SPATIOTEMPORAL GRAPH CONVOLUTIONAL NEURAL NETWORKS FOR MOTOR IMAGERY EEG CLASSIFICATION

by

Ahmed Mohamed Mohamed Mohamed Alramly

M.D., in Medicine, Mansoura University, 2014

Submitted to the Institute of Biomedical Engineering
in partial fulfillment of the requirements
for the degree of
Master of Science
in
Biomedical Engineering

Boğaziçi University
2023

SPATIOTEMPORAL GRAPH CONVOLUTIONAL NEURAL NETWORKS FOR MOTOR IMAGERY EEG CLASSIFICATION

APPROVED BY:

Prof. Dr. Ahmet Ademoğlu .....
(Thesis Advisor)

Prof. Dr. Burak Güçlü .....

Assist. Prof. Dr. İpek Şen .....

DATE OF APPROVAL: 30 December 2022

ACKNOWLEDGMENTS

First of all, I would like to thank Prof. Ahmet Ademoğlu for everything he has done for me. I still remember the first time I contacted him, he was very welcoming and kind since the first moment. When I joined the program, I did know nothing. "Ahmet Hoca" kept encouraging, guiding, and coaching me all the time to this point when I defended my thesis. I would like to thank him specifically for the very friendly environment in the lab. I enjoyed the scientific and general discussions we used to have over coffee breaks through the years.

I would like to thank all BME professors for my wonderful and enjoyable experience through the coursework of the program. I would like to especially thank Dr. İpek Şen for everything. She was an elder sister more than a professor.

I would also like to thank my labmates for the nice, and friendly lab environment. I like to give a special thanks to Hüden and Lina. We shared so many coffee breaks together and had many discussions on our round coffee table.

Finally, all the credits of what I have achieved go in the first place to my family and my friends back in Egypt. Without their support I could not have made it through the hard times of the pandemic.

ACADEMIC ETHICS AND INTEGRITY STATEMENT

I, Ahmed Mohamed Mohamed Mohamed Alramly, hereby certify that I am aware of the Academic Ethics and Integrity Policy issued by the Council of Higher Education (YÖK) and I fully acknowledge all the consequences due to its violation by plagiarism or any other way.

Name :

Signature:

Date:

ABSTRACT

SPATIOTEMPORAL GRAPH CONVOLUTIONAL NEURAL NETWORKS FOR MOTOR IMAGERY EEG CLASSIFICATION

Electroencephalography (EEG) has various applications in medicine, neuroscience, and neural engineering. It records the electrical activity of the brain tissues caused by the interaction among different neuronal communities. Numerous algorithms for the automatic classification of EEG signals have been developed. These algorithms work via extracting unique and non-redundant features from EEG signals. However, the majority of the proposed algorithms employ temporal components of EEG signals while disregarding the rich spatial network structure that underlies in the EEG. In this study, we propose a classification pipeline that uses the network structure of EEG data for a simultaneous representation of spatial and temporal features in the EEG signals. First, graph theory is utilized to model the EEG networks in two spatial domains; i) the sensor space and ii) the cortical source space. Second, a spatiotemporal graph convolutional neural network (STGCNN) classification model is employed combining both temporal and spatial features of EEG data for its classification under motor imagery conditions. Additionally, the model is tested using the cortical source space data in each of the seven resting state brain networks individually to estimate their performance on classification accuracy. The results show that STGCNN model performs slightly better than the temporal convolutional neural network (CNN) models by 1.25%.

Keywords: EEG, Spatiotemporal Graph Convolutional Neural Networks, Brain Networks.

ÖZET

MOTOR İMGELEME EEG SINIFLANDIRMASINDA UZAY-ZAMAN ÇİZGE EVRİŞİMLİ SİNİR AĞLARI

Elektroensefalografi (EEG) tıpta, nörobilimde ve nöral mühendislikte çeşitli uygulamalara sahiptir. Farklı nöron toplulukları arasındaki etkileşimin neden olduğu beyin dokularının elektriksel aktivitesini kaydeder. EEG sinyallerinin otomatik olarak sınıflandırılması için çok sayıda algoritma geliştirilmiştir. Bu algoritmalar, EEG sinyallerinden özgün ve yinelenmeyen özellikler çıkararak çalışır. Bununla birlikte, önerilen algoritmaların çoğu, EEG sinyallerinin zamansal bileşenlerini kullanırken onu üreten zengin uzamsal ağ yapısını göz ardı etmektedir. Bu çalışmada, EEG sinyallerinde uzamsal ve zamansal özelliklerin eşzamanlı kullanımı için EEG verilerinin ağ yapısını kullanan bir sınıflandırma ardışık düzeni önermekteyiz. İlk olarak, EEG ağlarını iki ayrı uzaysal alanda; i) sensör uzayı ve ii) kaynak uzayında modellemek için çizge kuramı kullanıldı. İkinci olarak, sınıflandırma işleminde EEG verilerinin hem zamansal hem de uzamsal özelliklerini birleştirmek için bir uzay-zaman çizge evrişimli sinir ağı (STGCNN) sınıflandırma modeli uygulandı. Ek olarak model, her bir dinlenme durumu beyin ağında oluşan kortikal kaynak verisi kullanılarak sınıflandırma doğruluk performansı incelendi. Çalışmanın sonuçları, STGCNN modelinin zamansal evrişimli sinir ağı (CNN) modellerine göre, % 1.25 oranında, az da olsa daha iyi performans sergilediğini göstermektedir.

Anahtar Sözcükler: EEG, Uzay-Zaman Çizge Evrişimli Sinir Ağı, Beyin Ağları

TABLE OF CONTENTS

ACKNOWLEDGMENTS . . . iii
ACADEMIC ETHICS AND INTEGRITY STATEMENT . . . iv
ABSTRACT . . . v
ÖZET . . . vi
LIST OF FIGURES . . . ix
LIST OF TABLES . . . xi
LIST OF SYMBOLS . . . xii
LIST OF ABBREVIATIONS . . . xiv
1. INTRODUCTION . . . 1
2. BACKGROUND . . . 3
2.1 Brain Computer Interface (BCI) . . . 3
2.2 EEG . . . 5
2.3 Why EEG is suitable for motor imagery BCI? . . . 6
2.4 EEG Classification Problem . . . 7
3. METHODS . . . 10
3.1 Graph Theory . . . 10
3.1.1 What are graphs? . . . 10
3.1.2 Applications of graph theory . . . 13
3.2 Modeling EEG as Graphs . . . 15
3.2.1 Why to model EEG as a graph? . . . 15
3.2.2 What are brain networks? . . . 17
3.2.3 How to model EEG data as graphs? . . . 20
3.2.3.1 Nodes . . . 20
3.2.3.2 Edges . . . 24
3.3 Graph Neural Networks (GNNs) . . . 25
3.3.1 Problem Setting . . . 26
3.3.2 Rethinking Convolutional Neural Networks . . . 29
3.3.3 From Convolution to Attention . . . 33
3.3.4 Graph Isomorphism Networks . . . 35

3.3.5 Message Passing GNNs . . . 36
3.3.6 Spatio-Temporal Graph Convolutional Neural Networks . . . 37
4. EXPERIMENTS . . . 39
4.1 Dataset . . . 39
4.2 Pipeline . . . 41
4.2.1 SVM Classification . . . 41
4.2.2 Preprocessing . . . 42
4.2.3 Nodes . . . 42
4.2.4 Edges . . . 43
4.2.5 Models . . . 43
4.2.5.1 STGCNN Model . . . 43
4.2.5.2 CNN Model . . . 44
4.2.6 Experiments . . . 46
5. RESULTS . . . 48
6. DISCUSSION . . . 54
7. CONCLUSION AND FUTURE WORK . . . 57
REFERENCES . . . 58

LIST OF FIGURES

Figure 2.1	An illustration of the components of BCI system [1].	4
Figure 2.2	The first recorded human EEG signal by Hans Berger [2].	6
Figure 3.1	The Seven Bridges Problem and Euler's Solution [3].	11
Figure 3.2	The Adjacency matrix, Degree Matrix, and the Laplacian Matrix.	12
Figure 3.3	A Caffeine molecule structure and its graph representation [4].	13
Figure 3.4	Graph representation of images [4].	14
Figure 3.5	Networks at different spatial and temporal scales as recorded by different neuroimaging methods [5]	16
Figure 3.6	Changing the order of the adjacency matrix does not change the structure of the graph.	17
Figure 3.7	Different data acquisition and analysis pipelines lead to different kinds of networks [3].	19
Figure 3.8	Schaefer's brain atlas [6].	21
Figure 3.9	Scalp EEG lacks the anatomical precision of the cortical EEG.	22
Figure 3.10	Functional brain networks defined by absolute Pearson correlation in different spaces.	23
Figure 3.11	Structural Networks defined on scalp EEG sensors and on cortical level in Schaefer's atlas	26
Figure 3.12	GNNs learn permutation equivariant representations in node-level tasks and invariant representations in graph-level tasks [7].	29
Figure 3.13	Regular convolution on images [7].	30
Figure 3.14	Attention mechanism on graphs [8].	34
Figure 3.15	The message passing scheme for GNNs [9].	37
Figure 3.16	An illustration of STGCNN layer.	38
Figure 4.1	EEG electrode positions [10].	40
Figure 4.2	STGCNN model architecture (n refers to the number of nodes).	44
Figure 4.3	CNN model architecture (n refers to the number of nodes).	45
Figure 4.4	A summary of the classification pipeline.	47

Figure 5.1 A summary of STGCNN and CNN accuracy results in both sensor and source spaces. 51

LIST OF TABLES

Table 4.1	STGCNN model architecture(n refers to the number of nodes).	44
Table 4.2	CNN model architecture (n refers to the number of nodes).	45
Table 4.3	STGCNN and CNN models hyperparameters configuration.	46
Table 5.1	SVM Classification Accuracy Results.	48
Table 5.2	Classification accuracy percentages in the sensor space (5-fold cross validation).	49
Table 5.3	STGCNN classification accuracy percentages on raw data in the source space (average parcellation).	50
Table 5.4	STGCNN classification accuracy percentages on raw data in the source space (sum parcellation).	50
Table 5.5	CNN classification accuracy percentages on raw source data.	51
Table 5.6	STGCNN classification accuracy percentages on delta band data in the source space (average parcellation).	52
Table 5.7	STGCNN classification accuracy percentages on delta band data in the source space (sum parcellation).	53
Table 5.8	CNN classification accuracy percentages on delta band source data.	53

LIST OF SYMBOLS

G	A graph
N	The set of nodes of a graph
E	The set of edges of a graph
A	The adjacency matrix
D	The degree matrix
L	The Laplacian matrix
U	The eigenvectors of the Laplacian matrix
Λ	The eigenvalues of the Laplacian matrix
Y	Scalp EEG data
H	The forward model
S	Source EEG data
ϵ	Error vector
$\hat{S}$	Estimate of the source EEG data
T	The solution matrix for the inverse problem
T_{wMNE}	minimum norm estimate of T
R	The covariance matrix of the source EEG data
C	The covariance matrix of the scalp EEG data
λ	Regularization parameter
ρ_{xy}	The Pearson correlation coefficient between x and y
cov	The covariance function
σ	The standard deviation function
μ	The mean of a vector
A_s	The structural connectivity matrix
f_{ij}	The number of white matter streamlines between areas i and j
X	Features matrix of a graph G
d	The dimensionality of X / the degree of the Laplacian polynomial
n_i	A target/receiver node
n_j	A source/sender node

x_i	The feature vector of the target/receiver node n_i
x_j	The feature vector of the source/sender node x_j
e_{ij}	The weight of the edge between n_i and n_j
P	A permutation function
M	An image matrix
m_{ij}	The value of the pixel ij
g	A learnable filter on images
$P_w(L)$	A polynomial of the Laplacian matrix
w	The coefficients of the polynomial of the Laplacian matrix
$\hat{X}$	The learned embedding of the graph feature matrix X
$\hat{x}_i$	The learned embedding of the feature vector of the node x_i
Z	The Laplacian polynomials as in ChebNet
W	A learnable weight matrix in GNNs
$\mathcal{N}(i)$	The degree of node i
α_{ij}	The attention coefficient between x_i and x_j
a^T	A learnable linear transformation to learn α_{ij}
β_i	A gating parameter in GT
h_θ	A differentiable learnable function
k	A learnable temporal convolution kernel

LIST OF ABBREVIATIONS

BCI	Brain Computer Interface
BMI	Brain Machine Interface
EEG	Electroencephalography
MI	Motor Imagery
GNN	Graph Neural Networks
STGCNN	Spatio-temporal Graph Convolutional Neural Networks
GCN	Graph convolutional networks
GAT	Graph attentional networks
GIN	Graph isomorphism networks
GT	Graph transformers
CNN	Convolutional Neural Networks
DL	Deep Learning
BOLD	Blood Oxygen Level Dependent
fMRI	functional Magnetic Resonance Imaging
RSFC	Resting-state functional connectivity

1. INTRODUCTION

Electroencephalography (EEG) has many applications in medicine, neuroscience, and neural engineering. In medicine, EEG is the go-to choice in diagnosis of epilepsy and assessing mental states in patients with deep coma. In neuroscience, EEG is widely used in sleep research. It is also a fundamental tool in studying cognitive functions such as memory, vision, and decision making. In neural engineering, EEG is the main non-invasive method used for developing brain computer interface devices (BCI). It is widely adopted in emotion recognition BCI applications and in motor imagery BCI applications for robotic limbs to aid in patient rehabilitation.

There are many algorithms developed for the purpose of automatic EEG classification. These algorithm work by extracting distinctive non-redundant features from EEG time series then feeds those features into a machine learning classification algorithm. The majority of these algorithms utilize the temporal components of EEG time series for feature extraction. However, EEG signals have a highly structured spatial components that are often neglected by many of these algorithms which may play a crucial role in modulating EEG dynamics. The brain is a highly interconnected structure that has connections various levels as single neurons, populations of neurons, and brain regions. These interactions which are called brain networks are thought to be the origin of EEG signals recorded on the scalp. Therefore, a classification model that can leverage this network structure in the learning process might produce better results. The main goal of this project is to address the role of brain networks in EEG classification. This goal can be divided into sub-goals to address three fundamental questions:

1. will a model employing a learning algorithm that can learn the network structure of spatiotemporal EEG data produce higher classification accuracy?

2. what kind of brain networks can be suitable for EEG classification? and how can we reconstruct these networks?

3. can we use brain networks as tools for dimensionality reduction in EEG classification?

In this study, these questions were investigated within the context of motor imagery (MI) EEG classification task. The questions have been addressed in two steps:

1. Modeling brain networks using EEG data: A graph theory model is adopted that is flexible and powerful to represent the desired networks.
2. Designing a neural network classification model that can learn on graphs: Specifically, a spatiotemporal graph convolutional neural network model (STGCNN) is employed for this purpose.

The thesis structure goes as follows; This chapter introduces the problem and the fundamental goals. Chapter 2 presents the relevant background information and the literature review of the EEG data classification. Chapter 3 discusses the detailed methods used in three sections. The first section provides a brief summary of graph theory and its relevance to the problem. The second one illustrates how to represent EEG data as graphs. The third section discusses graph neural network (GNN) as a learning algorithm. Chapter 4 describes the the dataset and the experimental protocol. Results, discussion, and conclusion are given in Chapters 5, 6, and 7, respectively.

2. BACKGROUND

2.1 Brain Computer Interface (BCI)

A brain-computer interface (BCI) or brain-machine interface (BMI) is a computer-based system that acquires brain signals, interprets them, and translates them into commands executed by an output device, which is typically another computer or a robotic equipment. Theoretically, any sort of brain signal, such as electrophysiological or metabolic activity, can be utilized as an input to the system, but in practice, electrical signals are the most common input. A functional BCI system has three components (Figure 2.1): a signal acquisition device, a signal decoding and translation software, and an output device, such as a robotic limb [1],[11].

We can categorize BCI systems into three distinct categories according to the signal acquisition step [11]:

1. **Invasive BCI**

   This type of BCI system requires surgically implanting an electrode array inside the brain, typically into the grey matter. This type of application has the advantage of recording the most accurate and least noisy brain signal, which simplifies subsequent processing and translation steps, but it is an invasive method which can possibly cause the tissue to develop a scar or an and immune response against the implanted electrode array [12].

2. **Partially Invasive BCI**

   In this kind of BCI system, an electrode array is implanted inside the skull on the surface of the brain. The signals recorded via such systems are more noisy than those recorded from invasive BCI systems, but less noisy than those recorded via non-invasive BCI systems.

