance level in Chen et al. [30] on the same dataset used here (BNCI2014-001 [53]). Even though RNNs do not represent

the state-of-the-art for EEG decoding, in order to provide an exhaustive comparative analysis spanning across diverse network architectures, we also include RNNs in our comparative analysis. Specifically, we use the 3-layer RNNs adopted at first in Nakagome et al. [28] and then in Borra et al. [23]. We consider both recurrent layers designed with LSTM and GRU cells (*StackedLSTM* and *StackedGRU*, respectively).

- Hybrid convolutional-recurrent neural networks. Hybrid designs are used for improving the RNN performance, by including convolutions in the RNN architecture (i.e., 'hybrid' convolutional-recurrent designs). In our analysis we include the network proposed by Chen et al. [29], based on the combination between EEGNet [15] and the sequence of two recurrent layers. Specifically, we consider both recurrent layers designed with LSTM and GRU cells (*EEGNetLSTM* and *EEGNetGRU*, respectively).
- Transformers. Finally, we include *EEGConformer* [38], a recent innovative transformer designed for decoding the EEG activity by focusing not only on local features, but also on long-term relationships, by employing self-attention.

These MOABB and MOABB+braindecode pipelines are designed by exploiting the dedicated YAML files available in the `pipelines/` folder of MOABB [41], using the hyperparameters provided (replicating the solutions proposed in the reference publications). Please refer to the original publications, and to MOABB and braindecode for further details about the contrasted decoders. The pipelines are trained and evaluated in a leave-one-session-out cross-validation scheme, as done with SpeechBrain-MOABB pipelines, to provide a fair evaluation.

The comparison is illustrated in Fig. 8, showcasing the performance of pipelines based on MOABB (dark cyan), MOABB+braindecode (dark orange), and SpeechBrain-MOABB (red). Please note that the distributions obtained with SpeechBrain-MOABB are the same of Fig. 5b. Pairwise Wilcoxon signed-rank tests [60] are performed to compare the top-performing algorithm with each of the other algorithms. The results from the statistical analysis are corrected for multiple tests (a total of 17) using the Benjamini–Hochberg procedure [97]. For transparency, we provide also a tutorial in the SpeechBrain-MOABB repository for replicating our results. As depicted in Fig. 8, SpeechBrain-MOABB pipelines are the most accurate. The top-performing algorithm (EEGNet-based pipeline using SpeechBrain-MOABB) significantly outperforms other traditional machine learning pipelines designed in MOABB ($p < 0.01$), even when including hyperparameter search via grid search (TS+SVM pipeline), and other deep learning pipelines designed in MOABB+braindecode ($p < 0.05$). Moreover, SpeechBrain-MOABB pipelines are significantly more accurate than the counterpart developed in MOABB+braindecode ($p = 0.0078$, pairwise Wilcoxon signed-rank test [60]) based on the same neural network (see EEGNet, ShallowConvNet, and EEGConformer distributions). Indeed, on average across these pipelines, an accuracy increase of 10.2 ± 3.3% (mean ± standard error of the mean across participants) is achieved with SpeechBrain-MOABB, with improvements spanning from 3.5% (ShallowConvNet-based pipeline) to 19.9% (EEGNet-based pipeline). RNNs are the worst deep neural networks for decoding the EEG (see StackedGRU-based and StackedLSTM-based pipelines), matching the findings of past benchmark studies [7,11–13,23,30,32,33]. Indeed, these models are almost at the chance level (25%), as also obtained by Chen et al. on the same dataset [30]. Finally, another important results is that pipelines based on EEGNet, ShallowConvNet, and EEGConformer are the most accurate ones with the alternative toolkit for deep learning-based EEG decoding (MOABB+braindecode). Thus, as stated in Section 3.3, these networks represent the state-of-the-art for EEG decoding, and these results further support our choice of relying on these networks when validating SpeechBrain-MOABB. Overall, these results suggest that SpeechBrain-MOABB, with its experimental protocol, is capable not only of providing a more trustworthy EEG decoding using deep neural networks, but also of significantly improving the decoding performance compared to existing libraries.

4.4. Benchmark results

Finally, as an additional result, we accompany our toolkit with benchmark results obtained when applying it with EEGNet, ShallowConvNet, and EEGConformer on 9 MOABB datasets (including 204 participants and 26 recording sessions, overall), covering *motor imagery-based BCIs* (BNCI2014-001 [53], BNCI2014-004 [71], BNCI2015-001 [72], Lee2019-MI [73], Zhou2016 [74]), *P300-based BCIs* (BNCI2014-009 [75], EPFLP300 [76], BI2015a [77]), and *SSVEP-based BCIs* (Lee2019-SSVEP [73]). The main properties of these datasets are summarized in Table 4 and the number of EEG trials are reported in Table 5. A description of these public datasets is provided in the following. Please refer to the original publications for further details about data recording and acquisition.

i. BNCI2014-001 [53]. This dataset consists of 22-channel EEG recorded from 9 healthy participants across 2 recording sessions. Electrodes were placed according to 10–10 international system at Fz, FC3, FC1, FCz, FC2, FC4, C5, C3, C1, Cz, C2, C4, C6, CP3, CP1, CPz, CP2, CP4, P1, Pz, P2, POz. The task consisted in the imagination of four movements for 4 s: left hand, right hand, feet, and tongue. For each participant and each session, 288 trials were recorded, balanced across classes. EEG signals were sampled at 250 Hz and band-pass filtered between 0.5 and 100 Hz. This dataset is also known as 'dataset IIa' from BCI competition IV [53].

ii. BNCI2014-004 [71]. Here, 3-channel EEG was collected from 9 healthy participants across 5 recording sessions. Electrodes were placed according to 10–20 international system at C3, Cz, C4. The task consisted in the imagination of two movements for 4.5 s: left hand, and right hand. For each participant and each session, approx. 145 trials were recorded (on average across participants and sessions), balanced across classes. EEG signals were sampled at 250 Hz and band-pass filtered between 0.5 and 100 Hz. This dataset is also known as 'dataset IIb' from BCI competition IV [53].

iii. BNCI2015-001 [72]. This dataset consists of 13-channel EEG recorded from 12 healthy participants across 2 recording sessions. Electrodes were placed according to 10–10 international system in correspondence of three Laplacian derivations at C3 (FC3, C5, CP3, C1), Cz (FCz, C1, CPz, C2), and C4 (FC4, C2, CP4, C6). The task consisted in the imagination of two movements for 5 s: right hand, and feet. For each participant and each session, 200 trials were recorded, balanced across classes. EEG signals were sampled at 250 Hz and band-pass filtered between 0.5 and 100 Hz.

iv. Lee2019-MI [73]. In this dataset, 62-channel EEG was recorded from 54 healthy participants across 2 recording sessions. Electrodes were placed according to 10–10 international system. The task consisted in the imagination of two movements for 4 s: left hand-grasping, and right hand-grasping. For each participant and each session, 100 trials were recorded, balanced across classes. EEG signals were sampled at 1000 Hz.

v. Zhou2016 [74]. Here, 14-channel EEG was recorded from 4 healthy participants across 3 recording sessions. Electrodes were placed according to 10–10 international system at Fp1, Fp2, FC3, FCz, FC4, C3, Cz, C4, CP3, CPz, CP4, O1, Oz, O2. The task consisted in the imagination of three movements for 5 s: left hand, right hand, and feet. For each participant and each session, approx. 150 trials were recorded (on average across participants and sessions), balanced across classes. EEG signals were sampled at 250 Hz and band-pass filtered between 0.1 and 100 Hz.

