SAPIENZA
UNIVERSITÀ DI ROMA

Graph neural networks for classification and task-conditioned brain connectivity estimation of electrophysiological data

Department of Computer, Control and Management Engineering

Master of science in Artificial Intelligence and Robotics

Candidate

Lorenzo Mattia

ID number 1793272

Thesis Advisor
Prof. Laura Astolfi

Co-Advisor
Prof. Nicola Toschi

Academic year 2021-2022

Summary

Introduction

Overview and State of the Art

The key idea of the work presented here is the application of artificial intelligence methods to neuroscience. Indeed, even if the former is an ever-expanding field, that covers an extremely wide range of application contexts and the latter is a fundamental scientific branch for understanding complex brain processes, research studies aimed at combining these two fields are still at a preliminary stage.

In neuroscience, a key role is played by the study and analysis of electroencephalographic (EEG) data, recorded using electrodes placed on subjects' scalps. The main advantages of this technique are the noninvasiveness and cost-effectiveness of the collection procedure, combined with the appreciable quality of the collected signals.

Therefore, it is precisely on them that in this thesis work it has been chosen to develop and test some artificial intelligence methods.

As mentioned, the quantity of research articles that can be found in the literature about this topic is relatively modest and mainly focused on disorder classification, such as epileptic seizure [1], sleep [2] and attention [3] disorders, or emotion recognition tasks.

According to this 2021 review [4], the research effort is focused on the development of EEG-based convolutional, recurrent and hybrid CNN+LSTM neural networks. Actually, a small number of works have investigated the application of graph neural networks (GNNs) to EEG data [5] to exploit their strong topological component.

Indeed, as will be further explained later on, EEG signals recorded by the scalp electrodes are not to be considered independent, since they influence each other according to the brain functional connectivity conditioned to the ongoing situation. Therefore, these data are perfectly suitable for representation in graph form.

In many study cases it is useful and interesting to estimate the functional connectivity graph of the brain to be used both to derive some indices quantifying graph's properties, or to represent the input of machine learning or deep learning algorithms aimed, for instance, at classification.

In traditional neuroscience, there are a variety of state-of-the-art methods to compute brain connectivity both as correlations (e.g., ordinary coherence [6]) or causality relationships between signals (e.g., Granger causality [7] or PDC [8]).

However, the possibility of deriving such graphs using deep learning techniques has been recently explored to exploit their great ability in extracting expressive features from EEG signals and to try to overcome some limitations of traditional methods used in neuroscience. In fact, these methods often rely on statistical assumptions about signals, such as stationarity and linearity, that frequently do not hold for EEG data.

In [9], the use of artificial neural networks (ANNs) as classic autoregressive models (MVARs) that are used in traditional neuroscience methods for brain connectivity estimation was studied and tested. Indeed, these networks have been proven to be extremely effective in time series analysis thanks to their flexibility in modeling nonlinearities between variables and because no assumptions have to be made a priori about signal stationarity and connectivity patterns [10]. In the latter work, it is also concluded that LSTM-based neural networks with nonuniform embeddings are the most promising method for brain connectivity analysis and are even able to compete with state-of-the-art approaches.

Something similar has been developed in [11] using graph neural networks. After having trained a GNN model on a classification task, the authors used the GNNExplainer framework [12] to determine some features and properties of the analyzed graphs that have been crucial in the GNN training. For instance, it has been used to derive node and edge importance, which can be interpreted as a measure of brain connectivity.

Aim of the thesis

In this thesis work, the application of graph neural networks to electroencephalographic signals was investigated. This is mainly due to two factors: first, because of their aptitude for capturing and exploiting the fundamental topological and structural characteristics of this type of data, which, as mentioned, are perfectly suited for being represented in graph format; second, because of the relative novelty of the application of this approach in the EEG field. In this sense, the main goals of the thesis have been the followings:

- First, to build an effective classification pipeline based on a strong feature extraction phase and on GNN architectures to test and validate their effectiveness when applied to complex electrophysiological data, such as the ones here chosen.
- Second, to equip the developed pipeline, and in particular the GNN architecture, with an automatic mechanism for graph structure learning. This is done to avoid the arbitrariness of the choice about how to determine graph edges, allowing the model itself to learn the graph topology. In addition, such a mechanism would considerably improve the interpretability of the obtained results.

This was done by using and comparing, in the built pipeline, three GNN architectures, based on three different well-known frameworks for graph convolutions, with the aim of performing a subject-dependent classification task on single trials of EEG signals. A particular novelty lies in the choice of data and, in particular, in the condition that was sought to be discriminated. As will be explained in detail below, the chosen EEG data were collected during a working memory task conditioned by a social component. In one condition, the experiment was conducted in the presence of a human avatar, representing the social cue; alternatively, it was conducted in the presence of a cylindrical stick, standing for the nonsocial cue.

It was precisely the type of stimulus presented in the course of the experiment that was sought to be discriminated in this work, outlining a problem of good complexity, given the little difference, also noted by the dataset's authors, between the chosen conditions.

In addition, the same problem was then solved by equipping the chosen classifier with a task-conditioned mechanism that automatically learns the structure of the graphs (in the form of their adjacency matrices). This information in "standard" GNNs is fixed and calculated a priori according to an arbitrary strategy. This mechanism, which was already present in the literature, was adapted and modified for the objective of this work, building a classifier that in an end-to-end fashion would have been capable of simultaneously classifying between conditions and learning for each training sample, and thus by aggregation for each condition, the corresponding functional connectivity graph.

The use of such a mechanism does not necessarily imply an increase in classification performances, but its aim is to increase the interpretability of the results obtained, with the idea of studying the resulting brain connectivity maps, for instance, from a physiological point of view.

As will be explained in detail in the following, since these graphs are learned at the same time as the classification task, with its outcome impacting the learning of the graphs themselves, they are not to be understood exactly as a measure of connectivity in the classical sense but rather as a representation of the brain connectivity that the network has found best while trying to improve classification results.

The automatic learning of brain connectivity maps is, at least theoretically, the best strategy for graph learning, since it excludes any arbitrary decision that is made a priori. However, it significantly increases the training complexity, sometimes leading to a lack of convergence.

Thesis outline

In this section, the content of each of the chapters that compose this thesis will be briefly presented.

The first chapter, named "Basics of electrophysiology and brain connectivity", will provide insight into how EEG electrical signals are physiologically generated, along with a presentation of the main features of this kind of data and an introduction to brain connectivity. Moreover, since it has been crucial for the development of this work, some hints about basic graph theory will also be provided.

In the second chapter, "Dataset description", I will give a thorough description of the dataset that has been used in this work, with particular attention to the dynamics of the experiment conducted for data collection and to the dataset structure and preprocessing steps performed by its authors.

Chapter three, "Artificial intelligence for classification: machine learning and convolutional neural networks", attempts to give an overview of some of the most well-known artificial intelligence methods for classification, which is the key task in this work. The methods described in this chapter have all been important in the thesis development; indeed, they have been either concretely used in it or are at the basis of some project aspects.

The "Proposed approach" chapter contains all the main concepts proposed in the thesis. First, it introduces the methodology used, graph neural networks, on a theoretical level, also emphasizing the extension of the classic convolution operation to graph-shaped data, which underlies all the architectures proposed here. Then, it explains, again from a theoretical point of view, all the frameworks and operators that have been used. Finally, it presents the experiments that have been carried out based on all the concepts and tools analyzed previously.

In the "Results" chapter, the results of all the performed simulations will be presented. Initially, the results of the "standard" GNNs will be shown and compared with those of the baselines implemented to validate them. Then, the results obtained through the graph structure learning mechanism will also be shown, both from the point of view of classification accuracy and learned connectivity maps.

In the "Conclusion" chapter, I will summarize the main aspects of the proposed thesis work, emphasizing the achieved results and presenting possible future developments.

Basics of electrophysiology and brain connectivity

The nervous system and the generation of brain electrical activity

The nervous system is the set of organs and structures that enables the transmission of signals between the different parts of the human body. It also controls the coordination of voluntary and involuntary physical and psychological functions. It can be divided into two basic parts: the central nervous system and the peripheral nervous system.

The central nervous system, consisting of the brain and spinal cord, is responsible for collecting and processing stimuli from the peripheral nervous system, generating appropriate responses from them to be spread back to the rest of the body. The cells that represent the basic anatomical units of the central nervous system are called neurons.

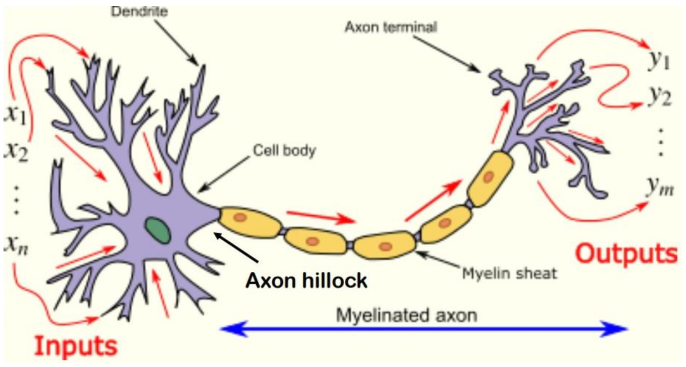


Figure 1: Anatomical representation of a neuron

Three are the basic functions of these cells:

- Collection. Each neuron is responsible for collecting multiple pieces of information from multiple sources.
- Integration. Neurons must process the incoming information, both in time and space, to provide accordingly a binary decision.
- Generation and Propagation. The binary decision is generated and propagated up to target cells, which may include muscle cells and other neural cells.

Thus, the multiple inputs $x_1, \ldots x_n$ collected by a neuron are processed to obtain one single binary decision, which is then propagated to target cells. Each of these functions is performed in a different part of the neural cell.

Main parts of neurons can be surely identified in its body, in which it is located the nucleus, and in its branches: the axon, which allows signal transmission, and the dendrites, which are devoted to signal reception from other neurons. Each part is then covered and protected by the neural membrane, which is a fundamental structure for each of the cell's functions.

The membrane is characterized by a resting potential that ranges between *-60* mV and *-70* mV. This value is guaranteed by the electrochemical equilibrium expressed, for a given ion family, through the Nernst equation:

$$\Delta\mu = RT\ \ln\frac{[X]_A}{[X]_B} + zF(E_A - E_B)$$

where the first addend represents the diffusional forces caused by different concentrations of ions in the intra- and extracellular fluid, while the second addend represents electrical forces generated by the attraction of positive ions toward negative potential regions. The neuron membrane is provided with fundamental structures called ion pumps, which control the crossing of the membrane by ions. In certain cases, the currents of ions crossing the membrane can generate variations in the membrane potential that, if a certain potential value is reached, cause the so-called action potential.

It is a variation of the membrane potential that only appears in neural, muscle and cardiac cells. The effect of this variation spreads to other cells and carries information. It is a process binarized by a threshold: if the stimulus does not reach a certain threshold, it does not occur. If, however, the threshold is reached, it will always have the same form, duration, and intensity, regardless of the amplitude of the stimulus. Action potentials represent the neurons' binary decision, which, starting from a cell, propagates to many others.

Electroencephalographic data

The electrical signals of the brain thus generated represent measurable brain activity. Their measurement can be performed both intracellularly and extracellularly.

Electroencephalography (EEG) belongs to the latter family and is a noninvasive technique. It records cerebral electrical activity through the use of electrodes placed on the scalp. The EEG signal is obtained as a potential difference between two electrodes (at least one of which is placed on the scalp).

The nerve structure that contributes the most to the generation of EEG signals is the cerebral cortex, which has a well-defined structure. It is divided into six layers, which consist of two types of neurons: nonpyramidal and pyramidal. The former has very short dendrites that develop in all directions, allowing communication between neighboring neurons; the pyramidal, instead, are so called because of the shape of their soma, make up approximately the $\frac{3}{4}$ of the cortical neurons and are located in layers II-III and V-VI. This type of neural cell has an apical dendrite that allows the transmission of information to the more superficial neurons. The dendrites are arranged parallel to each other and perpendicular to the surface of the cortex. Therefore, the pyramidal neurons, with their postsynaptic activity, contribute mainly to the generation of the EEG signal, given the particularly orderly arrangement of their dendrites.

However, the electrical signal detected by the electrodes still has a very low voltage, so that the acquisition system must include several stages, including amplification, to enable this signal to be visualized through a trace. The main problems with EEG are the low amplitude of the signals, together with a high sensitivity to noise (such as muscle artifacts or eye movement artifacts), as well as a spatial resolution that is inherently impaired by the passage of signals through multiple layers of tissue, especially through the skull. However, the tracing recorded by means of EEG represents the graphic, time-continuous recording of the electrical activity generated by populations of thousands of neurons and is formed by a succession of waves with different frequencies and amplitudes.

This technique has an excellent temporal resolution, can be used to detect signals in real time, to assess in which cortical areas the most activity is present and is also useful for diagnosing brain disorders. Different EEG acquisition systems differ mainly in the type of electrodes used, their number, sampling frequency and much more.

The bandwidth of the EEG signal is approximately between 1-50 Hz, although most of the information content lies within a frequency of 40 Hz. Depending on the different oscillation frequencies of the waves of neuronal activity, one can distinguish different brain rhythms that characterize the EEG signal. Each of these rhythms is particularly solicited in different cognitive tasks, reflecting the synchronous and coherent activity of groups of neurons participating in it.

