Analyzing brain connectivity dynamics during sleep

Mélanie Bernhardt

Data Science M.Sc. student
Student ID: 17-945-676

Research in Data Science Semester Project
Supervisor: Joachim Buhmann
Advisor: Djordje Miladinovic
Summer 2018

Abstract In this report, we show how graph neural models can be applied on multi-frequency brain connectivity data derived from MEG recordings in the context of sleep stage classification. Indeed, the brain can be viewed as a graph where brain regions are nodes and connectivity measures between regions represent weighted edges of this graph. However, many standard brain connectivity measures are defined in the Fourier domain yielding one connectivity value per frequency in the spectrum. In other words, at every time-step, the brain can be described by several weighted graphs (each corresponding to a different frequency of the spectrum). Consequently, here we propose an adaptation of a graph convolutional network capable of simultaneously taking several adjacency matrices as an input in order to classify sleep stages. The performance of this model was evaluated on a binary classification task consisting of discriminating the Rapid Eye Movement (REM) sleep stage from the other sleep stages. The model was compared to standard baselines in three different cross-validation settings.

Institute for Machine Learning
ETH Zurich

Contents

Introduction

Sleep is of utmost importance for human health. Hence, understanding the functioning of the brain in sleep phase is an eminent neuroscience challenge. In this context, it is of prime interest to investigate how state-of-the-art machine learning techniques can enhance brain connectivity analysis. For this purpose, MEG (Magnetoencophalogram) recordings provide extremely fine-grained temporal information about the brain activity. MEG data is therefore particularly suitable for analyzing brain connectivity with machine learning techniques. MEG measurements consist of multiple time series recorded at millisecond precision by several sensors dispatched on the head's surface. However, it is not trivial to identify the true source of a signal in the brain from these measurements. Indeed, several sensors can record a signal coming from the same source in the brain [SG09]. In order to solve this issue, neuroscientists have developed several source localization methods and brain connectivity measures to disentangle different brain signals [SG09]. One of these measures is the *imaginary coherence* which quantifies how much different regions in the brain communicate with each other at every point of time.

On the other hand, neural approaches on graph data have attracted great attention in the last years in the machine learning community, in particular with the introduction of graph convolutional neural networks [Bru+13]. In this context, it is important to note that the brain can be viewed as a graph where each region of interest (ROI) is a node of the graph and each connectivity measure between two ROIs represents a weighted edge between the corresponding nodes.

In this project, we will show how these cutting-edge methods can be applied to sleep stage classification. More precisely, we will focus on discriminating the last sleep stage, namely the Rapid Eye Movement (REM) sleep stage from the other sleep stages (known as non-REM sleep stages). In the first section, we will review standard brain connectivity analysis methods described in the neuroscience literature. In the second section, we will present the data that was used for this project in the sleep stage classification task. In the third section, we will first review graph neural approaches presented in the recent literature. Then, we will present a model capable of taking five different graphs as an input to classify sleep stages. In the fourth section, we will compare the performance of our graph neural model to the performance of standard baselines in three different cross validation settings. In particular, we will assess the predictive power of the model on classifying observations from a unseen subject through across-subjects cross-validation. Finally, in the last section, results and possible extensions to this project will be discussed.

1 Brain connectivity analysis with EEG/MEG signals

Brain connectivity analysis is a branch of neuroscience which studies interactions between different regions of the brain. As explained in [SG09], two types of brain connectivity have to be distinguished. First, "functional connectivity" quantifies only statistical dependencies between brain regions without dealing with the notion of causality. In functional connectivity analysis, only undirected interaction are analyzed. On the other hand, "effective connectivity analysis" aims to analyze the direction (i.e. causal relation) in the brain region dependencies. [SG09] details the various connectivity measures that can be used for both types of connectivity analysis.

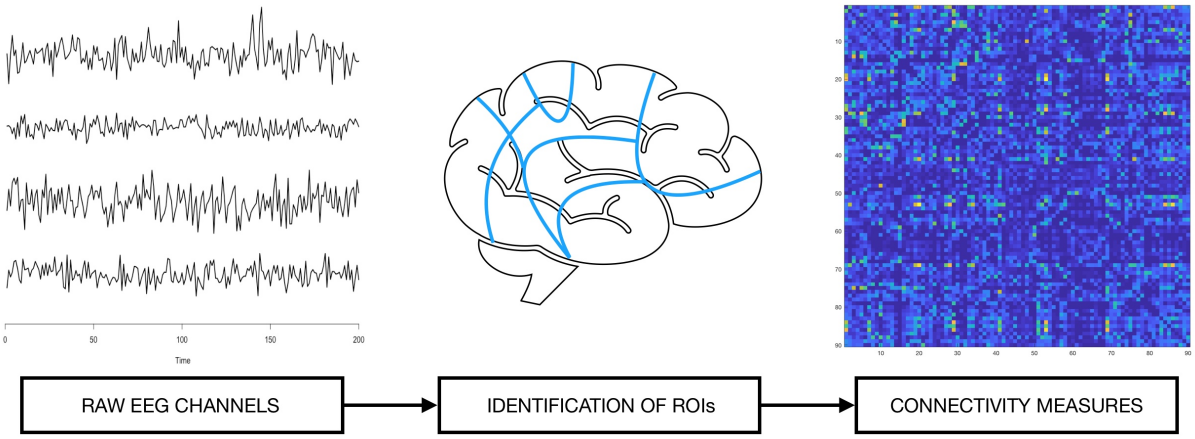


Figure 1: *2-step procedure for brain connectivity analysis*

1.1 Principles of brain connectivity analysis with EEG/MEG signals

1.1.1 EEG/MEG recordings and the problem of field spread

As explained in [KB12], EEG (Electroencephalogram) is a device that reads electrical potentials of the brain. Several electrodes are disposed on the scalp, each of them corresponding to one EEG channel. Similarly, MEG (Magnetoencophalogram) records the magnetic field of the brain. These two brain imaging techniques allow for millisecond whole-head measurements. The main problem associated with the interpretation and analysis of EEG/MEG for brain connectivity analysis is the so-called "volume conduction" or "field-spread" effect. As explained in [SG09], the main issue with connectivity analysis from EEG arises from the fact that a signal caused by a specific source in the brain is not only recorded by the electrode which is placed nearest to the source of the signal but also by all neighboring sensors. In [SG09], they illustrated this problem of field spread with simulated data showing how field spread corrupts estimates of brain connectivity at the sensor level. Hence, without addressing this issue, what could be interpreted as an interaction between two brain regions is likely to merely be noise resulting from the fact that a given source activity is recorded by both sensors [Nol+04]. This problem has given rise to so-called "source localization methods" that aim to identify the true underlying source of a signal.

1.1.2 Two-step procedure for connectivity analysis

The most common procedure for brain connectivity analysis is a two-stage procedure consisting of a source modeling phase followed by a connectivity analysis phase. Several methods and measures have been presented in the literature for each of these two stages. A survey of the most-widely used techniques can be found in [SG09]. The source modeling phase consists of identifying regions of interest (ROIs) of the brain. Then, in the second step, connectivity measures are computed to identify interactions between these ROIs.

One of the method that can be used for source modeling is based on brain neural activity maps and the Minimum Norm Estimate (MNE). This method was introduced by [Dav+02]. In their paper, they explain that *the principle consists in reducing the source space in an iterative way in order to minimize estimation biases in both spatial and temporal dimensions.* Hence, this method allows to identify neurons that can be grouped together for the connectivity analysis, these groups form the ROIs.

After having identified regions of interest, brain connectivity measures can be computed to estimate interactions between these regions over time. As summarized in [SG09], several connectivity measures are used across the literature. Two main characteristics distinguish these measures. The first distinction is between time-domain and frequency-domain measures. As explained in [BS16], it can be useful to work in the frequency domain as connectivity analysis aims to identify *rythmic components* of the signal. The second distinction is with regard to the directed or undirected aspect of the measure. In the next subsection, we will focus on one particular (undirected, frequency-based) connectivity measure: *imaginary coherence* (as this is the measure used for this project).

Finally, it is worth mentioning that the main shortcoming of this two-step method is the fact that the connectivity measures derived in the second step of the procedure heavily rely on the ROIs identified in the first step. Different ROIs definitions imply different connectivity values in the second step. The problem is that the identification of ROIs in the source modeling phase strongly relies on assumptions made about the underlying sources. If these assumptions are wrong, the source localization (i.e. ROI identification) can be wrong [Nol+04]. For this reason, recent developments aimed to perform connectivity analysis in a one-single step procedure in order to avoid the separate source modeling step. Examples of such procedures can be found in [Yan+16] or in [Ulr+14]. For example, in [Yan+16], they used an auto-regressive model applied on the original EEG time-series to derive a measure of the cross-region dynamic connectivity. In their model, they avoid computing separately the ROIs and the associated measures. The reader is referred to their paper for more details about these single-step methods as they are out-of-scope of this project regarding the data available for the analysis (raw MEG time-series are not available, see section 2).

