BCI: MI classification

Priscilla Cortese and Alessandro Piani

Università degli studi di Milano Bicocca

Abstract. Classification of Electroencephalogram (EEG) signals during the performance a two class Motor Imagery (MI) tasks by different subjects. In this report we present different methods to handle such problem, using both ML and DL approaces, and test them on data from 2008 BCI competition data set 2b.
The first method relies on feature extraction through Filter Bank Common Spatial Pattern (FBCSP) and classification via Support Vector Machine (SVM). The second one uses the features extracted with FBCSP and classifies them with the use of a Long Short Term Memory (LSTM) architecture.
A brief overview of the results obtained using CNN on spectrograms of the analyzed signals is also provided.

Keywords: Motor Imagery · EEG · Classification · FBCPS

1 Introduction

Electroencephalogram (EEG) records the brain electric potentials deriving from the activation of the cerebral cortex. In particular, as the present report regards techniques for MI classification, we will exploit EEG signals measured by three electrodes (C3, Cz and C4) which are placed on the scalp of the subject close to the motor cortex, which is the area devoted to movement control.

EEG signals are characterized by different frequency bands, called rhythms: in the following, the methods proposed will be based on the fact that μ and β rhythms are the most insightful for MI as they regard motor cortex functionalities, alert state and active thinking.

Even though EEG signals are filled with both spacial and frequency information, they present many issues which can have a negative effect on the classification's performance.
First of all, they are quite sensitive to noise, both physiological (e.g. ocular artifacts) and non-biological (e.g. power line noise): these problems will be handled by appropriately filtering the signals.

EEG data is also subject-specific and presents intra-subject variability, meaning that both the same subject in a different situation and different subjects performing the same task will produce different signals: such issue will be attenuated by normalizing data. Because of these characteristics, we've decided to evaluate our models using a Leave One Subject Out (LOSO) cross validation.

Such validation technique gives the opportunity of seeing how the model performs on an ”unseen” subject, as it uses as training data signals coming from different subject with respect to the one used for test.

2 Materials and Methods

2.1 Data set

Data used in the proposed work comes from the 2008 BCI competition data set 2b.[1] Such data set consists of six channel signals from 9 subjects during 5 different sessions, the first two without feedback (screening) and the last three with feedback. Based on a cue presented during the experiment, subjects had to perform a MI task of either the left or right hand and their neural activity was tracked by three bipolar recordings (C3, Cz and C4) designed for EEG signal collection and three monopolar electrodes for EOG signal.

For sake of simplicity, only the screening sessions were considered. Each session consists of several runs: the first one is devoted to estimate ocular artifacts (e.g. blinking, rolling, moving the eyes...) whereas in the last six the MI tasks are performed. These last sessions consisted of ten trials for each of the two classes of imagery, yielding a total of 120 trials per person per session.

Note that, due to technical problems during the recording of data, no EOG signal is available for the second session of the first subject so there isn't a calibration run in this particular session. This matter is handled in the code. During each trial a visual cue (that showed to the subject the requested class) was presented for 1.25s, then the subject had to imagine the corresponding hand movement for 4s and lastly a break of at least 1.5s was given.

The sampling frequency of the EEG channels is of 250Hz; each recording was band-pass filtered between 0.5Hz and 100Hz, and a notch filter at 50Hz was applied in order to remove the power-line noise.

It has to be noted that the considered data set was partially available in mat and GDF format.[2] The files in the former format are called ”B0sT.mat”, where s stands for the considered subject and goes from one to nine, each one contains a structure with three cells: the first two are the screening sessions and the last is the first feedback session, which isn't of our concern. Inside each cell EEG and EOG signals for each trial are stored in the variable X and the corresponding label in the variable y. Measurements of each run stored in X are separated by 100 NaN values which will be removed when re-organizing data.
Key information for signal segmentation such as the time-stamps and the duration of the different events (start of a new run, start of a trial, cue onset...) was stored only in the GDF files. Such files have to be opened in MATLAB using BioSig toolbox, which makes the programming platform less comfortable to use: for this reason relevant data was loaded only once with the help of the above mentioned toolbox to then be re-organized saved as ”headers.mat”. This

file, submitted with the present report, is a structure of 9x2 cells where each cell contains the header of a given subject for a given session.