These rhythms are:

- Alpha rhythm. It has a frequency band ranging from 8 Hz to 12 Hz. It has an average voltage of approximately 40 μV. Its presence is greater in the occipital areas of the cortex, where it has a very similar sinusoidal pattern. This rhythm is typical in a state of mental relaxation, in the waking state with eyes closed, and in environments where there are few outside stimuli. Moreover, it is particularly elicited in memory tasks.
- Beta rhythm. It is characterized by frequencies between 13-30 Hz and has an average voltage of approximately 19 μV. It is easily detectable in the frontal areas of the cortex. It is typical of conditions of attention, concentration, and alertness and plays a role in the coordination of motor activities. It is detected when there are many stimuli from the surrounding environment.
- Gamma rhythm. It is a rhythm with frequencies above 30 Hz but low amplitudes of approximately 15 μV. The gamma rhythm occurs in the

presence of high cognitive functionality, including processes that allow the integration of information from different cortical areas.

- Delta rhythm. It is characterized by waves oscillating between 1-3 Hz and having amplitudes of up to 200 μV. This rhythm is associated with deep sleep conditions but can also be detected in wakefulness or vague dream states.
- Theta rhythm. It is characterized by frequencies oscillating between 4-7 Hz. It plays a role in deep sleep states and in the retrieval and encoding of episodic memories. Theta rhythm is also assigned a role in visual imagery and hypnopompic imagery.

As anticipated, EEG signals are extremely sensitive to noise and artifacts. Various sources can generate these distortions: eye movements and blinks, muscle contractions, sweating, and cardiac activity are among the most common artifact sources caused by the human body. In addition, improperly applied electrodes, electrodes, small movements and alternating current-related artifacts can also affect the signal. Therefore, the experiments during which EEG signals are recorded are usually repeated multiple times, each of which represents a so-called trial. Moreover, a fundamental step before using and analyzing EEG signals is a strong preprocessing procedure aimed at obtaining a much cleaner signal.

Brain connectivity

The brain is a complex system, and in it, it is possible to identify dynamic brain networks at different scales: neurons, brain areas and social systems. The brain's analysis from this perspective is called network neuroscience and aims at recording, mapping and modeling the elements and the interactions between neurobiological systems.

So-called brain connectivity is a description of brain networks. In particular, three are the connectivity natures that characterize these networks:

- Anatomical connectivity. It describes physical or structural connections linking sets of neurons or neuronal elements. It is relatively stable at short time scales; thus, during a recording session of a few hours, it is not expected to change. Anatomical connectivity is at the basis of functional and effective connectivity but cannot explain them.

- Functional connectivity. It is essentially a description of what is happening in the brain in terms of similarity between the activity in different regions. Usually it is based on correlation, coherence and transfer entropy.
- Effective connectivity. It refers explicitly to the influence that one neural system exerts over another, either at a synaptic or population level. It is based on the concept of causality.

Graph theory for brain data representation

As has been anticipated in the previous paragraph, in the brain it is possible to identify networks and one of the most suitable ways of representing a network is certainly using graphs. While analyzing EEG data, each of the brain areas contained in these networks can be identified by an electrode, which will be represented in the graph as a node.

A graph can be formally defined as a tuple $G = (V, E)$, where V is the set of $N = |V|$ nodes and $E \subseteq V \times V$ is the set of $M = |E|$ edges linking the nodes. In many cases, this definition is extended, including the matrix $W \in R_{+}^{V \times V}$, in which each entry w_{ij} represents the strength of the connection between nodes i and j. The network topology is represented through the use of the adjacency matrix $A \in R_{+}^{V \times V}$. In the case of unweighted graphs, each entry a_{ij} of the adjacency matrix is equal to 1 if an edge linking node i to node j exists and is 0 otherwise. In weighted graphs, the value of the entries in which a link between nodes exists is $a_{ij} = w_{ij}$ and is 0 otherwise. Except for the values of the entries, these two definitions of the adjacency matrix will share the same sparsity pattern.

Another way of representing a graph is to use its Laplacian matrix

$$L = D - A$$

where D is the graph's degree matrix and its normalized Laplacian matrix $\tilde{L}$. This consists of pre-imposed multiplications by the inverse of the square root of D, resulting in

$$\tilde{L} = I - D^{-\frac{1}{2}} A D^{\frac{1}{2}}$$

The usefulness of graphs for representing networks, and in particular brain networks, is not only linked to the graphical support it provides. Indeed, they are associated with their (weighted or unweighted) adjacency matrix, which allows us to compute quantifiable and objective measures, called indices, which are able to describe properties of the network that sometimes can be difficult to extract just looking at the graph.

Before introducing some examples of graph indices that have been used for this thesis work, it is worth mentioning a couple of opposite graph properties that some of these indices measure:

- Integration, a network is more integrated when the parts that make it are more strictly interconnected.
- Segregation, a network is more segregated the more it can be divided into subnetworks.

As anticipated, some examples of graph indices are listed below:

- Weight of intra- and interconnections, given at least two groups of nodes to consider as clusters. They are computed as the sum of the weights of the connections inside and between the nodes of the specified groups.
- Modularity measures the tendency of specified subnetworks to form communities. A community is defined as a group of nodes that have more connections between them with respect to the average number of connections they have with any nodes in the network. Positive modularity means that the network can be divided into communities that behave as a group and not as random nodes. Negative modularity instead means that

communities do not behave as communities. Finally, modularity assumes values around zero when there is a random division between nodes.

The higher the modularity is, the higher the segregation, and the lower the modularity is, the higher the integration.

- Divisibility, which given two communities, measures their segregation, i.e., how much they are connected. Its value is generally contained in the interval [0.5, 1], where values close to 1 mean that the analyzed communities are completely not connected.

  The higher the divisibility is, the higher the segregation, and the lower the divisibility is, the higher the integration.

Dataset description

Description of the experiment

The chosen dataset [13, 14] collects electroencephalogram and behavioral data belonging to 47 subjects acquired during a working memory task presented in virtual reality.

The recruited participants were 49, divided into 33 females (67%) and 16 males (33%). They were all aged between 18 and 32 with a mean age of 21 and a standard deviation of 3.1. Three of the subjects were left-handed. It must be noted that 2 female participants in the initial group did not have their EEGs recorded.

The experiment's task consisted of remembering the status and the location of some objects placed above a table. Moreover, subjects were previously informed that a cue, a human avatar or a stick, would have been shown with the only aim of distracting them, without giving any help about the task. The human avatar represented a social stimulus, while the stick, a very simple cylindrical object, was a nonsocial one. There were 10 practice trials and 112 experimental trials for each cue type, for a total number of 224 trials for each subject. The cue condition was counterbalanced so that both the social avatar and the nonsocial stick condition were shown first, and all trials were completed before seeing the other cue.

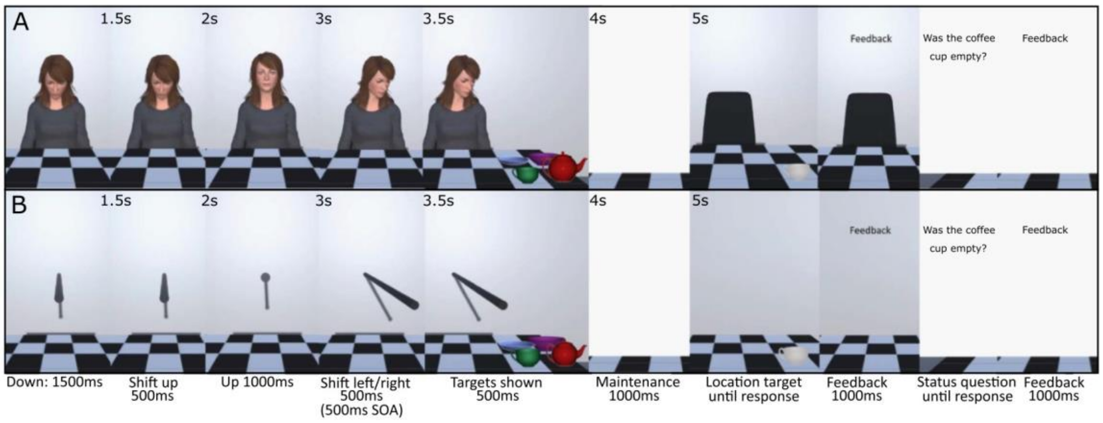


Figure 2: Illustration of the trial procedure showing both cues

As shown in Figure 2, one trial is composed of the following phases: first, a fixation cross is shown for 1 second, representing the intertrial interval; then, the cue is shown pointing or looking down for 1.5 seconds, after which, with a transition of 0.5 seconds, it starts pointing to the participant for 1 second. At this point, the cue starts looking/pointing to the left or right with a transition time of 0.5 seconds, and when it stops, the target objects are shown on the table for 0.5 seconds. The time interval between 2.5 and 3.5 seconds from the beginning of the experiment is called the cueing phase, in which there is direct contact between the cue and the subject; the interval between 3.5 and 4 seconds is called the encoding phase, in which the subject has to store all the information about the target objects' location and status. At the end of the encoding phase and after 1 second of the maintenance phase in which a blank screen was presented, two questions were asked of the subject. The first concerns the location of one of the target objects, and the second concerns its status (e.g., the cup was full or empty). All the objects presented on the table were objects that could be commonly found on a table.

Overview of dataset structure

Available data consist of complete EEG recordings belonging to 47 subjects and behavioral data of 49 participants. These include the reaction time and accuracy of the answers to questions about objects' location and status presented at the end of each trial. EEG data are given both in a raw and processed form.

The raw data that have not been used in this work are displayed in BIDS format (*.eeg .vhdr .vmrk*). Accordingly, there is a separate file for each participant in tsv format that contains the channels recorded (the same for each subject), as well as the data in brain vision format, which consists of the following:

- The binary data file (*.eeg*) that contains the recorded time series of the EEG;
- The header file (*.vhdr*), which contains data-related information such as amplifier settings, software filters, and channel count;
- The marker file (*.vmrk*), which includes the markers contained in the EEG data along with their data point positions, timestamps, marker type, descriptions, and references to the .eeg file.

The preprocessed data that have been used in this thesis, are organized in a similar structure.

Data preprocessing

As mentioned, in this work, preprocessed data have been used. For completeness, the steps that the dataset's authors have followed for data preprocessing are listed below using a MATLAB toolbox:

1. Data were loaded into MATLAB software and any unused channel, i.e., mastoids and EOG, was removed, selecting only relevant ones.
2. Data were detrended
3. Data were filtered 0.5 and 36.0 Hz with a bandpass filter.
4. To ensure that the cue was onset at time 0, a custom trial function was utilized to epoch the data from 1 second precue onset to 1 second post-probe response (including also the feedback period). This implies that each trial had a minimum duration of 7000 *ms*.
5. The training trials were removed, and accuracy information was included from the behavioral data.
6. A visual inspection was carried out on trials to detect artifacts. In this phase, trials corrupted with large artifacts were deleted. Trials affected by the presence of corrupted electrodes were interpolated using the average method.
7. Data were rereferenced after the visual inspection phase.
8. Noise, eye-blink, saccade, heartbeat and muscle components were manually identified and removed using independent component analysis (*fastica*). On average, 11 components were removed from each subject.

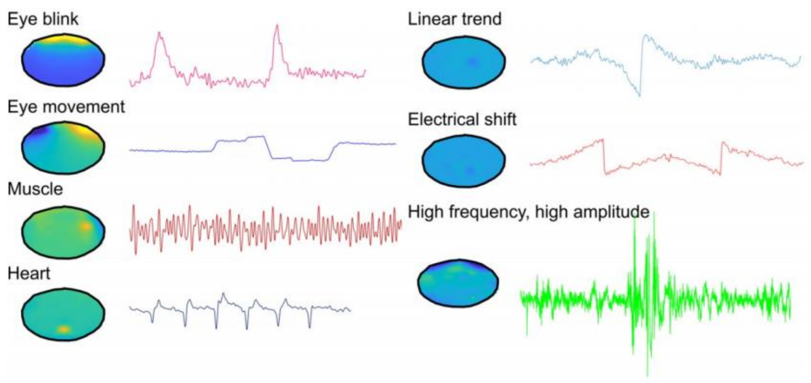


Figure 3: Example of the artifacts identified in the preprocessing procedure

In the work presented here, as will be further discussed later on, the focus has been on the cueing phase, training the graph neural networks to distinguish, starting from the recorded EEG signals of a given trial, whether the cue was social or nonsocial.

EEG data were recorded with a 64-channel eego™ sports mobile EEG system, but for the experiments conducted here, a subset was chosen. Further details will be given in the "Experiments" chapter.

Artificial intelligence for classification: machine learning and convolutional neural networks

Classification is one of the most widespread problems that intelligent systems are able to solve. The goal of this family of problems concerns associating a label representing the group to which they belong with dataset instances. In this work, a binary classification problem has been the object of study. It can be formalized as follows: given a dataset $D = \{(x_n, t_n)_{n=1}^{N}\}$ where x_n is a sample and $t_n \in \{1, -1\}$ is its associated target label, the goal is to learn a function $y: x \rightarrow \{1, -1\}$ that maps each input sample to a label.