1.2 Imaginary coherence

1.2.1 Definitions

Coherence is a connectivity measure which is the equivalent of cross-correlation in the frequency domain. As mentioned in [Nol+04], it measures the linear relationship between two signals at the specific frequency. Coherence can be derived in the following steps (mathematical definitions are taken from [Nol+04]):

- Given two time-series t_i and t_j (from two EEG sensors), compute their complex Fourier transforms $x_i(f), x_j(f)$ to map these signals to the frequency domain.
- Use the obtained representation of the signals to compute the *cross spectrum* defined as :

  $$S_{ij}(f) = \langle x_i(f), \overline{x_j}(f) \rangle$$

  where $\langle\rangle$ means expectation (in practice computed by taking the average over sufficiently large number of epochs).
- Using the above-defined cross-spectrum, the *coherency* between channels i and j can finally be computed for each frequence. Coherency is defined as the normalized cross-spectrum of the signals:

  $$C_{ij}(f) = \frac{S_{ij}(f)}{\sqrt{S_{ii}(f) S_{jj}(f)}}$$
- Taking the absolute value of the coherency we get the *coherence*:

  $$\mathrm{Coh}_{ij}(f) = |C_{ij}(f)|$$

- Taking the imaginary part of the coherency we finally get the *imaginary coherency* between signal i and j:
$$\mathrm{Imcoh}_{ij}(f) = \mathrm{Im}(\mathrm{C}_{ij}(f))$$
where Im denotes the imaginary part. Similarly we get the *imaginary coherence* by taking the absolute value of the imaginary coherency.

Hence, for each specific frequency, an imaginary coherence matrix can then be derived:

$$M(f) := (|\mathrm{Imcoh}_{i,j}(f)|)_{1 \leq i,j \leq R}$$

where R denotes the number of ROIs identified by the source modeling step and i, j are the indices corresponding to these regions. Given the above definitions, this matrix contains the imaginary coherence values for each pair of ROIs and is symmetric.

1.2.2 Advantage of imaginary coherence measures

In [Nol+04], the idea of using the imaginary part of coherency is motivated by the fact that the (standard) coherence is affected by the field spread problem mentioned earlier. Indeed, they show that (standard) coherence values between close-by electrodes are always very high. But these high values reflect the effect of volume conduction rather than a true interaction.

To understand the idea behind how imaginary coherence solves this problem, it is useful to recall that the cross-spectrum can be rewritten

$$S_{ij}(f) = \langle r_i r_j \exp(i(\Phi_i - \Phi_j)) \rangle := \langle r_i r_j \exp(i\Delta\Phi) \rangle$$

where $\Delta\Phi$ is the phase shift between both signals. As explained in [Nol+04], a phase-shift in the frequency domain corresponds to a time-lag in the time domain. Suppose that a signal captured by two different electrodes is only caused by volume conduction. In that case, the signal is simply noise with regard to brain interaction measures. Moreover, this signal is not time-lagged between both sensors implying that the imaginary coherence of this volume conduction signal is null. For this reason, imaginary coherence captures only true interactions, only coordinated activity can be captured by this measure. A superposition of independent signals yields a real coherence, hence will not be captured by the imaginary coherence which is what is desired in connectivity analysis as the goal is to identify interacting brain regions.

Note that, even if imaginary coherence is showed to be more robust to the effect of field spread compared to coherence, it can nevertheless vanish if there is a symmetric delay of the signals between two brains regions. And it is strongly dependent on the source modeling phase and the identified ROIs. This is the main motivation behind using new models that avoid this two-step procedure (as e.g. in [Yan+16], see section 1.1.2).

1.3 Viewing brain connectivity as a graph

The connectivity matrix, i.e. in our context the symmetric imaginary coherence matrix, can be viewed as the adjacency matrix of a graph. Indeed, if we view the brain as a (fully connected) graph were each node corresponds to a brain region, the imaginary coherence matrix is exactly the weighted adjacency matrix describing this graph. The stronger the interaction between two brain regions, the higher the connectivity value and the stronger the edge between the corresponding nodes in the graph. Note that as the adjacency matrix is symmetric, the corresponding graph is undirected.

1.4 Sleep stages

Neuroscientists have defined a classification of different sleep stages. Each of them is characterized by specific patterns in EEG recordings. There are typically five sleep stages, four of them are known as "non-REM" sleep stages whereas the last one is the "REM" (Rapid Eye Movement) sleep stage. A description of sleep stages and sleep stage annotation methodology is detailed in [KA14]. The five sleep stages' main characteristics presented in this article can be summarized as follow:

1. Stage Wake: consists of the first minutes of the record when the subject is still awake.
2. Stage N1 NREM: known as "transitional" or "light sleep". The subject starts falling asleep. This stage is characterized mainly by theta waves with intermittent alpha activity in the EEG signals and eyes began to slowly roll.
3. Stage N2 NREM: "sigma", "spindle" or "intermediate sleep". This sleep stage can represent up to half of the recording for adults. It is characterized by theta waves with minimal alpha activity.
4. Stage N3 NREM: also know as "deep sleep" stage or "slow wave sleep" (SWS). It is the *most refreshing and restorative sleep type which tends to diminish with age* [KA14].
5. Stage REM sleep: also called "paradoxical" or "active" sleep. The breath is irregular and rapid eye movements are observed (as in the Wake stage). The dreaming process happens in this sleep phase.

Note that all sleep stages can be further broken down into sleep substages, see [KA14].

Characteristics of the REM sleep stage are very different than those from previous sleep stages. Hence, sleep stages can be separated in two groups: REM and non-REM sleep stages (Wake, N1, N2, N3). In this project, the classification task presented in the experiments section will consist of attributing a REM or non-REM label to each observation (i.e. for each timestep).

2 Data

2.1 Data acquisition

The data available for this project came from whole-night MEG signals recorded on ten different subjects. This data was collected by the lab of Prof. Steffen Gais from the University of Tübingen. Additionally, 2 EEG channels were recorded for sleep stage scoring by hand. Furthermore, for each subject, fMRI images were taken for the source modeling phase (see section 1.1.2). As mentioned earlier in this report (see section 1.2), imaginary coherence is a commonly used measure for quantifying brain region interactions. For this project, we did not have access to the raw MEG signals but only to the final imaginary coherence matrices derived from them. More precisely, data preprocessing, source modeling and computation of imaginary coherence matrices were taken care of by Emily Coffey from the University of Tübingen. During the source modeling phase, the subjects' MRI were first segmented with the Freesurfer software, then weighted Minimum Norm Estimate source models were applied. Based on these models, 90 regions of interest (ROIs) were identified per subject. Finally, imaginary coherence values were calculated on these identified ROIs, in 5s time windows. The acquisition frequency resolution was 0.1 Hz ranging from 1 to 48 Hz, grouped into 50 frequency bins. Hence, for each frequency bin, for each 5s time-window, one imaginary coherence symmetric matrix was computed using the Brainstorm software [Bra]. Recall that each element $M_{i,j}$ of the imaginary coherence

matrix contains the value of the imaginary coherence between ROI i and ROI j. Hence, for each frequency bin, for each timestamp, the symmetric imaginary coherence matrix contains $\frac{90\cdot(90-1)}{2} + 90 = 4095$ distinct values, corresponding the 4095 distinct pairs of regions of interest identified during the source modeling phase. Taking advantage of the symmetric nature of the matrix, it is only needed to store the values of the lower triangular connectivity matrix (including the diagonal). Hence, for each frequency bin, the corresponding connectivity matrix can be stored as a 4095-dimensional vector. In other words, each observation in the dataset consisted of a 2-dimensional matrix of size $[4095, 50]$ where each column corresponds to one specific frequency bin.

Two subjects (S01 and S03) were excluded from the analysis as it was reported (by E. Coffey) that these subjects had poor head positioning during the night which could potentially lead to poor results in the source modeling phase and hence to corrupted imaginary coherence measures. Hence, the final dataset used contains a total of 7,363 observations recorded on eight different subjects.

2.2 Labels

The data was labeled by hand (by Emily Coffey) in 5s epochs in 6 different sleep (sub)-stages based on the EEG recordings for each subject. As we were primarily interested in discriminating REM sleep stages from non-REM stages, these substages were then grouped into two classes: REM and non-REM sleep stages (as described in section 1.4). Table 1 summarizes the class labels distribution for each subject in our dataset. As we can see in this table, the class distribution between REM and non-REM observations varies a lot across the subjects. In particular, the classes are imbalanced since we have far less REM observations than non-REM observations.

Subject	Number of REM observations	Number of non-REM observations	Percentage of REM observations
S04	322	502	39%
S05	154	430	26%
S06	330	457	42%
S07	333	479	41%
S08	245	896	21%
S10	482	671	42%
S11	311	749	29%
S12	235	767	23.5%

Table 1: *Summary statistics of class distribution per subject*

2.3 Data preprocessing: standard frequency band aggregation

For each observation, we had a total of $50 \times 4095 = 204,750$ features. In order to reduce the high dimensionality of the data, a "standard frequency band" aggregation step was performed as a preprocessing step. Indeed, neuroscientists have defined five standard MEG/EEG frequency bands, described in [KB12] as follows:

- Delta band: $[0; 4[$ Hz.
- Theta band: $[4; 8[$ Hz.
- Alpha band: $[8; 13[$ Hz.

- Beta band: $[13; 30[$ Hz.
- Gamma band: over 30 Hz.

As mentioned in the previous subsection, the dimension of the original dataset containing the (imaginary) coherence values is $[7363, 4095, 50]$. With this preprocessing step, the original frequency bins were aggregated over these five standard frequency bins (taking the average coherence value over all frequencies in each standard bin). In other words, the resulting feature matrix is of dimension $[7363, 4095, 5]$. Hence, this step reduced the dimensionality of the data from total of 204,750 to 20,475 features per observation.