While testing the different methods, we noticed how the accuracy increased significantly when removing the second and the third subject. This difference was particularly evident in the second method, as shown by the following table.

Table 1: Accuracy on each subject when performing LOSO validation with method 2.

Subj 1	Subj 2	Subj 3	Subj 4	Subj 5	Subj 6	Subj 7	Subj 8	Subj 9	Mean
0.5626	0.49583	0.51667	0.5375	0.5375	0.55417	0.5125	0.52083	0.54167	0.5310
0.6208			0.6125	0.6500	0.6333	0.6292	0.6792	0.6417	0.6381

Since the improvement in accuracy is not negligible, we've decided to discard data from subject 2 and 3 in all the proposed methods.

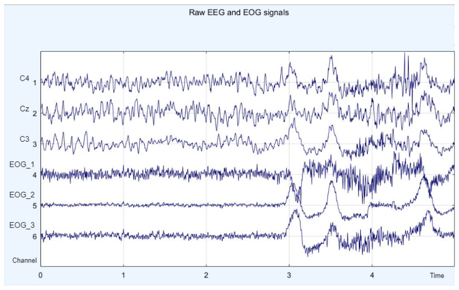
2.2 Segmentation and preprocessing

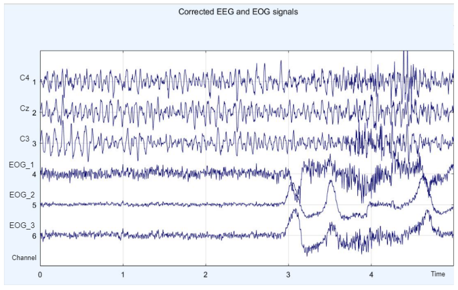
This first steps of the classification pipeline are in common for all the methods that will be presented, so they're explained only once here.

Data of each session has to be segmented in windows of the same size to then be processed: we have chosen to make such windows cover the cue presentation, the motor imagery task and the 1.5s following it. The latter probably allows to record part of the neural signal which would otherwise be lost as there is a delay between signal emission and collection; in any case this set of intervals was the one which seemed to give the best results in the next steps.

Ocular artifacts usually cover the frequency band of 0.5-3Hz, in order to remove them an high-pass 6-th order Butterworth filter was applied to EEG signals and EOG signals were discarded.

To reduce inter-subject differences the EEG signals were normalized using the z-score normalization.





Raw EEG and EOG signal Corrected EEG and EOG signal

Fig. 1: Comparison of the same window signal before and after high-pass filtering EEG channels

2.3 First method

This method follows a classic machine learning pipeline for data classification: after segmenting and preprocessing data, features are extracted and fed to a classifier. The first and part of the second step of the pipeline have been described in section 2.2. The remaining part of the pipeline is shown below.

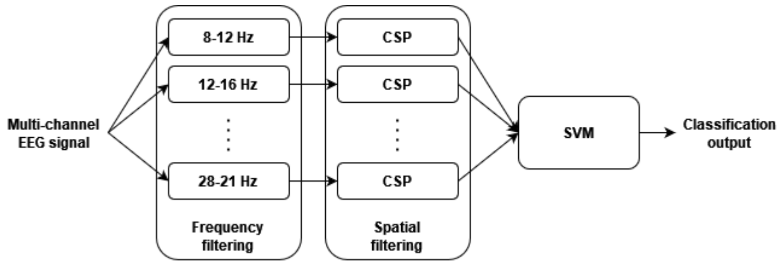


Fig. 2: Method 1 implementation schema

Feature extraction In order to extract relevant features from EEG signals, the Filter Bank Common Spatial Pattern (FBCSP) algorithm was applied.[3] Such algorithm enhances the performance of the CSP algorithm by selecting some characteristic frequency bands and applying the CSP algorithm to them. Note that FBCSP algorithm is applied separately to each measurement session of each subject.
The first step of the algorithm decomposes the EEG signals in multiple frequency bands using a filter bank of 4-th order Butterworth band pass filters. As already stated in the introduction, the relevant rhythms for MI are μ (8-13 Hz) and β (13-30 Hz): we decided to uniformly cover such frequencies using six band pass filters in the ranges 8-12, 12-16, ... 28-32 Hz.