Because of the importance and the huge number of applications that such problems may have, many techniques have been developed in the field of artificial intelligence to solve them.

In the following subchapters, some of these methods that are necessary to explain and introduce concepts and experiments of this thesis work will be presented.

Machine learning

Machine learning is a branch of artificial intelligence that focuses on programming machines to improve a performance criterion using example data and past experience. Below, we will explain two machine learning algorithms that are widely used to solve classification problems and that have been used as a baseline in this thesis.

Support Vector Machine

Support vector machine, proposed in the first half of the 1990s in [15], is one of the most effective and spread supervised machine learning algorithms for classification.

Given the labeled training samples of a dataset, it learns an optimal hyperplane in the feature space that discriminates dataset samples. In two-dimensional spaces, the hyperplane can be seen as a simple line.

Given a binary classification problem formalized as above, it learns a linear model of the form:

$$y(x) = w^T x + w_0$$

where the parameters w and w0 are learned to maximize the distance between the closest samples (the so-called support vectors) and the model.

However, some problems are intrinsically nonlinearly separable in the input feature space, and to solve them, nonlinear transformations must be applied. In this way, the feature space dimension increases, requiring more complex solutions. To solve this kind of problem, it is possible to use the kernel method based on different kernel functions: the most common are *linear*, *polynomial* and *Gaussian radial basis* functions. Moreover, another parameter of great importance in SVM functioning is the constant regularization term C. It controls the tradeoff between training sample accuracy and the generality of the solution: large values of C will lead to smaller margin solutions that attempt to classify most of the training samples well; small C values instead will cause a more general larger margin solution, even if it costs the misclassification of some training samples.

Random Forest

Classifiers based on decision trees are often used because of their very advantageous trade-off between performance and computational speed. However, one of their main limitations is the impossibility of arbitrarily growing their complexity to fit the dataset. Some of the methods aimed at this task were able to achieve very high performances on the training data, but due to the assumptions made in increasing the complexity of the trees, what had been learned was generalized with great difficulty to the test data, causing a radical drop in accuracy. The random forest method [16] is based on the use of multiple trees. a forest, with the idea that multiple classifiers would counterbalance the single errors made by each of them taken alone. However, to guarantee great functioning, it is fundamental that each model is uncorrelated, or at least with a low correlation, with all the other models. The proposed method consists of creating models in randomly selected subspaces of the feature space using the entire training set. Starting from the t trees created in the random features subspaces, the discriminant function to combine their predictions and obtain the final classification result is:

$$g_c(x) = \frac{1}{t}\sum_{j=1}^{t} \hat{P}(c|v_j(x))$$

where $\hat{P}(c|v_j(x))$ is the probability for sample x to belong to a certain class c according to the final node $v_j(x)$ predicted by tree j. This value is computed for each class c, and node x is assigned to the class that maximizes $g_c(x)$.

Note that thanks to the trees' splitting criterion, they are fully split; thus, in most cases, $\hat{P}(c|v_j(x)) = 1$, and the formula simply states that the belonging class c of a node x is the one predicted by more trees with respect to all the others.

Convolutional Neural Networks

Convolutional neural networks (CNNs) are a particular family of neural networks that are extremely effective in processing data characterized by a known grid-like structure. Some examples are time series, images and videos.

Their name immediately suggests that CNNs make use of the well-known convolution operation. Moreover, most of these networks also use operations such as pooling and nonlinear activation functions [17].

In machine-learning applications, the definition of convolution operation that it is used usually refers to discretized signals

$$s(t) = x(t) * w(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

Moreover, in these applications, convolutions are often used over multidimensional axis inputs that are filtered contemporarily. For instance, in the case of a two-dimensional input image I, it can also be worth using a two-dimensional filter K:

$$S(i,j) = I(i,j) * K(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m,j-n)$$

What practically happens is that the filter, or kernel, usually much smaller with respect to the image, slides along the input computing in each position the dot product between its parameters and the values in the corresponding patch of the image. The results obtained by performing this operation on the whole image are aggregated in the layer's output, usually called the feature map. It is worth emphasizing that one single convolutional layer can have multiple kernels, each of which will learn different parameters, extracting, as a consequence, different features.
The width of the kernel sliding is determined by a value called stride. When the filter reaches the ends of the input, it goes off the edge and has no values with which to perform the operation. Thus, it cannot return a value, so the size of the output will be smaller than the input. As a consequence, this would impose an upper limit on the number of successive convolutional layers that can be used. In such cases, it is possible to use so-called padding, which extends the size of the input according to a certain strategy, counteracting the abovementioned problem.

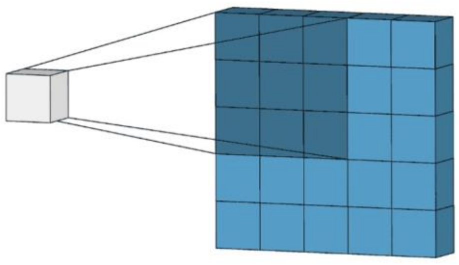


Figure 4: Illustrative example of how the convolution operation is performed

As anticipated, two layers that are commonly used to build a CNN are pooling layers and activation function layers:

- The first is mainly aimed at reducing the dimension of feature maps extracted by convolutional layers by aggregating information belonging to spatial neighborhoods of their input. Usually, the aggregation strategies are simple mathematical functions such as taking the mean, the maximum value or the sum of each patch.

- Activation layers instead are used to introduce nonlinearities in the feature maps. Indeed, convolution is a linear operation, but relationships between image pixels are usually nonlinear. Therefore, to extract more meaningful features, functions such as sigmoid and ReLU are commonly used.

For classification purposes, the output of the last convolutional layer, which embeds the features extracted from the input, is usually fed into a dense layer (a fully connected layer) or into a multilayer perceptron, which is a sequence of dense layers.

The success of CNNs in processing Euclidean grid-like data relies on their ability to exploit properties that are intrinsic in these kinds of data, such as stationarity and locality. For what concerns images, these two properties allow us to use filters that slide along the entire image analyzing small patches: indeed, stationarity refers to the presence of shift-invariant features, allowing us to extract meaningful features independently from their location; locality instead is related to the concept that even in a single small patch of the image, it is possible to find extremely significant features.

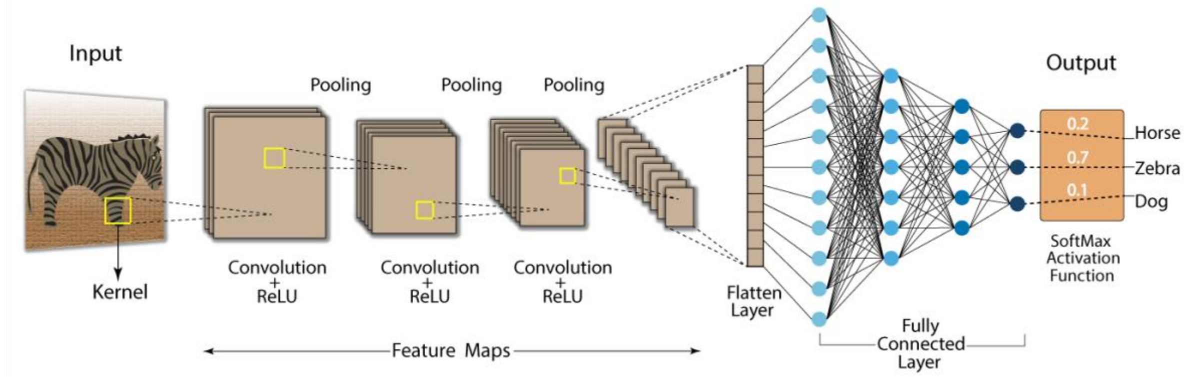


Figure 5: Example of a complete CNN architecture

Proposed approach

This chapter will first introduce the application of artificial intelligence to graph-structured data, and in particular graph convolutional neural networks, the methodology studied and applied in this work. Moreover, before describing the experiments carried out, I will describe some operators and frameworks on which they are based.

Introduction to graph learning

In recent years, deep learning methods, and in particular convolutional neural networks, have become the state of the art in a wide range of tasks involving many kinds of data, such as images and videos. Among the main factors supporting their success are the ever-increasing availability of large amounts of data, which are indispensable for architectures of this kind, and the technological development of increasingly powerful computational means, such as GPUs. Alongside these external factors, the ability to capture and exploit the statistical properties of all those types of data that have an underlying Euclidean structure is certainly fundamental in the spread of these methods.

However, there are an increasing number of fields of application where data have a non-Euclidean but a graph structure. For instance, in social networks, each user represents a node of the graph, and all his connections (friends, followers) are the edges of the graph itself; in neuroscience, as in the work presented here, the anatomical and functional structures of the brain are represented by means of graphs; in chemistry, the graph structure is perfectly suited for representing molecules, with atoms as nodes and their connections as edges.

The complex non-Euclidean nature of graph data makes the extension and application of some operations that are common on grid-shaped data, such as pooling or convolutions, very difficult. Indeed, for example, graphs can be irregular in terms of shape, number of nodes and neighborhoods of each node [18].

Some early attempts to apply 'classical' deep learning techniques to graphs involved their encoding in simpler structures, for example, in vectors or reals, in which, however, much information, such as some aspects of their topological structure, could be lost.

The first efforts devoted to the proposal of new, unseen methodologies designed explicitly for graphs, and in particular the proposal of graph neural networks, date back to 2005, when Scarselli et al. [19] proposed a method for learning on graphs, presented as an extension of recurrent neural networks.

Most graph neural network applications that can be found in the literature can be broadly summarized into three main classes:

- Graph-focused applications. The task is referred to as the entire graph. This often involves the use of a readout function (mean, sum…) that aggregates the node embeddings previously obtained to obtain an overall representation of the entire graph. Some examples are graph classification, graph property estimation, and graph generation.
- Node focused applications. The task is directly associated with single nodes of the graph. Node classification and recommendation are examples of these applications.
- Edge-focused applications. Despite being less explored, it can also be identified as one of the classes of graph deep learning applications. The main goal of works belonging to this class concerns edge classification and prediction. Typically, this involves the use of a similarity function (e.g., cosine similarity) or of a neural network to compute the similarity measure between two nodes’ embeddings to predict the label or the strength of the edge linking those nodes.

In graph learning applications, graph nodes typically have some features representing them.

Graph convolutional neural networks

The growing success of convolutional neural networks as one of the state-of-the-art methods for solving deep learning tasks has led to their generalization to non-Euclidean data in the form of graphs and manifolds [20].

In general, graph convolutions allow, following a message passing scheme, aggregating features of neighboring nodes, creating a sort of node-level embedding of the initial features.

Graph convolutional neural networks can be classified as either spectral-based or spatial-based, according to the definition of graph convolution they use. The first group of methods strongly relies on graph signal processing theory and defines the graph convolutional operator as a sort of filter to remove noise from graph signals. The second group instead takes direct inspiration from recurrent graph neural networks (RGNNs), inheriting the idea of information propagation and message passing through graph nodes.

In 2013, Bruna et al. [21] used the definition of convolutions in the spectral domain to develop the first definition of CNNs on graphs. Although conceptually significant, their paper had serious computational flaws that prevented it from being a truly effective solution. However, thanks to the recent growing interest in this hot field, many new architectures exploiting an extension of this concept have been published, reaching first state-of-the-art results in [22].

In parallel, the first efforts in the direction of graph neural networks based on the spatial definition of the convolution were made by Masci et al. [23], who presented an extension of the classic convolution operator to manifolds.

Many different operators belonging to both definitions have been developed in recent years, some of which will be explained in detail later on because of their cruciality in this work.

Spectral convolutions

In this paragraph, the main concepts behind spectral-based graph convolutions are briefly introduced. Previously, the normalized graph Laplacian matrix $\tilde{L}$, which has the property of being real symmetric positive semidefinite, was defined. Thus, it can be factored as $\tilde{L} = U\Lambda U^T$, where Λ is the diagonal matrix of eigenvalues and $U \in R^{N\times N}$ is the matrix of eigenvectors ordered by eigenvalues. According to graph signal processing theory, a graph signal $x \in R^N$ is a vector containing one element from the features of each node: x_i will contain one of the features of node i. It is now possible to define the graph Fourier transform on a graph signal x as $F(x) = U^T x$ and its inverse as $F^{-1}(\hat{x}) = U\hat{x}$, where $\hat{x}$ is the original signal transformed through Fourier transform. Given this definition, it is possible to define the graph convolution operator of a signal x with a filter g as:

$$x * g = F^{-1}(F(x) \odot F(g)) = U(U^T x \odot U^T g)$$

where $\odot$ represents the elementwise product. Defining the filter as $g_\theta = diag(U^T g)$, it is possible to simplify the graph convolutional operator as:

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

Most of the proposed spectral-based graph convolutional neural networks follow this operator definition and differ according to the choice of filter g_θ.

The main limitations of this approach are strictly related to computational issues for the eigen decomposition of the normalized Laplacian matrix and with poor robustness to domain or graph changes. In this work, the graph convolutional network (GCN) operator, which derives from a strong approximation of the theoretical spectral graph convolution and is one of the most cited and used graph convolutional neural networks, was used.