3 Adaptation of graph convolutional neural networks to multi-frequency brain connectivity data

As already mentioned in section 1, a coherence matrix can be seen as a weighted adjacency matrix of a "brain graph" whose nodes are the ROIs. Hence, it seems natural to try to apply graph classification techniques to the sleep stage classification task. Recall that the goal is to classify these graphs (or equivalently connectivity matrices) as "REM" or "non REM" observations. Previous papers have also aimed to use graph classification techniques for tasks based on brain connectivity data. For example, in [Ars+18], the authors fed a symmetric connectivity matrix (derived from fMRI data) into a graph convolutional network to classify the sex of the subject. This section will first consist of a review on node-level graph convolutional networks (GCN). Then, we will describe how these can be used for whole-graph classification tasks. Finally, the model designed for this project will be detailed.

3.1 Related work: review of graph convolutional networks

3.1.1 Initial definition of spectral graph convolutions

Motivated by the successful applications of convolutional networks (CNN) in the last years, [Bru+13] have introduced an adaptation of traditional CNN for graph-based data. In this paper, they have introduced two constructions of a so-called *graph convolutions*. The first one, what they call the "spatial" construction is based on using the local structure of the graph to get localized receptive fields for the convolutions. In the second construction, they introduced "spectral" graph convolutions, based on the spectrum of the graph Laplacian. Precisely, the spectral graph convolution is formally defined as (definition from [KW16]):

$$g_\theta \star x = U g_\theta U^T x$$

where g_θ denotes the filter (a diagonal matrix parametrized by $\theta \in \mathbb{R}^n$), U is the matrix of eigenvectors of the normalized graph Laplacian $L = I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Delta U^T$, x is the signal (i.e. one value per node) and $U^T x$ its graph Fourier transform.

3.1.2 A simpler expression to approximate graph convolution

In [HVG09] it has been shown that a spectral convolution can be approximated using its truncated Chebyschev polynomial expansion. In [KW16], they introduced a simpler expression for graph convolution that they show to be an appropriate approximation for spectral graph convolutions. Our graph-convolution model will make use of this graph convolution layer expression. This layer is defined by the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)})$$

where $\tilde{A} = A + I_n$ with A the (possibly weighted) adjacency matrix of the graph, $\tilde{D}$ is the diagonal degree matrix of $\tilde{A}$ (i.e. defined as $\tilde{D}_{i,i} = \sum_{j=1}^{n} \tilde{A}_{i,j}$), σ is the activation function, $H^{(l)}$ the value of the preceding layer and $W^{(l)}$ the matrix of trainable weights of layer l. Naturally, graph convolutional layers are at core of every graph convolutional network (GCN).

3.2 From node classification to whole-graph classification

In the same paper [KW16], they used a GCN with 2 graph convolutional layers in a semi-supervised node-classification task. In the node classification task presented in this paper, the input dataset is composed of one single graph instance. The network aims to label all nodes of this input graph. The input if their network consists of the graph adjacency matrix A and a feature matrix X corresponding to $H^{(0)}$ in the previous convolution layer formula. The node labeling task is not the same task as whole-graph classification. In whole-graph classification, the input consists of a set of different graphs that we want to label i.e. we do not label the graph nodes but the graph itself. Nevertheless, the node classification graph convolution network described in [KW16], allows to derive multi-dimensional node features (or embeddings) for each node in the input graph. Indeed, the architecture of their model consists of one convolutional layer with ReLu activation followed by a second convolutional layer with Softmax activation. The final output of their network corresponded to the node labels. This yields the following formula for the forward pass of their network [KW16]:

$$Y = \text{softmax}\left(\hat{A}\text{ReLu}\left(\hat{A}XW^{(0)}\right)W^{(1)}\right) := \text{softmax}\left(Z\right)$$

using the same notations as above and defining $Z \in \mathbb{R}^{n \times h_2}$ as the node features matrix corresponding to the output of the network just before applying the softmax activation function (where n is the batch size and h_2 the dimension of the second hidden layer). In this case, removing the softmax activation function, the network outputs one feature vector of size h_2 for every node in the graph.

In order to perfom whole-graph classification, an additional step is needed in order to derive a graph embedding from the nodes' embeddings that were derived thanks to the node classification network described above. One simple way to construct a graph embedding from several node embeddings is to simply perform sum (or average) pooling on the node embeddings i.e. simply use the sum of all node embeddings as whole-graph embedding. This is the approach that was followed in [Ars+18]. More precisely, their model can be described as follows: first they used a network composed of several graph convolutional layers to get the node embeddings. Note that they used the Chebychev's polynomials approximation to construct the graph convolutional layers. Then, once they have derived the node embeddings for each node in the graph, they applied global average pooling on the nodes features to derive a graph embedding. This graph embedding was finally fed into a 2-output units dense layer with softmax activation to derive the class probabilities.

The procedure transforming a coherence matrix into a graph embedding is depicted in fig. 2.

3.3 Proposed model

3.3.1 Model architecture

The model used for our sleep stage classification task is inspired from the approach followed in [Ars+18] but required some modifications to apply it with the MEG data. Indeed, in the

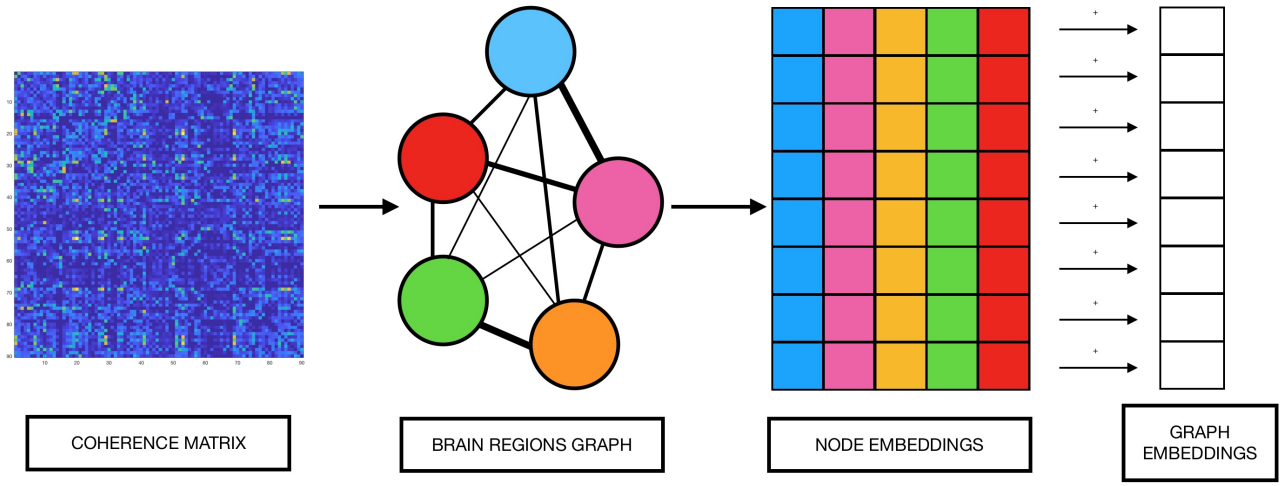


Figure 2: *From coherence matrix to graph embeddings. A coherence matrix can be viewed as the weighted adjacency matrix of a brain regions graph where each node is one ROI. The higher the coherence value between two ROIs the stronger the edge is between the corresponding nodes in the brain graph. A graph convolutional network is then used to derive one multi-dimensional node embedding for every node in this brain graph. In the second and third pictures of this figure each node is represented by another color. Finally, the whole-graph embedding is derived by summing all node feature embeddings, yielding a vector of the same dimension of a single node embedding.*

aforementioned paper, they use another connectivity measure (derived from fMRI images) which yields only one single connectivity matrix per observation (it is not a multi-frequency measure). Conversely, with multi-frequency imaginary coherence measures derived from MEG signals, there are several connectivity matrices per time-window (each corresponding to a different frequency band). In our case, there are originally 50 matrices (i.e. graphs) per observation. After the standard frequency bands aggregation step (described in section 2), this is reduced to 5 graphs per observation (i.e. per time window). However, the original formulation of the graph convolution network for whole-graph classification as presented in the [Ars+18] paper only takes one adjacency matrix as an input. Therefore, the model used in this paper was not directly applicable to our data. Consequently, we propose an adaptation of this model capable of using multi-frequency imaginary coherence matrices as input.

The model proposed for this project distinguishes itself from the cited model as follows:

- The graph classification model was modified in order to allow it to take simultaneously five adjacency matrices as input, transforming the original graph classification network into a variation of a "siamese" network. To achieve this goal, all five input adjacency matrices were, one at a time, fed into a node embedding network (denoted by "node GCN" in the rest of this report) to get one node embeddings matrix for each of the five input graphs. It is important to note that the same node GCN was used for all 5 graphs (i.e. all the weights in the node GCN are shared among all 5 graphs). This was done in order to make the training less prone to overfit (by reducing the number of parameters of the network). The architecture of the node GCN is described more precisely in the next paragraph. Subsequently to the node GCN step, graph embeddings were derived by applying sum pooling on the node embeddings, separately for each graph. The procedure per graph is schematized in figure 2. The five whole-graph embeddings were then concatenated to form one global embedding of the observation we want to classify. Finally, this global embedding was fed into a fully connected layer with two output units to get the class logits. The very