3. **Non Invasive BCI**

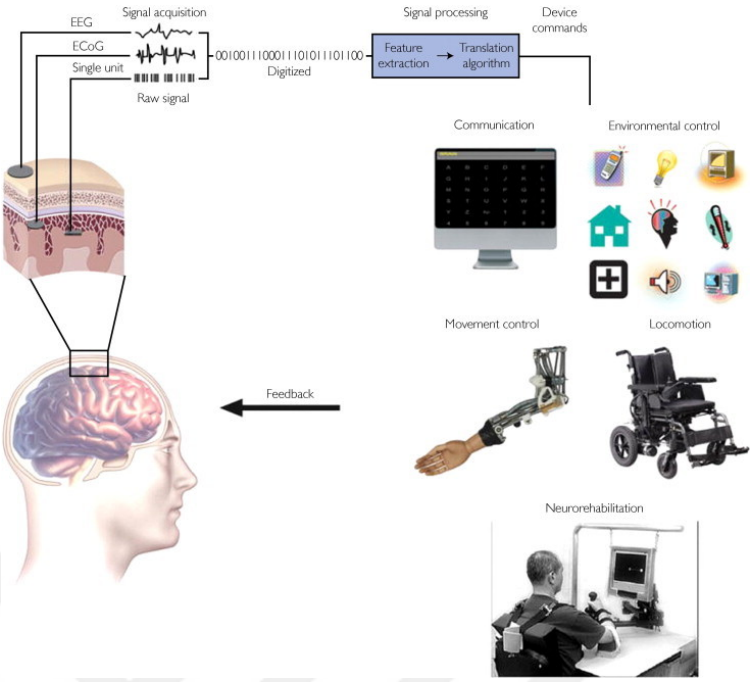

Figure 2.1 An illustration of the components of BCI system [1].

In this type of a BCI system, signals are recorded without surgical implantation of electrodes. Other neuroimaging modalities, such as fMRI and MEG, can also be used to acquire non-invasive brain signals. However, EEG is the most common method. Signals recorded are subject to measurement noise and have less spatial resolution than signals received invasively, but they are nonetheless, highly effective for a variety of BCI applications. This type of system can also leverage inputs other than brain signals, such as eye movement as a controller for an external device [13].

The main goal of BCI systems is to help handicapped people and patients with special needs to restore their normal lives. Examples of applications of BCI systems include:

1. **P300**

   A P300 is an electrophysiological response elicited by an internal cognitive or an external physical event that is time locked to the stimulus. It usually peaks around 300 ms after the stimulus, whence the name P300 comes. In a popular

P300 paradigm, the P300 speller, the subject looks at a screen where some characters are flashing, and the subject chooses a character by just attending to it [14],[15].

2. **Emotion Recognition**

   The objective of the emotion recognition BCI system is to identify and classify the emotional state of the subject. This could assist autistic patients in communicating effectively with others. It is also very important in our modern world where AI-operated machines need to understand the emotional state of humans to be more functional in daily life [16]

3. **Motor Imagery (MI) BCI**

   In this type of BCI application, brain signals are collected during the time the individual imagines executing motor activity with his limbs, such as moving one or both hands. It is commonly used for accelerating rehabilitation for stroke patients and controlling robotic limbs for amputees [17].

2.2 EEG

EEG refers to recording the brain's electrical activity using electrodes placed on the scalp. Typically, EEG is a non-invasive method. However, Electrocorticography (ECoG) or intracranial Electroencephalography (iEEG) is an invasive method that involves surgically placing electrodes on the outer surface of the brain [2]

EEG is measured in volts (typically, micro-volts μV). The signal measured in micro-volts at one EEG electrode is relative, such that it is indeed the change or fluctuation in electrical potential between this electrode and a reference electrode placed somewhere else on the scalp. In practice, it is common to use the average of all electrodes as a common reference for each electrode [18].

The first recording of human EEG was done in 1925 by Hans Berger, a German

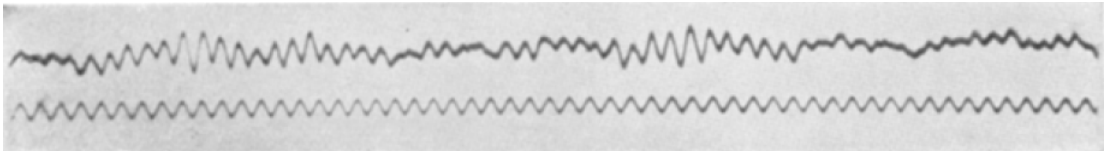

Figure 2.2 The first recorded human EEG signal by Hans Berger [2].

neuropsychiatrist. However, attempts to study the brain's electrical activity started way earlier than Berger's work. In 1875, Richard Caton found that there were very weak currents that could be detected by the galvanometer when placing the electrodes on two different points on the external surface of the skull of a rabbit, or one on the outer surface and one on the grey matter. In 1912, Pravdich-Neminsky recorded the EEG from the brain, dura, and the outer skull of a dog, and called it electrocerebrogram. In 1924, Hans Berger started his work on recording EEG from human patients with large skull bone defects, but later in 1925, he recorded EEG signals from healthy subjects and was able to obtain good EEG records with alpha waves (Figure 2.2). In 1934, Berger's observations were confirmed by two separate groups of researchers. The first was Edgar Douglas Adrian, who recorded his own alpha rhythm. The second one was A. J. Derbyshire, who was a graduate student of Hallowell Davis at Harvard. He was also able to record a good alpha rhythm from Davis. EEG research kept growing and flourishing from that point forward. In 1934, Fisher and Lowenbach were the first to identify the epileptiform spikes in EEG. In 1935, Gibbs, Davis, and Lennox identified interictal spike waves as well as the three cycles per second pattern that is characteristic of clinical absence seizures. These observations marked the beginning of the discipline of clinical electroencephalography. In 1953, Aserinsky and Kleitman studied rapid eye movement sleep (REM) with EEG. Furthermore, Berger with assistance from the physicist Dietsch was also the first to introduce computational techniques for the analysis of EEG waves. In 1932, they applied Fourier analysis to short segments of EEG [2],[19].

2.3 Why EEG is suitable for motor imagery BCI?

EEG is highly effective in the majority of motor imagery BCI applications due to its high temporal resolution, precision, and accuracy. Temporal resolution refers

to the number of samples collected per time unit. In EEG, it is determined by the sampling rate of the acquisition instrument. EEG typically has a very high temporal resolution with a sampling rate of a few hundred to a few thousand samples per second. This is very advantageous for nearly every task studied by EEG. High temporal resolution enables EEG to capture the underlying cognitive or neural dynamics in the time interval it already occurs. In general, cognitive and neural processes are fast. They happen within a very few milliseconds. Moreover, they span a few milliseconds to a few seconds. Thus, the high temporal resolution of EEG can capture this enormously fast and dynamic activity in the brain. Temporal accuracy refers to the relationship between the timing of EEG signals and the timing of the activity at the level of the cortical neurons. The electrical activity in the brain travels instantly from the populations of neurons that generate the electrical field to the electrodes placed on the scalp; there is no barrier that causes the signals to lag or delay.

We can differentiate three spatial scales on which brain networks are organized. The first is the microscopic scale. It refers to spatial areas that are less than a few cubic millimeters and represent neural columns, neurons, and synapses. EEG is totally blind to the dynamics that occur at that level. The second level is the mesoscopic level, which refers to spatial areas that range from several cubic millimeters to a few cubic centimeters. Dynamics at this scale can be captured with EEG, but it needs a higher number of electrodes (>64) and further analysis steps such as source space modeling and surface Laplacian filtering. The third scale is the macroscopic scale. It refers to broad areas in the cortex. Dynamics occurring at this level are easily captured by EEG. The spatial resolution of EEG can be improved by using high-density recordings, whereas improving spatial precision requires accurate individual head models and further spatial analysis in addition.

2.4 EEG Classification Problem

Due to its vast range of applications, the EEG classification problem has been intensively studied over the years. It has significant medical applications, such as the

prediction of epileptic seizures, as well as applications in brain-computer interfaces, such as motor imagery and emotion recognition. Numerous algorithms were developed for this purpose. Three unique steps comprise the pipeline of these algorithms: feature extraction, feature selection, and classification [17].

Feature extraction is using signal processing to extract discriminative and non-redundant information from EEG data to produce a feature set on which classification can be performed [17]. EEG data can be analyzed in the time domain, frequency domain, time-frequency domain, or spatial domain to extract features. Auto-regressive (AR) modeling has been utilized extensively to extract time-domain features [20],[21] where model coefficients are employed as features. In the frequency domain, fast Fourier transform (FFT) [22] and Welch's method [23] are used to obtain the power spectrum. Welch's method has a significantly lower noise component in the spectrum, but it has a lower frequency resolution. Time-frequency analysis relates the signal's spectrum information to the temporal domain, giving it an advantage over stand-alone time-domain and frequency-domain techniques. The short-time Fourier transform (STFT) [24], wavelet transform (WT) [25], and the discrete wavelet transform (DWT) [26] are utilized for MI EEG feature extraction. Finally, common spatial pattern (CSP) is a widely used feature extraction technique in the spatial domain for MI EEG classification [27],[28], and [29].

Feature selection is the process of selecting a subset of features that achieves the maximum classification accuracy. Dimensionality reduction techniques such as principle components analysis (PCA) [30],[31] and independent components analysis (ICA) [32],[25] are methods for feature selection. Others include evolutionary algorithms, which select features based on the optimization of classification accuracy. Numerous algorithms, such as particle swarm optimization (PSO) [33], artificial bee colony optimization (ABC) [34], and genetic algorithms (GE) [33] have been created for this purpose, as well.

Many classification algorithms have been used for MI EEG classification. In the literature, support vector machines (SVMs) and linear discriminant analysis (LDA)

are the most commonly used classification methods [35],[33],[36], and [37]. The performance of SVMs was found to be superior to that of other methods, including LDA, K-nearest neighbor (KNNs), naive Bayes, and regression trees [20],[28]. The overall performance of a classification pipeline is dependent not only on the classification algorithm, but also on the entire pipeline including feature extraction, selection, and classification. For instance, the KNN classifier can produce better results with a highly correlated feature set [38].

The adoption of deep learning (DL) architectures in EEG classification has many advantages over traditional methods. DL can perform the whole pipeline of feature extraction, selection, and classification in one single processing step. Additionally, it can handle raw EEG data without any preprocessing steps. Convolutional neural networks (CNNs) are the most prevalent DL architecture employed in MI EEG classification [39],[40],[41], and [42]. In [42], raw EEG time series were used as model input, whereas in [43] and [40], EEG images were utilised. Obtaining the images requires further preprocessing techniques, such as STFT or WT [40],[44]. Additional DL architectures, including recurrent neural networks (RNN), stacked auto-encoders (SAE), and deep belief networks (DBF), have been employed [43],[45]. Finally, graph neural networks (GNNs), a relatively new DL architecture, have been applied to a variety of EEG classification applications [46],[47], and [48].

3. METHODS

3.1 Graph Theory

The credit goes back to Euler for introducing graph theory to the mathematics community back in 1736. He was attempting to solve a problem involving the river Pregel in the city of Königsberg, which is now Kaliningrad, in East Prussia. There were seven bridges spanning the two branches of the river that connected four different parts of the city, including a small river island. The problem was to find a route to traverse each bridge precisely once and return to the starting point. It was agreed that there was no such route, but the problem was that there was no proof for such a claim. Euler provided not only a precise and concise demonstration that the route did not exist but also a general solution that can be applied to any arbitrary bridge and landmass problem (Figure 3.1). He showed that the exact geographical locations and physical distances did not matter for sketching a proof, what mattered was the relative locations of the bridges and the parts of the city they connected. He published his solution to the problem in the Proceedings of the Petersburg Academy in 1736. He is credited with what he referred to as the "geometry of position", which we now know as "graph theory" [3].

3.1.1 What are graphs?

A graph is a mathematical representation of a real-world network, or more generally, of some system composed of interconnected elements [3]. A graph consists of a set of nodes or vertices and a set of edges. Nodes represent the fundamental components of the system, such as atoms in molecules, individuals in a social network, or neurons in a neural network. The edges represent the connections between those elements. It could be the type of chemical bond in the case of a molecule, the type of relationship in a social network, such as between friends or coworkers, or the type of

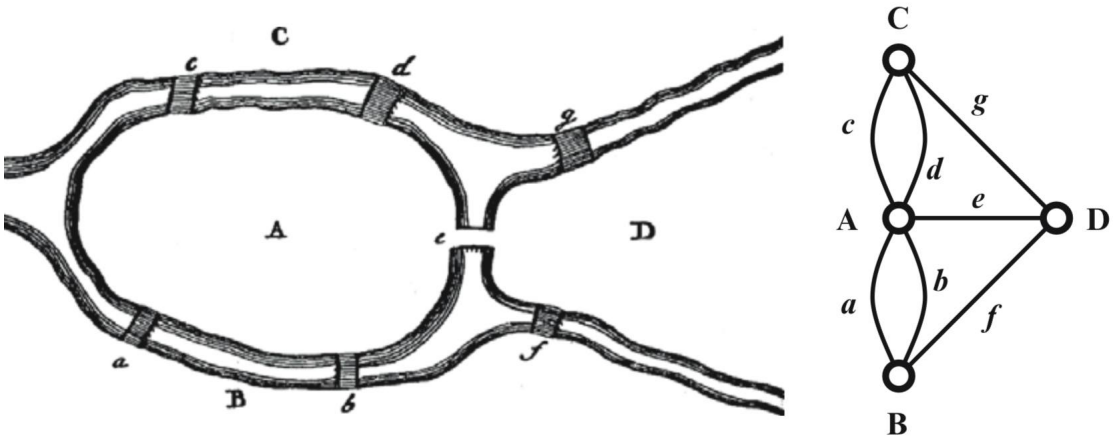

Figure 3.1 The Seven Bridges Problem and Euler's Solution [3].

synapse between neurons in a neural network [5],[49].

A graph G is described as a set of nodes or vertices N connected by a set of edges E and represented by an $n \times n$ square adjacency matrix A (Figure 3.2), the entries of which indicate whether an edge exists between two nodes such that $A_{ij} = 0$ if there is no edge between node i and node j, and $A_{ij} > 0$ if there is an edge between node i and node j. The graph can be a binary graph, meaning the adjacency matrix entries can consist of only ones and zeros $A_{ij} \in \{1, 0\}$. Another option is to weight the graph so that the adjacency matrix entries have arbitrary positive weights between zero and one $A_{ij} \in [0, 1]$. Additionally, a graph is said to be undirected when the adjacency matrix is symmetric such that $A_{ij} = A_{ji}$. It is a directed graph when the adjacency matrix is asymmetric such that $A_{ij} \neq A_{ji}$ [4]. Nodes can be connected together with a single edge or indirectly by a series of intermediate nodes and edges. Furthermore, if two nodes are connected by a unique series of nodes and edges, it is referred to as a path; if they are connected by a non-unique series of nodes and edges, it is referred to as a walk [3].

Another important matrix that describes a graph is the degree matrix D (Figure 3.2) which is also an $n \times n$ square matrix. The degree matrix contains the node degrees on its diagonal and zeros elsewhere. In undirected graphs, the node degree is the

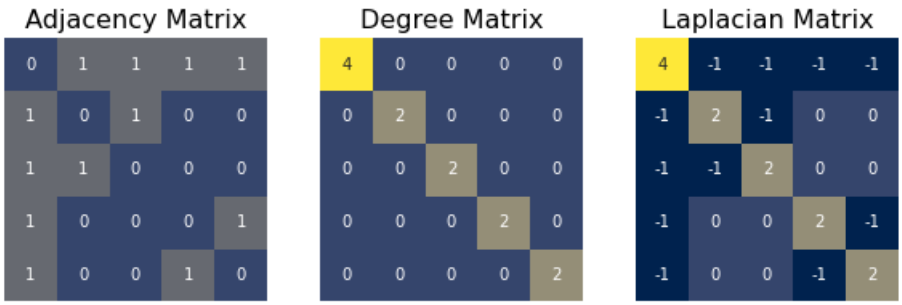

Figure 3.2 The Adjacency matrix, Degree Matrix, and the Laplacian Matrix.

number of all edges connected to this node as shown in Eq. 3.1. In directed graphs, we can describe a node by two kinds of degrees, the in-degree D_{in}, and the out-degree D_{out}. The in-degree of a node is the total number of incoming edges to that node (Eq. 3.2), while the out-degree is the total number of outgoing edges from that node (Eq. 3.3). Weighted graphs do not differ from unweighted graphs. The degree of a node in a weighted graph is defined as the sum of the edge weights attached to this node. The same logic applies to weighted and directed graphs as it does to binary graphs, but the sum of the edge weights attached to the node, whether incoming or outgoing, is used to compute the in-degree or the out-degree, respectively.

$$D_i = \frac{1}{2}\sum_{i=1}^{N} A_{ij} \tag{3.1}$$

$$D_j^{in} = \sum_{i=1}^{N} A_{ij} \tag{3.2}$$

$$D_i^{out} = \sum_{j=1}^{N} A_{ij} \tag{3.3}$$

Finally, another important graph representation matrix is the Laplacian matrix, which can be defined in Eq. 3.4 as follows:

$$L = D - A \tag{3.4}$$

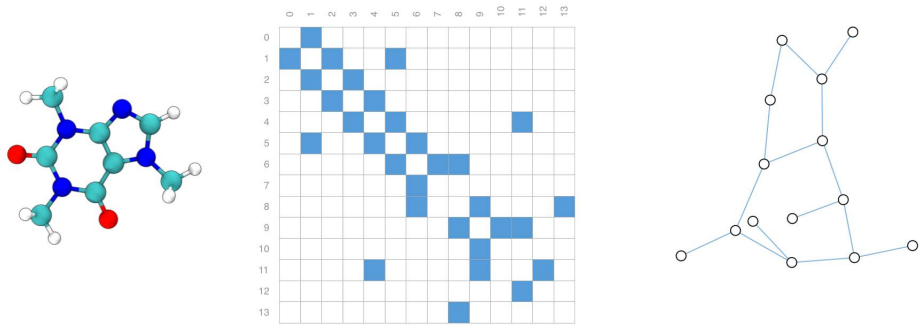

Figure 3.3 A Caffeine molecule structure and its graph representation [4].