Spatial convolutions

Directly inspired by the core operators of CNNs and by the message passing idea inherited from RGNNs, this family of methods relies on spatial relations between nodes. The key idea is that every kind of data can be represented by some sort of coordinates and that in this coordinates' space, it is possible to identify, for each node in the graph, its neighbors. Exploiting the graph topology, the convolutional operators propagate and aggregate node features to build an embedded representation of each node.

Architectures belonging to this group mainly differ from each other according to the way in which they aggregate the information. The main advantages are related to computational time, which is typically much shorter than spectral-based methods.

Among the three different GNN architectures that have been the object of test and study in this work, two are spatial-based. This choice mainly relies on the fact that in EEG data, the structural topology of the brain network is fundamental; thus, because of their spatial nature, these architectures should have been more appropriate to study this kind of data.

Operators and frameworks

Graph Convolution Network (GCN)

GCN was proposed in 2017 in [24]. As anticipated, it is a convolutional operator that derives from a strong approximation of the theoretical spectral-based graph convolution. Indeed, the authors identified in the eigen-decomposition of the Laplacian L and in the multiplication for the eigenvector matrix U, two very heavy computational steps, especially for large graphs. Therefore, they redefined the convolutional operator as:

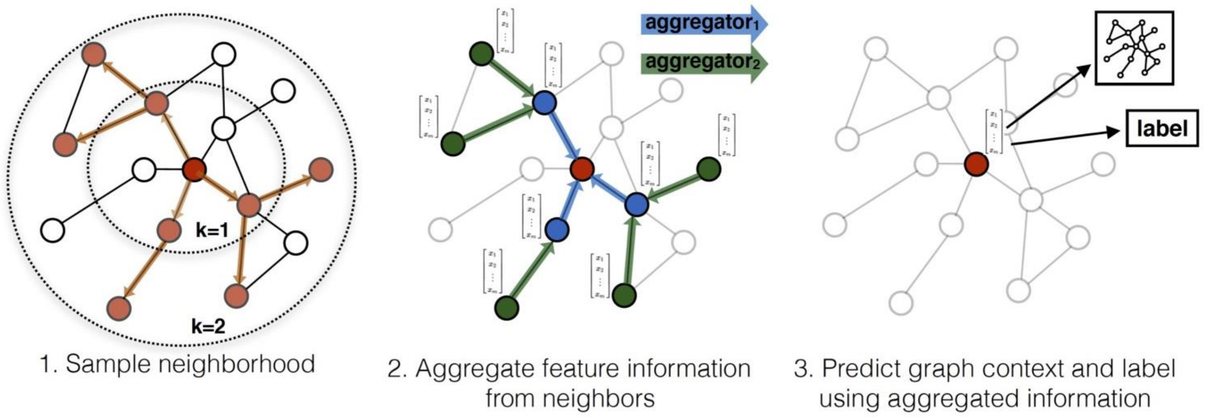
$$g_{\theta'} * x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L})\, x$$

where: $\tilde{L} = \frac{2}{\lambda_{max}} L - I_N$, θ represents the kernel parameters, T the Chebyschev polynomials and k defines the order of neighborhood to consider (how many nodes away from the current one). Thanks to two further simplifications, namely, choosing $\lambda_{max} \approx 2$ and the order $k = 1$, the final aggregation formula for a signal $X \in R^{N \times C}$,where C is the number of input channels, is:

$$X' = \widetilde{D}^{-1/2} \tilde{A} \widetilde{D}^{-1/2} X\Theta$$

Graph-SAGE

Proposed in 2017 in [25], GraphSAGE is a very effective and used framework for inductive graph learning whose name stands for *SAmple and aGgregatE*. Its functioning starts from sampling at each iteration a subset of the k-order neighbors of the current node, where the parameter k is chosen by the user. Then, the features from these nodes are aggregated in the feature space of the central node. Finally, according to the problem that it is being solved, embedded features are used to predict graph, node and edge properties.



k=1
k=2
1. Sample neighborhood
aggregator₁
aggregator₂
2. Aggregate feature information from neighbors
label
3. Predict graph context and label using aggregated information

Figure 6: Schematic representation of the GraphSAGE method

The main novelty of this architecture with respect to previous architectures is that it does not learn a different embedding for each node but an aggregation function over a subset of neighboring nodes.

$$h^k_{\mathcal{N}(v)} = f_k(\{h^{k-1}_u, \forall u \in \mathcal{N}(v)\})$$

where f_k is the learnable aggregation function, v is a node of the graph, $\mathcal{N}(v)$ is the set of neighbors of v and $h^k_{\mathcal{N}(v)}$ is the embedding at depth k of node v.

Then, the hidden representation obtained is concatenated with the node features at its previous layer ($k - 1$):

$$h_v^k = \sigma(W_k\ [h_v^{k-1}\ ||\ h_{\mathcal{N}(v)}^k])$$

where W_k is a matrix of parameters at layer k and σ is a nonlinearity function.

Various aggregation functions can be used to propagate information between nodes. One of the simplest and most effective, which has also been used in the work presented here, is the mean aggregator, which computes the elementwise average of the values in $\{h_u^{k-1}, \forall u \in \mathcal{N}(v)\}$.

Even if this architecture was thought for very large and time-changing graphs, it is also very effective in exploiting node features and the network topology, and this is the main reason for which it has been chosen to be used in this work.

Graph Isomorphism Network (GIN)

This framework, proposed in 2019 in [26], aims at maximizing the representational and discriminative power of graph neural networks. An ideal GNN would be able to discriminate between various graph structures by mapping them to different representations in the embedding space. This necessarily implies solving the nontrivial graph isomorphism problem: two graphs are isomorphic if they have the same topological structure and connections in a permutation of nodes. Thus, a maximally powerful GNN should embed two isomorphic graphs in the same representation and two nonisomorphic ones in different representations. The benchmark with which the authors evaluate the ability of GNNs in distinguishing graphs is the Weisfeiler-Lehman (WL) graph isomorphism test [27]. It is an iterative algorithm that transforms graphs in a canonical form: if two graphs have different canonical forms, they are surely nonisomorphic. However, this test cannot state with certainty if two graphs are isomorphic; indeed, two nonisomorphic graphs can still share the same canonical form.

The authors claim that a GNN is as powerful as the WL test in mapping nonisomorphic graphs to different embeddings if it iteratively aggregates node features according to the formula:

$$h_v^k = \sigma(h_v^{k-1}, f(\{h_u^{k-1}, \forall u \in \mathcal{N}(v)\}))$$

where both functions σ and f are injective. Moreover, the graph-level readout function must also be injective.

The intuition of the authors of the paper is to use an MLP to learn functions σ and f using the following node aggregation:

$$h_v^k = MLP^k((1-\epsilon^k) \cdot h_v^{k-1} + \sum_{u \in \mathcal{N}(v)} h_u^{k-1})$$

Regarding the graph-level readout function, the authors propose three different solutions: MAX, MEAN and SUM.

In this way, they theoretically build a graph convolutional layer that is as powerful as the WL test.

Adaptive and task-conditioned graph structure learning

In 2020, Jia et al. [28] proposed a novel architecture for sleep stage classification based on spatial-temporal graph convolutional neural networks. A key part of this architecture, which has been the object of interest and is also studied in this work, concerns the graph structure necessary for feature aggregation in graph convolutions. Indeed, although GNNs commonly use a fixed a priori graph structure for this purpose, this is not perfectly suitable for sleep stage classification because graph structures may differ among different sleep stages. The authors' idea is that while learning how to classify sleep stages, it would be more effective to dynamically learn also the graph's topology, without building it a priori.

The assumption behind the mechanism created for this purpose is that two nodes will be tied by an arc of the greater weight the more similar their features are. Therefore, they defined a nonnegative function

$$A_{mn} = g(x_m, x_n), \forall m, n \in \{1, \dots, N\}$$

where *N* is the number of nodes of the graph, which represents the intensity of the relationship between nodes *m* and *n* according to their input feature vectors x_m and x_n.

This is implemented in the Tensorflow library through a neural network layer with a learnable weight $w \in R^{F\times 1}$, where F denotes the number of features. The formula used by this layer in order to compute one entry of the adjacency matrix is the following:

$$A_{mn} = g(x_m, x_n) = \frac{\exp\left(ReLU(w^T|x_m - x_n|)\right)}{\sum_{n=1}^{N} \exp\left(ReLU(w^T|x_m - x_n|)\right)}$$

The ReLU function is used to guarantee A_{mn} nonnegativity, while the softmax operation normalizes each row of the matrix to have a sum equal to 1.

It is important to emphasize that the matrix thus calculated would be symmetrical; however, the chosen normalization strategy breaks its symmetry.

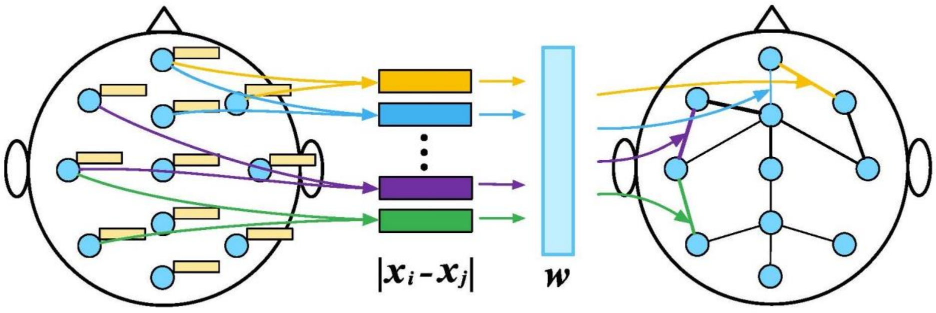


Figure 7: Schematic representation of the adaptive graph structure learning mechanism

The weight w is updated according to the minimization of the following loss function:

$$\mathcal{L}_{graph\_learning} = \sum_{m,n=1}^{N} \|x_m - x_n\|_2^2 \, A_{mn} + \lambda \|A\|_F^2$$

that, as anticipated, assigns smaller connection weights A_{mn} to nodes that have larger features distances $\|x_m - x_n\|$. Moreover, to control the matrix's sparsity, the second term is added and multiplied by the regularization parameter λ.

To avoid the trivial null solution, this loss function must use a regularized term in the complete loss function, which also contains the classification loss term.

In this thesis, it has been the object of interest to study the possibility of using such a mechanism even in a different classification context. Indeed, if, as stated in the reference paper, sleep stages may strongly differ in their graph structure, the conditions that are discriminated in this work are not theoretically supposed to be too different, since the only difference is in the stimulus social nature, while keeping the task structure fixed.

Experiments

In this subchapter, the experiments carried out in this thesis will be presented to solve the chosen classification task. First, a deep explanation of the task to be solved and of the data preprocessing steps that have been performed will be given. Then, the details of the GNN architectures developed to solve this problem are presented.

Task description

In the dataset-related chapter, the extreme relevance of the social component in the experiment performed for data collection is underlined. Indeed, the dataset's authors wanted to investigate the effects of dynamic social and nonsocial stimuli on participants' working memory.

Driven by the same interest in the social aspect of these data, this thesis work has also been chosen to examine this topic, developing and training models with the aim of discriminating whether the social or nonsocial cue was presented in a given trial.

In particular, the trials' window of interest, on which the classification was carried out, is the 1000 ms long *cueing* phase, in which half of the time is characterized by direct eye contact between the subject and the cue.

Because of the great intersubject variability that characterizes EEG data, the classification was performed in a subject-dependent fashion, meaning that the models were trained and tested on trials all belonging to the same person.

Thus, to summarize, the classification was focused on identifying whether, given a specific subject, his/her trials were conditioned by the presence of the avatar cue (social) or the stick cue (nonsocial).

Data preprocessing and feature extraction

The whole project has been developed using PyTorch and PyTorch Geometric [29], a very effective framework for applying deep learning to input data that are characterized by an irregular structure, such as graphs or manifolds. In this library, a graph with N nodes is formalized as a tuple $G = (X, (I, E))$, where $X \in R^{N \times F}$ is a node feature matrix, $I \in N^{2 \times E}$ is a couple of lists containing node indices of starting and ending points of each edge and $E \in R^{E \times D}$, which stores the optional edge features. The set X contains, for each node, a number F of properties, or node features, that characterize it. Set I is necessary to the network to build the network topology and to identify nodes' neighbors, which is fundamental to aggregating their features.

As mentioned, in this work, the training procedure was carried out in a single trial. This means that one training instance was constituted by the EEG signals, one for each electrode, of the cueing phase of each trial mapped into a graph to be in a suitable format for being the input of the GNNs.

To this aim, graph nodes were identified by electrodes; then, it was necessary to choose two strategies to extract node features and determine graph edges starting from the EEG signals of the electrodes.

Concerning electroencephalographic signals, it is extremely difficult, if not impossible, to determine a ground truth of how brain areas, here represented by electrodes, are connected. Thus, to provide the GNN with information about how nodes are connected, it is necessary to estimate the connections using certain measures, for example, related to the correlation or synchronization of the electrodes' signals.

In this work, three different alternative approaches for determining the strength of the link between nodes, given their EEG signals, have been considered: Pearson [30] correlation, partial correlation and phase locking value (PLV) [31]. The latter is a strategy for measuring the frequency-specific synchronization between the neuroelectrical signals of the EEG.