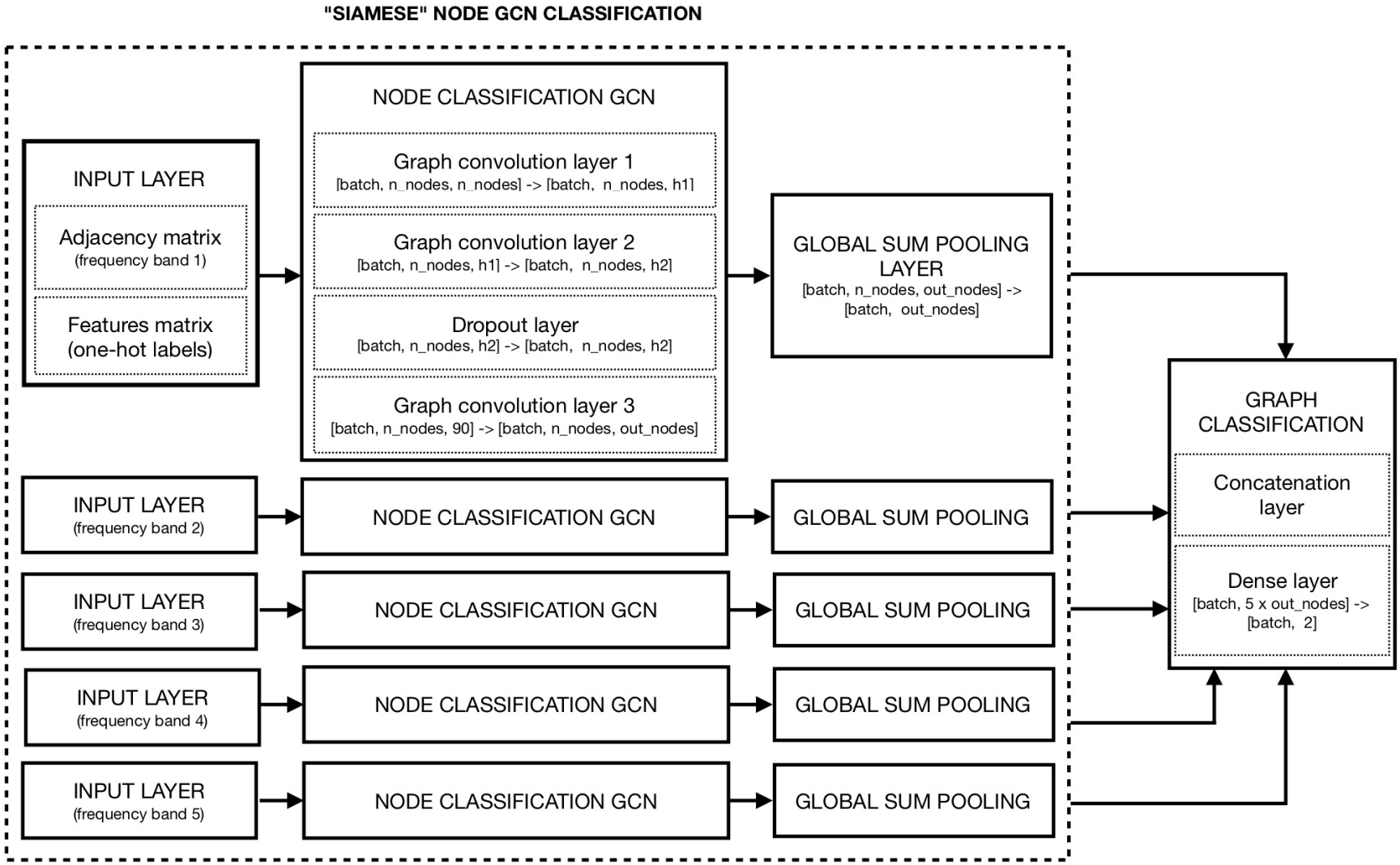


Figure 3: *Architecture of the model. The same node classification GCN is used for all five input matrices (i.e. weights sharing).*

Algorithm 1 Forward pass of the graph classification network

procedure FORWARD(Adj_1, Adj_2, Adj_3, Adj_4, Adj_5)
  for i in $1:5$ **do**
    $\tilde{A}_i \leftarrow \text{Adj}_i + I$
    $\tilde{D}_i \leftarrow \text{DEGREE}(\tilde{A}_i)$
    $\hat{A}_i = \tilde{D}_i^{-\frac{1}{2}} \tilde{A}_i \tilde{D}_i^{-\frac{1}{2}}$
  for i in $1:5$ **do**
    $\text{NodesFeats}_i \leftarrow \text{ReLu}\left(\hat{A}_i \left(\text{DropOut}\left(\text{ReLu}\left(\hat{A}_i \text{ReLu}\left(\hat{A}_i W^{(0)}\right) W^{(1)}\right)\right)\right) W^{(2)}\right)$
    $\text{GraphFeats}_i \leftarrow \sum_{j=1}^{n} \text{NodesFeats}_{i,j}$
  $\text{Global} \leftarrow \text{CONCAT}(\text{GraphFeats}_1, \text{GraphFeats}_2, \text{GraphFeats}_3, \text{GraphFeats}_4, \text{GraphFeats}_5)$
  $\text{Logits} \leftarrow \text{FC}(\text{Global}, 2)$ ▷ Fully Connected Layer with 2 output units.
  $\text{Probas} \leftarrow \text{SOFTMAX}(\text{Logits})$

last step consisted of applying the softmax activation function to derive the corresponding class probabilities. A schema of the architecture of the final model can be found in figure 3.

- The simplified formulation (also called renormalization trick) from [KW16] was applied to define the convolutional layers instead of the Chebyschev approximation used in [Ars+18]. In other words, a graph convolution layer is defined as $H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)})$ (see previous subsection).

The "node GCN" is the sub-network used to derive node embeddings for one input graph. The chosen architecture for this node GCN is composed of 3 different graph convolutional layers with ReLu activation. The output of the node GCN network is a node embedding matrix i.e. for each node in the graph one feature vector of size n_f is produced. In other words, for each input graph, the dimension of the output of the node GCN is $[n_{in}, n_f]$, where n_{in} is the number of nodes in the input graph. In addition to the adjacency matrix, an input node features matrix is required by the graph convolutional network (as explained in 3.2). Our dataset merely provides us with brain regions but no features associated to them. In [KW16] the advice given for this particular case (where no input feature is available) is to simply use the identity matrix as the input feature matrix. Indeed, each row of the identity matrix then corresponds to a one-hot encoding of each node in the graph which can be used as input node representation. To summarize this mathematically, the following formula describes the forward pass of the node GCN chosen for our classification model:

$$\text{NodesFeats} = \text{ReLu}\left(\hat{A}\text{ReLu}\left(\hat{A}\text{ReLu}\left(\hat{A} I_n W^{(0)}\right) W^{(1)}\right) W^{(2)}\right)$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, $\tilde{A} = A + I$, $\tilde{D}_{i,i} = \sum_j \tilde{A}_{i,j}$. With n the number of nodes in the input graph, h_1 the dimension of the first hidden layer, h_2 the dimension of the second hidden layer and n_f the dimension of the output features for each node $\hat{A} \in \mathbb{R}^{n \times n}$ and the trainable weights are $W^{(0)} \in \mathbb{R}^{n \times h_1}$, $W^{(1)} \in \mathbb{R}^{h_1 \times h_2}$ and $W^{(2)} \in \mathbb{R}^{h_2 \times n_f}$.

In the final classification network, an additional dropout layer (with dropout probability of $p = 0.5$) was added between the second and the third convolutional layer inside the node GCN for regularization of the network. Algorithm 1 summarizes the forward pass of the whole network.

3.3.2 Training procedure

The cost function used to train the network is the weighted cross-entropy loss. The loss is weighted in order to take class imbalance into account. More formally, the definition of the weighted cross-entropy loss for one observation is:

$$l(y, p) = -\left(y \cdot \text{weight}(1) \cdot \log(p) + (1 - y) \cdot \text{weight}(0) \cdot \log(1 - p)\right)$$

where p denotes the predicted probability for class 1, y $\in \{0, 1\}$ denotes the true class label and weight(i) denotes the weight attributed to class $i \in \{0, 1\}$. At each training step, the average loss over all observations in the batch is used to optimize the network's parameter with backpropagation. The weights of the loss function are calculated during initialization of the network as follows:

$$\text{weight}(i) = \frac{n}{n_i}$$

where n_i is the number of observation in class i and n the total number of observations in the training set. Hence, the total weight associated to class i is weight$(i) * n_i = n$ i.e. identical for each class. Therefore, weighting the loss solves the class imbalance problem during training of the network.

The model was implemented with PyTorch. The batch size used was 64. The network was trained using the Adam optimizer. Moreover, weight decay was applied with a regularization parameter of $5e^{-4}$. The loss and the balanced accuracy were computed and reported on the training and validation set every 5 steps. The validation and training loss were then plotted to evaluate the number of steps needed until convergence of the network and/or to detect overfitting patterns (see section 4.5).

4 Experiments

In order to assess the predictive power of the graph classification neural network described above, we performed several experiments to compare the results obtained in terms of balanced accuracy against those obtained by standard classifiers (presented below in section 4.1). Three different types of cross-validation schema (described in section 4.4) were used to compare the performance of the estimator.

4.1 Baselines

The performance of the model described above was compared to the performance of two standard baselines used in the brain connectivity analysis literature.

- In the first baseline, principal component analysis was applied on the data and 500 components were kept. Then, a linear Support Vector Machine classifier was used to classify the observations. This is a common classification pipeline used in neuroscience as for example in [DA12] where they applied this procedure to classify patients with autism and control patients (also using coherence matrices as input data).
- In the second baseline first a "univariate feature selection" procedure was applied to keep only 50% of the features. More precisely, this feature selection procedure consists of computing the p-value of one ANOVA F-test performed on each individual feature using the sleep stage labels as categorical predictor. The features with the smallest associated p-values are kept for classification. The resulting feature matrix was then fed in a random

forest classifier with 10,000 trees (the minimum number of samples per split was set to 30). Random forest is also a standard estimator in the field. It was for example used in [Wes+18] in an across-subjects classification setting using MEG data.