The Laplacian matrix (Figure 3.2) is symmetric and positive semi-definite, which means it has non-zero real eigenvalues and can be decomposed as in Eq.3.5. This is a very important characteristic for defining the notion of graph frequency and graph convolution as the eigenvalues of the Laplacian act as graph frequencies.

$$L = U\Lambda U^T \tag{3.5}$$

3.1.2 Applications of graph theory

In fact, graphs are around us everywhere and can be seen almost in every science or engineering application. Graphs are flexible, generic, and powerful structures that can be used to model any system with interconnected components [4].

In chemistry, graphs are used to model molecules (Figure 3.3). Molecules are the building blocks of matter. They consist of atoms that are connected together with various types of chemical bonds. The atoms of a molecule are considered the nodes in a graph, while the bonds are considered the edges of the graph. This graph representation of molecules is very intuitive and widespread, particularly in computational applications in chemistry, such as using machine learning to examine the interactions between different molecules, for example, in the context of drug discovery [49].

In social science, social networks are a tool of social science to study patterns in the behavior of individuals, organizations, etc [4]. We can easily consider individuals

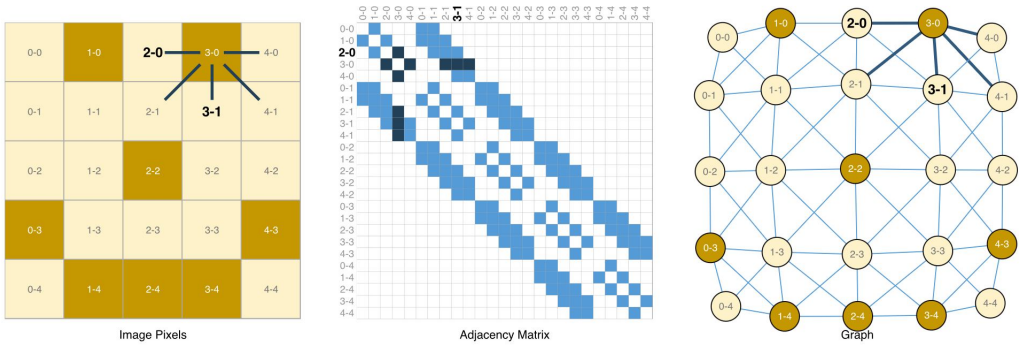

Figure 3.4 Graph representation of images [4].

as the nodes in a graph, and the edges represent the kind of relationship or interaction between them. For instance, the Facebook social network is typically modeled as a graph, with platform members serving as the graph's nodes. Additionally, if two individuals are Facebook friends, there is an undirected edge between them, whereas if one person follows another, there is a directed edge between them. This suggests that social network graphs are not required to be symmetric, otherwise stated they can be asymmetric.

Graph theory has numerous applications in the biological and biomedical sciences as well. In molecular biology, for example, graphs are used to represent gene-to-gene interaction networks, whereas, in computational neuroscience, graphs are utilized to model the connectivity patterns among brain areas [3].

In the previous examples, graphs act as models for natural phenomena that happen to be natural networks, but graphs can be extended further to model systems that do not occur to be natural networks or whose network structure is not clear from the first inspection. For instance, graphs can be used to model images [4] (Figure 3.4), with each pixel acting as a node with edges connecting it to all of its neighbors. Another way is to consider each scene in an image as a node, and the edges represent the relations between different scenes in the image. Even though the typical grid structure of images is a much better way of representing images, it is a good illustration of how grids are actually graphs with some unique properties.

3.2 Modeling EEG as Graphs

3.2.1 Why to model EEG as a graph?

To answer this question most effectively, we should consider it from two distinct perspectives: the biological perspective and the engineering perspective.

From a biological standpoint, biologically, the brain is a highly interconnected system that can be defined as a network of networks. The brain organizes networks at several levels (Figure 3.5) [3]. We can discern three distinct tiers. At the molecular and cellular level, examples of networks include protein interactions and gene-regulatory networks, as well as synaptic connections between populations of neurons that shape the neural circuitry within the brain. The networks at the systems level of the brain, such as the visual network and the somatomotor network, are characterized by anatomical projections between discrete brain regions and functional patterns of interaction between these areas. Finally, on a broader level, the interaction between brain systems and the environment shapes the behavior among social groups, which defines social networks.

Similar to other neuroimaging or electrophysiological techniques, EEG records the activations and interactions among distinct brain parts in parallel (at a specific scale), resulting in data sets that are rich in hidden dynamics and interactions between different brain systems [3]. Consequently, a model capable of capturing these latent dynamics will produce more accurate predictions than models unable to capture the network structure and dynamics. Graphs are ideal for capturing these latent interactions and dynamics in neural data because they represent a realistic network model.

From an engineering and computational viewpoint, graphs provide a rigorous method for describing the structure of EEG data. Typically, EEG classification pipelines necessitate two steps: feature extraction from raw data and feeding those features into a classification algorithm to predict the output. Typically, these features

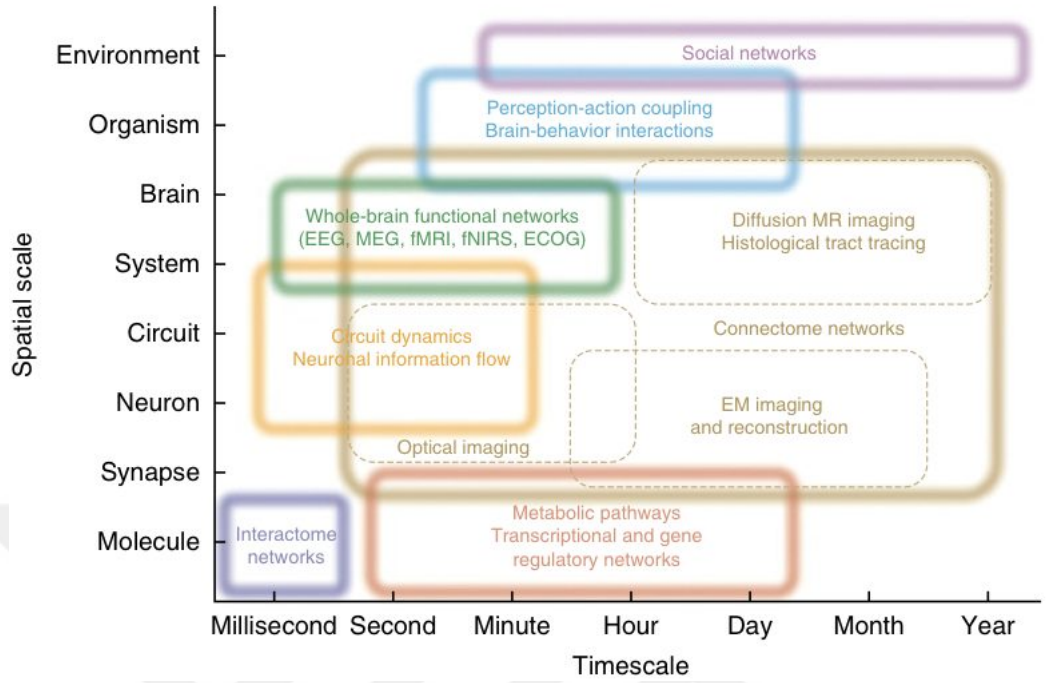

Figure 3.5 Networks at different spatial and temporal scales as recorded by different neuroimaging methods [5]

fall into two major categories: temporal features and spatial features. Spatial features capture the spatial dependence between EEG electrodes, yet the majority of spatial feature extraction methods are extremely sensitive to the ordering of EEG electrodes. Given that the order of EEG electrodes is completely arbitrary, this poses a challenge and may result in significant variation in learned representations. This issue is resolved in a very unique and subtle manner via graphs. Graphs are naturally characterized by the adjacency matrix as non-ordered structures. Changing the order of the electrodes may alter the order of the adjacency matrix, but it will not affect the structure of the graph (Figure 3.6). Moreover, any learning method that uses the graph structure for feature extraction will be unaffected by any change in the order of electrodes, so learning an invariant representation of features. Theoretically, this unique criterion puts graph-based learning algorithms ahead of other methods for spatial feature extraction for EEG classification.

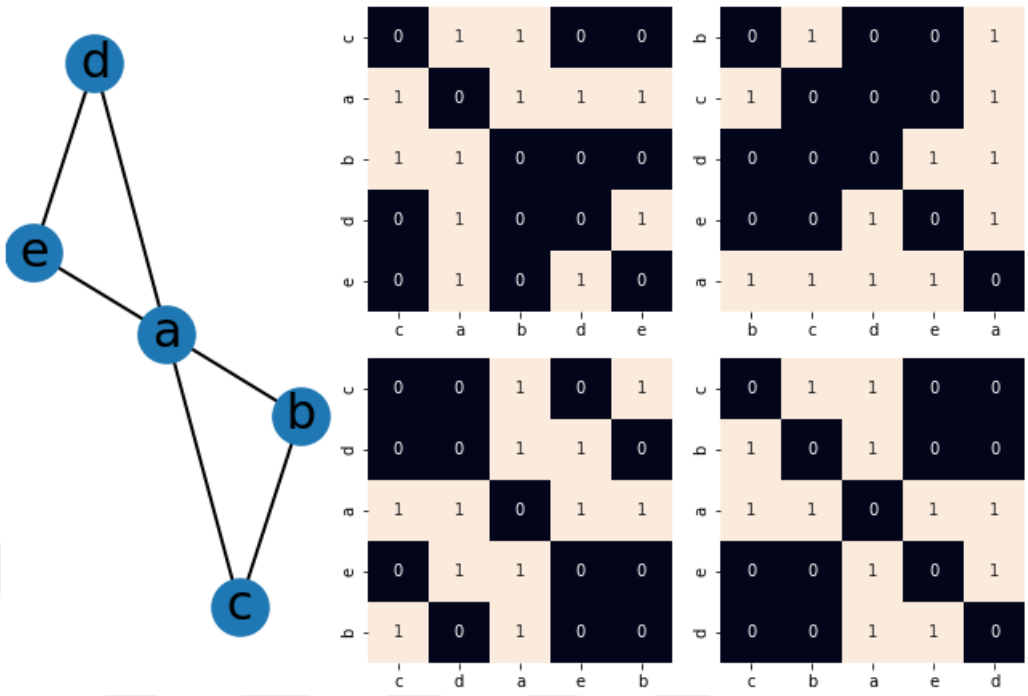

Figure 3.6 Changing the order of the adjacency matrix does not change the structure of the graph.

3.2.2 What are brain networks?

We have established that graphs are perfect models for representing EEG data in classification tasks due to their flexibility and power in capturing the dynamics of brain networks. But what exactly are brain networks, and how do we have to construct them from EEG data?

In order to define a network inside the brain, we need to make two choices, defining the nodes and defining the way those nodes connect via edges. Networks, in essence, describe the connectivity pattern among elements of a complex system. In our case, this system is the brain. From the choices above, we can distinguish three types of connectivity that can be inferred from neural data (Figure 3.7):

1. **Structural Connectivity**

   In this kind of connectivity, networks are defined by the physical or anatomical connections between neural elements [3]. The scale of these networks depends on the method used in acquiring the data. Neural circuits connecting discrete

neurons can be detected using histological scans of neural tissue, while neural pathways connecting areas of the brain are detected via brain structural imaging techniques like MRI. These networks are thought of as relatively static at shorter time periods, whereas they can be dynamic at longer time scales [5]. Finally, structural networks can be directed or undirected and binary or weighted depending on the method of data acquisition.

2. **Functional Connectivity**

   Functional connectivity refers to the statistical dependency between distinct neural units. Apart from structural networks, it depends entirely on neural time series data recordings such as single-cell recordings, EEG, MEG, or fMRI. Functional connectivity is very dynamic and often changes its pattern in tens to hundreds of milliseconds. Additionally, in most cases, it is symmetric, which means the graphs are undirected. Various statistical measures can be used to infer functional connectivity from time series data. The most widely adopted methods in the literature fall into three categories: correlation-based metrics, information-theory-based metrics, and phase-coupling and coherence methods.

3. **Effective Connectivity**

   Effective connectivity refers to the causal influence among neural elements [50]. Like functional connectivity, it is inferred from neural time series data. It can also be inferred via statistical modeling or experimental perturbations. Unlike functional connectivity, it derives directed interactions between neural elements, but like functional connectivity, it is very time-dependent and can change patterns in very short time scales, from tens to hundreds of milliseconds. The estimation of effective connectivity requires careful data processing. It can be estimated via model-free methods or by explicitly setting a causal model with structural parameters.

It is important to note that no single mode of connectivity is sufficient to fully explain the mechanism of brain networks [3]. There is always interaction and interference between them. For instance, we can not just neglect the structure while studying the

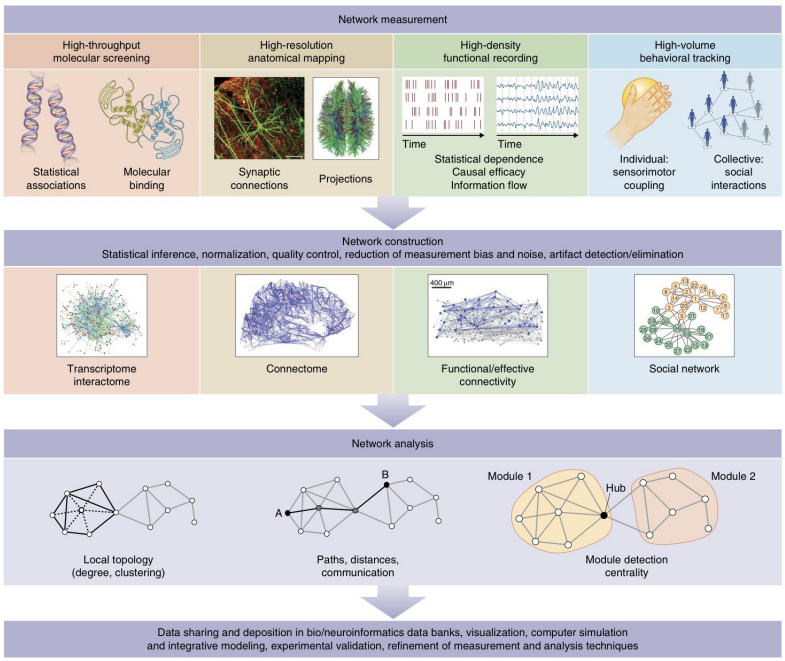

Figure 3.7 Different data acquisition and analysis pipelines lead to different kinds of networks [3].

functional patterns, or vice versa, we cannot rely on the structure only to understand the neurophysiological properties of the brain. A central goal in systems neuroscience is the parcellation of the cerebral cortex into discrete neurobiological atoms or units [6]. Plenty of studies have approached this problem using rs-fMRI data sets. The majority of these studies used one of two main approaches:

1. **The local gradient or boundary mapping approach [51],[52],[53]**

   This approach tries to cluster the cortical voxels or vertices of rs-fMRI based on the fact that resting-state functional connectivity (RSFC) abruptly changes from one spatial cortical location to a nearby one [6]. These abrupt changes can be detected by computing the local gradients in whole brain RSFC patterns. This method has the benefit that it correlates with the cortical regions identified by histological delineation, thus the name boundary mapping.

2. **The global similarity approach [54],[55],[56]:**

   This approach groups rs-fMRI voxels or vertices together, resulting in homogeneous functional connectivity patterns regardless of their spatial position on

the cortex. This type of parcellation is highly beneficial in studies involving dimensionality reduction problems since the parcels created have relatively homogeneous functional connectivity patterns that can be averaged into a single time series.

Schaefer et al. [6] mixed both approaches together into a unique one (Figure 3.8). They developed a gradient-weighted Markov random field (gwMRF) model which mixes the local gradient and global similarity approaches. Markov random field models (MRFs) are used for global similarity parcellation approaches. They modified the objective function of the model so it forces the vertices to share the same parcel only in the case of low local RSFC gradients. The resulting parcellation is superior to previous ones because it retains both advantages, the homogeneous RSFC of the parcels and the respect for the histologically-defined boundaries of the cortex. They developed their parcellation on the GSP data set which consists of structural MRI and rs-fMRI data from 1489 subjects. They provided their atlas in different resolutions, ranging from 100 to 1000 parcels. They also mapped each parcel to either one of the 7 or 17 brain networks defined by Yeo et al. parcellation [56].

3.2.3 How to model EEG data as graphs?

In order to model EEG data as graphs, we need to define a set of nodes with edges connecting them. Additionally, we need to define a space or level to capture brain networks. This section is dedicated to the data processing steps required in order to transform scalp EEG data into graph time series data describing brain networks captured at the scalp and the cortex levels.