To determine all the edge intensities of a graph, these measures have been computed in a pairwise approach, leading to a NxN matrix where each row contains the intensity of the links between the node corresponding to that row and all the other nodes. This matrix can be seen as the weighted adjacency matrix of the graph based on partial correlation. However, it must be emphasized that in this work, the classification was performed using unweighted edges; therefore, this weighted adjacency matrix had to be binarized, imposing a certain threshold.

Tests showed a higher classification accuracy when links between nodes were computed using the partial correlation coefficient. This measure quantifies the correlation between two signals through a number contained in the interval [-1, 1], where the extremes represent maximum negative or positive correlation between the two signals, while values approximately 0 represent no correlation between them.

To binarize the weighted adjacency matrix thus obtained, its elementwise absolute value was first computed and then binarized using a threshold of 0.5. The choice of such a high value as a threshold is motivated by the need to reduce the number of connections in the graphs. Indeed, experiments showed that using lower thresholds, the matrix would have been highly dense.

For what instead concerns node features, various approaches have been attempted:

- A first attempt involved the use of all electrode signals as node features. However, the lack of meaningful features extracted from them caused many difficulties in network convergence, making it impossible to overfit the training data.
- Similar results were obtained using some very simple statistical properties (mean, standard deviation) of the electrode signals as node features.

- Great improvements have been made by choosing features that not only focus on nodes' intrinsic properties but also represent their relationship with the other ones. Indeed, GNNs were first introduced to deal with large graphs and contexts of application in which node features are certainly well defined, while a precise definition of the edges could not exist. As mentioned, EEG data belong to these contexts since no ground truth of brain connections is available.

  Therefore, even if information about how nodes are connected are already given to the network in the "standard" training approach, it proved very useful to give to it also redundant information about this also in the node features. This choice is also useful when dealing with the automatic learning of graph-structures. Indeed, in this way, node features, that are essential for the learning process, also contain some insights about the edges that have to be learned.

  Tests have shown the best performances when using the Pearson correlation to compute node features. In particular, similar to what has been done with partial correlation for edge determination, the pairwise Pearson correlation coefficient has been computed for each pair of electrode signals, obtaining the Pearson correlation-based weighted adjacency matrix of dimension $N \times N$. As before, for instance, the first row contains the correlation of the first electrode with all the other ones and here has been chosen to constitute the first node features. Note that in this case, no binarization has been performed.

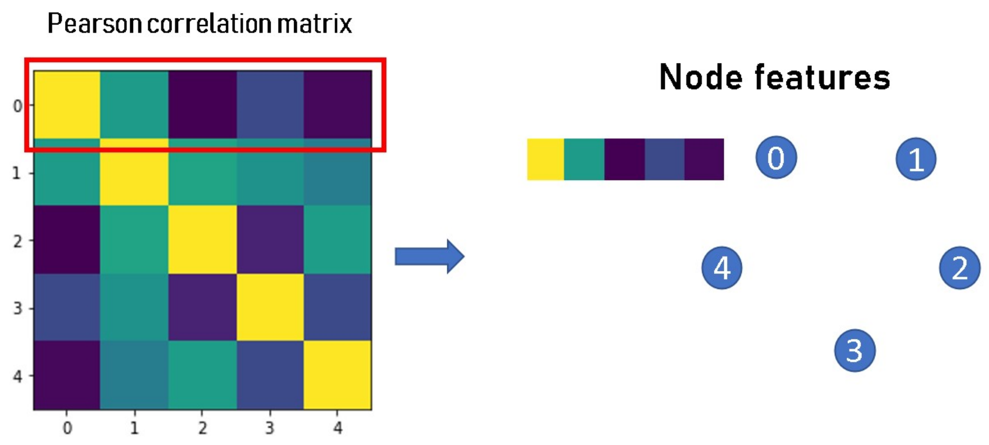


Figure 8: Illustrative toy example of how node features are extracted for node 0 from the Pearson correlation matrix

In the same direction, an alternative that has been tried involved the use of the phase locking value coefficient in place of the Pearson correlation coefficient. However, the use of this measure as node features caused a drop in terms of accuracy performance with respect to the Pearson correlation.

In the dataset chapter, it has also been described structure, underlying that the recorded electroencephalographic signals consisted of 64 different channels, including mastoids and EOG then deleted in postproduction, reducing the total number of relevant channels to 61.

In this work, to reduce the computational complexity while still maintaining a good representation of the brain activity, we chose to further reduce the number of electrodes, choosing a subset of 29 electrodes such that the whole scalp surface was covered. Below, the list of chosen electrodes is presented:

```
['Fp1','Fpz','Fp2',
 'AF3','AF4',
 'F7','F3','Fz','F4','F8',
 'FC5','FC1','FC2','FC6',
 'C3','Cz','C4',
 'CP5','CP1','CP2','CP6',
 'P5','P3','Pz','P4','P6',
 'POz','O1','O2']
```

Moreover, a downsampling of factor 4 was performed on every trial's signal to clean it from noise. The original signals were recorded with a sampling rate of 500 Hz, which was reduced to 125 Hz through this downsampling procedure.
It is worth emphasizing that the number of subjects analyzed in this work has been reduced from the 47 available ones to 44. Indeed, the three excluded subjects had a much smaller number of trials than the other subjects.
One last key point to underline is the limited amount of data available for each subject. In fact, there were only 224 samples per subject, equally divided between the two conditions.
To provide the network with as many samples as possible in the training set, while maintaining a reliable amount of data in both the test and validation sets, the 224 trials of a subject were split as follows:

	Split in percentage	Split in numbers
Train set	75%	167
Test set	15%	34
Validation set	10%	23

Table 1: Dataset split parameters

After the tests, the batch size for the training set was set to 64, while both the test and validation sets were analyzed in a single batch because of their reduced dimensions.

Classifiers

To solve the presented classification problem, three GNN architectures were designed, each of which was characterized by a different graph convolutional layer:

- The graph-SAGE layer is chosen because it is based on spatial convolutions, which are theoretically more suitable for exploiting the topological information of the analyzed graphs.

- Graph Isomorphism Network (GIN) layer, whose use is motivated again for its spatial nature and for being, at least in theory, one of the best performing GNNs
- The graph convolution network (GCN) layer is one of the basic spectral convolutional operators. Chosen to allow the comparison between spatial and spectral convolutions

The three networks definitely share an extremely similar structure, which will be described later on, and differ mainly in the graph convolutional layer only.

It has been chosen to design multiple GNN architectures for two main reasons: first, using different architectures, and in particular different graph convolution operators, it has been possible to test the effectiveness of this methodology applied to electroencephalographic data; then, it has allowed the comparison between these known frameworks for graph convolutions through the evaluation of their performances in this classification task.

Chronologically, the first designed architecture was based on the GraphSAGE layer. Below are the main features of this classifier's structure, chosen after some tests:

- double Graph-SAGE layer for feature aggregation with *mean* aggregator
- to introduce nonlinearities to the aggregated features returned by each convolutional layer, a ReLU activation function is applied.
- to obtain a graph-level embedding, node-level embeddings are aggregated using the global add pooling readout function
- the obtained graph features are fed into a multilayer perceptron composed of three dense layers interspersed with dropout layers.
- last dense layer's output is fed into the sigmoid activation function

Moreover, the following table reports the chosen hyperparameters:

Hyperparameters	**Values**	**Hyperparameters**	**Values**
Learning rate	0.00007	Loss function	Binary Cross Entropy
Optimizer	RMSProp	Dense Layers output dimension	100 - 16
Dropout	0.2	Embedding size	150

Table 2 Graph-SAGE based classifier hyperparameter set

The other two architectures, GCN- and GIN-based ones, have been designed starting from the same structure of the Graph-SAGE-based one, just substituting the convolutional layer with their respective ones. Moreover, some small modifications have been performed in some cases to improve performance; for instance, to reduce overfitting, in the GIN-based architecture, the layers of the MLP have been reduced from three to two.

As anticipated, despite the many attempts that have been made to find a fixed set of hyperparameters that would guarantee intersubject stability using these two architectures, this was not possible. Indeed, the performances of the architectures developed based on GCN and GIN layers using fixed hyperparameters were characterized by excessive intersubject variability in terms of classification accuracy. However, this is a common and well-known problem for electroencephalographic data.

Thus, to obtain reliable accuracy values on every subject, improving the architectures' performances, an optimization of the hyperparameters has been performed, aimed at finding the best hyperparameter set for each subject.

This has been done using Weights and Biases, a tool for performance visualization and experimental tracking of machine learning models.

Even if the GraphSAGE-based classifier stability across subjects was reached with fixed hyperparameters, this architecture was also optimized, leading to a great improvement in the results.

The optimization procedure practically consists of repeating the training on the same data multiple times, choosing a different set of hyperparameters each time according to a certain strategy (random, grid, Bayesian). Before the starting of the procedure, it is only necessary to prepare a configuration dictionary in which for each parameter to be optimized is specified a list of possible values or an interval in which to search for its best.

Ultimately, the model obtaining the best performance according to the chosen metric (for instance, validation loss) can be picked and tested.

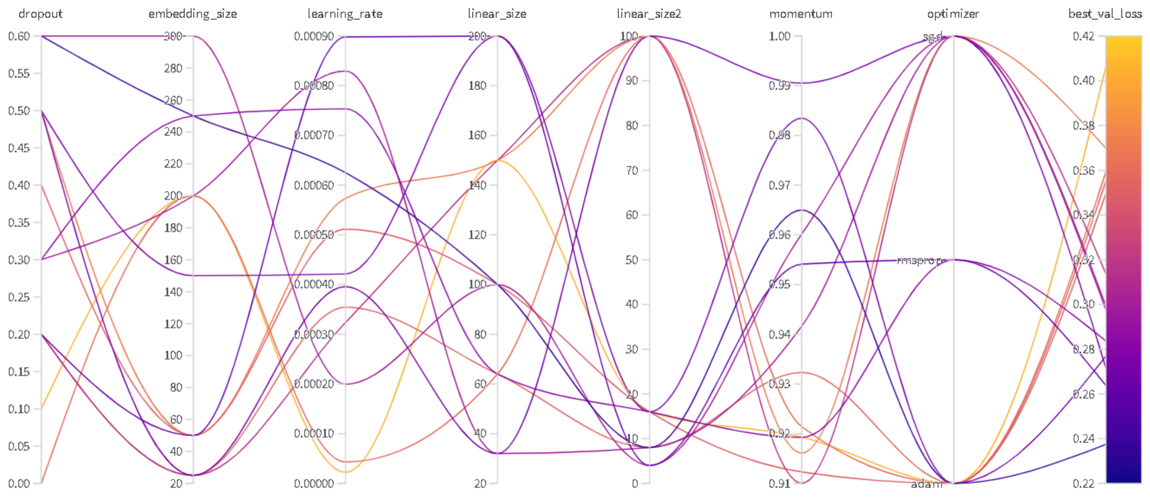


Figure 9: Graphical representation of the multiple training attempts with different hyperparameter choices aimed at minimizing the validation loss

The whole architecture of the proposed classifiers can be summarized, starting from input data to classification output, in the figure below:

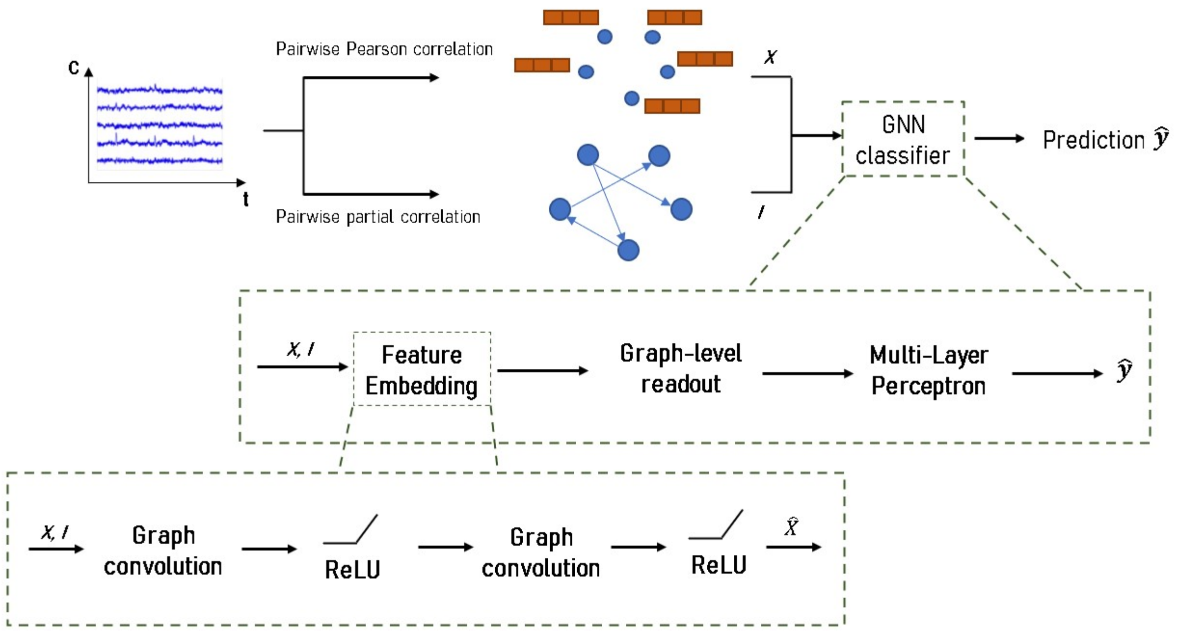


Figure 10: Complete architecture of the proposed classification pipeline

To validate the GNNs' performances, two machine learning baselines have been implemented and tested: SVM and random forest. To guarantee the most reliable comparison, these two algorithms were provided with the same set of features that constituted the input of the GNNs. To improve the performances of these methods, an empirical optimization of their hyperparameters was performed:

- For SVM, a *linear* kernel was used, and the C value was set to 1. Experiments have also been carried out using the *rbf* kernel, which has the worst performance.
- For random forest, the number of decision trees to use was set to 100.