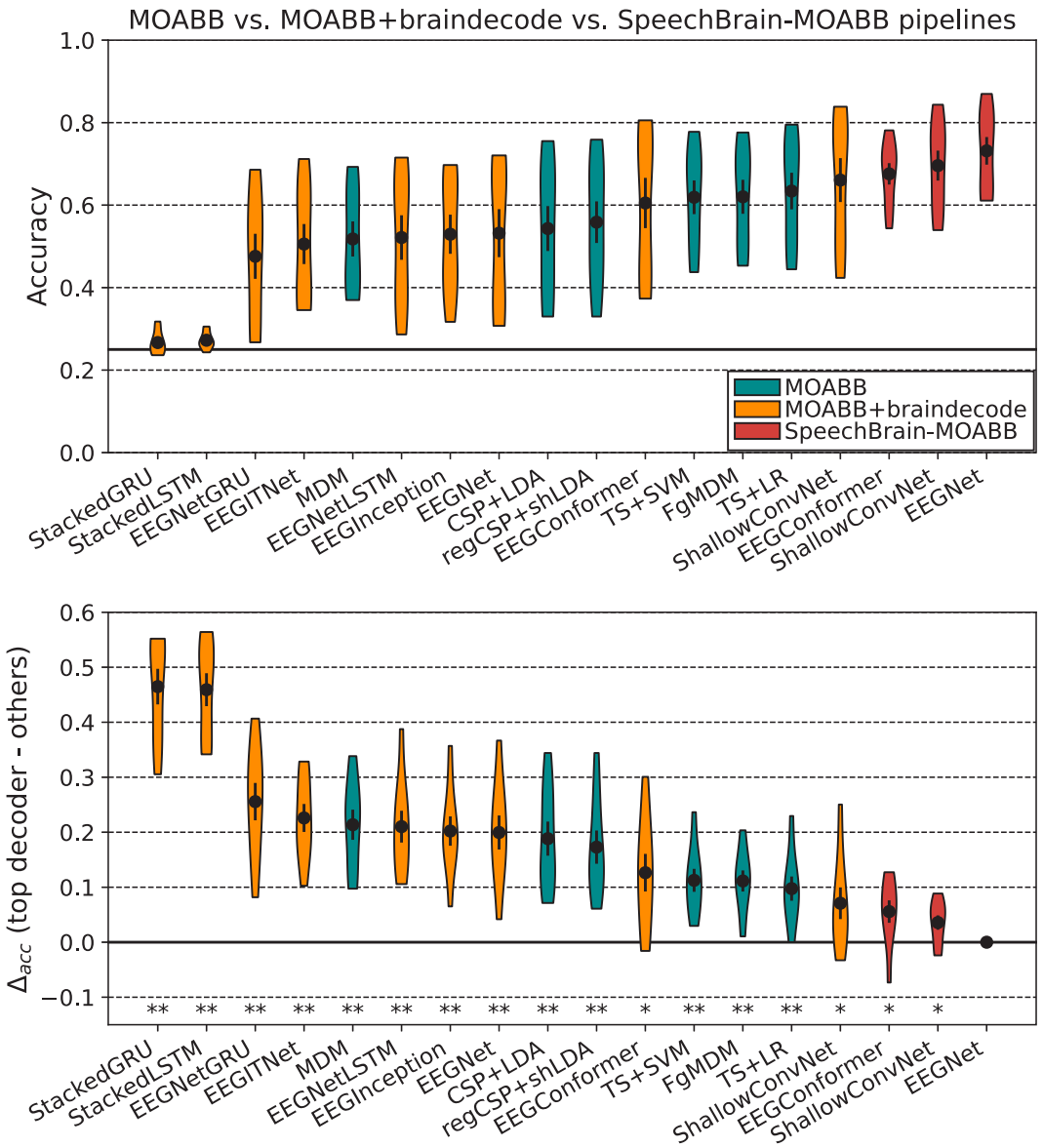

Fig. 8. Performance comparison between MOABB (dark cyan), MOABB+braindecode (dark orange), and SpeechBrain-MOABB (red) pipelines. The top panel displays the performance distributions scored by decoders. The bottom panel displays the performance difference (Δ_{acc}) from the top-performing decoder (EEGNet using SpeechBrain-MOABB), along with results from statistical analysis ($*$ $p < 0.05$, $**$ $p < 0.01$). Distributions are shown as violin plots; black dots indicate mean values and error bars the standard errors of the mean, computed across participants. For improving readability, pipelines are sorted from the least to the most accurate.

vi. BNCI2014-009 [75]. This dataset consists of 16-channel EEG recorded from 10 healthy participants across 3 recording sessions. Electrodes were placed according to 10–10 international system at Fz, FCz, Cz, CPz, Pz, Oz, F3, F4, C3, C4, CP3, CP4, P3, P4, PO7, PO8. During the task, participants used a P300 speller (6 × 6 matrix, 36 possible characters in total) organized as in Farwell and Donchin [105]. After the presentation of a stimulus (standard or target stimulus), the post-stimulus response was recorded for 0.8 s. For each participant and each session, 576 trials were recorded, with a strong class unbalance between standard and target stimuli (5:1, having indicated standard:target), by construction of the P300 speller paradigm. EEG signals were sampled at 256 Hz and band-pass filtered between 0.1 and 20 Hz.

vii. EPFLP300 [76]. Here, 32-channel EEG was collected from 9 participants across 4 recording sessions. Five out of nine participants were disabled (suffering from amyotrophic lateral sclerosis), while the remaining four participants were healthy. Electrodes were placed according to 10–10 international system. During the task, participants used an interface to choose one out of six images. In this six-choice P300 paradigm, the six different images flashed in a random order, eliciting the P300 response on the target image. After the presentation of a stimulus (standard or target stimulus), the post-stimulus response was recorded for 1 s. For each participant and each session, approx. 822 trials were recorded (on average across participants and sessions), with a strong class unbalance between standard and target stimuli (5:1, having indicated standard:target), by construction of the P300 recording paradigm. EEG signals were sampled at 2048 Hz.

Table 4
Datasets: general properties. Signal characteristics (e.g., the maximum trial length, the sampling frequency, and the applied band-pass filtering) are defined by the authors releasing each dataset. Note that in case of P300 datasets, the class unbalance ratio corresponds to standard:target stimuli ratio.

Dataset	No. of participants	No. of sessions	No. of channels	Max. trial length (s)	Sampling frequency (Hz)	Band-pass filter (Hz)	No. of classes	Class unbalance
BNCI2014-001 [53]	9	2	22	4	250	0.5–100	4	no
BNCI2014-004 [71]	9	5	3	4.5	250	0.5–100	2	no
BNCI2015-001 [72]	12	2	13	5	512	0.5–100	2	no
Lee2019-MI [73]	54	2	62	4	1000	None	2	no
Zhou2016 [74]	4	3	14	5	250	0.1–100	3	no
BNCI2014-009 [75]	10	3	16	0.8	256	0.1–20	2	5:1
EPFLP300 [76]	9	4	32	1	2048	None	2	5:1
BI2015a [77]	43	3	32	1	512	None	2	5:1
Lee2019-SSVEP [73]	54	2	62	4	1000	None	4	no

Table 5
Datasets: number of trials. For each dataset, this table lists the number of trials recorded during each recording session (on average across participants) and number of trials used in the performed leave-one-session-out experiments for the training, validation and test sets (on average across participants and cross-validation folds).

Dataset	No. of trials					No. of training trials	No. of valid. trials	No. of test trials
	Session 1	*Session 2*	*Session 3*	*Session 4*	*Session 5*			
BNCI2014-001 [53]	288	288	–	–	–	232	56	288
BNCI2014-004 [71]	124.4	124.4	160	155.6	160	463.6	115.9	144.9
BNCI2015-001 [72]	200	200	–	–	–	160	40	200
Lee2019-MI [73]	100	100	–	–	–	80	20	100
Zhou2016 [74]	153.5	146.5	150	–	–	240	60	150
BNCI2014-009 [75]	576	576	576	–	–	922	230	576
EPFLP300 [76]	813.5	827.9	826.4	821.8	–	1974	493.1	822.4
BI2015a [77]	558	561.8	542.6	–	–	886.7	221.6	554.1
Lee2019-SSVEP [73]	100	100	–	–	–	80	20	100

viii. BI2015a [77]. This dataset consists of 32-channel EEG recorded from 43 healthy participants across 3 recording sessions. Electrodes were placed according to 10–10 international system. During the task, participants played a videogame (Brain Invaders) by using a P300-based BCI. The interface exploited the oddball paradigm using a grid of 36 symbols (organized in a 6 × 6 matrix), randomly flashing during the task to elicit the P300 response. For each participant and each session, approx. 554 trials were recorded (on average across participants and sessions), with a strong class unbalance between standard and target stimuli (5:1, having indicated standard:target), by construction of the P300 recording paradigm. EEG signals were sampled at 512 Hz.

ix. Lee2019-SSVEP [73]. In this dataset, 62-channel EEG was recorded from 54 healthy participants across 2 recording sessions. Electrodes were placed according to 10–10 international system. The paradigm was designed following a conventional SSVEP-based control of BCI systems that require four-direction movements. SSVEP stimuli were presented to the user for 4 s at four different positions (down, right, left, up), flickering at four different frequencies: 5.45, 6.67, 8.57, 12 Hz. For each participant and each session, 100 trials were recorded, balanced across classes. EEG signals were sampled at 1000 Hz.