In the second step, spatial filtering is performed using CSP algorithm. The CSP algorithm is applied separately to each of the band-pass filtered signals obtained in step 1: this allows to extract CSP features specific to each frequency range.

The CSP algorithm linearly transforms the considered signals through W_b called CSP projection matrix, which is specific to each frequency band b. Such matrix can be found by solving the eigenvalue decomposition problem

$$S_{b,1} W_b = (S_{b,1} + S_{b,2}) W_b D_b, \tag{1}$$

where $S_{b,1}$ and $S_{b,2}$ are the covariance matrices of the b-th band-bass filtered EEG measurements of respectively class 1 (left) and 2 (right), D_b is the diagonal matrix containing the eigenvalues of $S_{b,1}$.

In order to compute the spatial filtered EEG signals $Z_{b,i}$ for each band b and each trial i, a simple matrix multiplication has to be performed:

$$Z_{b,i} = W_b^T E_{b,i}. \tag{2}$$

The spatial filtered signal $Z_{b},_{i}$ in the previous equation maximizes the differences in the variance of the two classes of band-pass filtered EEG signals.
We obtained a pair of CSP features of the i-th trial for the b-th band-pass filtered EEG signals by computing

$$v_{b},_{i} = log \frac{diag(\hat{W}_b^T E_{b},_{i} \, E_{b},_{i}^{T} \, \hat{W}_b)}{tr(\hat{W}_b^T E_{b,i} E_{b,i}^T \hat{W}_b)}, \tag{3}$$

where $\hat{W}_b^T$ represents the first and the last column of W_b.[3]
To form the feature vector of the different trials for a given band b, it is sufficient to vertically concatenate the values obtained in equation 3:

$$\bar{v_b} = \begin{bmatrix} \mathrm{v}_{b,1} \\ \mathrm{v}_{b,2} \\ \vdots \\ \mathrm{v}_{b,120} \end{bmatrix} \tag{4}$$

Lastly, to obtain the complete matrix of features for the given subject in the given session we must store the columns obtained in (4) in a single matrix:

$$V = \begin{bmatrix} \bar{v_1} \; \bar{v_2} \ldots \bar{v_6} \end{bmatrix} \tag{5}$$

Classification To be able to classify the features extracted in the previous section, we used a Support Vector Machine (SVM). In particular, we used a cubic polynomial SVM as it was the one which yielded the best results. Lastly, we performed LOSO cross validation.

2.4 Second method

This method follows a deep learning approach to perform data classification, specifically it uses the features extracted with FBCSP and feds them to a Long Short-Term Memory (LSTM) architecture in order to classify them. Even in this case we performed LOSO cross validation, so the LSTM network was trained seven times (one for each considered subject) using data from a given subject as test and the data from the remaining ones as training data.

A LSTM neural network is a type of recurrent neural network (RNN) that is capable of learning long-term dependencies between time steps of sequence data. It is able to remember information for long periods of time and connect previous information to the present task, in the same way as using previous words of a sentence might inform the understanding of the present word [5].

The diagram below illustrates the architecture of our LSTM neural network used for sequence-to-label classification. The network starts with a sequence input layer, used to input sequence or time series data into the neural network, and is followed by a BiLSTM layer. A bidirectional LSTM (BiLSTM) layer is an RNN layer that learns bidirectional long-term dependencies between time steps of time series or sequence data.
Then a dropout layer is added to prevent overfitting. In order to predict class labels, the neural network ends with a fully connected layer, a softmax layer, and a classification output layer.