In the rest of this report, these two baselines will be referred to as "PCA SVM" and "RF". The hyperparameters (e.g. number of PCA components in the first baseline, percentage of feature to keep in the second) were chosen with standard cross validation on the full dataset. For the experiments, the baselines were implemented using estimators defined in the Python Sklearn package.

4.2 Class imbalance

As it was mentioned in section 2, there is less observations available for the REM class compared to the non-REM class. Hence, we are facing a class imbalance problem. In the neural approach presented in the last section, a weighted loss was used in order to take this issue into account. To make a fair comparison between baselines and the neural approach, this class imbalance problem also has to be approached for the baselines. To this end, there are two possibilities: either using class weighted version of the estimators or perform upsampling of the REM class on the training set. The main issue with the first approach is that we want to use the same estimator (hence the same class weights) for all cross-validation schemes and for all subjects but the ratio between both classes varies a lot from subject to subject (see 2). For this reason, the second approach i.e. upsampling the minority class in the training set was chosen.

Upsampling was performed by drawing with replacement n_0 observations from the set of REM observations (in the training set), where n_0 is the number of non-REM observations in the training set. Note that it is very important to perform the upsampling procedure *after* having splitted the data into a training and a test set because otherwise we could have exact replicates of observations from the training set in the test set leading to over-optimistic results in terms of performance. The initial train and test split is a "stratified" split by the labels i.e. the ratio of REM/non-REM observations is the same in the original dataset, in the training set and in the test set. Hence, the upsampling procedure had to be implemented within the cross validation procedure. The detailed pseudo-code of this procedure is presented in algorithm 2.

In the results section, we denote "RF UP" (resp. "PCA SVM UP") the random forest (resp. PCA SVM) baseline mentioned earlier but trained with upsampling.

4.3 Metrics used

The main metric chosen to compare our estimators was the balanced accuracy defined as :

$$\text{mean}\left(\frac{TP}{TP+FN}\ ;\ \frac{TN}{TN+FP}\right)$$

where TP true positive (resp. TN true negative) means that the predicted label is REM (resp. non-REM) and the true label is REM (resp. non-REM). Similarly, FP false positive (resp. FN false negative) means that the predicted label is REM (resp. non-REM) and the true label is non-REM (resp. REM). By these definitions, $TP + FN$ (resp. $TN + FP$) is the total of observations with true label REM (resp. non-REM). This metric is more representative of the performance than the standard accuracy because of the class imbalance in our dataset. Indeed, for most of the subjects, predicting always the non-REM label would already yield a standard

Algorithm 2 Stratified K-fold cross validation with upsampling

procedure K-FOLD-UPSAMPLING(data, y, estimator, up=True, K)
  Split data into K fold stratified by y.
  for k in $1 : K$ **do**
    $S_{test} \leftarrow$ data[fold==k]
    $S_{train} \leftarrow$ data[fold!=k]
    $S_{0,train} \leftarrow S_{train}[y == 0]$ ▷ 0 is non-REM
    $S_{1,train} \leftarrow S_{train}[y == 1]$ ▷ 1 is REM
    $n_0 \leftarrow \text{LENGTH}(S_{0,train})$
    if up **then**
      $S^{new}_{1,train} \leftarrow$ Draw n_0 observations with replacement in $S_{1,train}$.
      $S^{new}_{train} \leftarrow S_{0,train} \cup S^{new}_{1,train}$
      Fit estimator on S^{new}_{train}
    else
      Fit estimator on S_{train}
    Save performance of predicting labels in S_{test}. ▷ Test set is untouched

accuracy over 70% as there are less than 30% REM observations. Hence, in this case, the balanced accuracy is a more appropriate metric to evaluate the performance of the estimators.

For within-all-subjects and across-subjects cross-validation confusion matrices were also computed. A confusion matrix is defined as:

		Predicted label: 0	Predicted label: 1
True label	0	TN $\left(\frac{TN}{TN+FP}\%\right)$	FP $\left(\frac{FP}{TN+FP}\%\right)$
	1	FN $\left(\frac{FN}{TP+FN}\%\right)$	TP $\left(\frac{TP}{TP+FN}\%\right)$

4.4 Different types of cross-validation

When predicting sleep stage labels three types of cross-validation can be distinguished: "within-one-single subject" CV, "within-all-subjects" CV and "across-subjects" CV.

In *within-one-single-subject* cross-validation, both the training folds and the test fold contain observations from only one selected subject. In other word, this is similar to standard cross-validation procedure (stratified by the labels) but using only the observations from one chosen subject as input dataset. Hence, a separate within-one-single-subject cross-validation can be run on each subject. The number of fold was set to four in each within-one-single-subject CV procedure.

In *within-all-subjects* cross-validation, all observations are used and the data set is split into k (stratified) folds (here k=4) regardless of the subject's number. Hence, this corresponds to a standard k-fold cross-validation (stratified by labels) on the whole dataset. In this setting, we have observations from every subject as well in the training set as in the test set.

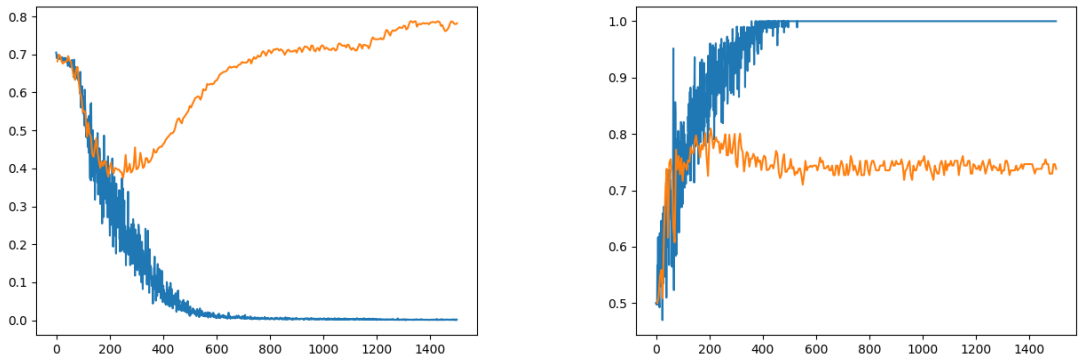
Conversely, the *across-subjects* CV assesses the predictive power of the method on an unseen subject. This predictive power is important for real applications as it is more likely to have to predict labels for new subjects rather than for the same subjects that were used for training. In across-subject cross-validation, each subject corresponds to one fold of the CV procedure. Hence, the splits are not random and if there is n subjects it means the procedure is a n-fold CV procedure. In this procedure, the observations from $(n-1)$-subjects are used as the training fold and the observations from the hold-out subject are used as the test fold. Note that achieving good performance on across-subject prediction is harder than on within-subject(s) prediction, as there is no dependency (induced by the subject) between the training set and the test set.

The experiments of this project consisted of evaluating the performance of all estimators (described above) in terms of balanced accuracy for each cross-validation schema. In other words, we performed:

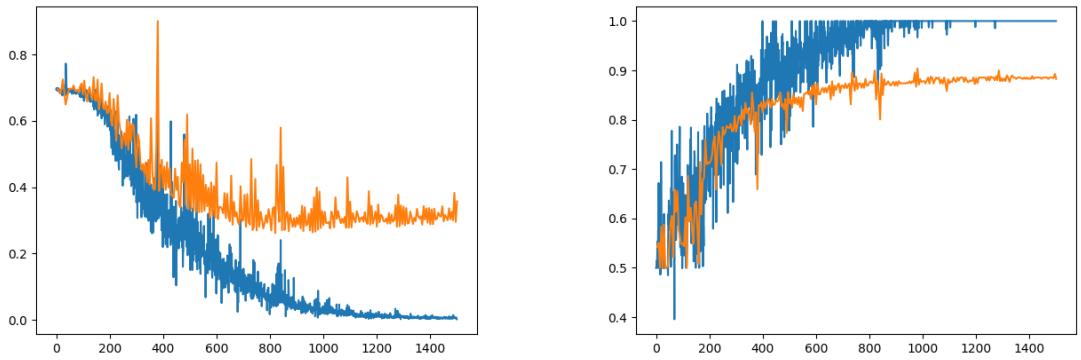
- Eight different 4-fold within-one-single-subject cross-validations (the CV procedure was run separately for every subject)
- One 4-fold within-all-subjects cross-validation
- One across-subjects cross-validation.