The benchmark results obtained on these datasets are reported in Table 6. The same experimental setup used in Section 4.1 is adopted here. The results of EEGNet, ShallowConvNet, and EEGConformer on BNCI2014-001 [53] reported in Table 6 are the same as those displayed in Fig. 5b. The table reports the average performance across cross-validation folds (i.e., held-out sessions) and the 10 random seeds used during multi-seed training and evaluation, in its mean value and standard error of the mean across participants. ShallowConvNet and EEGConformer results are provided for motor imagery as these networks were mainly proposed and validated for motor imagery decoding [14,38]. Indeed, ShallowConvNet design is specific for sensorimotor rhythm decoding [14], learning power-related EEG features. On the other hand, EEGNet was designed and applied for general EEG decoding on a variety of BCI recording paradigms [17,19–24,32,33,48,61–66], thus we extended the application of EEGNet also to P300 and SSVEP datasets. P300 decoding performance are quantified by means of the F1-score, as this decoding problem is inherently class unbalanced (see Table 4), by construction of the P300 BCI paradigm. From our results on motor imagery applications, an EEGNet-based decoding pipeline is the most accurate one, consistently across most of the datasets (4 out of the 5 motor imagery datasets tested). The experiments with this pipeline take 321.8 h (13.4 days, computational time expressed as the wall-clock time) on average across dataset, spanning from approx. 34 h (Zhou2016 [74] dataset) to approx. 1300 h (BI2015a [77] dataset). The high computational time required for the most time-demanding dataset (BI2015a [77]), is due to the high number of participants (43) and training trials recorded (approx. 887, on average across participants). Importantly, as anticipated in Section 3.6, using less participants during hyperparameter search reduces the computational time by 82.6% (2.3 days), on average across datasets, and by 95.2% in the most time-demanding dataset. Here, we have reported the computational time changes (in percentage) between using all participants and a subset of 3–5 participants during hyperparameter search (as described in Section 3.6).

4.5. Challenges, limitations and future directions

Our toolkit is designed for providing the neuroscience community a robust and trustworthy benchmarking software for EEG decoding via deep neural networks. In the following, we discuss the main challenges that have been addressed during the development of SpeechBrain-MOABB. This discussion could be useful for new users to better understand our toolkit, and for experienced researchers interested into contributing to our toolkit.

i. Data. The first challenge regarded the selection of the datasets to use in our toolkit. We aspired to use our toolkit with the most relevant datasets adopted in the literature, recorded across diverse BCI recording paradigms, e.g., motor imagery, P300, and SSVEP. Besides the type of EEG recording, the datasets needed

Table 6
Benchmark results obtained with SpeechBrain-MOABB (mean value ± standard error of the mean across participants) exploiting decoding pipelines based on EEGNet [15], ShallowConvNet [14], and EEGConformer [38] on 9 MOABB [41] datasets. The experimental protocol is based on 2-step sequential model-based search (TPE-based) using 10-seed final training and evaluation. Results are obtained with a leave-one-session-out cross-validation. For each participant, the decoding accuracy was averaged across cross-validation folds (i.e., held-out sessions) and across the 10 runs performed in the 10-seed final training and evaluation. Bold values represent the most accurate metrics, across architectures.

	Dataset	Evaluation metric	Architecture		
			EEGNet	*ShallowConvNet*	*EEGConformer*
Motor imagery	BNCI2014-001 [53]	accuracy	**0.732 ± 0.033**	0.696 ± 0.036	0.676 ± 0.026
	BNCI2014-004 [71]	accuracy	**0.812 ± 0.025**	0.785 ± 0.028	0.799 ± 0.024
	BNCI2015-001 [72]	accuracy	0.811 ± 0.026	**0.829 ± 0.025**	0.752 ± 0.031
	Lee2019-MI [73]	accuracy	**0.694 ± 0.015**	0.658 ± 0.015	0.651 ± 0.014
	Zhou2016 [74]	accuracy	**0.844 ± 0.007**	0.827 ± 0.010	0.840 ± 0.016
P300	BNCI2014-009 [75]	F1-score	0.755 ± 0.024	–	–
	EPFLP300 [76]	F1-score	0.635 ± 0.040	–	–
	BI2015a [77]	F1-score	0.724 ± 0.015	–	–
SSVEP	Lee2019-SSVEP [73]	accuracy	0.916 ± 0.017	–	–

to satisfy the following requirements. Based on our design principles (see Section 3.1), the datasets had to be public with no need to sign data sharing agreements (as in Obeid et al. [106]), for ensuring an easy replication of the results and facilitating the use of the toolkit for new users (providing a 'plug-and-play' user experience). Of course, the data needed to be anonymized, to avoid privacy concerns [107]. Moreover, datasets had to be recorded by research groups in accordance with the Declaration of Helsinki, with an informed consent collected from each research participant, and with ethical approval obtained from their local ethical committees [108]. This ensured that the data have been collected while protecting the rights, safety, dignity, and well-being of research participants (that is, avoiding ethical concerns). Therefore, we relied on the datasets of MOABB [41], as it conveniently collects EEG datasets that fit all these requirements.

ii. Usability and scalability. Our toolkit needed to be easy-to-use and, at the same time, easy to be scaled to new datasets and models, for assisting and accelerating its spread in the neuroscience community. This represented another important challenge that has been addressed as follows. We conveniently exposed in a single YAML file (the 'hyperparameter file') the hyperparameters and the hyperparameter search settings, including complex Python objects (e.g., the neural network Python class), for running the entire EEG decoding pipeline. See Fig. A.2 of Appendix for the code snippet of a representative hyperparameter file. A new model can be designed in a Pythonic way using PyTorch, and stored in a separate Python file in the dedicated SpeechBrain-MOABB folder (see Section 3.3, and see the code snippet in Fig. A.1 of Appendix for an example). A new EEG dataset can be selected from the ones available in MOABB – that continuously adds new datasets in a rolling release fashion – matching all our data requirements (as discussed at point i.). Both the new model and dataset can be easily added into the hyperparameter file together with the other hyperparameters, for being utilized in the experiments. This way, researchers can just focus on: (a) the design of a new innovative model, (b) the selection of a target MOABB EEG dataset, and (c) the writing of a single file (hyperparameter file), listing all the relevant hyperparameters for controlling the entire EEG decoding pipeline. To the best knowledge of the authors, no other existing software allows this easy-to-use definition of complex deep learning-based EEG decoding solutions, maintaining at the same time a high level of scalability across new EEG applications (i.e., new datasets) and new decoding models (see also Section 2.2 for a comparison with existing toolkits).

iii. Computational demand. The search of the optimal hyperparameters across the entire decoding pipeline, which is one of the key novel aspects of SpeechBrain-MOABB (see Section 3.6), poses a high computational demand. Therefore, to introduce and validate SpeechBrain-MOABB, we ran our comprehensive battery of experiments on high-computing clusters. However, even though high-computing clusters were employed, we exploited computational nodes equipped with an Intel Gold 6148 Skylake CPU (2.4 GHz), an NVIDIA V100 GPU (16 GB of memory), and 12 GB of RAM, to be in line with the resources used in prior EEG studies [14,15,17,19,20,23,38]. Overall, our experiments took 7.61 months to complete (wall-clock time), across all the considered datasets, neural networks, and ablation studies (see Sections 4.2, 4.3, and 4.4). To the best knowledge of the authors, our results represent a unique collection of robust and trustworthy EEG decoding results. Despite our experiments took several months to provide an exhaustive set of results, a single complete experiment (i.e., one EEG dataset decoded with one neural network) required 13.4 days (on average), which could be reduced to 2.3 days (on average) by leveraging on less participants during hyperparameter search (as discussed in Sections 3.6 and 4.4). Therefore, neuroscientists can conveniently use SpeechBrain-MOABB to obtain robust and trustworthy results within a few days of computation, using computational resources similar to ours. Of course, the computational time depends on the used resources, thus, we recommend neuroscientists to consider their resources in relation to ours while planning their experiments. To better support researchers, future releases of our toolkit will also include a comparative analysis of the computational time on different GPUs and CPUs.