3.2.3.1 Nodes. There are two ways we can model EEG nodes. The first is to consider each scalp electrode as a node and its corresponding time series as its feature vector. This approach is straightforward and does not need further data processing. However, it might not be the best way to model brain networks because it lacks the

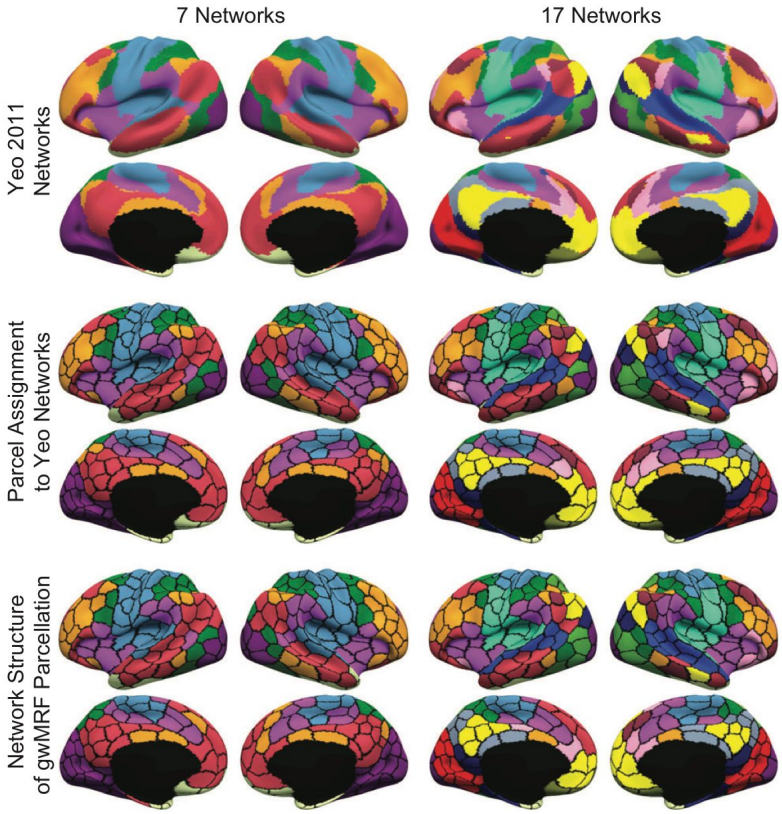

Figure 3.8 Schaefer's brain atlas [6].

anatomical precision of cortical networks (Figure 3.9). Additionally, the resulting networks show higher correlations due to the volume conduction problem. The other approach is to project scalp EEG data onto the cortical source space. We know that EEG captures the neural dynamics in the cerebral cortex at the mesoscopic level. Our goal is to transform scalp EEG data (registered by electrodes) into cortical time series by projecting them onto the 100 nodes located on 7 resting state brain networks given by Schaefer's parcellation atlas.

In order to achieve this goal, we need a model that describes how cortical dynamics is represented by scalp EEG data. Field currents are produced by the interaction of different populations of neurons on the cortical surface. These currents traverse a distance from one area of the brain to another via cortical structures like grey matter, white matter, and CSF. At the same time, they pass through via non-cortical structures such as the dura, skull, and scalp, where EEG electrodes can detect them. We

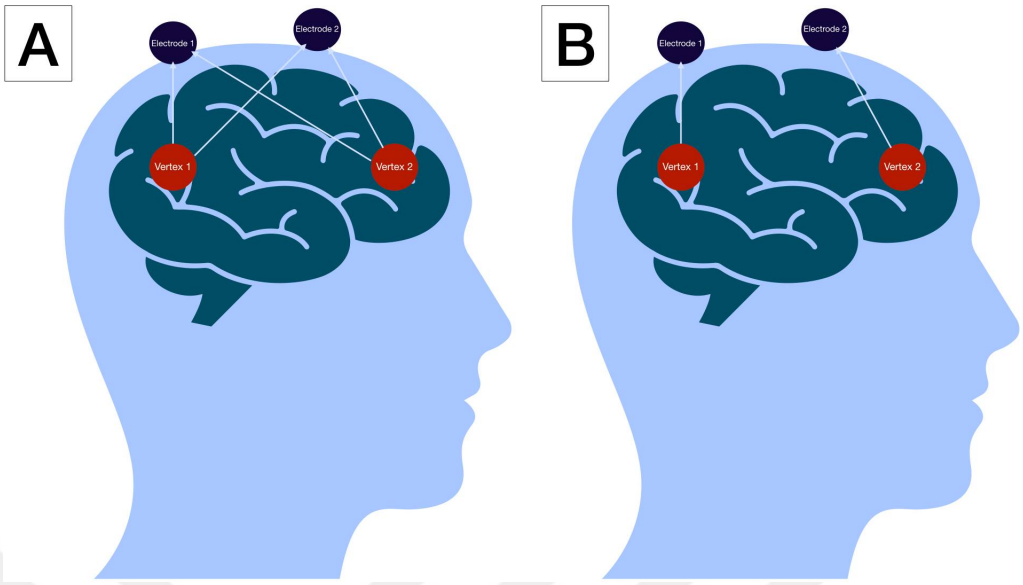

Figure 3.9 Scalp EEG lacks the anatomical precision of the cortical EEG.

can describe this process mathematically:

$$Y = H \cdot S + \epsilon \tag{3.6}$$

where Y $(m \times t)$ is the scalp EEG time series, S $(n \times t)$ is the cortical sources time series, m is the number of electrodes, n is the number of cortical sources, t is the time dimension, H $(m \times n)$ is the lead field matrix, and ϵ refers to the noise vector. This equation defines the EEG forward model. It describes how cortical dynamics from distinct sources on the cortex contribute to the scalp EEG data. The lead field matrix H maps each cortical source contribution to the scalp EEG data. Our goal is to estimate the cortical source activations S. From the forward model, we can define an inverse solution:

$$\hat{S} = T \cdot Y \tag{3.7}$$

where $\hat{S}$ is an estimate of the cortical sources activations, Y is the scalp EEG time series, and T is a matrix that provides a solution to estimate the inverse mapping of the electrical currents from scalp EEG to the cortical surface. This problem is ill-posed and does not have a unique solution. We need to impose some structural constraints in order to find a unique solution. We need three kinds of structural information as constraints. First, we need the exact positions of the electrodes placed on the scalp.

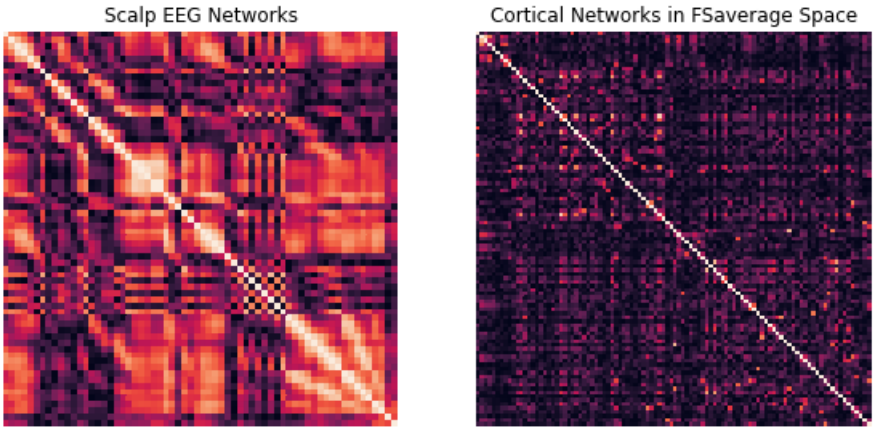

Figure 3.10 Functional brain networks defined by absolute Pearson correlation in different spaces.

Second, we need prior information about the cortical source distribution. Finally, we need information about the non-cortical structures that conduct the currents, like the dura, skull, and scalp. Electrode positions are normally recorded exactly during the data acquisition process. Additionally, the advent of neuroimaging methods with high spatial resolution, such as fMRI, gives us substantial information about cortical and non-cortical structures in anatomy and function. This prior information is encoded in the lead field matrix H, so we can now solve the inverse problem to acquire an estimate of the cortical source activations.

There are multiple algorithms developed in the literature for solving the inverse problem, such as wMNE [57], dSPM [58], eLORETA [59], and sLORETA [60]. In this project, weighted minimum norm estimates wMNE [57] was utilized to solve the inverse problem. Here, T is estimated to produce the source distribution with the minimum power that fits the measurements in a least-square error [61]:

$$T_{wMNE} = RH^T(HRH^T + \lambda C)^{-1} \tag{3.8}$$

Where T_{wMNE} $(n \times m)$ is the estimate of T, H is the lead field matrix, C is the scalp EEG noise covariance matrix, λ is a regularization parameter. R is the source covariance matrix. If $R = I$, wMNE will assign the same weight for each cortical vertex. Additionally, λ is computed based on the signal-to-noise ratio as $\lambda = 1/SNR$ or chosen empirically.

After solving the inverse problem and reconstructing the source time series, we can simply group the time series of the cortical vertices that belong to the same parcel by averaging or summing. Thus, we will end up with a hundred time series that represent the nodes of our desired graph.

3.2.3.2 Edges. To define the edges of the graph, we need to choose a measure of connectivity between nodes as defined earlier in this section. In this project, two types of connectivity were utilized to define the edges of the graph.

1. **Functional Connectivity: Pearson Correlation [62]**

   Pearson correlation or correlation coefficient is the simplest way to measure the statistical dependency between two time series. It is defined as the linear correlation between two random variables. The value of Pearson correlation always ranges between (-1, 1). A value of 1 means a total positive correlation between the two variables, whereas a value of -1 corresponds to a total negative correlation between the two variables. A value of 0 means there is no linear correlation between the two variables.

   $$\rho_{xy} = \frac{cov(x, y)}{\sigma_x \sigma_y} \tag{3.9}$$

   where $cov(x, y)$ is the covariance between x and y, and σ is the standard deviation of x and y. The covariance between x and y is calculated as:

   $$cov(x, y) = E[(x - \mu_x)(y - \mu_y)] \tag{3.10}$$

   Where E is the expectation, and μ_x and μ_y are the means of x and y respectively.

   Usually, the absolute value of the Pearson correlation coefficient is used when constructing brain networks. The resulting correlation matrix acts as a weighted adjacency matrix, defining the edges that connect the nodes or the parcels together. It is also common to define a threshold and drop every edge whose weight is lower than this threshold. This is useful to account for the fact that Pearson

correlation tends to produce highly dense networks and it helps to decrease the noise contribution to the networks.

2. **Structural Connectivity** In the sensor space, structural connectivity of a node can be defined by its first degree neighboring electrodes physically located on the surface of the scalp. In the source space, Schaefer's atlas is defined based on functional connectivity networks as discussed earlier. In theory, a significant portion of its parcels should share a physical or anatomical connection, too. Luppi and his colleagues [63] addressed this particular question using nine different anatomical and functional atlases of the brain including Schaefer's atlas (Figure 3.11). Using diffusion tensor imaging data, they were able to define structural networks that connect different parcels defined on these atlases. They constructed a structural connectivity matrix A_s such that:

$$a_{s_{ij}} = log_{10}(1 + f_{ij}) \tag{3.11}$$

where f_{ij} is the number of white matter streamlines between areas i and j. Furthermore, this matrix can be binarized such that $a_{s_{ij}} = 1$ when there is a structural link between areas i and j and 0 elsewhere.

Finally, we can combine both structural and functional networks together by keeping the weights of the functional connectivity matrix only if there is a structural edge. It is simply achieved by element-wise multiplying binarized structural and weighted functional connectivity matrices.

3.3 Graph Neural Networks (GNNs)

We have established the suitability of graphs for modeling EEG data for learning in classification tasks. In this section, we are going to discuss the methods of machine learning on graphs. We start by stating the problem and defining the task and the challenges of learning on graphs. Then, we will dive step-by-step into the formulation of graph neural networks (GNNs), a powerful neural network architecture for learning

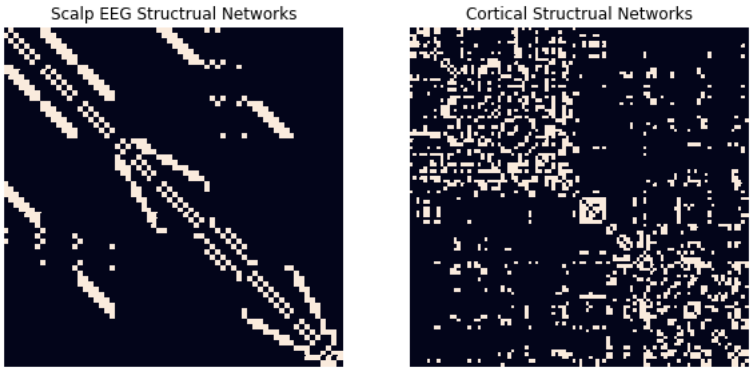

Figure 3.11 Structural Networks defined on scalp EEG sensors and on cortical level in Schaefer's atlas

.

on graphs.

3.3.1 Problem Setting

In the upcoming sections, we will be using the notation $G = (N, E, A, X)$ to define a graph (one EEG trial in our case) where N represents the nodes of the graph, E the edges of the graph, A the adjacency matrix of the graph and $X \in R^{N \times d}$, is an array of nodes attributes with d attribute per node (the EEG time dimension in our case). We refer to a target/receiver node of the graph as n_i, and consequently, its feature vector is x_i. Similarly, n_j refers to a source/sender node of the graph and its feature vector becomes x_j. The entries of the adjacency matrix A define the presence of an edge between two nodes such that A_{ij} indicates there is an edge connecting node i and node j, whereas e_{ij} refers to the features associated with the edge between node i and node j. If e_{ij} is a scalar, in that case, it is considered an edge weight and can also be an n-dimensional feature vector. There are GNN variants that can handle only one-dimensional edge features and others that can handle multi-dimensional edge feature vectors as discussed later.

There are three different task levels that can be formulated over graphs [9],[64],[7]:

1. **Graph-Level Tasks:**

   The goal of these kind of tasks is to predict a label for the entire graph. For instance, we might want to predict the smell of a particular molecule or its effect on the treatment of a certain disease. This is analogous to image classification tasks when our goal is to predict a label for the entire image, e.g., a dog or a cat.

2. **Node-Level Tasks:**

   The goal here is to predict a label for each node distinctly. A classic example of these tasks is the classification of social networks. For instance, we may predict the social group or the political party a person belongs to from his social media connections on Facebook or Twitter. The analogy for these kinds of tasks is image segmentation tasks, where the goal is to predict a label for each pixel separately.

3. **Edge Prediction Tasks:**

   In these kind of tasks, we want to predict the existence of an edge or a link between two nodes. A good example is image scene understanding, when our goal is to predict the relationship between objects in a scene rather than identify the objects.

Learning on graphs is not as straightforward as learning on images for instance. Instead, there are many challenges that need to be considered while formulating a learning algorithm. These challenges are include [64]:

1. **Lack of consistent graph structure**

   Traditional machine learning and deep learning algorithms only accept fixed-size input data. On the other hand, graphs are very flexible structures that can be of any arbitrary size, which means they lack consistent structure across instances [64]. For example, consider the task of chemical molecule classification, whether toxic or non-toxic. Molecules can have different atoms, which vary in number, and various kinds of bonds between atoms, which in turn have various strengths. Thus, we need our algorithm to be able to handle such a lack of structural consistency.

Traditional machine learning and deep learning algorithms only accept fixed-size input data. On the other hand, graphs are very flexible structures that can be of any arbitrary size, which means they lack consistent structure across instances [64]. If we again consider the task of chemical molecule classification, whether it is toxic or non-toxic. Molecules can have different atoms, which vary in number, and various kinds of bonds between atoms, which in turn have various strengths. Thus, we need our algorithm to be able to handle such a lack of structural consistency.

2. **Node-order equivariance**

   Graphs do not inherently assume any ordering of the nodes. We need our algorithm to consider such a criterion. More specifically, we need our algorithm to learn permutation-equivariant and permutation-invariant representations. Permutation invariance means that the learning algorithm does not depend on the arbitrary ordering of the rows and columns of the adjacency matrix such that [9]:

$$f(PAP^T) = f(A) \tag{3.12}$$

   Where f is the learning algorithm, P is the permutation function, and A is the adjacency matrix. Moreover, permutation equivariance means that the output of the learning algorithm is permuted in a consistent way with the permutation function P.

$$f(PAP^T) = Pf(A) \tag{3.13}$$

3. **Scalability**

   The size of a graph ranges from a small number of nodes to hundreds, thousands, or even millions of nodes, such as social network graphs, *i.e.*, Facebook. Because graphs are defined primarily by the adjacency matrix, the problem becomes computationally more complex and costly. Luckily, most of the graphs in nature are sparse. This means that we can utilize sparse representations of the adjacency matrix and save on computational costs.