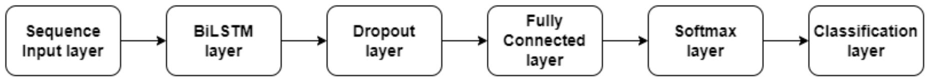


Fig. 3: LSTM neural network architecture

LSTM layer operations The LSTM network layer has a chain-like structure of repeating modules (LSTM blocks) each containing 4 neural network layers (gates) interacting in a special way. [5]
At each time step t, the LSTM block uses the current state of the RNN $(\boldsymbol{c_{t-1}}, \boldsymbol{h_{t-1}})$, denoted respectively by the current cell state and by the current hidden state (or output state), and the next time step of the sequence $\boldsymbol{x_t}$ to compute the output $\boldsymbol{h_t}$ and the updated cell state $\boldsymbol{c_t}$.

The hidden state is also called output state because at time step t it contains the output of the LSTM layer for this time step, while the cell state contains information learned from the previous time steps.

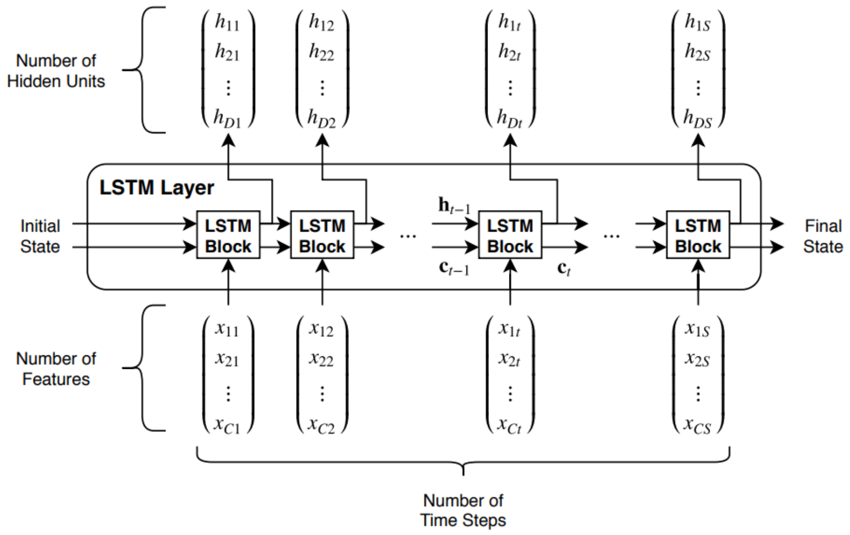

Number of Hidden Units
h_{11} h_{21} h_{D1}
h_{12} h_{22} h_{D2}
h_{1t} h_{2t} h_{Dt}
h_{1S} h_{2S} h_{DS}
LSTM Layer
Initial State
LSTM Block
$\mathbf{h}_{t-1}$
$\mathbf{c}_{t-1}$
$\mathbf{c}_t$
Final State
Number of Features
x_{11} x_{21} x_{C1}
x_{12} x_{22} x_{C2}
x_{1t} x_{2t} x_{Ct}
x_{1S} x_{2S} x_{CS}
Number of Time Steps

Fig. 4: This diagram illustrates the flow of a time series X with C features (channels) of length S through an LSTM layer. [6]

The key to LSTM layer is that, at each time step, it can add or remove information from the cell state, carefully regulated by structures called *gates*. These components allow to optionally let information through, and are composed out of a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between 0 and 1, describing how much of each component should be let through (a value of 0 means "let nothing through", while a value of 1 means "let everything through") [5].

Each LSTM block of the LSTM layer has 4 of these gates, to control the cell state and hidden state:

1. Forget gate layer $\boldsymbol{f}$: a sigmoid layer used to decide what information is going to be thrown away from the cell state. Here the formula to compute it at time step t:
$$f_t = \sigma_g(W_f \boldsymbol{x}_t + R_f \boldsymbol{h}_{t-1} + b_f), \tag{6}$$
where σ_g denotes the gate activation function.
2. Input gate layer $\boldsymbol{i}$: a sigmoid layer used to decide which values will be updated.
$$i_t = \sigma_g(W_i \boldsymbol{x}_t + R_i \boldsymbol{h}_{t-1} + b_i) \tag{7}$$
3. Cell candidate layer $\boldsymbol{g}$: a sigmoid layer used to create a vector of new candidate values that could be added to the state.
$$g_t = \sigma_c(W_g \boldsymbol{x}_t + R_g \boldsymbol{h}_{t-1} + b_g), \tag{8}$$
where σ_c denotes the state activation function.