4.5 Final neural model parameters

The best parameters (chosen via standard cross-validation) for the dimension of the hidden layers used in the node GCN were: 128 output units in the first hidden layer, 32 in the second and 16 for the dimension of the node features. Observing the training and validation loss plots, it appeared that the model needed 1500 training steps in order to converge in the within-all-subjects and across-subjects cross-validation settings (fewer steps decreased the mean balanced accuracy substantially). In these two settings, training for more steps did not improve the validation balanced accuracy and the model began overfitting if more steps were used. However, in the within-one-single subject setting, depending on the subject's ratio of REM:non-REM observations, the model was often clearly overfitting with 1500 steps as this can be see in the upper plots of figure 4. This difference can mainly be explained by the fact that we are using less than 20% of the observations in each single-subject cross-validation as we restrict the dataset to one chosen subject. With so few observations the network is naturally prone to overfit the training set. However, first it did mainly happen for subjects where the number of REM observation was very low (e.g. S12 depicted in the first plot of the figure). Secondly, even though the overfitting pattern was clearly observable on the loss plot, it often did not decrease the balanced accuracy of the validation set much. Hence, even if the model was trained for less training steps (i.e. stopping before the overfitting pattern appears) the average results in terms of balanced accuracy over all cross-validations did not improve greatly. Indeed, gain in balanced accuracy due to less overfitting for some subjects was annihilated by a significant decrease in the balanced accuracy for subjects where the ratio REM:non-REM was higher (i.e. subject where more training steps were needed until convergence). Therefore, it was decided to keep the same number of training steps (i.e. 1500 steps) in every cross-validation setting. The choice to take the same number of training steps in each cross-validation setting was especially made in order to allow for a fair comparison between all estimators. Indeed, as we do not change the baselines' hyperparameters between the different experiments it was not suitable to change the parameters of the neural model between the different experiments. To choose the optimal number of training steps, we focused on the within-all-subjects (i.e. standard cross-validation)



Training and validation loss (left) and balanced accuracy (right) for subject 12 (fold 1) by number of steps. The overfitting pattern is clearly recognizable on the loss plot but less on the balanced accuracy plot. The percentage of REM observations for this subject is 23.5%



Training and validation loss (left) and balanced accuracy (right) for subject S6 (fold 2) by number of steps. For this subject, the percentage of REM observation is 42%. We can't clearly see the U-shape of the validation loss characterizing overfitting as above. It also shows that if training would have been stopped at 250 steps (corresponding the minimum point of the loss function for the overfitting subject above) the balanced accuracy for this subject would have been nearly 10% less than the one achieved with more than 1000 steps.

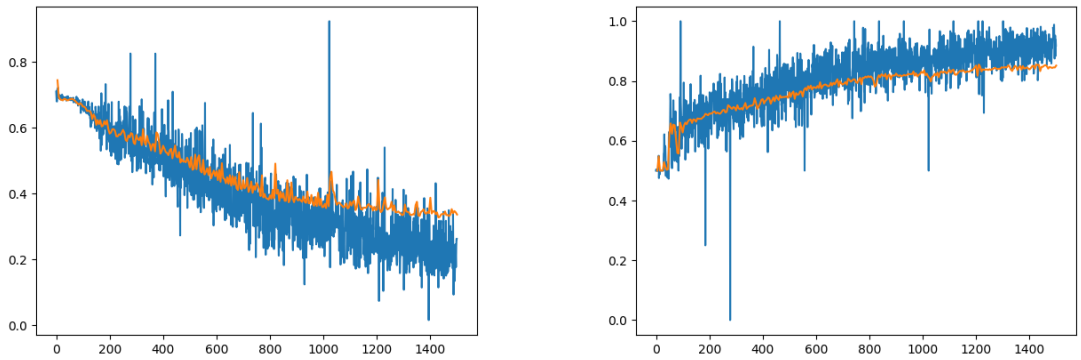
Figure 4: Examples of plots of loss and balanced accuracy evolution during training in the within-one-single subject cross-validation setting.

and across-subject performance (as within-one-single-subject predictions are unlikely to be used in real applications).

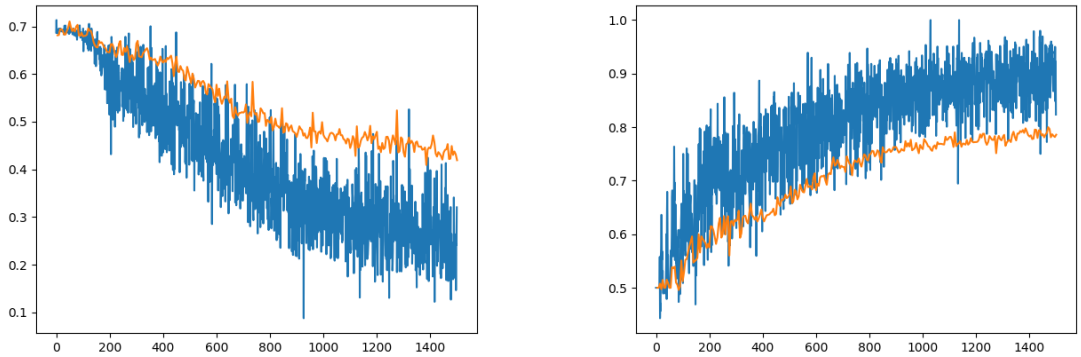
4.6 Results

4.6.1 Within-one-single-subject cross-validation

In this setting, we performed 8 separate 4-fold cross-validation procedures, one per subject. For each cross-validation, i.e. for each subject, the average balanced accuracy over the 4-folds is depicted on the left plot of figure 6. On the right plot of the same figure, the average balanced accuracy over all cross-validations is plotted for each estimator. The same results are detailed in figure 7. Note that the standard deviation computed in this table is the standard deviation among all mean balanced accuracy obtained for each cross-validation (and not the standard deviation across fold as in the usual cross validation setting). Detailed descriptive statistics of



Training and validation loss (left) and balanced accuracy (right) in fold 5 of the within-all-subjects cross validation.



Training and validation loss (left) and balanced accuracy (right) in fold S04 of across-subjects cross validation.

Figure 5: Examples of plots of loss and balanced accuracy evolution during training in the within-all-subjects and across-subjects cross-validation settings.

the balanced accuracy for each cross-validation can be found in table 2 of the appendix. The point estimate of the average performance over all within-one-single-subject cross-validation is slightly higher for the neural approach compared to the other baselines. However, given the wide standard deviation of the estimates, we can not say that the neural approach is significantly better than the SVM baseline. The mean balanced accuracy (over all folds and all subjects) achieved by the proposed neural model is 80.5% (with standard deviation of 5.8% between the subjects). It is worth noting that the random forest baseline performs systematically worse than SVM or GCN in this cross validation setting. Moveover, the graph on the left of figure 6 shows that the performance of the neural approach is much more stable than the performance of the baselines across subjects. Indeed, the mean balanced accuracy for each cross-validation for the neural model lies between 0.681 and 0.826 whereas it varies between 0.496 and 0.853 for the PCA SVM with upsampling baseline (see table 2 in the appendix). The baselines' performance appears to be more sensitive to the initial REM:non-REM ratio in the training set than the GCN model.

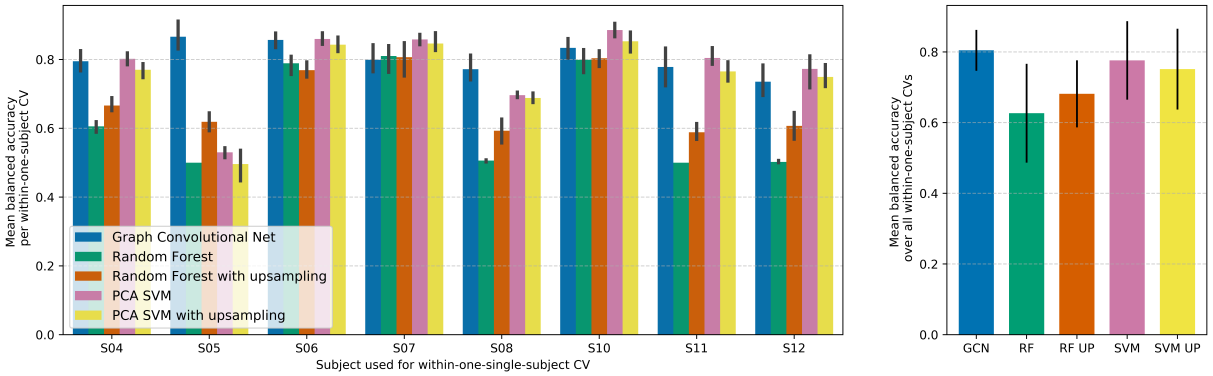


Figure 6: *Within-one-single subject cross-validation performance. Left plot: mean balanced accuracy over the 4 folds in each cross-validation procedure (one per subject), per estimator. Right plot: overall mean of balanced accuracy over all cross-validations, per estimator.*

Estimator	Mean balanced accuracy over all CVs	Standard deviation of mean balanced accuracy between CVs
Graph Convolutional Net	**0.805**	0.058
Random Forest	0.627	0.140
Random Forest with upsampling	0.682	0.095
PCA SVM	0.776	0.111
PCA SVM with upsampling	**0.751**	0.114

Figure 7: *Overall performance among all within-one-single subject cross-validations.*

4.6.2 Within-all-subjects cross-validation

Recall that within-all-subject cross-validation denotes here the standard 4-fold cross validation setting i.e. we split the dataset randomly regardless of the subject number. Figure 8 shows the results obtained in this setting. The plot on the left shows the balanced accuracy obtained for each fold in the cross-validation procedure. The plot right shows the global results of the cross-validation procedure, per estimator. We see here that the standard deviation across folds is much smaller than the standard deviation observed across the different within-one-single cross-validations. This could be explained by the fact that, in this setting, all folds have the same REM:non-REM ratio. In this cross-validation setting, the graph convolutional network performs better than the baselines. Indeed, the point estimate of the mean balanced accuracy lies higher than the upper bound of the (normal) approximate confidence intervals of the other's estimator mean balanced accuracy. We can also see on the mean confusion matrices that the performance of SVM and GCN are better in terms of balance of the performance between both classes than the performance of the random forest (see figure 10). The mean balanced accuracy achieved by the neural model is 84.3% (with standard deviation of 0.9%, see figure 9). Moreover, it systematically yields the best balanced accuracy for each of the individual folds of the cross-validation. Detailed descriptive statistics of the balanced accuracy in this cross-validation setting can be found in table 3 of the appendix.