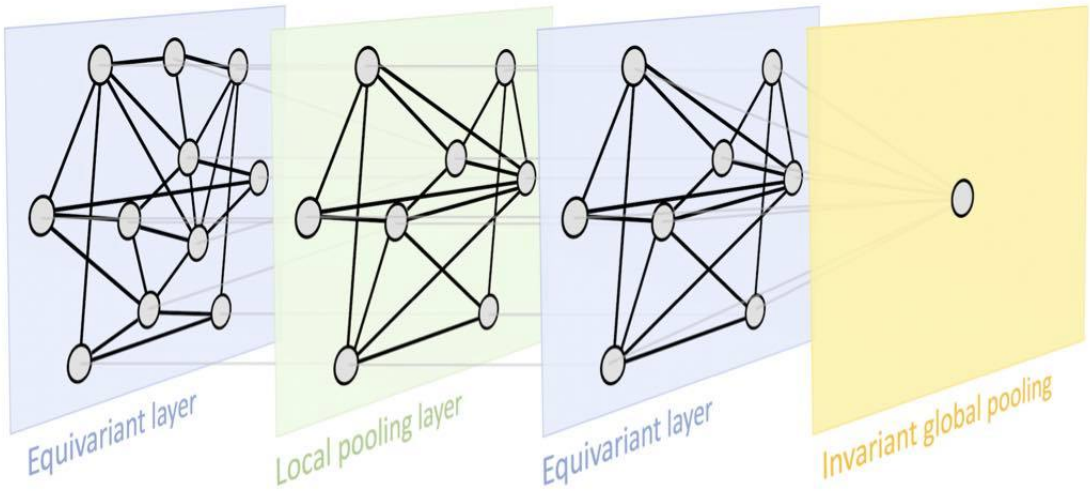

Figure 3.12 GNNs learn permutation equivariant representations in node-level tasks and invariant representations in graph-level tasks [7].

Node representation learning is a common precursor for solving the aforementioned tasks [64]. It refers to learning fixed-size real-valued vectors that are called representations or embeddings. Graph neural networks (GNNs) provide a subtle way of learning node representations. A GNN is an optimizable transformation on all attributes of the graph (nodes, edges, global-context) that preserves graph symmetries (permutation invariances) [4]. The general idea behind GNNs is the neural message passing mechanism, in which vector messages are exchanged among nodes and updated via neural networks [9]. In the upcoming sections, we are going to derive GNN from scratch.

3.3.2 Rethinking Convolutional Neural Networks

Convolutional neural networks (CNNs) are the most widely used family of neural networks in machine learning applications. They achieved state-of-the-art results in multiple domains, especially in computer vision applications. The idea behind CNNs is to learn multiple localized translation equivariant kernel filters (Figure 3.13 [7],[65]). Consider an image as an $m \times n$ matrix M and a learnable filter g as an $l \times l$ matrix,

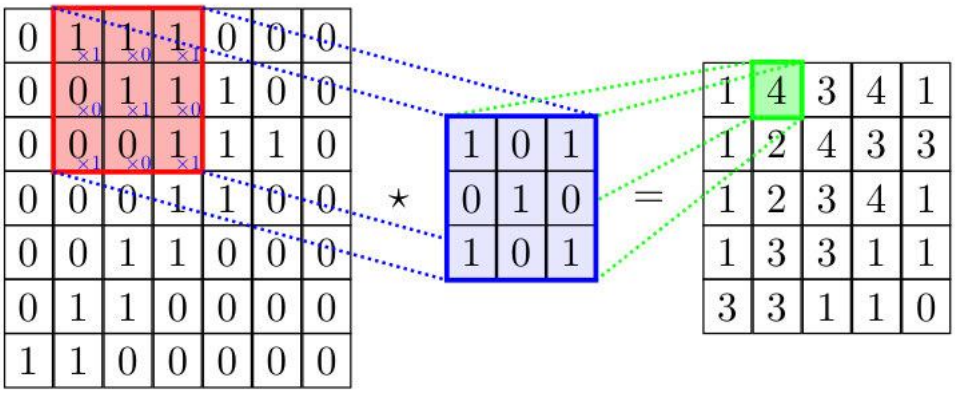

Figure 3.13 Regular convolution on images [7].

the convolution can be expressed as follows:

$$\hat{m}_{ij} = \sum_{a=1}^{l} \sum_{b=1}^{l} g_{ab} m_{i+a,j+b} \tag{3.14}$$

M is a gray-scale image for simplicity and $\hat{m}_{ij}$ is the updated value of a pixel in the image. There are two ideas from convolution on images that can help us formulate a general convolution on non-euclidean structures: learning localized filters over neighboring pixels and updating the value of each pixel with information from the neighboring pixels.

The Graph Laplacian provides a robust way of defining localized filters on graphs. Recall that it encodes exactly the same information as the adjacency matrix about the graph structure. We can construct polynomials of the Laplacian as follows:

$$\mathbf{p}_w(L) = w_0 I_n + w_1 L + w_2 L^2 + ... + w_d L^d = \sum_{i=0}^{d} w_i L^i \tag{3.15}$$

Alternatively, each polynomial can be written as a vector of its coefficients:

$$w = [w_0, w_1, ..., w_d] \tag{3.16}$$

We can think of these polynomials as analogous to filters or kernels in regular CNNs. In this case, we can apply graph convolution by convolving with the polynomial filters

as follows:

$$\hat{X} = \mathbf{p}_w(L)X \tag{3.17}$$

This process is exactly the same as the localized filters of CNNs, it aggregates information from the first-degree neighbors. We can see this clearly when we derive the embedding computation for a single node x_i in the simplest case when $w_1 = 1$ as follows:

$$\begin{aligned} \hat{x}_i &= (Lx)_i = L_i X \\ &= \sum_{j \in G} L_{ij} x_j \\ &= \sum_{j \in G} (D_{ij} - A_{ij}) x_j \\ &= D_i x_i - \sum_{j \in \mathcal{N}(i)} x_j \end{aligned} \tag{3.18}$$

This equation can be generalized to higher degrees as follows:

$$\begin{aligned} \hat{x}_i &= (\mathbf{P}_w(L)X)_i \\ &= (\mathbf{P}_w(L))_i X \\ &= \sum_{k=0}^{d} w_k L_i^k X \\ &= \sum_{k=0}^{d} w_k \sum_{j \in G} L_{ij}^k x_j \end{aligned} \tag{3.19}$$

where d is the degree of the Laplacian polynomial. The degree of the polynomial determines the locality of the filter such that the convolution of node i occurs only with nodes j which are not more than d hops away. ChebNet [66] used this idea of polynomial filters with a slight modification to build one of the earliest GNN architectures. It is formulated as follows:

$$\hat{X} = \sum_{k=0}^{d} Z^k \cdot W^k \tag{3.20}$$

where Z is the Laplacian polynomials of X computed as aforementioned and W is a learnable weight matrix. The only modification they introduce is that they use a

normalized version of the Laplacian for polynomial computations:

$$\hat{L} = \frac{2L}{\lambda_{max}} - I \tag{3.21}$$

where λ_{max} is the largest eigenvalue of the Laplacian matrix. ChebNet made a breakthrough in extending convolution to graphs and learning localized filters that are node-order equivariant, but it also motivated another broader perspective for thinking about convolution.

From this new perspective, we can reconsider the convolution on images. In the regular convolution on images (Eq. 3.14), what the filter actually does is updating the value of each pixel in the image in two steps: It applies a linear transformation to each close-neighbor pixel and then sums all the values of these pixels. In a more abstract way, it updates the value of each pixel with some update function (linear transformation in this case) and then aggregates the information through some aggregation function (sum function in this case). The same applies to ChebNet. What if we consider different kinds of aggregation functions beyond what is done with regular convolution or polynomial filters? There is only one constraint for such a function, it must be a node-order equivariant to ensure the learned node representation is permutation equivariant.

With this perspective in mind, Kipf and his colleagues introduced a modern version of graph convolutional neural networks (GCN) [67]. They used the mean as an aggregation function, but they kept using linear transformation as the update function. The node embedding can be computed as follows:

$$\hat{x}_i = W \cdot \sum_{j \in \mathcal{N}(i)} \frac{e_{ji}}{\sqrt{|\mathcal{N}(j)| \cdot |\mathcal{N}(i)|}} \cdot x_j \tag{3.22}$$

where W is a learnable weight matrix, $|\mathcal{N}(i)|$ and $|\mathcal{N}(j)|$ is the degree of nodes i and j respectively, and e_{ji} is the edge weight from the source node j to the target node i. It is set to 1 in case the adjacency matrix is binary. Furthermore, the normalization step by the node degree ensures we use the mean of the first-degree neighbor embeddings as an update function for the node embedding.

Although ChebNet and GCN are a breakthrough in generalizing the power of CNN to graphs, they have their own drawbacks, too. A major drawback for both models is that they are not able to handle multi-dimensional edge features. They can only handle scalar edge weights. Additionally, they both utilize a single linear transformation layer for the update function. In the upcoming sections, we are going to discuss some of the ideas to derive more general and robust GNNs.

3.3.3 From Convolution to Attention

Graph Attention Networks (GAT) [8] build upon GCN and introduce a new way of aggregating information from neighboring nodes. The main idea is to implicitly learn a weight for each node in the neighborhood by allowing nodes to attend over their neighborhoods' features (Figure 3.14). This approach fixes the minimized ability of GCN as a message-passing scheme. Besides, it is powerful in situations when we do not know the graph structure in advance yet we know that there is a significant structural dependency among the features. The computation of the embedding is as follows:

$$\hat{x}_i = \alpha_{ii} W x_i + \sum_{j \in \mathcal{N}(i)} \alpha_{ij} W x_j \tag{3.23}$$

Where α_{ij} the attention coefficients are computed as follows:

$$\alpha_{ij} = \frac{exp(LeakyReLU(a^T[Wx_i||Wx_j]))}{\sum_{w \in \mathcal{N}(i)} exp(LeakyReLU(a^T[Wx_i||Wx_w]))} \tag{3.24}$$

Where a^T is a learnable linear transformation that is applied to the features from both source and target nodes concatenated together. This is the original implementation introduced in the GAT paper [??]!!!!!. However, it cannot handle edge features yet. It can be modified to include edge features in learning node embeddings by concatenating the edge features with source and target node features while computing the attention coefficients as follows:

$$\alpha_{ij} = \frac{exp(LeakyReLU(a^T[Wx_i||Wx_j||W_e e_{ij}]))}{\sum_{w \in \mathcal{N}(i)} exp(LeakyReLU(a^T[Wx_i||Wx_w||W_e e_{iw}]))} \tag{3.25}$$

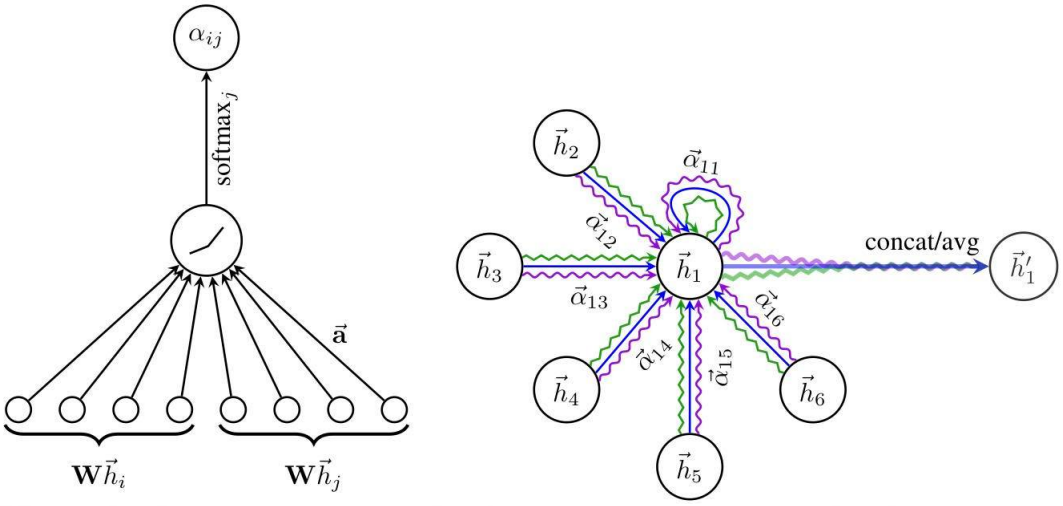

Figure 3.14 Attention mechanism on graphs [8].

W_e is a separate linear transformation that maps edge features of arbitrary dimensionality to match the dimensionality of node embeddings. In practice, this process of computing attention coefficients and node embeddings is repeated many times, and the final node embedding is computed by either averaging or concatenating the intermediate embeddings. This iteration is called multi-head attention, and each iteration is called an attention head. Finally, this kind of attention, where attention coefficients are computed by concatenating the embeddings from source and target nodes and from the edges, is called additive attention.

Another attentional flavor was introduced in the Graph Transformer (GT) architecture [68]. GT differs from GAT in two ways. Instead of using additive attention, it uses dot product or multiplicative attention, and it employs a learnable gating mechanism to weigh the contribution of the source and target nodes in the computation of the target node embedding. Embedding computation goes as follows:

$$\hat{x}_i = \beta_i W_1 x_i + (1 - \beta_i) \left(\sum_{j \in \mathcal{N}(i)} \alpha_{i,j} (W_2 x_j + W_3 e_{ij}) \right) \tag{3.26}$$

with the attention coefficient α_{ij} computed as follows:

$$\alpha_{ij} = \text{softmax} \left(\frac{(W_4 x_i)^T (W_5 x_j + W_3 e_{ij})}{\sqrt{(\mathcal{N}(i) \mathcal{N}(j}} \right) \tag{3.27}$$

and the learnable gating parameter β as follows:

$$\beta_i = \text{sigmoid}(W_6^T[W_1 x_i, m_i, W_1 x_i - m_i]) \tag{3.28}$$

where:

$$m_i = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} W_2 x_j \tag{3.29}$$

GAT and GT introduce a further step ahead of GCN by utilizing edge features of arbitrary dimensionality and by combining messages from both source and target nodes in embedding computation. However, all introduced flavors use a single linear transformation layer for neural message passing, which in some cases might not be enough to learn useful patterns. In the next section, we will discuss Graph Isomorphism Networks (GIN) that provide a general solution for this limitation.

3.3.4 Graph Isomorphism Networks

Graph Isomorphism Networks (GIN) [69] follow the same message-passing scheme, yet take it a step further. The main idea is that a single linear transformation layer is not enough for the learning process in a variety of tasks. So, instead of using a single layer, it uses any arbitrary function that can be applied to each node embedding individually. Additionally, it is similar to GT in the sense that it assigns a weight for the contribution of the source and target embeddings in the target node embedding update. The vanilla version of GIN can be computed as follows:

$$\hat{x}_i = h_\theta \left((1 + \epsilon) \cdot x_i + \sum_{j \in \mathcal{N}(i)} x_j \right) \tag{3.30}$$

where h_θ is any arbitrary differentiable function that can be applied to each node embedding individually and ϵ is a scalar weight that can be assigned empirically or learned during training. This vanilla form does not handle edge features, so it has

been modified to incorporate them as follows [70]:

$$\hat{x}_i = h_\theta \left((1+\epsilon) \cdot x_i + \sum_{j \in \mathcal{N}(i)} \text{ReLU}(x_j + e_{ji}) \right) \tag{3.31}$$

3.3.5 Message Passing GNNs

All the discussed models can be generalized in the sense that they follow the message passing scheme. The message passing scheme consists of two steps:

1. An aggregation function that aggregates the information from the source nodes and combines them into a single message. The aggregation function must be differentiable and permutation invariant. We have seen the sum and mean functions as aggregation functions, but we can employ any other function that retains the same criteria, such as the max, min, or std functions.
2. An update function that applies some transformation to the embeddings of the target node and the message from source nodes to update the embedding at the target node at the current iteration. The only constraint on the update function is that it must be applicable to each node embedding individually. This constraint ensures the idea of parameter sharing in GNN.

The general scheme for message-passing neural networks can be formulated as follows:

$$\hat{x}_i^{l+1} = UPDATE^{(l)} \left(\hat{x}_i^l, AGGREGATE^{(k)}(\{\hat{x}_j^l, \forall j \in \mathcal{N}(i)\}) \right) \tag{3.32}$$

$$\hat{x}_i^{l+1} = UPDATE^{(l)} \left(\hat{x}_i^l, m_{\mathcal{N}(i)}^l \right) \tag{3.33}$$

where $\hat{x}_i^{l+1}$ is the target node embedded in the $l+1$ iteration, $m_{\mathcal{N}(i)}^l$ is the message aggregated from neighboring nodes from the previous iteration, and $UPDATE$ and $AGGREGATE$ functions are any differentiable functions that preserve permutation invariance and permutation equivariance.

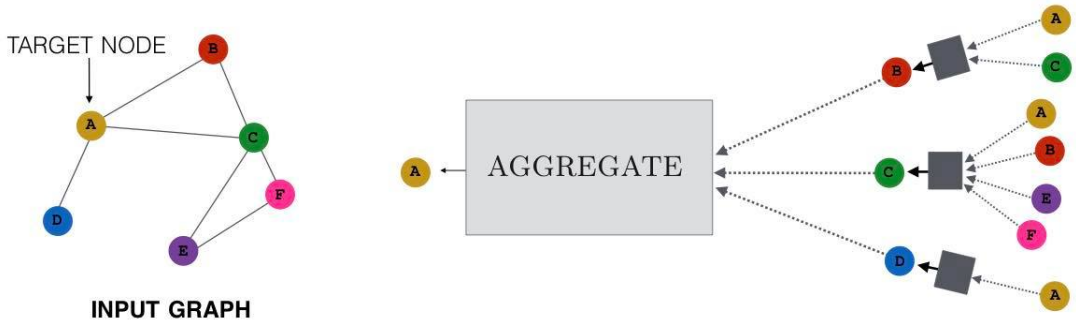

Figure 3.15 The message passing scheme for GNNs [9].