4. Output gate layer $\boldsymbol{o}$: a sigmoid layer used to decide what parts of the cell state will be output.

$$o_t = \sigma_g(W_o \boldsymbol{x}_t + R_o \boldsymbol{h}_{t-1} + b_o) \tag{9}$$

The outputs of the first 3 gates are combined to create an update to the cell state:

$$\boldsymbol{c}_t = f_t \odot \boldsymbol{c}_{t-1} + i_t \odot g_t, \tag{10}$$

where $\odot$ denotes the element-wise multiplication of vectors, while the hidden state is given by:

$$\boldsymbol{h}_t = o_t \odot \sigma_c(\boldsymbol{c}_t) \tag{11}$$

It is worth noting that the learnable weights of an LSTM layer are the input weights $\mathbf{W}$, the recurrent weights $\mathbf{R}$ and the bias $\mathbf{b}$, which are concatenated and stored in matrices in this way:

$$W = \begin{bmatrix} \mathrm{W}_i \\ \mathrm{W}_f \\ \mathrm{W}_g \\ \mathrm{W}_o \end{bmatrix}, \ R = \begin{bmatrix} \mathrm{R}_i \\ \mathrm{R}_f \\ \mathrm{R}_g \\ \mathrm{R}_o \end{bmatrix}, \ b = \begin{bmatrix} \mathrm{b}_i \\ \mathrm{b}_f \\ \mathrm{b}_g \\ \mathrm{b}_o \end{bmatrix}, \tag{12}$$

where $\boldsymbol{i}$, $\boldsymbol{f}$, $\boldsymbol{g}$ and $\boldsymbol{o}$ denote the 4 gates. [6]

As previously stated, we used a BiLSTM layer and it works similarly to the LSTM layer. Basically, BiLSTM adds one more LSTM layer, which reverses the direction of information flow: this means that the input sequence flows backward in such additional LSTM layer. Then the outputs from both LSTM layers are combined in several ways, such as average, sum, multiplication, or concatenation.

2.5 Possible alternatives

The former idea, when brainstorming for this project, was to compare a method based on ML and handcrafted feature extraction with a DL method that automatically extract features. In particular, we wanted to perform classification via a two-dimensional CNN that took as inputs the (processed) spectrograms of the signal's windows.
The spectrogram is a visual way of representing the signal strength of a signal over time at various frequencies, by our 1D signals in this representation we obtained a 2D images for each channel. To make the representation of each window's signal more compact we discarded the spectrogram relative to Cz channel and we subtracted the one obtained with signals from C4 to the one from C3. The idea behind this is that the Cz channel doesn't add many relevant information and since C4 and C3 record signals from the two different hemispheres of the brain, by subtracting the spectrograms we can still maintain the spatial information given by them [4].

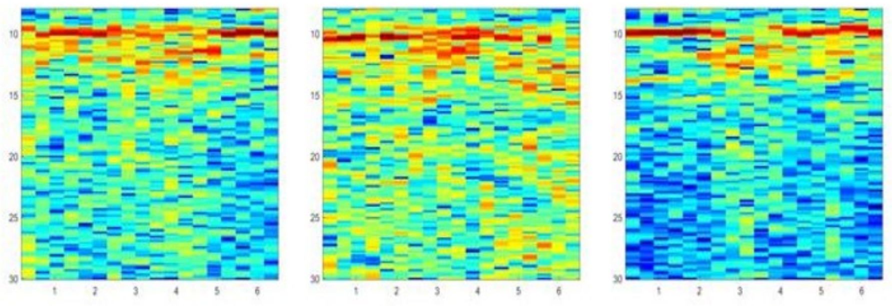


Fig. 5: Examples of extracted and processed spectrograms.

CNNs are neural networks mainly composed of convolutional layers (CL), pooling layers (PL) and fully connected layers (FC).
Given the small amount of data at our disposal, the idea was to exploit pretrained networks instead of creating one from scratch. We made several experiments using GoogleNet and ResNet50, and we followed two approaches.