Moreover, I will show in the "Results" chapter that these baselines have also been used for testing the meaningfulness of some features, such as PLV and the raw EEG signals.

Graph structure learning experiments

The next step has been to integrate the aforementioned task-conditioned graph structure learning mechanism, based on Jia et al. [28], in the classification process thus constructed. Indeed, using fixed, a priori calculated adjacency matrices as input to the networks introduces a great deal of arbitrariness into the training process and potentially reduces its interpretability. Therefore, at least theoretically, equipping the networks with a mechanism that is able to automatically learn how nodes are connected within the input graphs should be the best strategy.

The use of such a mechanism implies that, at each training epoch, the edges between the nodes, according to which the graph convolutional layers aggregate node features, are learnt thanks to the updating of certain weights on which the classification performance also impacts. Consequently, the structure of the graph that has been learnt can be interpreted as the structure that the network has 'deemed' best to solve the classification task.

The mechanism learns one different connectivity matrix for each of the training instances, in this case, one for each trial. Thus, combining the matrices obtained for all the instances belonging to the same class or task, for instance, by computing their average, it is possible to determine a unique "task-conditioned" matrix that is representative of that task or class.

As previously mentioned, even if this method was initially designed for the sleep stage classification task, in which the different conditions are characterized by very different brain connectivity maps, its inclusion within the task presented here remains of extreme interest given the interpretability it provides regarding the behavior of the neural network. Indeed, the representative adjacency matrices obtained for each condition can be studied to better understand the classification results. For instance, it would be possible to analyze them from a physiological point of view, trying to identify in them some patterns or structures that are expected for the condition to which they belong.

After the resolution of a few implementation issues related to the insertion of the graph structure learning layer within the classification pipeline, the tuning of the method was carried out:

- In the original paper, the node features fed into the network were represented by EEG signals' differential entropy computed at different frequency bands. Tests about this approach applied to this classification task did not give appreciable results. Thus, the same Pearson correlation-based features that were used in the "standard" classification process were chosen.
- An important role in the stabilization of the adjacency matrices learning process was played by tuning the regularization parameter λ. Through empirical tuning, it was initially set to 10^{-3}. Successively, hyperparameter optimization confirmed that values around 10^{-3} allow to learn biologically plausible matrices.

As underlined in the paragraph that introduces this method, the normalization strategy chosen by the authors fixes the sum of each row of the matrix at one. This choice, however, introduces a purely artifactual asymmetry that breaks the conceptual symmetry of the method. Therefore, in this work, the use of a new normalization strategy, which fixes the sum of the entire matrix at one, thus maintaining the symmetry, was tested.

The GNN architecture in which the use of this mechanism has been tested is the classifier based on the GCN layer. Indeed, even if it is the simplest graph convolutional operator among the chosen ones, it is the only one allowing the use of edge weights. This implies both to not waste any information learned about the matrices and to eliminate the element of arbitrariness that is introduced by binarizing matrices with a chosen empirical threshold.

It is worth emphasizing that, as mentioned earlier, one possible application of graph learning is edge prediction. However, this could require having a ground truth value for learning. In contrast, the learning pipeline used here results in a semisupervised approach, in which the classification part is carried out in a supervised fashion; instead, since no ground truth is available, the graph structure learning is unsupervised and completely based on input features.

The "standard" classification pipeline shown in Figure 10 is modified by the addition of this graph structure learning mechanism as follows:

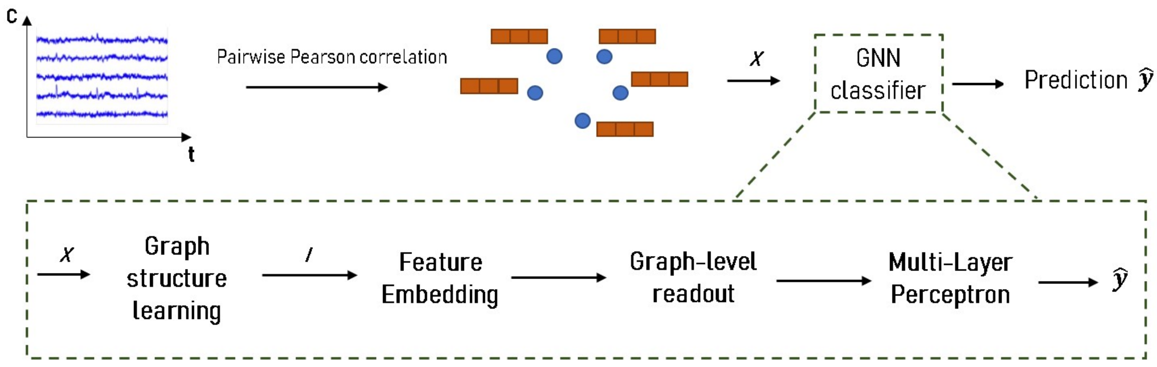


Figure 11: Complete architecture of the proposed classification pipeline with the addition of the graph structure learning mechanism

Results

In the following, I will show and compare the results obtained by the developed methods while solving the presented classification task, with particular emphasis given to their average accuracy across-subjects.

Machine learning baselines

It is crucial, when presenting the application of a method, to also provide a baseline against which the proposed approach can be compared to assess its actual effectiveness. In this work, as anticipated, two machine learning methods were chosen: SVM, with both *linear* and *rbf* kernels, and random forest.

To provide the most reliable comparison with the results obtained with the graph neural networks, in both cases, I used the same set of input features.

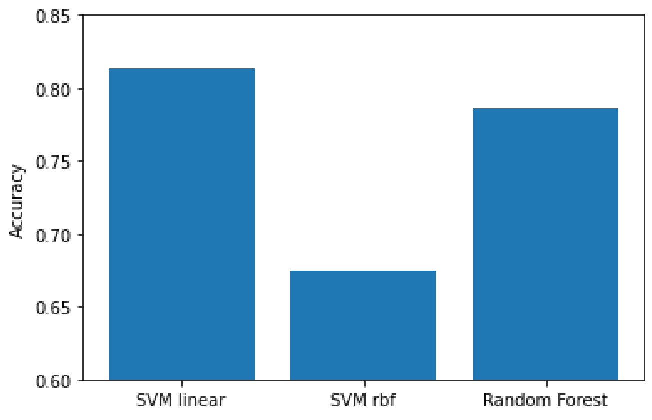


Figure 12: Average accuracy of machine learning methods given the Pearson correlation based features

The histogram above shows the performances obtained by the three baseline methods provided with the Pearson correlation-based features, which are the same input of the GNNs. The reached average accuracy across subjects of the three baselines is 75.8%, which is a quite good result, especially if considering the reduced conceptual and computational complexity of the methods.

Given the appreciable results obtained by the baselines, subsequent tests were aimed at investigating whether they were due to the methods or to the significance of the extracted features. Therefore, a first experiment was to input raw (unprocessed) EEG signals to the three baselines.

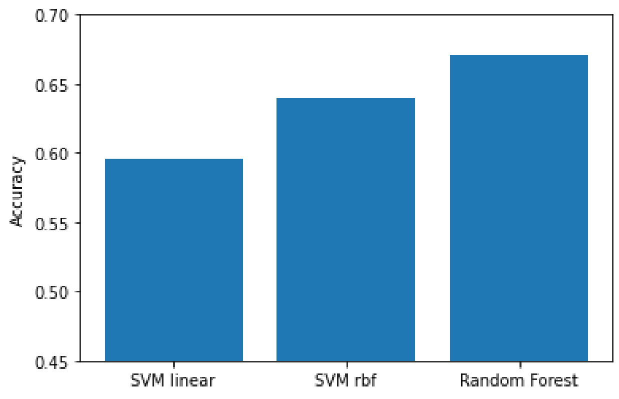


Figure 13: Average accuracies of machine learning methods given the raw EEG signals

As it is possible to see from the above histogram, using the raw signals as input has considerably reduced the baselines' performances, leading to an average accuracy between the three methods of 63.5%, more than 10% less with respect to before. This partially ensured the good level of expressiveness of the chosen features.

However, another more accurate test was carried out: it was chosen to provide the algorithms with features that are conceptually similar to the Pearson correlation-based ones but computed with the phase locking value method. Indeed, for a given node, they both give information about how strong the connection with the other nodes is.

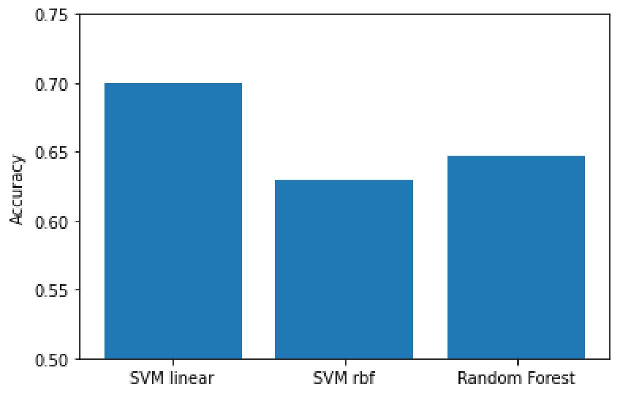


Figure 14: Machine learning baseline average accuracies given phase locking value features

With an average accuracy of 65.5% across the baselines, the use of these features slightly improved the raw signal-related results, which were not even close, however, to those obtained with Pearson correlation-based features. This confirmed their meaningfulness.

Below is a graphical comparison of the results of the three baselines given the three different input features:

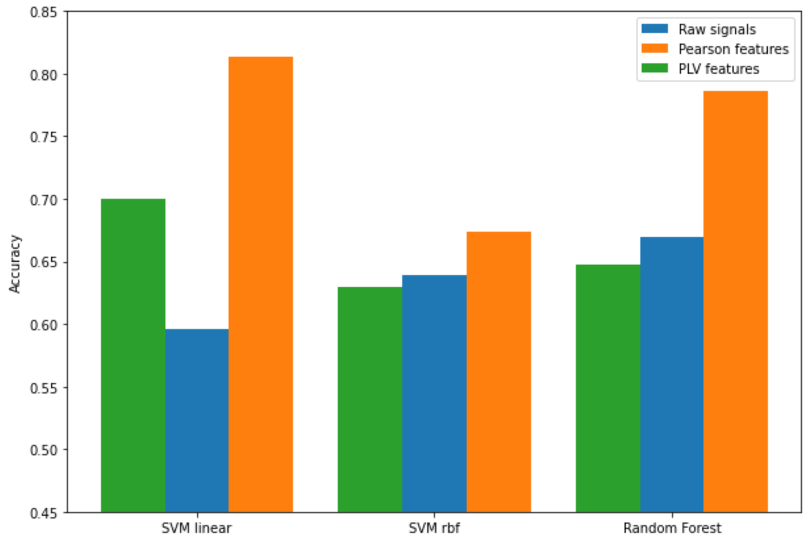


Figure 15: Machine learning methods feature comparison

Moreover, because of the extreme intersubject variability that characterizes physiological data, it is also interesting to look at the accuracy distributions obtained by the methods on the different subjects. Below will be presented two scatter plots representing the accuracy values obtained with the best performing features (Pearson-based) and, in particular, with the two methods providing the highest average accuracies (linear SVM and random forest):

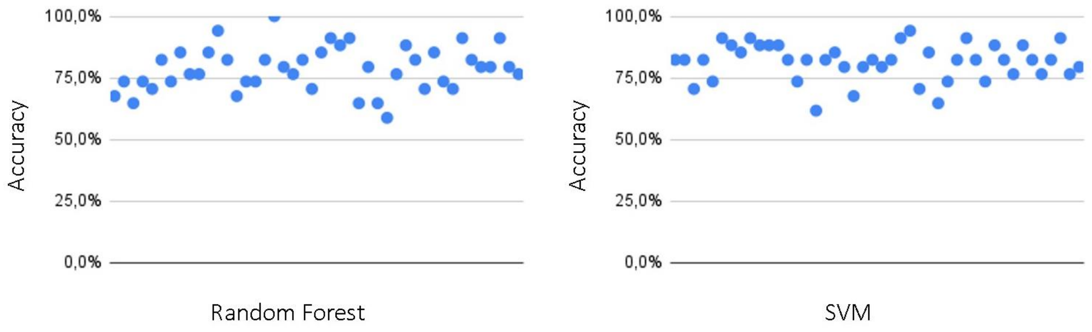


Figure 16: Accuracy distribution over subjects obtained with linear SVM and random forest using Pearson correlation based features

Graph Convolutional Neural Networks

As previously anticipated and described in the "Experiments" chapter, to solve the chosen classification task, I used three classifiers, each based on a different graph convolutional operator. These GNN architectures, whose details have been provided above, have been initially trained following a "standard" approach, in which the information about graphs' topology, in particular the edges linking the graphs' nodes, are fixed and computed a priori.