Finally, it is important to note that a single iteration aggregates information from the first-degree neighbors, while at each iteration l, information is aggregated from l hops away. In practice, this is achieved by stacking multiple layers of GNN. Additionally, each layer can have its own learnable weights or the weights can be shared across layers.

3.3.6 Spatio-Temporal Graph Convolutional Neural Networks

All GNN variants introduced so far are very powerful in tasks applied to static graphs but they are very limited when it comes to tasks related to dynamic graphs or graphs that change over time (EEG data for instance). This limitation comes from the fact that GNNs by default assume node features are discrete that don't have temporal dependencies. To overcome this limitation, researchers introduced mixed GNN models that utilize GNNs for spatial information learning and temporal CNNs or temporal attention models for temporal information learning. These models work mainly by stacking spatial (GNN) layers and temporal (CNN) layers in a cascade as in [71],[72].

In this project, a novel spatio-temporal graph convolutional neural network (STGCNN) layer is introduced. We can break it down in accordance with the message-passing scheme introduced in the previous section. We need to formulate update and aggregation functions that can learn both temporal and spatial representations simul-

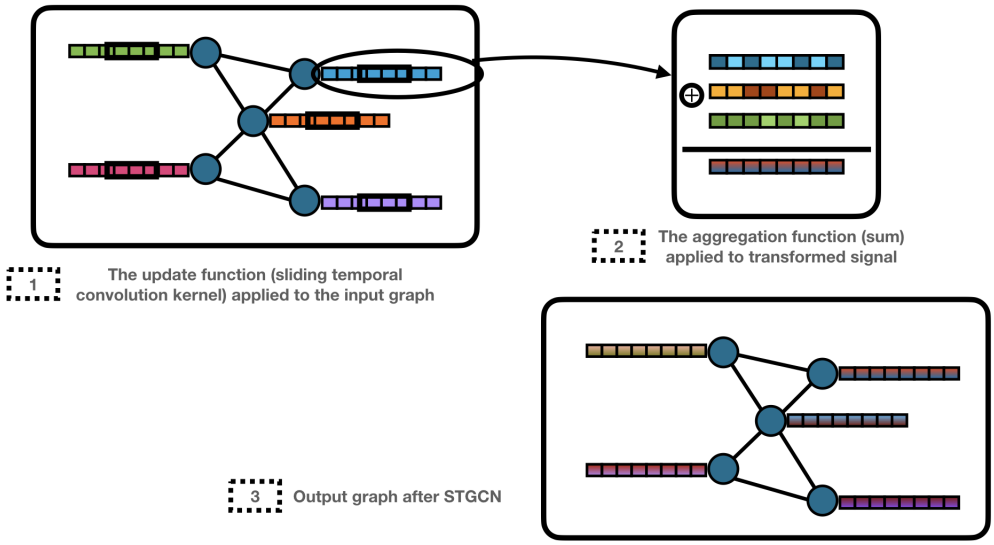

Figure 3.16 An illustration of STGCNN layer.

taneously. We introduce using sliding temporal convolution as an update function while keeping the sum function as the aggregation function. The node embedding can be computed as follows:

$$\hat{x}_i = x_i * k_1 + \sum_{j \in \mathcal{N}(i)} e_{ij}.(x_j * k_2) \tag{3.34}$$

Where $k_1$and k_2 are sliding learnable temporal convolution kernels. It can be written in the matrix form as follows:

$$\hat{X} = X * k_1 + A(X * k_2) \tag{3.35}$$

where A is the weighted adjacency matrix. An illustration of the algorithm is shown in Figure 3.16.

In this way, STGCNN can learn temporal features from EEG data while constrained to respect spatial dependencies at the same time. In other words, it can learn spatiotemporal dynamics from EEG data. Finally, typically in practice, we need to learn multiple filters or kernels to efficiently model the data. This can be achieved very easily with this formulation by using as many learnable kernels k as we need.

4. EXPERIMENTS

4.1 Dataset

The dataset used in this project is the motor imagery EEG dataset from Physionet [10],[73]. This dataset is the largest public BCI motor imagery dataset known today. It contains EEG recordings from 109 subjects while they performed 4 different conditions. Each subject performed 14 two-minute runs of recording. In the first two runs, the subjects did nothing. They were resting state runs, one with eyes open and the other with eyes closed. The subject did one of the following four conditions for three runs in the remaining 12 runs:

1. Right vs. Left Motor Imagery:

   A target appears either on the right or left side of the screen, and the subject responds by imagining opening and closing the corresponding fist until the target fades away.

2. Right vs. Left Motor Execution:

   A target appears either on the right or left side of the screen, and the subject responds by actually opening and closing the corresponding fist until the target fades away.

3. Hands vs. Feet Motor Imagery:

   A target appears either at the top or bottom of the screen, and the subject responds by imagining opening and closing either both fists if the target is on top, or both feet if the target is on the bottom, until the target fades away.

4. Hands vs. Feet Motor Execution:

   A target appears either at the top or bottom of the screen, and the subject responds by actually opening and closing either both fists if the target is on top, or both feet if the target is on the bottom until the target fades away.

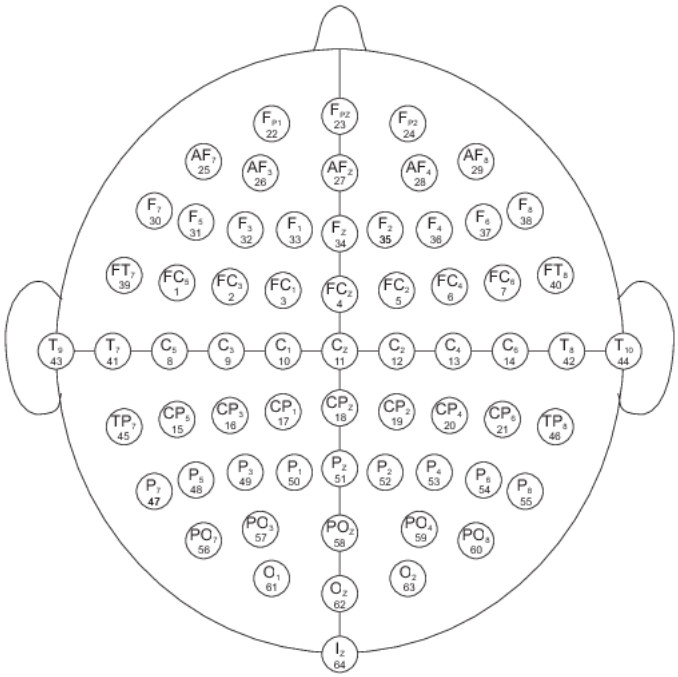

Figure 4.1 EEG electrode positions [10].

There are three dedicated runs for each condition. Each run was divided into successive equal-length blocks of resting state and a task. Each block consumes 4 seconds, so in total, for each run, we have fifteen blocks of resting state and fifteen blocks of tasks for two different conditions. In total, each subject performed 90 four-second long trials of four different motor imagery conditions and another 90 four-second long trials of four different motor execution conditions.

The EEG data was recorded using the BCI2000 system [73], from 64 electrodes and an annotation channel in accordance with the international 10-10 system at a sampling rate of 160 Hz and excluding electrodes Nz, F9, F10, FT9, FT10, A1, A2, TP9, TP10, P9, and P10 as shown in Figure 4.1.

4.2 Pipeline

4.2.1 SVM Classification

Motor imagery runs were downloaded then a common average reference was applied to the date before any further processing. The data for each subject was segmented into trials. Only trials that belong to the first two conditions (right hand vs left hand) were used so we have on average forty five four-seconds motor imagery trials per subject. The data from subjects 38, 43, 88, 89, 92, 100, and 104 were excluded due to inconsistencies in the trial lengths and damaged labels.

An SVM classifier was applied to each subject data individually in a 5-fold cross validation scheme as an exploration of the data for feature extraction. The classifier was applied to the raw data and to the data after bandpass filtering in the 5 frequency bands Delta (0-4 Hz), Theta (4-8 Hz), Alpha (8-12 Hz), Beta (12-30 Hz), and Gamma (30-80 Hz).

Two kinds of features were extracted for SVM classification:

1. **Temporal features**

   The average power of the signals over a non-overlapping 100 ms time intervals were extracted and fed to the classifier.

2. **Spatial features**

   The absolute Pearson correlation matrix was used and then the lower triangle of the correlation matrix was fed to the classifier.

The results of SVM classifier determined the later choice of processing steps.

4.2.2 Preprocessing

Based on SVM classification results, two versions of the data were used:

1. **Raw data**

   Raw data was used with a single preprocessing step, a notch filter at 60 Hz to eliminate the power line noise.

2. **Bandpass filtered data**

   A bandpass filter between 0 and 4 Hz (Delta band) was applied to the raw data.

4.2.3 Nodes

Two variants of node features (time series) were prepared as mentioned is Section 3.2.3.1:

1. **Sensor Nodes**

   Preprocessed scalp EEG time series were considered as connected nodes of a graph without any further processing.

2. **Source Space Nodes**

   Scalp EEG data were projected onto the source space using a precomputed forward model based on the 152-MNI template which has been generated using SPM12 (Statistical Parameter Mapping) software functions and MATLAB scripts developed for minimum norm source estimation (see Section 3.2.3.1). This model has 5124 cortical vertices yielding 5124 cortical time series. Each vertex was mapped to the parcel to which it belonged from the 100-parcels-7-networks Schaefer's brain atlas. Finally, cortical time series that belonged to the same parcel were averaged together in one version of the data and summed together in another, so that we had 96 cortical time series (Four parcels did not have any vertices as a result of this source space reconstruction).

4.2.4 Edges

Two types of connectivity metrics were estimated (as discussed in Section 3.2.3.2:

1. **Functional Connectivity**

   Absolute Pearson correlation (PC) maps were calculated for both sensor and source nodes.

2. **Structural-functional Connectivity**

   Structural connectivity matrices described in Section 3.2.3.2 were element-wise multiplied with PC matrices so that we ended up with an adjacency matrix that had a nonzero weight where there was a structural edge between two nodes and zero otherwise.

4.2.5 Models

4.2.5.1 STGCNN Model. An STGCNN model was designed for classification over EEG graphs. The model consisted of 5 STGCNN layers (introuced in Section 3.3.6). Each layer was followed by a *Batch Normalization*, *Tanh* non-linearity, *Dropout*, and *Average Pooling* functions. Each of the two successive layers was connected with *skip connections* or *residual connections* which meant that the output of the former layer was added to the output of the current layer before fed into the next layer. When the outputs sizes did not match, the output of the former layer was broadcast to the size of the current layer via 1×1 *convolution*. *Residual connections* were employed to prevent gradient vanishing or explosion during training and to prevent node embedding over saturation. After the STGCNN layers, the output was flattened and passed through two linear layers with *Batch Normalization*, *ReLU*, and *Dropout* functions in between. The detailed model architecture is illustrated in Figure 4.2 and in Table 4.1 and the model configuration is presented in Table 4.3.

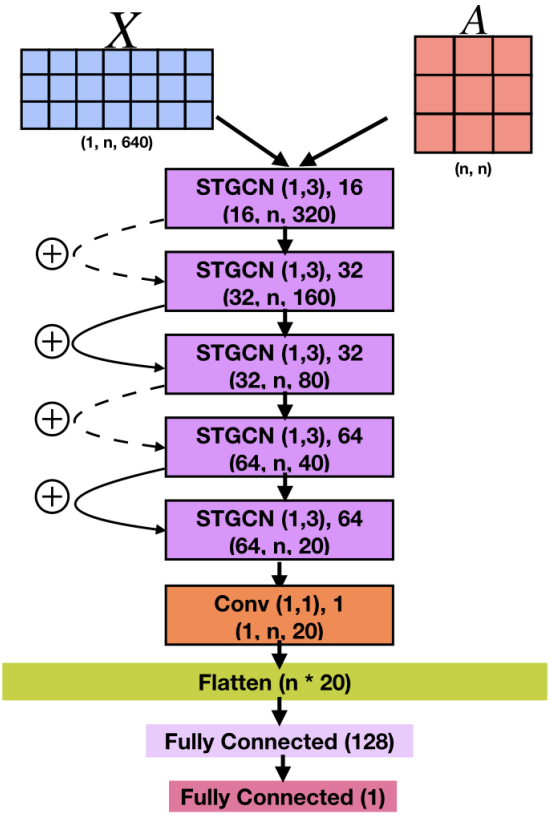

Figure 4.2 STGCNN model architecture (n refers to the number of nodes).

Table 4.1
STGCNN model architecture(n refers to the number of nodes).

Layer	Kernel size	Stride	Dropout	Pooling	Activation	Input size	Output Size
STGCNN	(1, 3)	1	0.12	Avg	Tanh	[1, n, 640]	[16, n, 320]
STGCNN	(1, 3)	1	0.12	Avg	Tanh	[16, n, 320]	[32, n, 160]
STGCNN	(1, 3)	1	0.12	Avg	Tanh	[32, n, 160]	[32, n, 80]
STGCNN	(1, 3)	1	0.12	Avg	Tanh	[32, n, 80]	[64, n, 40]
STGCNN	(1, 3)	1	0.12	Avg	Tanh	[64, n, 40]	[64, n, 20]
CNN	(1, 1)	1	-	-	-	[64, n, 20]	[1, n, 20]
Flatten	-	-	-	-	-	[1, n, 20]	[1, n*20]
Linear	-	-	0.12	-	ReLU	[1, n*20]	[1, 128]
Linear	-	-	-	-	ReLU	[1, 128]	[1, 1]

4.2.5.2 CNN Model. As a baseline, another identical model was employed but instead of using STGCNN layers, 1D Convolution layers were employed. The model architecture is shown in Figure 4.3 and in Table 4.2. The model has the same hyper-

parameters configuration as the STGCNN model as shown in Table 4.3.

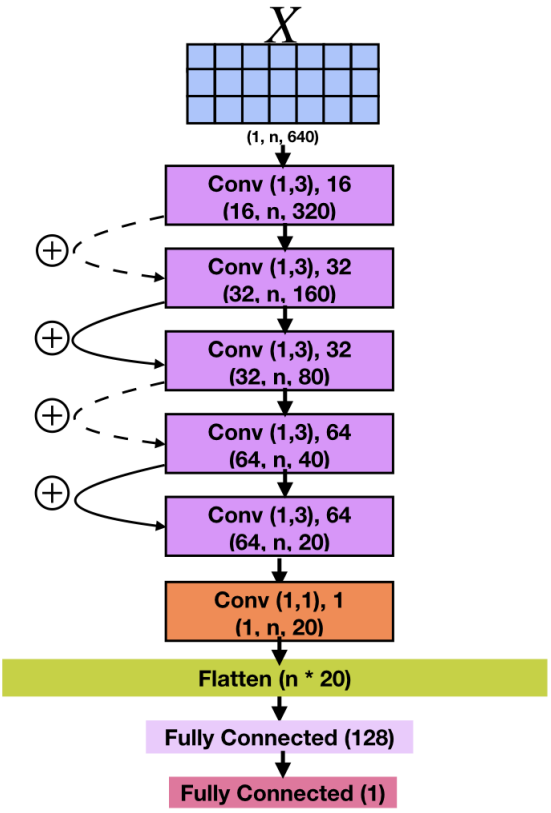

Figure 4.3 CNN model architecture (n refers to the number of nodes).

Table 4.2
CNN model architecture (n refers to the number of nodes).

Layer	Kernel size	Stride	Dropout	Pooling	Activation	Input size	Output Size
CNN	(1, 3)	1	0.12	Avg	Tanh	[1, n, 640]	[16, n, 320]
CNN	(1, 3)	1	0.12	Avg	Tanh	[16, n, 320]	[32, n, 160]
CNN	(1, 3)	1	0.12	Avg	Tanh	[32, n, 160]	[32, n, 80]
CNN	(1, 3)	1	0.12	Avg	Tanh	[32, n, 80]	[64, n, 40]
CNN	(1, 3)	1	0.12	Avg	Tanh	[64, n, 40]	[64, n, 20]
CNN	(1, 1)	1	-	-	-	[64, n, 20]	[1, n, 20]
Flatten	-	-	-	-	-	[1, n, 20]	[1, n*20]
Linear	-	-	0.12	-	ReLU	[1, n*20]	[1, 128]
Linear	-	-	-	-	ReLU	[1, 128]	[1, 1]

Table 4.3
STGCNN and CNN models hyperparameters configuration.