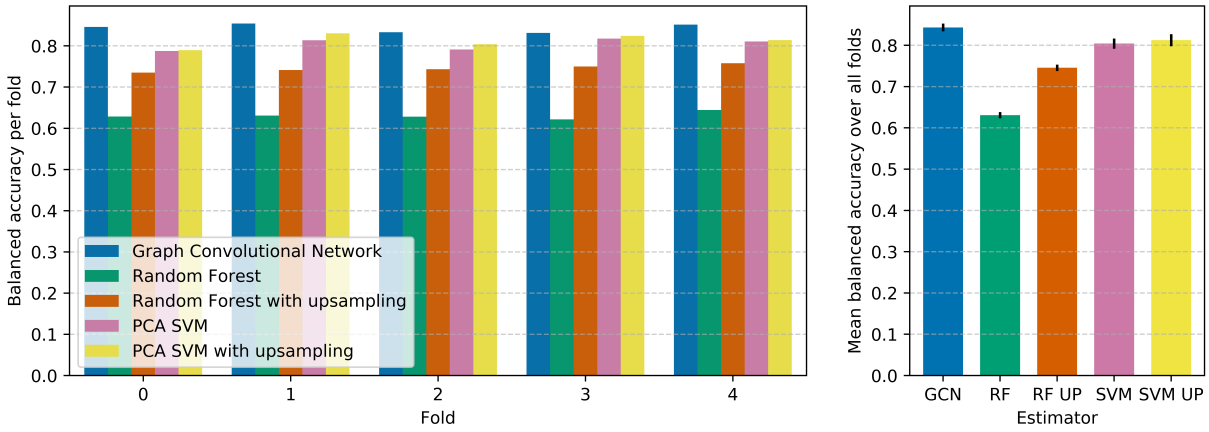


Figure 8: *Plots of within-all-subjects cross-validation performance. Left plot: balanced accuracy per fold and per estimator. Right plot: mean balanced accuracy over all folds per estimator.*

Estimator	Mean balanced accuracy over all folds	Standard deviation of balanced accuracy between folds
Graph Convolutional Net	**0.843**	0.009
Random Forest	0.631	0.007
Random Forest with upsampling	0.746	0.008
PCA SVM	0.804	0.012
PCA SVM with upsampling	**0.812**	0.015

Figure 9: *Mean balanced accuracy for the within-all-subjects cross-validation per estimator*

	0	1
0	859.6 (87%)	130.6 (13%)
1	87.6 (18%)	394.8 (82%)

(a) GCN

	0	1
0	951.4 (96%)	38.8 (4%)
1	226.6 (47%)	255.8 (53%)

(b) RF with UP

	0	1
0	838.8 (85%)	151.4 (15%)
1	107.2 (22%)	375.2 (78%)

(c) PCA SVM with UP

Figure 10: *Average confusion matrices (over all folds) for within-all-subjects cross-validation for Graph Convolutional Network, Random Forest with upsampling and PCA SVM with upsampling estimators.*

4.6.3 Across-subject cross-validation

Across-subject prediction is the hardest task among the three prediction settings evaluated in this project. Indeed, here, contrarily to what was done previously, we predict labels on a subject that has not been seen during the training phase. Figure 11 and figure 12 show the results of across-subjects cross-validation. Again, the neural model was the one yielding the highest mean balanced accuracy among the folds. However, as in the first cross-validation, the result obtained is not significantly better than the mean balanced accuracy achieved with the PCA SVM baseline. Nevertheless, it is noteworthy to point out that the neural network was the best estimator for 6 folds out of 8 folds in the cross validation (as this can be seen in fig.11). Also, in the confusion matrices we see that the precision in the REM class is better with the

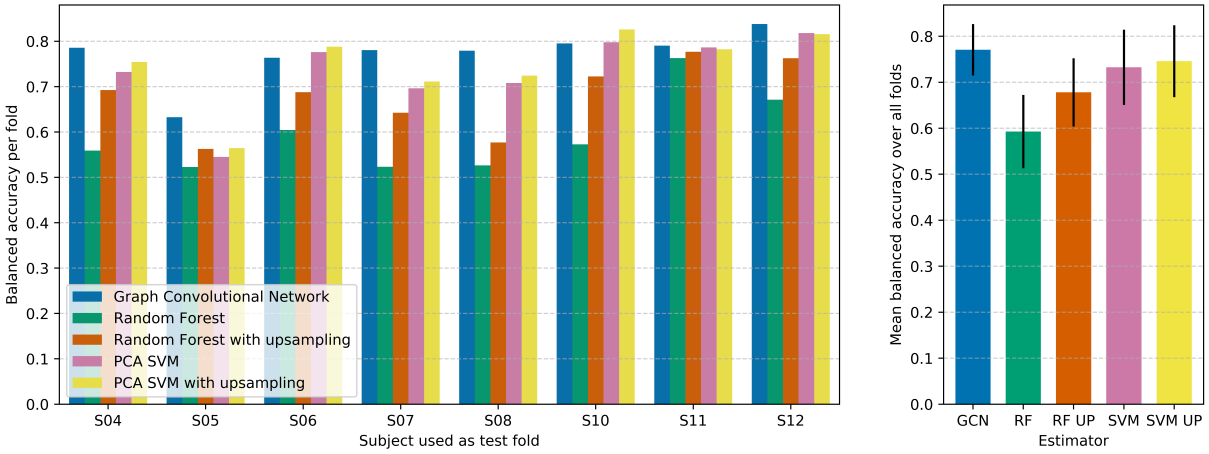


Figure 11: *Plots of across-subjects cross-validation performance. Left plot: balanced accuracy per fold (each subject corresponds to one fold in this setting). Right plot: mean balanced accuracy over all folds.*

Estimator	Mean balanced accuracy over all folds	Standard deviation of balanced accuracy between folds
Graph Convolutional Network	**0.771**	0.056
Random Forest	0.593	0.080
Random Forest with upsampling	0.678	0.074
PCA SVM	0.733	0.082
PCA SVM with upsampling	**0.746**	0.078

Figure 12: *Mean balanced accuracy for the across-subjects cross-validation per estimator*

	0	1
0	501.62 (82%)	117.25 (18%)
1	77.25 (27%)	224.25 (73%)

(a) GCN

	0	1
0	588.62 (96%)	30.25 (4%)
1	174.87 (60%)	126.62 (40%)

(b) RF with UP

	0	1
0	516.5 (85%)	102.37 (15%)
1	97.37 (35%)	204.12 (65%)

(c) PCA SVM with UP

Figure 13: *Average confusion matrices (over all folds) for across-subjects cross-validation for Graph Convolutional Network, Random Forest with upsampling and PCA SVM with upsampling estimators.*

neural model than with the SVM baseline (at a price of a slight decrease of the precision in the non-REM class) and far better than the results of the random forest for the REM class (see figure 13). The overall mean balanced accuracy achieved with our graph convolutional network in the across-subject setting is 77.1% (with standard deviation of 5.6%). Detailed descriptive statistics of the balanced accuracy in this cross-validation setting can be found in table 4 of the appendix.

5 Discussion and possible extensions

With a balanced accuracy around 80% in each cross-validation setting, the results presented in the previous section firstly show that imaginary coherence matrices derived from MEG signals have a predictive power for discriminating between REM and non-REM sleep stages.

The proposed neural model yielded the highest mean balanced accuracy over all folds in the within-all-subjects and in the across-subjects cross-validation settings (even though the difference with the best baseline was not significant in the across-subjects setting). For the within-one-single subject cross-validations, the best estimator (in terms of mean balanced accuracy over all folds) depended on the subject. However, taking the mean balanced accuracy over all folds and all within-one-subject CVs, it was again the neural model that achieved the best mean balanced accuracy (but the difference with the PCA SVM estimator was still not significant). These results are promising in the sense that further tuning of the network's architecture could yield a network that would significantly surpass the baselines. Further work might for example include testing deeper architectures of the node GCN or more fine-tuning of the regularization parameters (especially in the context where overfitting was observed). Another point that might improve the performance of the network would be to work on the technique used to transform the node embeddings into a single graph embedding (instead of the simple sum pooling technique used in the current model). This model could also be adapted in order to take even more adjacency matrices as input and thus allow for an even more fine-grained frequency band aggregation.

The results obtained with this example of graph neural model especially show that graph-based machine learning methods can be useful in the context of brain connectivity analysis. Therefore, another possible extension of this work might include the adaptation of graph clustering methods (e.g. one of those presented in [Sch07]) to brain connectivity measures in order to identify clusters of brain regions typically interacting with each other.

Finally, as it was already mentioned in section 1.1.2, single-step methods have recently been developed to analyze MEG/EEG time-series avoiding a separate source-modeling phase and without separately computing connectivity measures prior to brain connectivity analysis (see [Ulr+14], [Yan+16]). Provided that we could have access to the original MEG time-series, it would also be interesting to compare the performance of these methods (taking the raw time-series as an input) to the performance achieved by the graph neural network that uses imaginary coherence matrices as input.