In this chapter, the GNN results obtained using Pearson correlation-based features, which have been the best performing among the tested features, will be shown.

The classifier with the best results was based on the graph-SAGE convolutional operator. Moreover, as mentioned, it has been the only one with good performance stability across subjects with a fixed set of hyperparameters. However, the optimization process has still been performed, allowing further improvement of its performance:

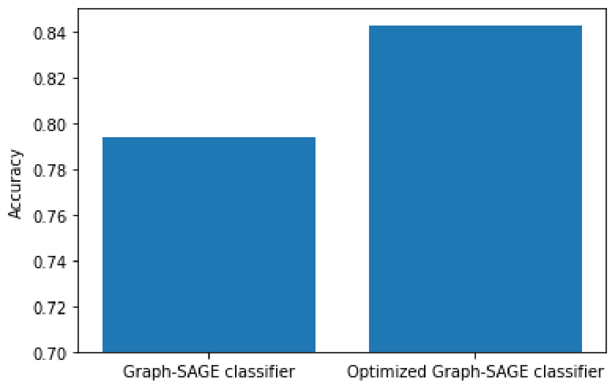


Figure 17: Average accuracies reached by the Graph-SAGE classifier with and without hyperparameter optimization

As it is possible to see from the above histogram, the results obtained with the classifier without hyperparameter optimization, with an average accuracy across subjects of 79.4%, are even lower than those obtained with the SVM baseline. The optimization process improved the performance by 4.9%, leading to an average accuracy of 84.3%, which overcomes all the baseline results.

As done before for the baselines, it is useful to look at the accuracy distributions over subjects both for the optimized and nonoptimized graph-SAGE-based classifiers:

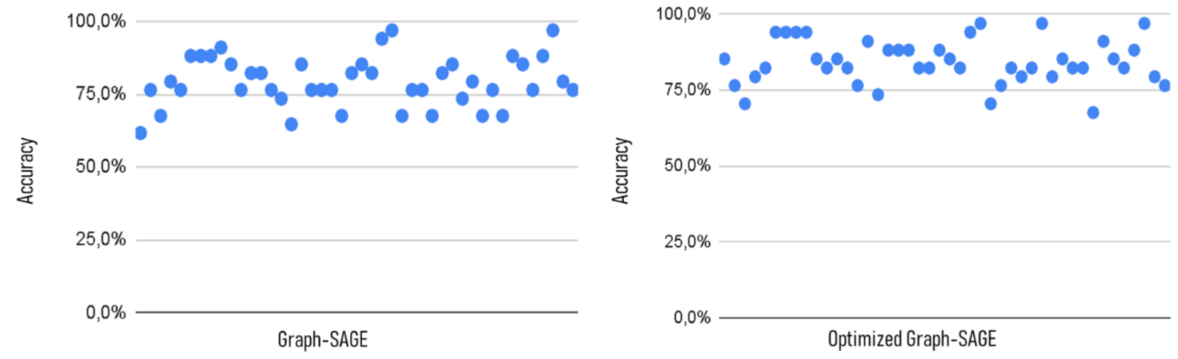


Figure 18: Accuracy distribution over subjects obtained with the optimized and nonoptimized versions of the Graph-SAGE classifier

From the scatter plots, it is possible to capture not only the lower results in terms of average accuracy but also, as predicted, the highest performance variability among subjects reached with the nonoptimized version of the classifier. Indeed, the standard deviation of the accuracies across subjects obtained with the standard version of the classifier is 8.5% with respect to the 7.3% obtained after optimization. In general, all three GNN architectures have achieved similar accuracy results, all contained in the range between the 80-85% of average across subjects:

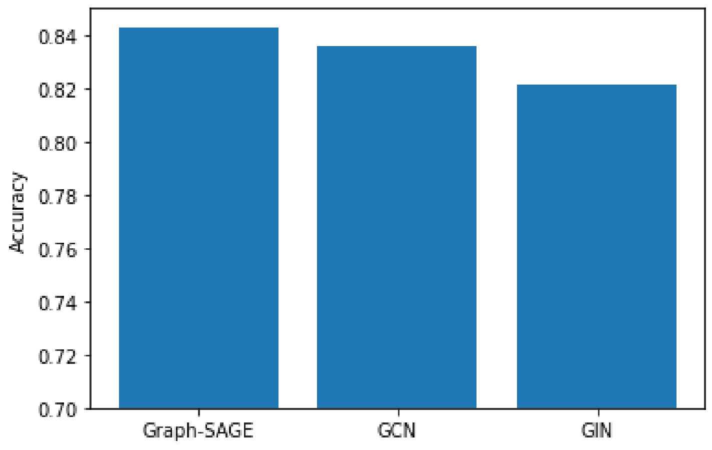


Figure 19: Average accuracy across subjects reached by the three classifiers

As mentioned, although the networks work extremely well on some subjects, even reaching 100% accuracy in certain cases, the extreme intersubject variability that characterizes electroencephalographic data is one of the main causes of such fluctuating results. Furthermore, another factor to be considered is the difficulty of the chosen classification task. In fact, although the social component certainly plays an important role in the experiment during which the data were collected, regardless of the sociality of the stimulus, the participants always had to perform the same working memory task, which made it challenging to discriminate the two conditions. As proof of this, even in the original article [14] presenting the dataset, from a study conducted on the cueing phase, the same analyzed in this thesis work, no significant differences were found in terms of behavioral data depending on the type of cue shown in the trials.

However, all the analyzed GNN architectures outperform the chosen machine learning baselines:

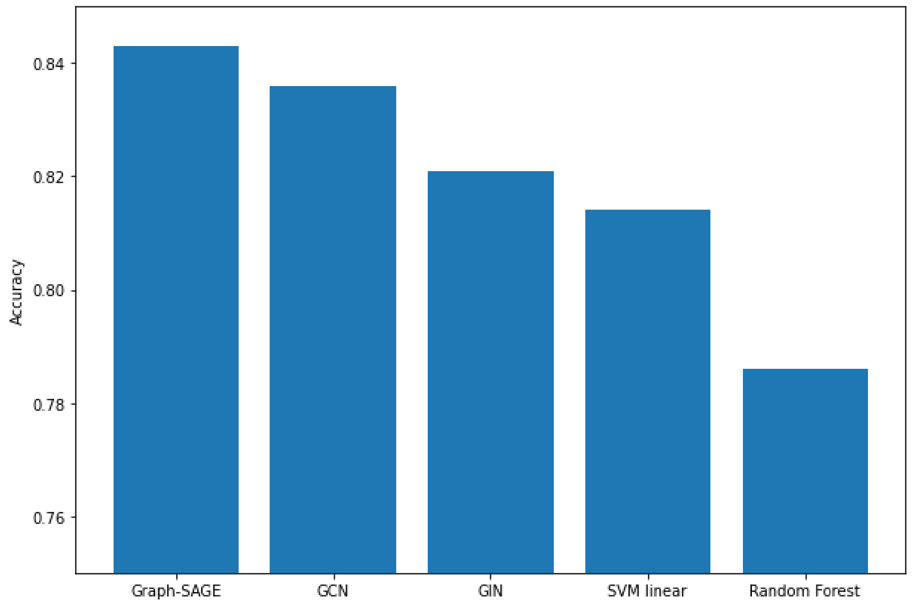


Figure 20: Average accuracy comparison between GNN architectures and the machine learning baselines

As anticipated, the graph-SAGE-based classifier achieves the best accuracy result, perhaps due to the spatial nature of the convolution it performs. From this point of view, it has been slightly unexpected to see the lowest results obtained with the GIN-based classifier, both for its theoretical power and for its spatial-based convolution operator.

Note, as mentioned, that the results of each GNN architecture shown in Figure 20 have been obtained after hyperparameter optimization.

A fundamental aspect to be underlined is the extremely low computational effort needed to complete the training using these architectures. Indeed, they all need a maximum time of 3 minutes to complete the training procedure on one subject's data.

To allow and facilitate more precise numerical comparisons between the presented results, below it is shown a table summarizing all the results obtained with the three GNNs and with the baselines methods, using Pearson correlation features as input:

Model	**Accuracy (%)**
Random Forest	78.6 ± 8.9
SVM	81.4 ± 7.5
GIN	82.1 ± 7.9
GCN	83.5 ± 8.0
Graph-SAGE	**84.3 ± 7.3**

Table 3: Average accuracy across subjects obtained with the presented methods. The best result is shown in ***bold.***

Analysis of graph structure learning results

As mentioned, the latest experiments involved solving the chosen classification problem through the use of a classifier equipped with a mechanism that, in parallel with classification, automatically learns the graph structures. In the following, this classifier is referred to with the name GCN*.

From the point of view of classification accuracy, the results obtained are inferior to those achieved with classifiers trained in a "standard" manner. Indeed, although the automatic learning of the connectivity matrices is theoretically the best strategy, as it eliminates external arbitrariness factors, it greatly increases the complexity of the training process, often leading to a lack of convergence.

However, the results obtained, with an average accuracy across subjects of 78.1%, remain comparable with the previous ones.

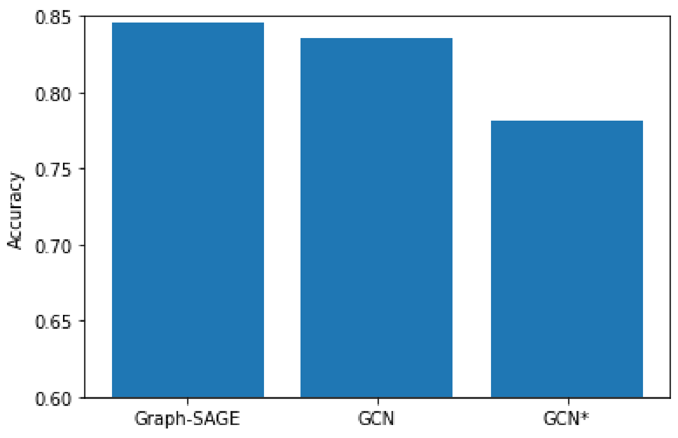


Figure 21: Comparison of the accuracies obtained with: the best performing GNN (graph-SAGE based), GCN-based trained in "standard" modality and GCN, which is the GCN-based classifier equipped with the graph-structure learning mechanism*

It should be emphasized that in the results obtained with the GCN* classifier, only subjects on which it was possible to learn noncompletely flat connectivity matrices were considered, reducing the number of subjects analyzed from 44 to 33.

Indeed, as mentioned, the complexity and instability of the training procedure are considerably increased by the use of this mechanism, sometimes causing the learning algorithm to be unable to achieve convergence. This primarily leads to the learning of totally flat adjacency matrices, which is also followed by a drastic drop in classification performance. A possible solution to this problem, which has also been attempted in this thesis without succeeding on all subjects, is a very thorough optimization of the training hyperparameters. Clearly, this tuning procedure must be repeated for each subject, looking for its specific set of best hyperparameters.

Moreover, in this case, it is possible to look at the results not only in terms of classification accuracy but also in terms of the learned adjacency matrices. Below are the average matrices across all subjects learned for the two conditions (social cue and nonsocial cue).

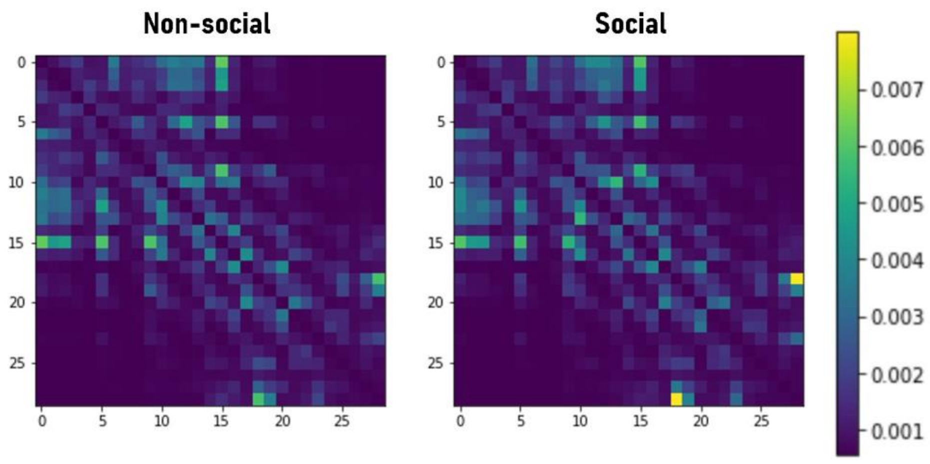


Figure 22: Average across subjects of the learned adjacency matrices for the two discriminated conditions

In the above matrices, the electrodes are positioned in the same order of the list shown in the previous chapters. Therefore, in the left-up corner, it is possible to find electrodes of the brain frontal area; in the right-bottom corner, instead, the electrodes of the brain parietal-occipital area are located.