Even though SpeechBrain-MOABB is characterized by unique properties that ease the definition and benchmarking of novel deep learning-based EEG decoding pipelines, the following limitations are currently affecting our toolkit. First, SpeechBrain-MOABB lacks feature visualization and network decision explanation functionalities. In the future, these will be integrated, enabling a straightforward, reproducible, and reliable understanding of the most relevant EEG neural signatures in the spatial, temporal, and frequency domains. This will be addressed by deriving representations based on explanation techniques, such as layerwise relevance propagation (LRP) [109] and deep learning important features (DeepLIFT) [110] – that resulted the most promising for EEG in a recent benchmark study [111] – but also saliency maps [112], local interpretable model-agnostic explanations (LIME) [113], gradient-weighted class activation mapping (Grad-CAM) [114], and Shapeley addictive explanations (SHAP) [115]. Second, SpeechBrain-MOABB focuses on supervised learning decoding pipelines only, and it does not currently support pipelines based on self-supervised learning. Importantly, self-supervised learning is gaining interest in the neuroscience community [116], for training neural decoders on small labeled datasets, exploiting the knowledge learned by learning systems on large unlabeled datasets. Future versions of our toolkit will extend

```python
class EEGNet(torch.nn.Module):
    def __init__(self):
        super(EEGNet, self).__init__()
        # Initialize your model components here

    def forward(self, x):
        """Forward pass of the EEGNet model.

        Args:
        - x (torch.Tensor): Input tensor (batch size, EEG time samples, EEG channels, 1).

        Returns:
        - torch.Tensor: Output tensor of the model.
        """
        # Perform your computations here
        return x
```

Fig. A.1. Model definition. In SpeechBrain-MOABB the model definition follows PyTorch standards, inheriting from `torch.nn.Module`. Users have to define initialization and forward methods.

```yaml
# Dataset and data pre-processing definition
dataset: !new:moabb.datasets.BNCI2014001
# Cut-off frequencies for band-pass filters
fmin: 1. # @orion_step1: --fmin~"uniform(0.1, 5.0)"
fmax: 40. # @orion_step1: --fmax~"uniform(20.0, 50.0)"
# Time interval for defining EEG epochs
tmin: 0.
tmax: 4.0 # @orion_step1: --tmax~"uniform(1.0, 4.0)"
original_sample_rate: 250 # Original sampling rate (Hz)
sample_rate: 125 # Target sampling rate (Hz)
# Number of steps used for spatial sampling (starting from Cz by default)
n_steps_channel_selection: 2 # @orion_step1: --n_steps_channel_selection~"uniform(1, 3,discrete=True)"
T: !apply:math.ceil
    - !ref <sample_rate> * (<tmax> - <tmin>)
C: 22 # Number of total channels
events_to_load: null # Use all events: ['left-hand', 'right-hand', 'feet', 'tongue']
n_classes: 4 # Number of classes when using all events

# Model definition
cnn_temporal_kernels: 8 # @orion_step1: --cnn_temporal_kernels~"uniform(4, 64,discrete=True)"
cnn_temporal_kernelsize: 62 # @orion_step1: --cnn_temporal_kernelsize~"uniform(24, 62,discrete=True)"
dropout: 0.25 # @orion_step1: --dropout~"uniform(0.0, 0.5)"
model: !new:models.EEGNet.EEGNet
    input_shape: [null, !ref <T>, !ref <C>, null]
    cnn_temporal_kernels: !ref <cnn_temporal_kernels>
    cnn_temporal_kernelsize: [!ref <cnn_temporal_kernelsize>, 1]
    dropout: !ref <dropout>
    dense_n_neurons: !ref <n_classes>
```

Fig. A.2. An excerpt of a YAML file (based on HyperPyYAML) defining the relevant hyperparameters for pre-processing a MOABB dataset (BNCI2014-001 [53]) and for defining a state-of-the-art neural network for decoding motor imagery (EEGNet [15]). Here, the variables `tmin`, `tmax`, `fmin`, `fmax`, `n_steps_channel_selection` denote t_0, t_1, f_0, f_1, C_{step}, respectively. In addition, `original_sample_rate` and `sample_rate` represent the original sampling frequency and target sampling frequency (f_s). The EEGNet architectural hyperparameters introduced in this example are the number of temporal kernels of the first convolutional layer (`cnn_temporal_kernels`) and their kernel size (`cnn_temporal_kernelsize`), and the dropout rate (`dropout`).

the SpeechBrain functionalities that support self-supervised learning with speech waveforms, adapting them for working on EEG signals, thus, providing neuroscientists the opportunity to design accurate deep learning-based decoders on small labeled datasets. This could be useful in BCI applications for reducing the time required for calibrating the BCI on a new user. Third, SpeechBrain-MOABB includes common pre-processing steps for preparing EEG signals for deep learning-based pipelines (see Section 3.2.1), aimed at leaving the learning system the ability of automatically learning the most relevant information contained in the activity by itself. Future SpeechBrain-MOABB releases will also enrich the possible EEG pre-processing steps, for example including artifact removal via independent component analysis, providing neuroscientists a wider range of possible data preparation steps.

5. Conclusion

In this study, we introduced SpeechBrain-MOABB, a user-friendly Python library for EEG decoding using deep neural networks. Our toolkit is not intended as a replacement or competitor for existing software. Instead, its strength lies in bridging existing tools specific to EEG decoding (MOABB and braindecode) and speech decoding (SpeechBrain) within a unified framework for benchmarking neural networks applied to EEG data in a user-friendly and reliable manner. Our experimental results demonstrated that SpeechBrain-MOABB outperforms both MOABB and MOABB+braindecode pipelines, achieving performance improvements of 14.9% and 25.2% (on average across all pipelines), respectively.

SpeechBrain-MOABB pipelines are designed to be comprehensive, encompassing all necessary processing steps from raw data to final decoding results (i.e., from data pre-processing, augmentation to network architecture and training). SpeechBrain-MOABB not only focuses on creating easy-to-use and easy-to-share decoding pipelines but also implements a robust experimental protocol that manages crucial aspects of the pipeline, including multi-step hyperparameter search, for handling hyperparameter search in high-dimensional search spaces, and multi-seed training and evaluations, for addressing variability of performance measures depending on the used random seed. This standardized protocol offers several advantages. Firstly, neuroscientists using SpeechBrain-MOABB can transition from the trial-and-error selection of hyperparameters to an automated search for the most promising ones, facilitated by our command-line interface and a well-structured YAML file. Moreover, this transparent, trustworthy, and replicable protocol enables fair comparisons across deep learning-based decoding pipelines, crucial for properly benchmarking novel and existing approaches working with EEG signals. In this way, we hope that our efforts contribute to addressing the 'replicability crisis' currently faced by the neuroscience community.

CRediT authorship contribution statement

Davide Borra: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Francesco Paissan:** Validation. **Mirco Ravanelli:** Validation, Supervision, Software, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We gratefully acknowledge the support of Digital Research Alliance of Canada (alliancecan.ca) and of Mila - Quebec AI Institute for the use of their high-performance computing clusters while performing the computations. Providers were not involved in the study design, collection, analysis, interpretation of data, the writing of this article or the decision to submit it for publication.