6 Summary

In this report, we have proposed a neural model to classify sleep stages using multi-frequency brain connectivity measures derived from MEG data. This model is an adaptation of whole-graph classification networks with graph convolutional layers. Its particularity arises from its ability to take simultaneously five graphs as an input to classify one single observation (where each graph corresponds to one frequency band of connectivity measures). This model was evaluated on a binary classification task whose goal was to distinguish observations where the subject was in REM sleep stage from observations where the subject was in a non-REM sleep stage. We compared the performance of this model with two baselines commonly used in neuroscience applications: classifying with a linear SVM after having applied PCA to the original feature matrix and classifying with a random forest after having performed univariate feature selection on the original data. Three cross-validation experiments were conducted: standard cross-validation,

within-one-single subject cross validation and across-subject cross validation. Even though the differences in terms of balanced accuracy where not significant between the neural model and the baselines, the neural model systematically lead to the highest mean balanced accuracy in all three different cross-validation settings. It was also noted that the balanced accuracy was more regular across folds for the graph neural model than with the baselines. The results emphasize the fact that graph-based machine learning techniques can be beneficial for brain connectivity analysis when dealing with connectivity measures matrices. Hence, this suggests that other graph-based techniques (e.g. graph-clustering methods) could be used for further brain dynamics analysis with connectivity matrices from MEG data.

References

[Ars+18] S Arslan et al. "Graph Saliency Maps through Spectral Convolutional Networks: Application to Sex Classification with Brain Connectivity." In: 2018.

[Bra] "Brainstorm". URL: https://neuroimage.usc.edu/brainstorm/.

[Bru+13] Joan Bruna et al. "Spectral Networks and Locally Connected Networks on Graphs". In: *CoRR* abs/1312.6203 (2013). URL: http://arxiv.org/abs/1312.6203.

[BS16] André M. Bastos and Jan-Mathijs Schoffelen. "A Tutorial Review of Functional Connectivity Analysis Methods and Their Interpretational Pitfalls". In: *Frontiers in Systems Neuroscience* 9 (2016), p. 175. URL: https://www.frontiersin.org/article/10.3389/fnsys.2015.00175.

[DA12] Frank H. Duffy and Heidelise Als. "A stable pattern of EEG spectral coherence distinguishes children with autism from neuro-typical controls - a large case control study". In: *BMC Medicine* 10.1 (June 2012), p. 64. URL: https://doi.org/10.1186/1741-7015-10-64.

[Dav+02] O. David et al. "Estimation of neural dynamics from MEG/EEG cortical current density maps: application to the reconstruction of large-scale cortical synchrony". In: *IEEE Transactions on Biomedical Engineering* 49.9 (Sept. 2002), pp. 975–987. DOI: 10.1109/TBME.2002.802013.

[HVG09] D. K Hammond, P. Vandergheynst, and R. Gribonval. "Wavelets on Graphs via Spectral Graph Theory". In: *ArXiv e-prints* (Dec. 2009). arXiv: 0912.3848.

[KA14] Raman K. Malhotra and Alon Avidan. *Sleep Stages and Scoring Technique*. Dec. 2014, pp. 77–99. ISBN: 9781455712670.

[KB12] J. Satheesh Kumar and P. Bhuvaneswari. "Analysis of Electroencephalography (EEG) Signals and Its Categorization A Study". In: *Procedia Engineering* 38 (2012), pp. 2525–2536. URL: http://www.sciencedirect.com/science/article/pii/S1877705812022114.

[KW16] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *CoRR* abs/1609.02907 (2016).

[Nol+04] Guido Nolte et al. "Identifying true brain interaction from EEG data using the imaginary part of coherency". In: *Clinical Neurophysiology* 115.10 (2004), pp. 2292–2307.

[Sch07] Satu Elisa Schaeffer. "Survey: Graph Clustering". In: *Comput. Sci. Rev.* 1.1 (Aug. 2007), pp. 27–64. ISSN: 1574-0137. URL: http://dx.doi.org/10.1016/j.cosrev.2007.05.001.

[SG09] Jan-Mathijs Schoffelen and Joachim Gross. "Source connectivity analysis with MEG and EEG". In: *Human Brain Mapping* 30.6 (2009), pp. 1857–1865. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/hbm.20745.

[Ulr+14] Kyle R Ulrich et al. "Analysis of Brain States from Multi-Region LFP Time-Series". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2483–2491. URL: http://papers.nips.cc/paper/5624-analysis-of-brain-states-from-multi-region-lfp-time-series.pdf.

[Wes+18] Britta U. Westner et al. "Across-subjects classification of stimulus modality from human MEG high frequency activity". In: *PLOS Computational Biology* 14.3 (Mar. 2018), pp. 1–14. URL: https://doi.org/10.1371/journal.pcbi.1005938.

[Yan+16] Ying Yang et al. "A state-space model of cross-region dynamic connectivity in MEG/EEG". In: *NIPS*. 2016.

Appendix

		Fold balanced accuracy						
		mean	std	min	25%	50%	75%	max
Estimator	Subject							
Graph Convolutional Net	S04	0.795	0.033	0.757	0.781	0.792	0.806	0.838
	S05	0.866	0.050	0.826	0.833	0.851	0.884	0.937
	S06	0.857	0.026	0.823	0.844	0.861	0.874	0.883
	S07	0.799	0.044	0.756	0.781	0.790	0.808	0.861
	S08	0.772	0.044	0.731	0.746	0.761	0.787	0.834
	S10	0.834	0.035	0.790	0.816	0.836	0.854	0.873
	S11	0.778	0.069	0.692	0.739	0.789	0.828	0.843
	S12	0.735	0.053	0.681	0.707	0.727	0.755	0.806
PCA SVM	S04	0.802	0.020	0.784	0.785	0.799	0.816	0.825
	S05	0.530	0.018	0.506	0.522	0.534	0.542	0.546
	S06	0.860	0.021	0.843	0.845	0.854	0.869	0.888
	S07	0.858	0.019	0.835	0.848	0.860	0.871	0.877
	S08	0.696	0.009	0.687	0.692	0.695	0.699	0.709
	S10	0.886	0.024	0.864	0.867	0.882	0.901	0.913
	S11	0.805	0.030	0.784	0.790	0.793	0.807	0.848
	S12	0.773	0.056	0.691	0.759	0.789	0.803	0.820
PCA SVM with upsampling	S04	0.770	0.025	0.734	0.762	0.779	0.787	0.789
	S05	0.496	0.055	0.421	0.473	0.507	0.530	0.547
	S06	0.843	0.022	0.822	0.826	0.841	0.859	0.868
	S07	0.846	0.030	0.821	0.831	0.837	0.853	0.890
	S08	0.688	0.016	0.670	0.680	0.687	0.695	0.709
	S10	0.853	0.036	0.806	0.837	0.859	0.875	0.889
	S11	0.765	0.034	0.727	0.742	0.765	0.788	0.803
	S12	0.749	0.037	0.714	0.732	0.741	0.758	0.801
Random Forest	S04	0.606	0.018	0.581	0.600	0.609	0.614	0.623
	S05	0.500	0.000	0.500	0.500	0.500	0.500	0.500
	S06	0.789	0.033	0.740	0.784	0.803	0.808	0.811
	S07	0.810	0.049	0.737	0.801	0.831	0.839	0.842
	S08	0.506	0.004	0.500	0.506	0.508	0.508	0.508
	S10	0.799	0.039	0.745	0.787	0.807	0.819	0.838
	S11	0.500	0.000	0.500	0.500	0.500	0.500	0.500
	S12	0.502	0.004	0.500	0.500	0.500	0.502	0.508
Random Forest with upsampling	S04	0.666	0.023	0.648	0.654	0.658	0.670	0.700
	S05	0.619	0.032	0.580	0.599	0.625	0.645	0.645
	S06	0.769	0.029	0.732	0.759	0.771	0.781	0.801
	S07	0.807	0.053	0.739	0.778	0.819	0.848	0.850
	S08	0.593	0.038	0.547	0.578	0.592	0.607	0.639
	S10	0.804	0.026	0.770	0.794	0.806	0.815	0.833
	S11	0.588	0.029	0.565	0.569	0.580	0.599	0.628
	S12	0.607	0.045	0.568	0.569	0.605	0.644	0.650

Table 2: Descriptive statistics of the balanced accuracy for each within-one-single subject cross-validation procedure (one per subject).

Estimator	Fold	Balanced accuracy
Graph Convolutional Network	0	0.846
	1	0.854
	2	0.833
	3	0.832
	4	0.852
PCA SVM	0	0.788
	1	0.814
	2	0.791
	3	0.818
	4	0.811
PCA SVM with upsampling	0	0.789
	1	0.831
	2	0.804
	3	0.824
	4	0.814
Random Forest	0	0.628
	1	0.631
	2	0.628
	3	0.622
	4	0.644
Random Forest with upsampling	0	0.735
	1	0.742
	2	0.743
	3	0.750
	4	0.758

Table 3: Balanced accuracy per fold in within-all-subjects cross-validation

Estimator	Fold (test subject)	Balanced accuracy per fold
Graph Convolutional Network	S04	0.786
	S05	0.633
	S06	0.764
	S07	0.781
	S08	0.779
	S10	0.795
	S11	0.790
	S12	0.838
PCA SVM	S04	0.732
	S05	0.545
	S06	0.776
	S07	0.696
	S08	0.708
	S10	0.798
	S11	0.787
	S12	0.818
PCA SVM with upsampling	S04	0.754
	S05	0.565
	S06	0.788
	S07	0.711
	S08	0.724
	S10	0.826
	S11	0.782
	S12	0.816
Random Forest	S04	0.559
	S05	0.523
	S06	0.605
	S07	0.523
	S08	0.527
	S10	0.573
	S11	0.763
	S12	0.671
Random Forest with upsampling	S04	0.693
	S05	0.563
	S06	0.688
	S07	0.643
	S08	0.577
	S10	0.723
	S11	0.777
	S12	0.763

Table 4: Balanced accuracy per fold in across-subjects cross-validation