	CNN model on scalp EEG	All other experiments
Learning Rate (lr)	0.01	0.006
Optimizer	Adam	Adam
L2 Regularization gamma γ	0.08	0.02
lr Scheduler	Exponential decay	Exponential decay
lr Scheduler gamma γ	0.67	0.5
Batch size	64	64

4.2.6 Experiments

Subjects were shuffled and divided into 5 folds to employ 5-fold cross validation. Both models were trained on scalp EEG and cortical EEG individually. Additionally, each model was trained on each network EEG series individually to test its contribution. Finally, STGCNN model was trained using two kinds of connectivity mentioned in Section 4.2 to test for the optimum network settings. All data analysis was performed using Python 3.7 programming language and its scientific computing ecosystem `NumPy`, and `SciPy` libraries. All models were written in `PyTorch` package and trained using `PyTorch Lightning` library. Finally, the models hyperparameters were optimized using the `Optuna` library [74].

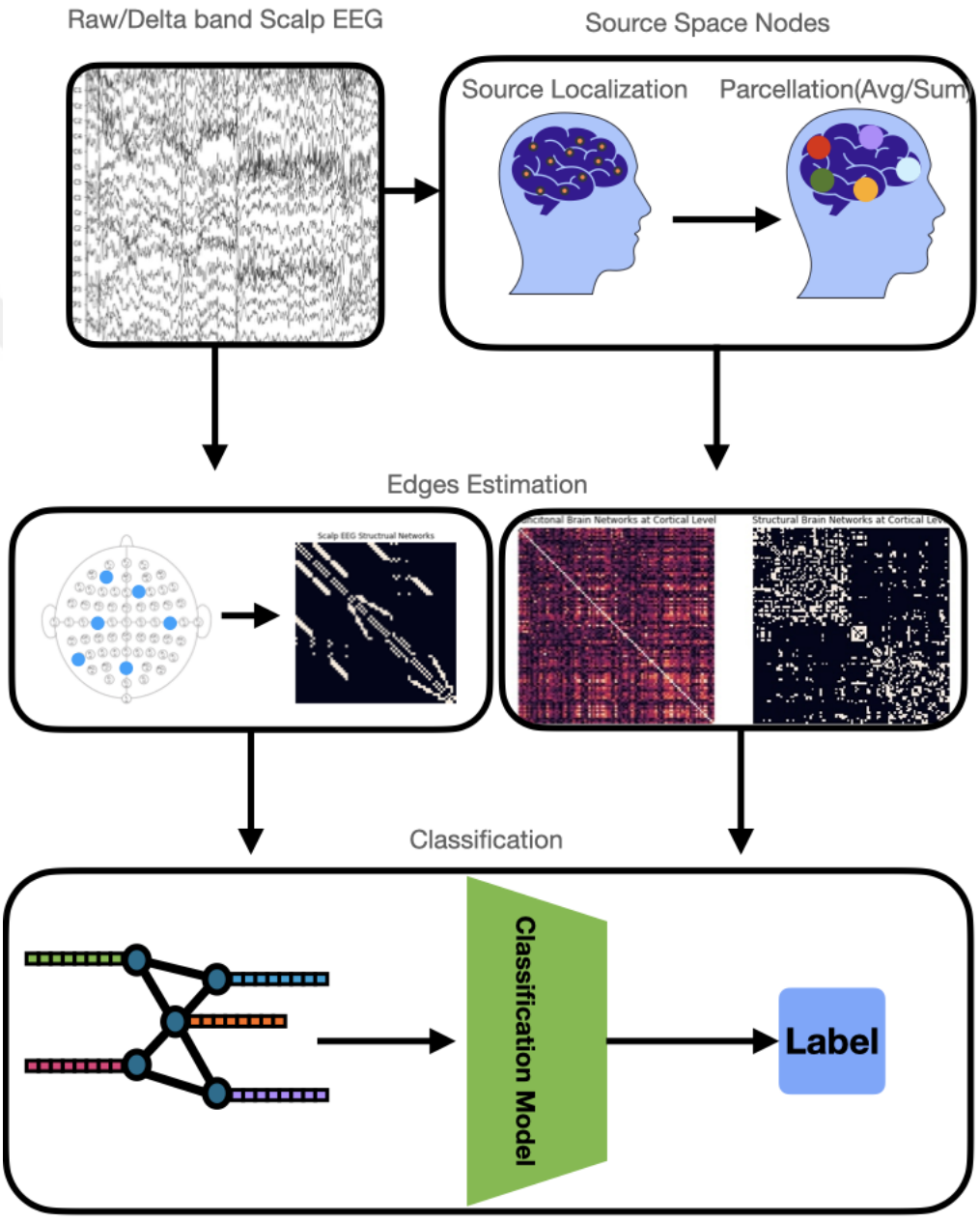

Figure 4.4 A summary of the classification pipeline.

5. RESULTS

SVM classifier yielded its highest classification accuracy rates using the mean power of the signals over non-overlapping 100 ms intervals of i) delta band filtered and ii) the raw scalp EEG data, as 73.83% and 72.28%, respectively. On the other hand, it yielded 60.21% and 58.22% using the absolute Pearson correlation maps applied to the above conditions of data. It yielded chance level accuracy rates on the other frequency bands with both types of features. Using these results as a guideline, STGCNN and CNN models were trained only on the raw and the delta band filtered EEG data in both the sensor and source spaces. The detailed SVM classification results are shown in Table 5.1.

Table 5.1
SVM Classification Accuracy Results.

Features	Average power of the signal		Absolute Pearson correlation	
type	Mean	STD	Mean	STD
Raw data	72.28	14.92	58.22	13.72
Delta band	73.83	14.67	60.21	13.92
Theta band	52.38	8.59	56.18	11.97
Alpha band	50.50	8.09	56.12	11.23
Beta band	49.92	6.49	54.22	11.25
Gamma band	48.76	6.27	47.00	8.27

In the sensor space, the highest classification accuracy was achieved by training the STGCNN model using structural-functional connectivity on the raw EEG data and on delta band EEG data as 82.13% and 81.71%, respectively. The lowest result achieved was 80.88% and was obtained by training the CNN model on the raw EEG data. The detailed results in the sensor space are shown in Table 5.2.

In the source space, training STGCNN using the structural-functional connectivity on the raw cortical EEG data yielded the highest average accuracy of 79.27% and 79.18% when the data were summed or averaged over the vertices in each parcel,

Table 5.2
Classification accuracy percentages in the sensor space (5-fold cross validation).

	Raw Data		Delta Band	
	Mean	STD	Mean	STD
STGCNN - Functional	81.60	2.11	81.56	2.72
STGCNN - Structural-Functional	82.13	2.39	81.71	2.57
CNN	80.88	2.37	81.65	2.35

respectively. These results outperformed the CNN model results in the same settings which yielded 79.17% and 79.04% by summing and averaging, respectively. Training STGCNN and CNN models on each network individually yielded variable results. The highest performance was achieved by the control network with an average accuracy of 78.18% with training the CNN model and by summing the data over the vertices of each parcel. The second highest network was the ventral attention network with an average accuracy of 74.94% by training the CNN model and summing. The third highest network was the default mode network with an average accuracy of 74.73% by training the STGCNN model and summing. Additionally, the somatomotor and limbic networks yielded significantly lower results than the aforementioned networks with an average accuracy of 67.5% and 65.34% for the somatomotor and limbic networks, respectively by training the CNN model and averaging over vertices. On the other hand, the visual and dorsal attention networks almost achieved a chance level accuracy levels with both models. Finally, using the top three achieving networks for training both models achieved the highest accuracy rate of 79.17% while training the CNN model and summing. The detailed results for training STGCNN model on the raw data using the average parcellation and sum parcellation is shown in Table 5.3 and Table 5.4, respectively. The results of training the CNN model on the raw data are shown in Table 5.5.

In the source space, when trained using the structural-functional connectivity and averaging, STGCNN using the delta band source space EEG data yielded the highest average accuracies of 79.13% and 78.99%, respectively. These results outperformed

Table 5.3
STGCNN classification accuracy percentages on raw data in the source space (average parcellation).

	Functional		Structural-Functional	
	Mean	STD	Mean	STD
Cortical EEG	78.71	2.23	79.18	1.30
Visual network	54.71	2.24	55.56	1.90
Somatomotor network	66.08	3.28	66.71	2.81
Default mode network	73.92	3.21	74.52	2.75
Control network	77.49	1.50	77.48	1.85
Limbic network	64.48	2.06	63.60	2.12
Ventral attention network	74.64	2.86	74.27	2.38
Dorsal attention network	50.09	2.64	49.81	2.28
Top 3 networks	78.84	1.95	79.11	2.35

Table 5.4
STGCNN classification accuracy percentages on raw data in the source space (sum parcellation).

	Functional		Structural-Functional	
	Mean	STD	Mean	STD
Cortical EEG	78.96	1.83	79.27	1.55
Visual network	54.09	2.20	55.73	2.91
Somatomotor network	66.06	3.24	66.66	3.70
Default mode network	73.80	3.06	74.73	2.75
Control network	78.14	1.51	77.63	1.99
Limbic network	64.62	2.22	63.01	2.33
Ventral attention network	74.44	2.97	74.42	2.68
Dorsal attention network	50.36	1.85	49.18	2.10
Top 3 networks	78.77	2.67	78.90	2.27

the CNN model results in the same settings which achieved 77.73% and 78.56% by averaging and summing, respectively. Similar to training on unfiltered EEG data, training STGCNN and CNN models on each network individually yielded variable results. The control network with an average accuracy of 78.28% by training the STGCNN model using the functional connectivity and summing yielded the best result. The second

Table 5.5
CNN classification accuracy percentages on raw source data.

	Average Parcellation		Sum Parcellation	
	Mean	STD	Mean	STD
Cortical EEG	78.90	2.47	78.59	2.40
Visual network	53.67	1.79	54.11	1.75
Somatomotor network	67.50	2.39	67.43	2.30
Default mode network	74.09	3.22	74.37	2.93
Control network	78.16	2.20	78.18	2.32
Limbic network	65.34	2.98	64.86	3.38
Ventral attention network	74.93	3.01	74.94	3.57
Dorsal attention network	51.50	2.12	51.69	2.44
Top 3 networks	79.04	2.14	79.17	2.33

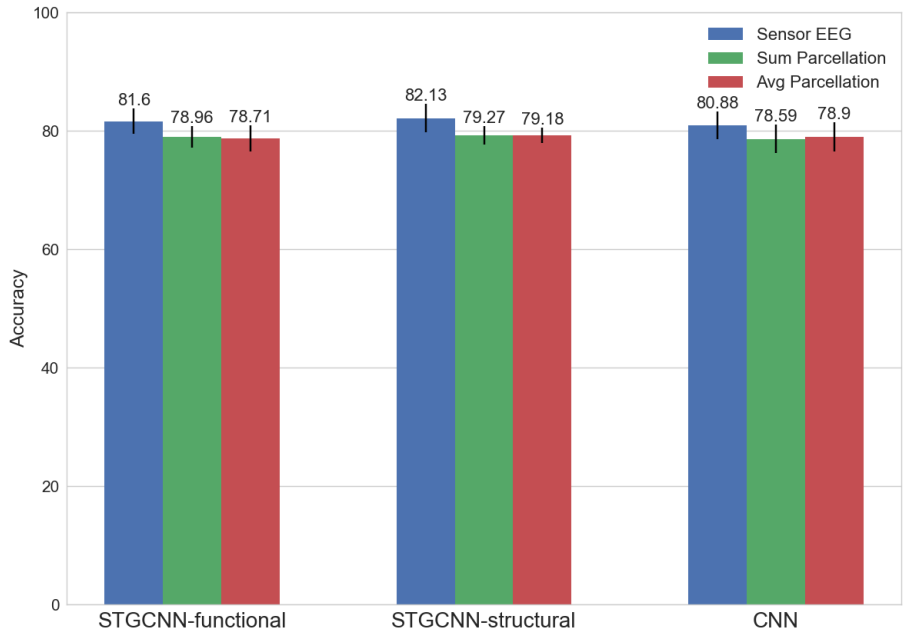

Figure 5.1 A summary of STGCNN and CNN accuracy results in both sensor and source spaces.

highest was the ventral attention network with an average accuracy of 75% by training the CNN model and summing. The third highest was the default mode network with an average accuracy of 74.15% by training the CNN model and summing. Additionally,

the somatomotor and limbic networks yielded significantly lower results than the aforementioned networks. The somatomotor yielded an average accuracy of 67.42% by the CNN and averaging and the limbic by STGCNN and the functional connectivity and the averaging yielded 63.96%. On the other hand, the visual and dorsal attention networks almost achieved a chance level accuracy levels with both models. Finally, using the top three achieving networks for training both models achieved its highest accuracy rate of 79.05% while training the STGCNN model using the functional connectivity and summing. The detailed results for training STGCNN model on the raw data using the averaging and summing are shown in Table 5.6 and Table 5.7, respectively. The results of training the CNN model on the raw data are shown in Table 5.8.

Table 5.6
STGCNN classification accuracy percentages on delta band data in the source space (average parcellation).

	Functional		Structural-Functional	
	Mean	STD	Mean	STD
Cortical EEG	78.80	1.97	79.13	1.41
Visual network	53.81	1.33	54.13	2.20
Somatomotor network	66.25	2.45	66.22	2.78
Default mode network	73.50	2.32	73.77	2.38
Control network	77.88	1.80	77.63	1.84
Limbic network	63.96	1.70	63.90	2.72
Ventral attention network	74.47	2.71	74.08	2.79
Dorsal attention network	49.02	1.78	49.90	2.95
Top 3 networks	78.82	2.00	78.91	1.87

Table 5.7
STGCNN classification accuracy percentages on delta band data in the source space (sum parcellation).

	Functional		Structural-Functional	
	Mean	STD	Mean	STD
Cortical EEG	78.38	2.00	78.99	1.47
Visual network	54.01	2.42	54.04	0.94
Somatomotor network	66.39	3.43	66.39	2.34
Default mode network	73.24	2.71	73.70	2.41
Control network	78.28	1.96	77.79	1.78
Limbic network	62.71	3.47	63.06	2.88
Ventral attention network	74.80	3.06	74.37	2.49
Dorsal attention network	49.44	2.43	49.67	3.13
Top 3 networks	79.05	1.83	78.90	1.98

Table 5.8
CNN classification accuracy percentages on delta band source data.

	Average Parcellation		Sum Parcellation	
	Mean	STD	Mean	STD
Cortical EEG	77.73	2.07	78.56	2.07
Visual network	53.72	1.97	54.71	1.90
Somatomotor network	67.42	2.00	67.28	1.93
Default mode network	73.95	2.24	74.15	2.46
Control network	78.16	2.11	77.75	1.84
Limbic network	63.89	4.08	63.72	4.59
Ventral attention network	74.25	3.35	75.00	3.45
Dorsal attention network	51.56	2.48	52.07	2.04
Top 3 networks	78.92	2.47	79.01	2.38

6. DISCUSSION

In this study, a brain-network based classification pipeline for EEG data was developed. First, a method was implemented to represent the brain networks using EEG data in both the sensor and source spaces using graph theory and neuroimaging. Second, a novel spatiotemporal graph convolutional neural network model (STGCNN) was designed for the classification of motor imagery EEG data. Additionally, a convolutional neural network (CNN) classification model was employed as a baseline for testing the proposed STGCNN model.

The results generally outline that STGCNN shows a slightly better Äerfromance than CNN in both sensor and source spaces ($\approx 1.25\%$ vs. $\approx 0.7\%$). It is also observed that sensor space results are slightly better than those in the source space by $\approx 2.9\%$ and $\approx 1.7\%$ for STGCNN and CNN, respectively. The results also show that employing combined structural and functional connectivity yields a minor ($\approx 0.4\%$) increase in performance than utilizing only functional connectivity. Additionally, the results show that training the model on individual brain networks produces almost identical results regardless of the model used or using only the functional or structural and functional together. However, when training either model using the data belonging to the top three performing networks, the results are very similar to those obtained using the entire data of all networks. Finally, the results show that filtering the data do not bring a considerable improvement over using raw data.

STGCNN outperforming CNN supports the main hypothesis of this study that models which can learn spatio-temporal features and leverage the hidden network structure in EEG data would yield higher classification accuracy rates. STGCNN operates by passing information among nodes in a dynamic way over time, that shows its capability of capturing changes in EEG dynamics with high temporal and spatial precision. Another advantage of STGCNN is that it can learn permutation invariant feature representations of EEG data. This means any change in the order of the nodes will not

affect the output of the model. On the contrary, spatiotemporal models based on 2D CNN are very sensitive to the ordering of the electrodes or the nodes of EEG data and can produce very different output feature representations with even any slight change in the order of the electrodes.

One of the major expectations of this study was that source space data classification would yield better results than those in the sensor space. There might be several plausible explanations for its failure to be fulfilled. First, there might be a source of error due to the forward and inverse methods employed. Typically, when source localization methods are employed in EEG classification, specific regions of interest that depend on the task are chosen. This approach minimizes the risk of anatomical errors affecting the classification accuracy. Second, Schaefer's parcellation groups together cortical vertices that share homogeneous resting state functional connectivity patterns. Hence, summing or averaging the cortical activations over the vertices belonging to the same parcel may reduce the main discriminative characteristics of the signals thereby introducing inaccurate functional connectivity links among different parcels. On the other hand, scalp EEG time series do not suffer from such a potential source of confusion.