This work was supported by #NEXTGENERATIONEU (NGEU) and funded by the Ministry of University and Research (MUR), National Recovery and Resilience Plan (NRRP), project MNESYS (PE0000006) – A Multiscale integrated approach to the study of the nervous system in health and disease (DN. 1553 11.10.2022).

Appendix. Additional code excerpts

In this section we present additional code excerpts. In Fig. A.1 we present an example of code for defining a neural network by following PyTorch convention. In Fig. A.2 we present an example of hyperparameter file (YAML file) defining the relevant hyperparameters for pre-processing a MOABB dataset and for defining the neural network.

References

[1] J.I. Glaser, A.S. Benjamin, R.H. Chowdhury, M.G. Perich, L.E. Miller, K.P. Kording, Machine learning for neural decoding, eneuro 7 (4) (2020) http://dx.doi.org/10.1523/eneuro.0506-19.2020.

[2] M. Filippini, D. Borra, M. Ursino, E. Magosso, P. Fattori, Decoding sensorimotor information from superior parietal lobule of macaque via convolutional neural networks, Neural Netw. 151 (2022) 276–294, http://dx.doi.org/10.1016/j.neunet.2022.03.044.

[3] D. Borra, M. Filippini, M. Ursino, P. Fattori, E. Magosso, Motor decoding from the posterior parietal cortex using deep neural networks, J. Neural Eng. 20 (3) (2023) 036016, http://dx.doi.org/10.1088/1741-2552/acd1b6.

[4] D. Borra, M. Filippini, M. Ursino, P. Fattori, E. Magosso, Convolutional neural networks reveal properties of reach-to-grasp encoding in posterior parietal cortex, Comput. Biol. Med. 172 (2024) 108188, http://dx.doi.org/10.1016/j.compbiomed.2024.108188.

[5] D. McFarland, C. Anderson, K.-R. Muller, A. Schlogl, D. Krusienski, BCI meeting 2005-workshop on BCI signal processing: feature extraction and translation, IEEE Trans. Neural Syst. Rehabil. Eng. 14 (2) (2006) 135–138, http://dx.doi.org/10.1109/tnsre.2006.875637.

[6] F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy, F. Yger, A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update, J. Neural Eng. 15 (3) (2018) 031005, http://dx.doi.org/10.1088/1741-2552/aab2f2.

[7] Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T.H. Falk, J. Faubert, Deep learning-based electroencephalography analysis: a systematic review, J. Neural Eng. 16 (5) (2019) 051001, http://dx.doi.org/10.1088/1741-2552/ab260c.

[8] N. Robinson, R. Mane, T. Chouhan, C. Guan, Emerging trends in BCI-robotics for motor control and rehabilitation, Curr. Opin. Biomed. Eng. 20 (2021) 100354, http://dx.doi.org/10.1016/j.cobme.2021.100354.

[9] J.D.R. Millán, Combining brain-computer interfaces and assistive technologies: state-of-the-art and challenges, Front. Neurosci. 1 (2010) http://dx.doi.org/10.3389/fnins.2010.00161.

[10] R. Abiri, S. Borhani, E.W. Sellers, Y. Jiang, X. Zhao, A comprehensive review of EEG-based brain–computer interface paradigms, J. Neural Eng. 16 (1) (2019) 011001, http://dx.doi.org/10.1088/1741-2552/aaf12e.

[11] A. Al-Saegh, S.A. Dawwd, J.M. Abdul-Jabbar, Deep learning for motor imagery EEG-based classification: A review, Biomed. Signal Process. Control 63 (2021) 102172, http://dx.doi.org/10.1016/j.bspc.2020.102172.

[12] A. Craik, Y. He, J.L. Contreras-Vidal, Deep learning for electroencephalogram (EEG) classification tasks: a review, J. Neural Eng. 16 (3) (2019) 031001, http://dx.doi.org/10.1088/1741-2552/ab0ab5.

[13] K.M. Hossain, M.A. Islam, S. Hossain, A. Nijholt, M.A.R. Ahad, Status of deep learning for EEG-based brain–computer interface applications, Front. Comput. Neurosc. 16 (2023) http://dx.doi.org/10.3389/fncom.2022.1006763.

[14] R.T. Schirrmeister, J.T. Springenberg, L.D.J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, T. Ball, Deep learning with convolutional neural networks for EEG decoding and visualization, Hum. Brain Mapp. 38 (11) (2017) 5391–5420, http://dx.doi.org/10.1002/hbm.23730.

[15] V.J. Lawhern, A.J. Solon, N.R. Waytowich, S.M. Gordon, C.P. Hung, B.J. Lance, EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces, J. Neural Eng. 15 (5) (2018) 056013, http://dx.doi.org/10.1088/1741-2552/aace8c.

[16] D. Zhao, F. Tang, B. Si, X. Feng, Learning joint space–time–frequency features for EEG decoding on small labeled data, Neural Netw. 114 (2019) 67–77, http://dx.doi.org/10.1016/j.neunet.2019.02.009.

[17] D. Borra, S. Fantozzi, E. Magosso, Interpretable and lightweight convolutional neural network for eeg decoding: Application to movement execution and imagination, Neural Netw. 129 (2020) 55–74, http://dx.doi.org/10.1016/j.neunet.2020.05.032.

[18] D. Borra, S. Fantozzi, E. Magosso, EEG motor execution decoding via interpretable sinc-convolutional neural networks, in: J. Henriques, N. Neves, P. de Carvalho (Eds.), XV Mediterranean Conference on Medical and Biological Engineering and Computing – MEDICON 2019, Springer International Publishing, 2020, pp. 1113–1122, http://dx.doi.org/10.1007/978-3-030-31635-8_135.

[19] D. Borra, S. Fantozzi, E. Magosso, A lightweight multi-scale convolutional neural network for P300 decoding: Analysis of training strategies and uncovering of network decision, Front. Hum. Neurosci. 15 (2021) http://dx.doi.org/10.3389/fnhum.2021.655840.

[20] D. Borra, E. Magosso, Deep learning-based EEG analysis: investigating P3 ERP components, J. Integr. Neurosci. 20 (4) (2021) 791–811, http://dx.doi.org/10.31083/j.jin2004083.

[21] D. Borra, E. Magosso, M. Castelo-Branco, M. Simões, A Bayesian-optimized design for an interpretable convolutional neural network to decode and analyze the P300 response in autism, J. Neural Eng. 19 (4) (2022) 046010, http://dx.doi.org/10.1088/1741-2552/ac7908.

[22] D. Borra, F. Bossi, D. Rivolta, E. Magosso, Deep learning applied to EEG source-data reveals both ventral and dorsal visual stream involvement in holistic processing of social stimuli, Sci. Rep. 13 (1) (2023) http://dx.doi.org/10.1038/s41598-023-34487-z.

[23] D. Borra, V. Mondini, E. Magosso, G.R. Müller-Putz, Decoding movement kinematics from EEG using an interpretable convolutional neural network, Comput. Biol. Med. 165 (2023) 107323, http://dx.doi.org/10.1016/j.compbiomed.2023.107323.

[24] N. Waytowich, V.J. Lawhern, J.O. Garcia, J. Cummings, J. Faller, P. Sajda, J.M. Vettel, Compact convolutional neural networks for classification of asynchronous steady-state visual evoked potentials, J. Neural Eng. 15 (6) (2018) 066031, http://dx.doi.org/10.1088/1741-2552/aae5d8.

[25] A. Farahat, C. Reichert, C.M. Sweeney-Reed, H. Hinrichs, Convolutional neural networks for decoding of covert attention focus and saliency maps for EEG feature visualization, J. Neural Eng. 16 (6) (2019) 066010, http://dx.doi.org/10.1088/1741-2552/ab3bb4.

[26] J.M. Mayor-Torres, M. Ravanelli, S.E. Medina-DeVilliers, M.D. Lerner, G. Riccardi, Interpretable SincNet-based deep learning for emotion recognition from EEG brain activity, 2021, http://dx.doi.org/10.48550/arXiv.2107.10790.