In the first one we implemented the so called "transfer learning", meaning that we loaded the pretrained network with its weights and we modified only the last layers of the architecture in order to adapt them to our task, while keeping the rest frozen. In particular, we replaced the last FC layer with another FC layer which has only two neurons, corresponding to the two classes, the classification layer was also modified. Spectrograms were given as input to the network and they were used to fine-tune the last layers.

The second approach relied on the idea of using the pretrained network (until its last FC layer) as an automatic feature extractor and giving its output to a traditional classifier (in our case the same SVM used in the first method). Since the imported networks extract a huge number of features, before classification we performed feature selection using the relief algorithm and picking the best 30 features (this is the number which, experimentally, yielded the best results).

3 Experiments

In this section the results obtained with the previously presented methods will be analyzed.

3.1 First method

As previously, stated we performed LOSO cross validation so the SVM was trained seven times (one for each considered subject) using data from a given subject as test and the data from the remaining ones as training data.
To evaluate the model performance, we computed the accuracy obtained in each fold (on each iteration) and then we derived a comprehensive confusion matrix considering the predictions made in all the seven folds.

Table 2: Accuracy on each subject when performing LOSO validation with method 1.

Subject 1	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Mean
0.6250	0.5542	0.5833	0.5792	0.6083	0.7792	0.7708	0.6429

Prediction accuracy changes when testing on different subject: this is probably be due to the fact that EEG signals are highly subject specific, so it is normal that the accuracy level is not constant even though the signals were normalized.

Table 3: Total confusion matrix of method 1 predictions.

	Predicted class	
Real Class	Left	Right
Left	537	303
Right	297	543

It's possible to notice how wrong predictions are made uniformly in both classes, so the difficulties faced with classification don't depend on the class of the signal.

3.2 Second method

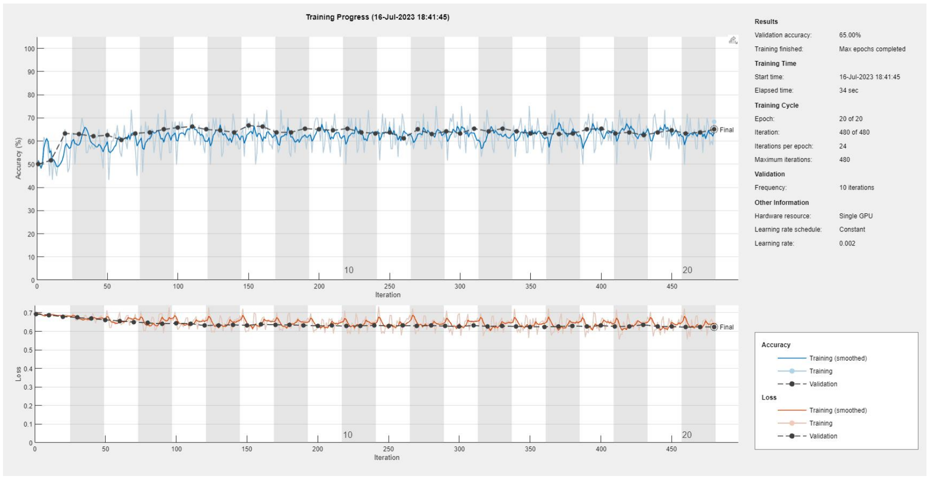
The LSTM architecture that we presented section 2.4 takes as input sequences of data; a first experiment was done by using as inputs the three channel EEG signals obtained after segmentation. Even in this case we performed LOSO cross validation, so we trained the architecture seven times and tested it on the only subject that was kept out of the training set.

Table 4: Accuracy on each subject when performing LOSO validation with preprocessed signals as input.

Subject 1	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Mean
0.4958	0.5166	0.4791	0.4750	0.4375	0.5000	0.4750	0.4827

The results weren't satisfactory at all, as the model performed worst than guessing at random: this may be due to the dimensions of the signals.