From a qualitative point of view, two main points have been identified:

- First, the two matrices are very similar. Actually, this is not particularly surprising, since the two conditions just differ for the social content of the presented stimulus, which makes them particularly challenging to be discriminated. Moreover, the intrinsic inter-subject variability makes a group analysis less informative. For this reason, in the following I focused on analysis at the single subject level.
- In both matrices the frontal and parietal-occipital areas define two clusters of strong intraconnections. This is in accordance with previous studies on Working Memory tasks [32] and with results presented in the original paper introducing the dataset [13].

The differences between the two average matrices learned for the two conditions, even if small, are shown in the comparison matrix below, normalized by means of the ratio between their elementwise difference and sum:

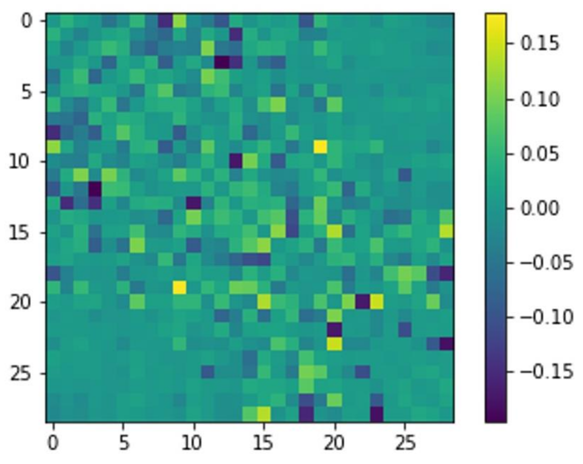


Figure 23: Comparison matrix between the average learned matrices in the nonsocial and social conditions

In addition, I performed quantitative and statistical tests to assess the validity of the learned adjacency matrices. These have been done using the subject-specific learned matrices in the two conditions and not their average, which are the ones shown above. In particular, an analysis of integration and segregation between two pairs of graph classes was performed: the two hemispheres (left and right side of the brain) and the frontal and parietal-occipital lobes. The indices that I have computed considering them as classes are the ones previously described in the chapter about graphs' theory: sum of intra- and inter-connections weights, modularity and divisibility.

In addition to calculating these indices on the subject-dependent adjacency matrices for both the social and nonsocial conditions, to investigate if the differences detected by the GNN between the social and nonsocial matrices were significant, index computation was also performed on the absolute value of their difference matrices. In the following, I will refer to this matrix as the "difference matrix".

After computing the mentioned indices, I carried out statistical tests of two types:

- Paired T test between the indices distributions over subjects of the social and nonsocial conditions, looking for physiological differences enhanced by the graphs' indices.

- Correlation of the indices computed on the three kinds of matrices (social, nonsocial, difference) with subjects' behavioral data, with the aim of searching for network properties directly linked with the subjects' behavior. In particular, the analyzed behavioral data were the reaction times and the accuracies of the answers to the question provided at the end of the data collection experiment.

Indices computed on social matrices have been correlated with behavioral data of the social condition, the ones computed on nonsocial matrices with the nonsocial behavioral data, and the ones calculated on the difference matrix with the absolute value of the difference between the behavioral data in the two conditions.

Two correlations returned a significant result: those between the absolute value of the difference between reaction times (as introduced in the "Dataset description" chapter) and the divisibility computed on the difference matrices, considering the two hemispheres (r = -0.338; p=0.05) and the frontal and parietal-occipital lobes (r = -0.473; p=0.004), respectively, as classes. This latter result is particularly relevant, as a strong significance value is associated with a physiologically meaningful index, i.e. a measure of the segregation between the frontal regions and the parieto-occipital ones.

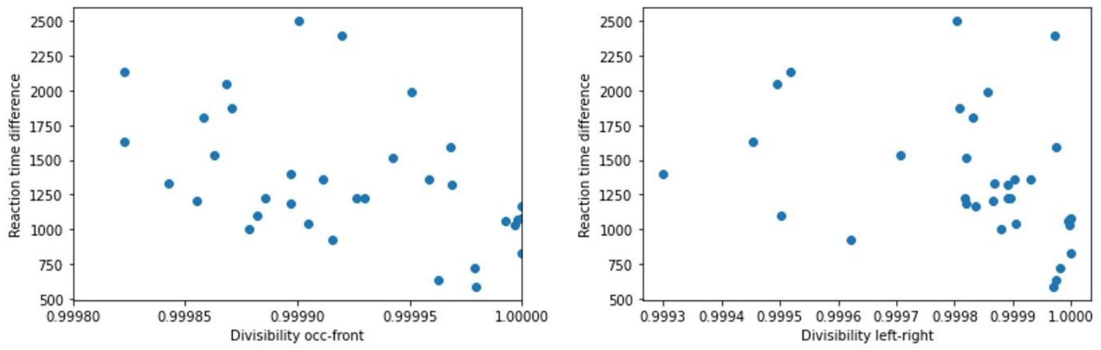


Figure 24: Scatter plots of the distributions of the two significant correlations. On the left, the one between reaction time differences and divisibility between parietal-occipital and frontal classes (r = -0.473; p = 0.004). On the right the plot of the correlation between reaction time differences and the divisibility between left and right hemispheres (r = -0.338; p = 0.05)

This reveals how the variations between the matrices learned by the automatic graph-structure learning mechanism for the two conditions are correlated with variations of the behavioral data, specifically with a reduction of the difference between the reaction time between the social and nonsocial conditions. This result indicates that when the difference matrices are more integrated (stronger communication between the frontal and parieto-occipital lobes, i.e. a lower divisibility between the related classes) there is a stronger difference between reaction times, suggesting that the social content of the stimulus affects the subject's performances. On the contrary, when the two matrices do not differ in terms of fronto-posterior integration (high divisibility) the subject's performances do not differ between the social and nonsocial conditions (small difference between reaction times). This result is physiologically meaningful and it is in line with the results shown in the original paper introducing the dataset, which shows an involvement of frontal and parieto-occipital electrodes. At the same time, it goes beyond the state-of-the-art by providing a measure based on the properties of the brain networks (and not just on the activations) able to explain and characterize the difference between a social and a nonsocial cue in a working memory task. Finally, the scatter plot also shows the inter-subject variability which is inherent to any task with a social content.

Conclusion

In this work, I examined the application of graph learning, particularly graph convolutional neural networks, to human electrophysiological data gathered during a cognitive task. Specifically, a classification problem focused on the social component of the experiment carried out for data collection was solved.

The novelty of this work lies in three aspects:

1. The choice of expressive and meaningful node features that characterize the nodes not by their intrinsic properties, but rather by information about how they are connected with other nodes. In particular, this information is computed through pairwise Pearson correlation between their EEG signals. This choice has proven to be effective in terms of classification accuracy, leading to the best obtained results. In addition, it is also useful while automatically learning the graph-structures, providing the mechanism with features that are already informative about how nodes are connected.
2. The insertion, in the proposed classification pipeline, of a framework for automatic graph-structure learning, with the aim of improving the interpretability of the obtained results. In particular, the aforementioned framework has been conceptually modified and adapted to the task studied here. Furthermore, differently from its original formulation, that used the learned matrices just for a visual and qualitative inspection, in this thesis they have been validated through statistical and quantitative tests related to their physiological meaning
3. The properties of the proposed approach (in terms of accuracy and interpretability) allowed its application to a particularly challenging dataset, that differs significantly from the classification of brain disorders, or of very distinctive motor tasks, that were the target of the previous literature on the topic

The results obtained, validated through the use of the most common baselines in the field of artificial intelligence, showed a good ability of the method to discriminate two conditions that are extremely challenging and variable across subjects, with a reduced computational effort. Future developments could include the use of more complex architectures to further improve the classification performance; for instance, spatial-temporal graph convolutional networks, which are sometimes used in EEG analysis, could represent a possible solution.

The automatic matrix learning mechanism showed very interesting insights from the point of view of the interpretability it adds to the classification process. Future efforts should be focused first on refining the learning of these brain connectivity maps, allowing us to carry out more precise and deep quantitative studies.

References

[1] Daoud, Hisham, and Magdy A. Bayoumi. "Efficient epileptic seizure prediction based on deep learning." IEEE transactions on biomedical circuits and systems 13.5 (2019): 804-813.

[2] Aboalayon, Khald Ali I., et al. "Sleep stage classification using EEG signal analysis: a comprehensive survey and new investigation." Entropy 18.9 (2016): 272.

[3] Chen, He, Yan Song, and Xiaoli Li. "A deep learning framework for identifying children with ADHD using an EEG-based brain network." Neurocomputing 356 (2019): 83-96.

[4] Amrani, Ghita, et al. "EEG signal analysis using deep learning: A systematic literature review." 2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS). IEEE, 2021.

[5] Demir, Andac, et al. "EEG-GNN: Graph Neural Networks for Classification of Electroencephalogram (EEG) Signals." 2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC). IEEE, 2021.

[6] Nunez, Paul L., et al. "EEG coherency: I: statistics, reference electrode, volume conduction, Laplacians, cortical imaging, and interpretation at multiple scales." Electroencephalography and clinical neurophysiology 103.5 (1997): 499-515.

[7] Granger, Clive WJ. "Investigating causal relations by econometric models and cross-spectral methods." Econometrica: journal of the Econometric Society (1969): 424-438.

[8] Baccalá, Luiz A., and Koichi Sameshima. "Partial directed coherence: a new concept in neural structure determination." Biological cybernetics 84.6 (2001): 463-474.

[9] Antonacci, Yuri, et al. "Estimation of brain connectivity through artificial neural networks." 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2019.

[10] Faes, Axel, Iris Vantieghem, and Marc M. Van Hulle. "Neural Networks for Directed Connectivity Estimation in Source-Reconstructed EEG Data." Applied Sciences 12.6 (2022): 2889.

[11] Zhdanov, Maksim, Saskia Steinmann, and Nico Hoffmann. "Investigating Brain Connectivity with Graph Neural Networks and GNNExplainer." arXiv preprint arXiv:2206.01930 (2022).

[12] Ying, Zhitao, et al. "Gnnexplainer: Generating explanations for graph neural networks." Advances in neural information processing systems 32 (2019).

[13] Gregory, Samantha EA, Hongfang Wang, and Klaus Kessler. "A dataset of EEG recordings from 47 participants collected during a virtual reality working memory task where attention was cued by a social avatar and nonsocial stick cue." Data in Brief 41 (2022): 107827.

[14] Gregory, Samantha EA, Hongfang Wang, and Klaus Kessler. "EEG alpha and theta signatures of socially and nonsocially cued working memory in virtual reality." Social Cognitive and Affective Neuroscience 17.6 (2022): 531-540.

[15] Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik. "A training algorithm for optimal margin classifiers." Proceedings of the fifth annual workshop on Computational learning theory. 1992.

[16] Ho, Tin Kam. "Random decision forests." Proceedings of 3rd international conference on document analysis and recognition. Vol. 1. IEEE, 1995.

[17] Kelleher, John D. Deep learning. MIT press, 2019.

[18] Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." IEEE transactions on neural networks and learning systems 32.1 (2020): 4-24.

[19] Gori, Marco, Gabriele Monfardini, and Franco Scarselli. "A new model for learning in graph domains." Proceedings. 2005 IEEE international joint conference on neural networks. Vol. 2. No. 2005. 2005.

[20] Bronstein, Michael M., et al. "Geometric deep learning: going beyond euclidean data." IEEE Signal Processing Magazine 34.4 (2017): 18-42.

[21] Bruna, Joan, et al. "Spectral networks and locally connected networks on graphs." arXiv preprint arXiv:1312.6203 (2013).

[22] Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." Advances in neural information processing systems 29 (2016).

[23] Masci, Jonathan, et al. "Geodesic convolutional neural networks on riemannian manifolds." Proceedings of the IEEE international conference on computer vision workshops. 2015.

[24] Kipf, Thomas N., and Max Welling. "Semisupervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).

[25] Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." Advances in neural information processing systems 30 (2017).

[26] Xu, Keyulu, et al. "How powerful are graph neural networks?." arXiv preprint arXiv:1810.00826 (2018).

[27] Weisfeiler, Boris, and Andrei Leman. "The reduction of a graph to canonical form and the algebra which appears therein." NTI, Series 2.9 (1968): 12-16.

[28] Jia, Ziyu, et al. "GraphSleepNet: Adaptive Spatial-Temporal Graph Convolutional Networks for Sleep Stage Classification." IJCAI. 2020.

[29] Fey, Matthias, and Jan Eric Lenssen. "Fast graph representation learning with PyTorch Geometric." arXiv preprint arXiv:1903.02428 (2019).

[30] Pearson, Karl. "VII. Mathematical contributions to the theory of evolution.—III. Regression, heredity, and panmixia." Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character 187 (1896): 253-318.

[31] Piqueira, José Roberto C. "Network of phase-locking oscillators and a possible model for neural synchronization." Communications in Nonlinear Science and Numerical Simulation 16.9 (2011): 3844-3854.

[32] Toppi J, Astolfi L, Risetti M, Anzolin A, Kober SE, Wood G, Mattia D. Different Topological Properties of EEG-Derived Networks Describe Working Memory Phases as Revealed by Graph Theoretical Analysis. Front Hum Neurosci. 2018 Jan 12;11:637. doi: 10.3389/fnhum.2017.00637.