[27] F. Paissan, V.P. Kumaravel, E. Farella, Interpretable CNN for single-channel artifacts detection in raw eeg signals, in: 2022 IEEE Sensors Applications Symposium, SAS, 2022, pp. 1–6, http://dx.doi.org/10.1109/SAS54819.2022.9881381.

[28] S. Nakagome, T.P. Luu, Y. He, A.S. Ravindran, J.L. Contreras-Vidal, An empirical comparison of neural networks and machine learning algorithms for EEG gait decoding, Sci. Rep. 10 (1) (2020) http://dx.doi.org/10.1038/s41598-020-60932-4.

[29] Y.-F. Chen, R. Fu, J. Wu, J. Song, R. Ma, Y.-C. Jiang, M. Zhang, Continuous bimanual trajectory decoding of coordinated movement from EEG signals, IEEE J. Biomed. Health Inf. 26 (12) (2022) 6012–6023, http://dx.doi.org/10.1109/jbhi.2022.3224506.

[30] X. Chen, X. Teng, H. Chen, Y. Pan, P. Geyer, Toward reliable signals decoding for electroencephalogram: A benchmark study to eegnex, Biomed. Signal Process. Control 87 (2024) 105475, http://dx.doi.org/10.1016/j.bspc.2023.105475.

[31] A. Salami, J. Andreu-Perez, H. Gillmeister, EEG-itnet: An explainable inception temporal convolutional network for motor imagery classification, IEEE Access 10 (2022) 36672–36685, http://dx.doi.org/10.1109/access.2022.3161489.

[32] M. Simões, D. Borra, E. Santamaría-Vázquez, M. Bittencourt-Villalpando, D. Krzemiński, A. Miladinović, T. Schmid, H. Zhao, C. Amaral, B. Direito, J. Henriques, P. Carvalho, M. Castelo-Branco, BCIAUT-p300: A multi-session and multi-subject benchmark dataset on autism for P300-based brain-computer-interfaces, Front. Neurosci. 14 (2020) http://dx.doi.org/10.3389/fnins.2020.568104.

[33] J. An, X. Chen, D. Wu, Algorithm contest of motor imagery BCI in the world robot contest 2022: A survey, Brain Sci. Adv. 9 (3) (2023) 166–181, http://dx.doi.org/10.26599/bsa.2023.9050011.

[34] E. Santamaría-Vázquez, V. Martínez-Cagigal, F. Vaquerizo-Villar, R. Hornero, EEG-inception: A novel deep convolutional neural network for assistive ERP-based brain-computer interfaces, IEEE Trans. Neural Syst. Rehabil. Eng. 28 (12) (2020) 2773–2782, http://dx.doi.org/10.1109/tnsre.2020.3048106.

[35] D. Xu, F. Tang, Y. Li, Q. Zhang, X. Feng, An analysis of deep learning models in SSVEP-based BCI: A survey, Brain Sci. 13 (3) (2023) 483, http://dx.doi.org/10.3390/brainsci13030483.

[36] N.-S. Kwak, K.-R. Müller, S.-W. Lee, A convolutional neural network for steady state visual evoked potential classification under ambulatory environment, in: F. Schwenker (Ed.), PLOS ONE 12 (2) (2017) e0172578, http://dx.doi.org/10.1371/journal.pone.0172578.

[37] T.-H. Nguyen, W.-Y. Chung, A single-channel SSVEP-based BCI speller using deep learning, IEEE Access 7 (2019) 1752–1763, http://dx.doi.org/10.1109/access.2018.2886759.

[38] Y. Song, Q. Zheng, B. Liu, X. Gao, EEG conformer: Convolutional transformer for EEG decoding and visualization, IEEE Trans. Neural Syst. Rehabil. Eng. 31 (2023) 710–719, http://dx.doi.org/10.1109/tnsre.2022.3230250.

[39] M. Hutson, Artificial intelligence faces reproducibility crisis, Science 359 (6377) (2018) 725–726, http://dx.doi.org/10.1126/science.359.6377.725.

[40] S. Saha, M. Baumert, Intra- and inter-subject variability in EEG-based sensorimotor brain computer interface: A review, Frontiers in Computational Neuroscience 13 (2020) http://dx.doi.org/10.3389/fncom.2019.00087.

[41] V. Jayaram, A. Barachant, MOABB: trustworthy algorithm benchmarking for BCIs, J. Neural Eng. 15 (6) (2018) 066011, http://dx.doi.org/10.1088/1741-2552/aadea0.

[42] R. Moriconi, M.P. Deisenroth, K.S. Sesh Kumar, High-dimensional Bayesian optimization using low-dimensional feature spaces, Mach. Learn. 109 (9–10) (2020) 1925–1943, http://dx.doi.org/10.1007/s10994-020-05899-z.

[43] W. Ma, Y. Gong, G. Zhou, Y. Liu, L. Zhang, B. He, A channel-mixing convolutional neural network for motor imagery eeg decoding and feature visualization, Biomed. Signal Process. Control 70 (2021) 103021, http://dx.doi.org/10.1016/j.bspc.2021.103021.

[44] R.R. Chowdhury, Y. Muhammad, U. Adeel, Enhancing cross-subject motor imagery classification in EEG-based brain–computer interfaces by using multi-branch CNN, Sensors 23 (18) (2023) 7908, http://dx.doi.org/10.3390/s23187908.

[45] B.E. Olivas-Padilla, M.I. Chacon-Murguia, Classification of multiple motor imagery using deep convolutional neural networks and spatial filters, Appl. Soft Comput. 75 (2019) 461–472, http://dx.doi.org/10.1016/j.asoc.2018.11.031.

[46] S. Roy, A. Chowdhury, K. McCreadie, G. Prasad, Deep learning based inter-subject continuous decoding of motor imagery for practical brain-computer interfaces, Front. Neurosci. 14 (2020) http://dx.doi.org/10.3389/fnins.2020.00918.

[47] I.H. de Oliveira, A.C. Rodrigues, Empirical comparison of deep learning methods for EEG decoding, Front. Neurosci. 16 (2023) http://dx.doi.org/10.3389/fnins.2022.1003984.

[48] D. Borra, S. Fantozzi, E. Magosso, Convolutional neural network for a P300 brain-computer interface to improve social attention in autistic spectrum disorder, in: J. Henriques, N. Neves, P. de Carvalho (Eds.), XV Mediterranean Conference on Medical and Biological Engineering and Computing – MEDICON 2019, Springer International Publishing, 2020, pp. 1837–1843, http://dx.doi.org/10.1007/978-3-030-31635-8_223.

[49] X. Bouthillier, P. Delaunay, M. Bronzi, A. Trofimov, B. Nichyporuk, J. Szeto, N. Mohammadi Sepahvand, E. Raff, K. Madan, V. Voleti, S. Ebrahimi Kahou, V. Michalski, T. Arbel, C. Pal, G. Varoquaux, P. Vincent, Accounting for variance in machine learning benchmarks, in: A. Smola, A. Dimakis, I. Stoica (Eds.), Proceedings of Machine Learning and Systems, 3, 2021, pp. 747–769.

[50] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R.D. Mori, Y. Bengio, SpeechBrain: A general-purpose speech toolkit, 2021, http://dx.doi.org/10.48550/arXiv.2106.04624.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, 2019, http://dx.doi.org/10.48550/arXiv.1912.01703.

[52] C. Liu, J. Jin, I. Daly, S. Li, H. Sun, Y. Huang, X. Wang, A. Cichocki, SincNet-based hybrid neural network for motor imagery EEG decoding, IEEE Trans. Neural Syst. Rehabil. Eng. 30 (2022) 540–549, http://dx.doi.org/10.1109/tnsre.2022.3156076.

[53] M. Tangermann, K.-R. Müller, A. Aertsen, N. Birbaumer, C. Braun, C. Brunner, R. Leeb, C. Mehring, K.J. Miller, G.R. Müller-Putz, G. Nolte, G. Pfurtscheller, H. Preissl, G. Schalk, A. Schlögl, C. Vidaurre, S. Waldert, B. Blankertz, Review of the BCI competition IV, Front. Neurosci. 6 (2012) http://dx.doi.org/10.3389/fnins.2012.00055.