In the second experiment we decided to feed to the network the features extracted with FBCSP from method 1: in this way the sequence input is simply a sequence of features, so it is mono-dimensional, and it captures important characteristics of the signals. BiLSTM layer further processes such features and which are then classified by the network. To speed up the process, the features extracted with FBCSP have been saved and submitted in the file named ”features_csp.mat” (”features_csp_all.mat”is also provided and it contains features from all 9 subject; it is used in section 2.1 to show how the accuracy increases when considering only 7 subjects).



Results
Validation accuracy: 65.00%
Training finished: Max epochs completed
Training Time
Start time: 16-Jul-2023 18:41:45
Elapsed time: 34 sec
Training Cycle
Epoch: 20 of 20
Iteration: 480 of 480
Iterations per epoch: 24
Maximum iterations: 480
Validation
Frequency: 10 iterations
Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.002

Fig. 6: Accuracy and loss over 20 epochs during one training session.

Table 5: Accuracy on each subject when performing LOSO validation with method 2 using FBCSP features as input.

Subject 1	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Mean
0.6208	0.6125	0.6500	0.6333	0.6292	0.6792	0.6417	0.6381

3.3 Possible alternatives

The possible alternative methods based on CNN that were presented in section 2.5 weren’t explained much in depth: the reason behind this choice is the fact that they didn’t lead to satisfactory results. In fact, both approaches (transfer learning and automatic feature extraction with traditional classifier), with both GoogleNet and ResNet50, reached results comparable to random guessing.

Table 6: Accuracy on each subject when performing LOSO validation using GoogleNet and transfer learning

Subject 1	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Mean
0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5708	0.5101

Table 7: Total confusion matrix of predictions using ResNet50 as feature extractor.

	Predicted class	
Real Class	Left	Right
Left	347	493
Right	338	502

The mean accuracy obtained with this last method was of 0.5054: this architecture basically guesses at random.

We tried to investigate the reasons behind his and in our opinion this is due to many factors:

1. Different tasks: The two considered networks are built for image classification, but they are trained to perform classification of 1000 classes of objects and animals, which is a task quite different from ours. This could lead to poor results since the features extracted with nets trained for such task could not be really significant to perform EEG signal classification.
2. Shortage of training data: Even though this problem was partially solved by using pre-trained nets, the amount of data at our disposal probably wasn't enough to allow a successful training.
3. Input images: The spectrograms used as inputs were processed in order to highlight important characteristics of EEG data, but they probably didn't capture them well enough. Also, looking at the images produced for the different subject, it's possible to notice that they are quite subject-specific: probably more sophisticated normalization methods are needed to make the images more "general".

4 Conclusion

In this project we explored different ways to solve the problem of classification of EEG signals regarding the performance of Motor Imagery tasks. We tried to handle the problems of such signals due to noise and subject dependency, to then create some methods able to classify data from new subjects.

Let's summarize the obtained results by comparing the average accuracy obtained by the proposed methods.

Table 8: Mean accuracy with different methods

FBCSP+SVM	LSTM based	CNN transfer	CNN features
0.6429	0.6381	0.5101	0.5054

The method which yielded the best results is the one based on FBCSP features and SVM classifier, followed by the LSTM architecture with FBCSP features as input and lastly the CNN based methods.
In the previous section we tried to unfold the reasons behind the poor results of CNN based approaches, so it is not surprising to see the other methods achieve much better results. With FBCSP it's possible to extract features that are relevant with respect to the considered task, so they are actually helpful to classify the signals.

References

1. Leeb R., Brunner C. et al (2008) BCI Competition 2008 - Graz data set B
2. BNCI Horizon 2020 `https://bnci-horizon-2020.eu/database/data-sets`
3. Ang K. K., Chin Z. Y. et al (2008) Filter Bank Common Spatial Pattern (FBCSP) in Brain-Computer Interface.
4. Chen Z., Wang Y., and Song Z. (2021) Classification of Motor Imagery Electroencephalography Signals Based on Image Processing Method.
5. Christopher Olah Colah's blog Understanding LSTM Networks (2015) `https://colah.github.io/posts/2015-08-Understanding-LSTMs/` .
6. Mathworks Long Short-Term Memory Neural Networks `https://it.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html` .