Employing both structural and functional connectivity in graph modeling of EEG data performs slightly better than the functional connectivity used alone. From a biological perspective, neurons that respond to the same stimulus tend to connect together in neuronal ensembles as in Hebb's assembly theory [75]. Utilizing both structural and functional connectivity metrics increases the chances of capturing the network dynamics giving rise to the task. From a technical point of view, this combination induces sparsity on the constructed graphs which in turn increases the efficiency of GNN models. GNN models learn by aggregating information from nodes that share edges together. When the graph is too dense, this increases the risk of GNN over-smoothing which means all nodes have similar or identical representations. This over-smoothing decreases the discrimination power of GNN models.

Training either STGCNN model or CNN model on individual networks data

yields equal results in different settings. Each individual network has a few number of parcels or nodes which in turn are highly connected. This results in graphs that are very dense and almost fully connected graphs. In this case, there is no difference between employing functional networks only or employing both functional and structural networks together. In practice, they tend to be the same in small dense graphs. Additionally, the dense structure of the graphs leads to over-smoothing of the GNN models which impairs the efficiency of GNN in classification. However, training either models on the top three networks yield accuracy rates that are comparable with those obtained by training the entire source space EEG data. The resulting graphs in this settings are sparse enough that they benefit from the GNN representing the most relevant information that can give rise to high accuracy. This is an important issue for dimensionality and complexity reduction affecting the computational costs.

Finally, filtering the data does not add improvement to STGCNN and CNN methods compared with the improvement obtained using SVM. This is simply because both STGCNN and CNN operate by learning spatiotemporal and temporal filters, repectively from the data. In essence, it is a filtering operation but instead of using a pre-defined filter, it learns the filter via training on the data.

7. CONCLUSION AND FUTURE WORK

In this study, a novel spatiotemporal graph convolutional neural network model is introduced for motor imagery EEG data classification. The model is evaluated on EEG data in the sensor space first. Then the data are projected upon the source space after source localization and spatial averaging over the cortical locations in each parcel defined by Schaefer's brain atlas. The model outperforms typical convolutional neural network models in classification accuracy.

In the future, the approach adopted in this study can be furthered in several directions. Source reconstruction algorithms can be refined to incorporate the spatiotemporal properties of the EEG data instead of using only their instantaneous values. Also, an alternative way of grouping can be employed for the cortical vertices which might be more specific to motor execution or imagery instead of the the parcellations obtained during the resting state. Additionally, the model can be tested on multiple datasets and different tasks like clinical or emotion recognition EEG data. Finally, the model can be combined with transfer learning to be trained on EEG data of real movements and then fine tuned on motor imagery EEG datasets. This might improve the EEG classification algorithms significantly.

REFERENCES

1. Shih, J. J., D. J. Krusienski, and J. R. Wolpaw, "Brain-computer interfaces in medicine," *Mayo Clinic Proceedings*, Vol. 87, p. 268, 2012.

2. Wikipedia contributors, "Electroencephalography — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Electroencephalographyoldid=1107189348, 2022. [Online; accessed 3-September-2022].

3. Sporns, O., *Networks of the Brain*, pp. 18–64. Feb 2016.

4. Sanchez-Lengeling, B., E. Reif, A. Pearce, *et al.*, "A gentle introduction to graph neural networks," *Distill*, Vol. 6, p. e33, 9 2021.

5. Bassett, D. S., and O. Sporns, "Network neuroscience," *Nature Neuroscience 2017 20:3*, Vol. 20, pp. 353–364, 2 2017.

6. Schaefer, A., R. Kong, E. M. Gordon, *et al.*, "Local-global parcellation of the human cerebral cortex from intrinsic functional connectivity mri," *Cerebral Cortex*, Vol. 28, pp. 3095–3114, 9 2018.

7. Bronstein, M. M., J. Bruna, T. Cohen, *et al.*, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," April 2021.

8. Veličković, P., G. Cucurull, A. Casanova, *et al.*, "Graph attention networks," *ArXiv Preprint arXiv:1710.10903*, 2017.

9. Hamilton, W. L., "Graph representation learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Vol. 14, no. 3, pp. 1–159.

10. Goldberger, A. L., L. A. Amaral, L. Glass, *et al.*, "Physiobank, physiotoolkit, and physionet," *Circulation*, Vol. 101, 6 2000.

11. Wikipedia contributors, "Brain-computer interface — Wikipedia, the free encyclopedia," 2022. [Online; accessed 3-September-2022].

12. Abdulkader, S. N., A. Atia, and M. S. M. Mostafa, "Brain computer interfacing: Applications and challenges," *Egyptian Informatics Journal*, Vol. 16, pp. 213–230, 7 2015.

13. Bozinovski, S., "Signal processing robotics using signals generated by a human head: From pioneering works to EEG-based emulation of digital circuits," in *Advances in Robot Design and Intelligent Control* (Rodić, A., and T. Borangiu, eds.), (Cham), pp. 449–462, Springer International Publishing, 2017.

14. Kirasirova, L., V. Bulanov, A. Ossadtchi, *et al.*, "A p300 brain-computer interface with a reduced visual field," *Frontiers in Neuroscience*, Vol. 14, p. 1246, 12 2020.

15. Fazel-Rezai, R., B. Z. Allison, C. Guger, *et al.*, "P300 brain computer interface: current challenges and emerging trends," *Frontiers in Neuroengineering*, Vol. 5, pp. 1–30, 6 2012.

16. Houssein, E. H., A. Hammad, and A. A. Ali, "Human emotion recognition from EEG-based brain-computer interface using machine learning: a comprehensive review," *Neural Computing and Applications*, Vol. 34, pp. 12527–12557, 5 2022.

17. Padfield, N., J. Zabalza, H. Zhao, *et al.*, "EEG-based brain-computer interfaces using motor-imagery: Techniques and challenges," *Sensors (Basel, Switzerland)*, Vol. 19, 3 2019.

18. Cohen, M. X., *Analyzing Neural Time Series Data*, pp. 15–28. Jan 17 2014.

19. Schomer, D. L., and F. L. da Silva, *Niedermeyer's Electroencephalography*, pp. 1–16. Dec 22 2010.

20. Kevric, J., and A. Subasi, "Comparison of signal decomposition methods in classification of EEG signals for motor-imagery bci system," *Biomedical Signal Processing and Control*, Vol. 31, pp. 398–406, 1 2017.

21. Krusienski, D. J., D. J. McFarland, and J. R. Wolpaw, "An evaluation of autoregressive spectral estimation model order for brain-computer interface applications," in *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1323–1326, IEEE, 2006.

22. Rodríguez-Bermúdez, G., and P. J. García-Laencina, "Automatic and adaptive classification of electroencephalographic signals for brain computer interfaces," *Journal of Medical Systems*, Vol. 36, no. 1, pp. 51–63, 2012.

23. Oikonomou, V. P., K. Georgiadis, G. Liaros, , *et al.*, "A comparison study on EEG signal processing techniques using motor imagery EEG data," in *2017 IEEE 30th International Symposium on Computer-based Medical Systems (CBMS)*, pp. 781–786, IEEE, 2017.

24. Graimann, B., B. Allison, and G. Pfurtscheller, "Brain–computer interfaces: A gentle introduction," in *Brain-computer Interfaces*, pp. 1–27, Springer, 2009.

25. Wang, Y., X. Li, H. Li, *et al.*, "Feature extraction of motor imagery electroencephalography based on time-frequency-space domains," *Journal of Biomedical Engineering*, Vol. 31, no. 5, pp. 955–961, 2014.

26. Xu, B. G., A. G. Song, and S. M. Fei, "Feature extraction and classification of EEG in online brain-computer interface," *ACTA ELECTONICA SINICA*, Vol. 39, no. 5, p. 1025, 2011.

27. Kumar, S., R. Sharma, A. Sharma, *et al.*, "Decimation filter with common spatial pattern and fishers discriminant analysis for motor imagery classification," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 2090–2095, IEEE, 2016.

28. Kumar, S., A. Sharma, and T. Tsunoda, "An improved discriminative filter bank selection approach for motor imagery EEG signal classification using mutual information," *BMC Bioinformatics*, Vol. 18, 12 2017.

29. Zhang, Y., Y. Wang, J. Jin, *et al.*, "Sparse bayesian learning for obtaining sparsity of EEG frequency bands based feature vectors in motor imagery classification," *International Journal of Neural Systems*, Vol. 27, no. 02, p. 1650032, 2017.

30. Yu, X., P. Chum, and K. B. Sim, "Analysis the effect of PCA for feature reduction in non-stationary EEG based motor imagery of BCI system," *Optik*, Vol. 125, no. 3, pp. 1498–1502, 2014.

31. Baig, M. Z., N. Aslam, H. P. Shum, *et al.*, "Differential evolution algorithm as a tool for optimal feature subset selection in motor imagery eeg," *Expert Systems with Applications*, Vol. 90, pp. 184–195, 2017.

32. Guo, X., X. Wu, X. Gong, *et al.*, "Envelope detection based on online ica algorithm and its application to motor imagery classification," in *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, pp. 1058–1061, IEEE, 2013.

33. Chen, G., J. Zhang, and L. Yao, "An improved feature extraction method for self-paced brain-computer interface application," in *2009 ICME International Conference on Complex Medical Engineering*, pp. 1–6, IEEE, 2009.

34. Karaboga, D., "An idea based on honey bee swarm for numerical optimization," tech. rep., Technical Report-tr06, Erciyes University, Engineering Faculty, 2005.

35. Ilyas, M. Z., P. Saad, M. I. Ahmad, *et al.*, "Classification of EEG signals for brain-computer interface applications: Performance comparison," *Proceedings of 2016 International Conference on Robotics, Automation and Sciences, ICORAS 2016*, 3 2017.

36. Batres-Mendoza, P., C. R. Montoro-Sanjose, E. Guerra-Hernandez, *et al.*, "Quaternion-based signal analysis for motor imagery classification from electroencephalographic signals," *Sensors*, Vol. 16, no. 3, p. 336, 2016.

37. Novi, Q., C. Guan, T. H. Dat, *et al.*, "Sub-band common spatial pattern (sbcsp) for brain-computer interface," in *2007 3rd International IEEE/EMBS Conference on Neural Engineering*, pp. 204–207, IEEE, 2007.

38. Sohaib, A. T., S. Qureshi, J. Hagelback, *et al.*, "Evaluating classifiers for emotion recognition using EEG," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 8027 LNAI, pp. 492–501, 2013.

39. Yang, H., S. Sakhavi, K. K. Ang, *et al.*, "On the use of convolutional neural networks and augmented CSP features for multi-class motor imagery of EEG signals classification," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, Vol. 2015-November, pp. 2620–2623, 11 2015.

40. Tabar, Y. R., and U. Halici, "A novel deep learning approach for classification of EEG motor imagery signals," *Journal of Neural Engineering*, Vol. 14, no. 1, p. 016003, 2016.

41. Tang, Z., C. Li, and S. Sun, "Single-trial EEG classification of motor imagery using deep convolutional neural networks," *Optik*, Vol. 130, pp. 11–18, 2 2017.

42. Schirrmeister, R. T., J. T. Springenberg, L. Dominique, *et al.*, "Deep learning with convolutional neural networks for EEG decoding and visualization," *Wiley Online Library*, Vol. 38, pp. 5391–5420, 11 2017.

43. Bashivan, P., I. Rish, M. Yeasin, *et al.*, "Learning representations from EEG with deep recurrent-convolutional neural networks," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 11 2015.

44. Dai, M., D. Zheng, R. Na, *et al.*, "EEG classification of motor imagery using a novel deep learning framework," *Sensors*, Vol. 19, no. 3, p. 551, 2019.

45. Ditthapron, A., N. Banluesombatkul, S. Ketrat, *et al.*, "Universal joint feature extraction for P300 EEG classification using multi-task autoencoder," *IEEE Access*, Vol. 7, pp. 68415–68428, 2019.

46. Li, Y., N. Zhong, D. Taniar, *et al.*, "Mutualgraphnet: a novel model for motor imagery classification," *ArXiv Preprint arXiv:2109.04361*, 2021.

47. Jang, S., S. E. Moon, and J. S. Lee, "EEG-based emotional video classification via learning connectivity structure," *IEEE Transactions on Affective Computing*, pp. 1–1, 11 2021.

48. Demir, A., T. Koike-Akino, Y. Wang, *et al.*, "EEG-GNN: Graph neural networks for classification of electroencephalogram (EEG) signals," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, pp. 1061–1067, 2021.

49. Samanta, P., "Introduction to graph theory," 03 2011.

50. Friston, K. J., "Functional and effective connectivity in neuroimaging: A synthesis," *Human Brain Mapping*, Vol. 2, pp. 56–78, 1 1994.

51. Cohen, A. L., D. A. Fair, N. U. Dosenbach, *et al.*, "Defining functional areas in individual human brains using resting functional connectivity mri," *NeuroImage*, Vol. 41, pp. 45–57, 5 2008.

52. Nelson, S. M., N. U. Dosenbach, A. L. Cohen, *et al.*, "Role of the anterior insula in task-level control and focal attention," *Brain Structure Function*, Vol. 214, p. 669, 2010.

53. Gordon, E. M., T. O. Laumann, A. Babatunde, *et al.*, "Generation and evaluation of a cortical area parcellation from resting-state correlations," *Cerebral Cortex (New York, N.Y. : 1991)*, Vol. 26, pp. 288–303, 1 2016.

54. Craddock, R. C., G. A. James, P. E. Holtzheimer, *et al.*, "A whole brain fmri atlas generated via spatially constrained spectral clustering," *Human Brain Mapping*, Vol. 33, pp. 1914–1928, 2012.

55. Honnorat, N., H. Eavani, T. D. Satterthwaite, *et al.*, "Grasp: Geodesic graph-based segmentation with shape priors for the functional parcellation of the cortex," *NeuroImage*, Vol. 106, p. 207, 2 2015.

56. Yeo, B. T. T., F. M. Krienen, J. Sepulcre, *et al.*, "The organization of the human cerebral cortex estimated by intrinsic functional connectivity," *Journal of Neurophysiology*, Vol. 106, pp. 1125–1165, 9 2011.

57. Hämäläinen, M. S., and R. J. Ilmoniemi, "Interpreting magnetic fields of the brain: minimum norm estimates," *Medical & Biological Engineering & Computing*, Vol. 32, no. 1, pp. 35–42, 1994.

58. Dale, A. M., A. K. Liu, B. R. Fischl, *et al.*, "Dynamic statistical parametric mapping: Combining fmri and meg for high-resolution imaging of cortical activity," *Neuron*, Vol. 26, pp. 55–67, 4 2000.

59. Pascual-Marqui, R. D., D. Lehmann, M. Koukkou, *et al.*, "Assessing interactions in the brain with exact low-resolution electromagnetic tomography," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 369, pp. 3768–3784, 10 2011.

60. Pascual-Marqui, R. D., *et al.*, "Standardized low-resolution brain electromagnetic tomography (sloreta): technical details," *Methods Find Exp Clin Pharmacol*, Vol. 24, no. Suppl D, pp. 5–12, 2002.

61. Hassan, M., and F. Wendling, "Electroencephalography source connectivity: Aiming for high resolution of brain networks in time and space," *IEEE Signal Processing Magazine*, Vol. 35, pp. 81–96, 5 2018.

62. Wikipedia Authors, "Pearson correlation coefficient - wikipedia."

63. Luppi, A. I., and E. A. Stamatakis, "Combining network topology and information theory to construct representative brain networks," *Network Neuroscience*, Vol. 5, pp. 96–124, 2 2021.

64. Daigavane, A., B. Ravindran, and G. Aggarwal, "Understanding convolutions on graphs," *Distill*, Vol. 6, p. e32, 9 2021.

65. Moka, S., and B. L. Yoni Nazarathy, "5 convolutional neural networks | the mathematical engineering of deep learning."

66. Defferrard, M., X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in Neural Information Processing Systems*, pp. 3844–3852, 6 2016.

67. Kipf, T. N., and M. Welling, "Semi-supervised classification with graph convolutional networks," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 9 2016.

68. Shi, Y., Z. Huang, S. Feng, *et al.*, "Masked label prediction: Unified message passing model for semi-supervised classification," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 1548–1554, 9 2021.

69. Xu, K., S. Jegelka, W. Hu, *et al.*, "How powerful are graph neural networks?," *7th International Conference on Learning Representations, ICLR 2019*, 10 2018.

70. Hu, W., B. Liu, J. Gomes, *et al.*, "Strategies for pre-training graph neural networks," *ArXiv Preprint arXiv:1905.12265*, 2019.

71. Wu, Z., S. Pan, G. Long, *et al.*, "Connecting the dots: Multivariate time series forecasting with graph neural networks," *CoRR*, Vol. abs/2005.11650, 2020.

72. Yu, B., H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting," *CoRR*, Vol. abs/1709.04875, 2017.

73. Schalk, G., D. J. McFarland, T. Hinterberger, *et al.*, "Bci2000: a general-purpose brain-computer interface (BCI) system," *IEEE transactions on bio-medical engineering*, Vol. 51, pp. 1034–1043, 6 2004.

74. Akiba, T., S. Sano, T. Yanase, *et al.*, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

75. Hebb, D. O., *The Organization of Behavior*, Psychology Press, Apr 2005.