[54] A. Gramfort, M. Luessi, E. Larson, D.A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, M.S. Hämäläinen, MEG and EEG data analysis with MNE-Python, Front. Neurosci. 7 (267) (2013) 1–13, http://dx.doi.org/10.3389/fnins.2013.00267.

[55] S.-P. Kim, Preprocessing of EEG, in: C.-H. Im (Ed.), Computational EEG Analysis, Springer Singapore, 2018, pp. 15–33, http://dx.doi.org/10.1007/978-981-13-0908-3_2.

[56] E. Lashgari, D. Liang, U. Maoz, Data augmentation for deep-learning-based electroencephalography, J. Neurosci. Methods 346 (2020) 108885, http://dx.doi.org/10.1016/j.jneumeth.2020.108885.

[57] J. Polich, Updating P300: An integrative theory of P3a and P3b, Clin. Neurophysiol. 118 (10) (2007) 2128–2148, http://dx.doi.org/10.1016/j.clinph.2007.04.019.

[58] A. Al-Saegh, S.A. Dawwd, J.M. Abdul-Jabbar, CutCat: An augmentation method for EEG classification, Neural Netw. 141 (2021) 433–443, http://dx.doi.org/10.1016/j.neunet.2021.05.032.

[59] J. Townsend, M. Westerfield, E. Leaver, S. Makeig, T.-P. Jung, K. Pierce, E. Courchesne, Event-related brain response abnormalities in autism: evidence for impaired cerebello-frontal spatial attention networks, Cogn. Brain Res. 11 (1) (2001) 127–145, http://dx.doi.org/10.1016/s0926-6410(00)00072-0.

[60] F. Wilcoxon, Individual comparisons by ranking methods, Biom. Bull. 1 (6) (1945) 80, http://dx.doi.org/10.2307/3001968.

[61] A. Vahid, M. Mückschel, S. Stober, A.-K. Stock, C. Beste, Applying deep learning to single-trial EEG data provides evidence for complementary theories on action control, Commun. Biol 3 (1) (2020) http://dx.doi.org/10.1038/s42003-020-0846-z.

[62] X. Deng, B. Zhang, N. Yu, K. Liu, K. Sun, Advanced TSGL-eegnet for motor imagery EEG-based brain-computer interfaces, IEEE Access 9 (2021) 25118–25130, http://dx.doi.org/10.1109/access.2021.3056088.

[63] M. Riyad, M. Khalil, A. Adib, MI-EEGNET: A novel convolutional neural network for motor imagery classification, J. Neurosci. Methods 353 (2021) 109037, http://dx.doi.org/10.1016/j.jneumeth.2020.109037.

[64] W. Huang, Y. Xue, L. Hu, H. Liuli, S-eegnet: Electroencephalogram signal classification based on a separable convolution neural network with bilinear interpolation, IEEE Access 8 (2020) 131636–131646, http://dx.doi.org/10.1109/access.2020.3009665.

[65] P. Li, J. Su, A.N. Belkacem, L. Cheng, C. Chen, Corrigendum: Multi-person feature fusion transfer learning-based convolutional neural network for SSVEP-based collaborative BCI, Front. Neurosci. 16 (2022) http://dx.doi.org/10.3389/fnins.2022.1024150.

[66] H. Yao, K. Liu, X. Deng, X. Tang, H. Yu, FB-eegnet: A fusion neural network across multi-stimulus for ssvep target detection, J. Neurosci. Methods 379 (2022) 109674, http://dx.doi.org/10.1016/j.jneumeth.2022.109674.

[67] S. Bai, J.Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018, http://dx.doi.org/10.48550/arXiv.1803.01271.

[68] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017, http://dx.doi.org/10.48550/arXiv.1706.03762.

[69] L.N. Smith, Cyclical learning rates for training neural networks, 2015, http://dx.doi.org/10.48550/arXiv.1506.01186.
[70] B.T. Polyak, A.B. Juditsky, Acceleration of stochastic approximation by averaging, SIAM J. Control Optim. 30 (4) (1992) 838–855, http://dx.doi.org/10.1137/0330046.
[71] R. Leeb, F. Lee, C. Keinrath, R. Scherer, H. Bischof, G. Pfurtscheller, Brain–computer communication: Motivation, aim, and impact of exploring a virtual apartment, IEEE Trans. Neural Syst. Rehabil. Eng. 15 (4) (2007) 473–482, http://dx.doi.org/10.1109/tnsre.2007.906956.
[72] J. Faller, C. Vidaurre, T. Solis-Escalante, C. Neuper, R. Scherer, Autocalibration and recurrent adaptation: Towards a plug and play online ERD-BCI, IEEE Trans. Neural Syst. Rehabil. Eng. 20 (3) (2012) 313–319, http://dx.doi.org/10.1109/tnsre.2012.2189584.
[73] M.-H. Lee, O.-Y. Kwon, Y.-J. Kim, H.-K. Kim, Y.-E. Lee, J. Williamson, S. Fazli, S.-W. Lee, EEG dataset and OpenBMI toolbox for three BCI paradigms: an investigation into BCI illiteracy, GigaScience 8 (5) (2019) http://dx.doi.org/10.1093/gigascience/giz002.
[74] B. Zhou, X. Wu, Z. Lv, L. Zhang, X. Guo, A fully automated trial selection method for optimization of motor imagery based brain-computer interface, in: B. He (Ed.), PLOS ONE 11 (9) (2016) e0162657, http://dx.doi.org/10.1371/journal.pone.0162657.
[75] P. Aricò, F. Aloise, F. Schettini, S. Salinari, D. Mattia, F. Cincotti, Influence of P300 latency jitter on event related potential-based brain–computer interface performance, J. Neural Eng. 11 (3) (2014) 035008, http://dx.doi.org/10.1088/1741-2560/11/3/035008.
[76] U. Hoffmann, J.-M. Vesin, T. Ebrahimi, K. Diserens, An efficient P300-based brain–computer interface for disabled subjects, J. Neurosci. Methods 167 (1) (2008) 115–125, http://dx.doi.org/10.1016/j.jneumeth.2007.03.005.
[77] L. Korczowski, M. Cederhout, A. Andreev, G. Cattan, P.L. Coelho Rodrigues, V. Gautheret, M. Congedo, Brain invaders calibration-less P300-based BCI with modulation of flash duration dataset (bi2015a), 2019, http://dx.doi.org/10.5281/ZENODO.3266929.
[78] X. Bouthillier, C. Tsirigotis, F. Corneau-Tremblay, T. Schweizer, L. Dong, P. Delaunay, F. Normandin, M. Bronzi, D. Suhubdy, R. Askari, M. Noukhovitch, C. Xue, S. Ortiz-Gagné, O. Breuleux, A. Bergeron, O. Bilaniuk, S. Bocco, H. Bertrand, G. Alain, D. Serdyuk, P. Henderson, P. Lamblin, C. Beckham, Epistimio/orion: Asynchronous Distributed Hyperparameter Optimization, 2023, http://dx.doi.org/10.5281/zenodo.3478592.
[79] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2012) 281–305, URL https://dl.acm.org/doi/10.5555/2188385.2188395.
[80] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, 24, 2011, URL https://dl.acm.org/doi/10.5555/2986459.2986743.
[81] T. Yu, H. Zhu, Hyper-parameter optimization: A review of algorithms and applications, 2020, http://dx.doi.org/10.48550/arXiv.2003.05689.
[82] M. Malu, G. Dasarathy, A. Spanias, Bayesian optimization in high-dimensional spaces: A brief survey, in: 2021 12th International Conference on Information, Intelligence, Systems and Applications, IISA, IEEE, 2021, http://dx.doi.org/10.1109/iisa52424.2021.9555522.
[83] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, http://dx.doi.org/10.48550/arXiv.1412.6980.
[84] Abdullah, I. Faye, M.R. Islam, EEG channel selection techniques in motor imagery applications: A review and new perspectives, Bioengineering 9 (12) (2022) 726, http://dx.doi.org/10.3390/bioengineering9120726.
[85] C. Rommel, T. Moreau, J. Paillard, A. Gramfort, CADDA: Class-wise automatic differentiable data augmentation for eeg signals, 2021, http://dx.doi.org/10.48550/arXiv.2106.13695.
[86] C. Rommel, J. Paillard, T. Moreau, A. Gramfort, Data augmentation for learning predictive models on EEG: a systematic comparison, J. Neural Eng. 19 (6) (2022) 066020, http://dx.doi.org/10.1088/1741-2552/aca220.
[87] O. George, R. Smith, P. Madiraju, N. Yahyasoltani, S.I. Ahamed, Data augmentation strategies for EEG-based motor imagery decoding, Heliyon 8 (8) (2022) e10240, http://dx.doi.org/10.1016/j.heliyon.2022.e10240.
[88] M.N. Mohsenvand, M.R. Izadi, P. Maes, Contrastive representation learning for electroencephalogram classification, in: E. Alsentzer, M.B.A. McDermott, F. Falck, S.K. Sarkar, S. Roy, S.L. Hyland (Eds.), Proceedings of the Machine Learning for Health NeurIPS Workshop, in: Proceedings of Machine Learning Research, 136, PMLR, 2020, pp. 238–253.
[89] E.S. Sadik, H.M. Saraoglu, S. Canbaz Kabay, M. Tosun, G. Akdag, Comparison of different data augmentation methods with an experimental EEG dataset, in: 2021 13th International Conference on Electrical and Electronics Engineering, ELECO, IEEE, 2021, http://dx.doi.org/10.23919/eleco54474.2021.9677865.
[90] M. Ravanelli, M. Omologo, Contaminated speech training methods for robust DNN-hmm distant speech recognition, 2017, http://dx.doi.org/10.48550/arXiv.1710.03538.
[91] K.S. Kamble, J. Sengupta, Emotion recognition using wavelet synchrosqueezing transform integrated with ensemble deep learning, IEEE Sens. J. 24 (1) (2024) 607–614, http://dx.doi.org/10.1109/jsen.2023.3335229.
[92] M. Diachenko, S.J. Houtman, E.L. Juarez-Martinez, J.R. Ramautar, R. Weiler, H.D. Mansvelder, H. Bruining, P. Bloem, K. Linkenkaer-Hansen, Improved manual annotation of EEG signals through convolutional neural network guidance, eneuro 9 (5) (2022) http://dx.doi.org/10.1523/eneuro.0160-22.2022.
[93] G. Pfurtscheller, F. Lopes da Silva, Event-related EEG/MEG synchronization and desynchronization: basic principles, Clin. Neurophysiol. 110 (11) (1999) 1842–1857, http://dx.doi.org/10.1016/s1388-2457(99)00141-8.
[94] P. Ofner, G.R. Muller-Putz, Using a noninvasive decoding method to classify rhythmic movement imaginations of the arm in two planes, IEEE Trans. Biomed. Eng. 62 (3) (2015) 972–981, http://dx.doi.org/10.1109/tbme.2014.2377023.
[95] A. Korik, R. Sosnik, N. Siddique, D. Coyle, Decoding imagined 3D hand movement trajectories from EEG: Evidence to support the use of mu, beta, and low Gamma oscillations, Front. Neurosci. 12 (2018) http://dx.doi.org/10.3389/fnins.2018.00130.
[96] J.-H. Kim, F. Biessmann, S.-W. Lee, Decoding three-dimensional trajectory of executed and imagined arm movements from electroencephalogram signals, IEEE Trans. Neural Syst. Rehabil. Eng. 23 (5) (2015) 867–876, http://dx.doi.org/10.1109/tnsre.2014.2375879.
[97] Y. Benjamini, Y. Hochberg, Controlling the false discovery rate: A practical and powerful approach to multiple testing, J. R. Stat. Soc. Ser. B Stat. Methodol. 57 (1) (1995) 289–300, http://dx.doi.org/10.1111/j.2517-6161.1995.tb02031.x.
[98] A. Tiwari, A logistic binary jaya optimization-based channel selection scheme for motor-imagery classification in brain-computer interface, Expert Syst. Appl. 223 (2023) 119921, http://dx.doi.org/10.1016/j.eswa.2023.119921.
[99] A. Tiwari, A. Chaturvedi, Automatic channel selection using multiobjective X-shaped binary butterfly algorithm for motor imagery classification, Expert Syst. Appl. 206 (2022) 117757, http://dx.doi.org/10.1016/j.eswa.2022.117757.
[100] Z.J. Koles, M.S. Lazar, S.Z. Zhou, Spatial patterns underlying population differences in the background EEG, Brain Topogr. 2 (4) (1990) 275–284, http://dx.doi.org/10.1007/bf01129656.
[101] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, K.-r. Muller, Optimizing spatial filters for robust EEG single-trial analysis, IEEE Signal Process. Mag. 25 (1) (2008) 41–56, http://dx.doi.org/10.1109/msp.2008.4408441.
[102] A. Barachant, S. Bonnet, M. Congedo, C. Jutten, Riemannian geometry applied to BCI classification, in: V. Vigneron, V. Zarzoso, E. Moreau, R. Gribonval, E. Vincent (Eds.), Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 629–636, http://dx.doi.org/10.1007/978-3-642-15995-4_78.
[103] A. Barachant, S. Bonnet, M. Congedo, C. Jutten, Multiclass brain–computer interface classification by Riemannian geometry, IEEE Trans. Biomed. Eng. 59 (4) (2012) 920–928, http://dx.doi.org/10.1109/tbme.2011.2172210.
[104] A. Barachant, S. Bonnet, M. Congedo, C. Jutten, Classification of covariance matrices using a Riemannian-based kernel for BCI applications, Neurocomputing 112 (2013) 172–178, http://dx.doi.org/10.1016/j.neucom.2012.12.039.
[105] L. Farwell, E. Donchin, Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials, Electroencephalogr. Clin. Neurophysiol. 70 (6) (1988) 510–523, http://dx.doi.org/10.1016/0013-4694(88)90149-6.
[106] I. Obeid, J. Picone, The temple university hospital EEG data corpus, Front. Neurosci. 10 (2016) http://dx.doi.org/10.3389/fnins.2016.00196.
[107] K. Xia, W. Duch, Y. Sun, K. Xu, W. Fang, H. Luo, Y. Zhang, D. Sang, X. Xu, F.-Y. Wang, D. Wu, Privacy-preserving brain–computer interfaces: A systematic review, IEEE Trans. Comput. Soc. Syst. 10 (5) (2023) 2312–2324, http://dx.doi.org/10.1109/tcss.2022.3184818.
[108] M.-J. Schneider, J.J. Fins, J.R. Wolpaw, Ethical Issues in BCI Research, in: Brain–Computer Interfaces: Principles and Practice, Oxford University Press, 2012, http://dx.doi.org/10.1093/acprof:oso/9780195388855.003.0024.
[109] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, in: O.D. Suarez (Ed.), PLOS ONE 10 (7) (2015) e0130140, http://dx.doi.org/10.1371/journal.pone.0130140.
[110] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences, 2017, http://dx.doi.org/10.48550/arXiv.1704.02685.
[111] A. Sujatha Ravindran, J. Contreras-Vidal, An empirical comparison of deep learning explainability approaches for EEG using simulated ground truth, Sci. Rep. 13 (1) (2023) http://dx.doi.org/10.1038/s41598-023-43871-8.
[112] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013, http://dx.doi.org/10.48550/arXiv.1312.6034.
[113] M.T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?": Explaining the predictions of any classifier, 2016, http://dx.doi.org/10.48550/arXiv.1602.04938.
[114] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual explanations from deep networks via gradient-based localization, 2016, http://dx.doi.org/10.48550/arXiv.1610.02391.
[115] S. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, 2017, http://dx.doi.org/10.48550/arXiv.1705.07874.
[116] H. Banville, O. Chehab, A. Hyvärinen, D.-A. Engemann, A. Gramfort, Uncovering the structure of clinical EEG signals with self-supervised learning, J. Neural Eng. 18 (4) (2021) 046020, http://dx.doi.org/10.1088/1741-2552/abca18.