PRECLINICAL ALZHEIMER’S DISEASE PREDICTION USING GRAPH NEURAL NETWORKS

A Master's Thesis
submitted to the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by
Oscar Pina

In partial fulfilment
of the requirements for the degree of
MASTER IN ADVANCED TELECOMMUNICATION TECHNOLOGIES

Advisor: Verónica Vilaplana

Barcelona, July 2020

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

telecos BCN

Abstract

Alzheimer's disease (AD) is the most common form of dementia and it is considered as a biological continuum that can begin decades before the first cognitive symptoms. The detection of healthy but amyloid positive individuals is an opportunity for the prevention of the disease but non-invasive and cost-efficient amyloid detection techniques are needed to reduce the number of unnecessary, invasive, expensive PET/CSF tests. The aim of this project is to study the state of the art of Deep Learning on graphs or Geometric Deep Learning and its most known models: Graph Neural Networks in order to use them to predict the preclinical stage of Alzheimer's disease with parcelled and processed MRI, which have been expressed as graphs using the regions of interest defined by the brain parcellation atlases as nodes and their volumes and other features as node signals. Two different datasets have been used and addressed as two independent graph classification tasks. Furthermore, the results have been interpreted carrying out a class activation mapping technique that determines what are the most relevant brain regions for the models to predict the preclinical stage.

To my grandparents, my parents and my brother.

Acknowledgements

I would like to express my sincere gratitude and deep appreciation to Verónica Vilaplana, the supervisor of this project, for giving me the opportunity to do the job and her unconditional support. It would have not been possible to carry out this project without her help and guidance, from the research process and implementations to the writing of this thesis. Her willingness to learn and her enthusiasm for the field are a source of inspiration and motivation.

Moreover, although I have not had the pleasure to meet them in person due to external circumstances we are living nowadays, I would like to thank the whole BarcelonaBeta Brain Research Center (BBRC) for both providing me with the data necessary to do this project and for their work to improve people lives and facing a disease such as Alzheimer.

Table of Contents

List of figures

List of tables

1 Introduction

1.1 Context and motivation

Alzheimer’s disease (AD) is the most common form of dementia and it is considered a biological continuum that can begin decades before the first cognitive symptoms. The detection of the preclinical stage of the Alzheimer’s disease is an opportunity to develop therapies focused on the prevention of the disease in earlier stages, before the loss of part of the brain tissue and neurons. In fact, “the continuous failure of clinical trials for drug development targeting mild-to-moderate AD dementia subjects combined with the increasing evidence of a long asymptomatic stage of AD supported by new biomarker developments have produced a shift towards AD prevention initiatives”. [1], [2]

Patients in the preclinical stage do not present any cognitive symptom but have abnormal amyloid biomarkers, which can be measured either in cerebrospinal fluid (CSF) or by positron emission tomography (PET). However, the accumulation of the amyloid is a slow process and more subjects and longer follow-up periods screening the evolution of those individuals would be needed to obtain robust statistical conclusions. Moreover, the cost of these clinical trials is high and it is a challenge to find and examine alternatives to face amyloid-positive detection in cognitive normal individuals.

Magnetic resonance imaging (MRI) can detect changes in the brain structure damaged by the AD and it is considered an important biomarker for the evolution of the disease. Furthermore, the technique is non-invasive and most cost-effective than PET and CSF. However, whereas brain structural damages are notable in later stages of the disease, the changes and differences are subtle in earlier stages when no cognitive symptoms have been detected yet.

Casamitjana et al. 2018 [3] proposed the usage of MRI processed images in order to be able to classify the preclinical stage of AD versus normal subjects using Machine Learning (ML) techniques. Concretely, they used a basic linear model to classify the subjects as either normal or preclinical. In this work, we study and analyse more recent and powerful Deep Learning (DL) models to face this challenge.

Deep Learning has revolutionized many Machine Learning tasks in recent years in a wide range of applications. Convolutional Neural Networks (CNNs) have achieved the state-of-the-art of the computer vision field and they are applied to many image processing challenges such as image classification or image segmentation. At the same time, (Gated) Recurrent Neural Networks are widely used in Speech Processing, Natural Language Processing and Time-Series analysis. Data used in these applications is usually represented in an Euclidean space, using vectorization techniques when necessary. CNNs and RNNs are able to exploit the locality, either in space or time, of the data and capture those hidden patterns that are relevant for the task.

However, there is an increase number of applications where the data is better suitable in a non-Euclidean space and that contain more complex relationships between objects, such as graphs and manifolds. Therefore, new models and algorithms able to exploit the information contained by these new spaces and these sophisticated relationships are required.

The complexity of graph data has imposed significant challenges on existing ML and DL techniques. In recent years, a new family of DL models called Graph Neural Networks (GNN) has achieved the state-of-the-art of many applications whose information can be represented using graph structures. This kind of models is characterized by its flexibility since they do not impose any constraint regarding the input size and the same model can be used with graphs of any dimensionality. Moreover, graph structures are composed by two basic entities, nodes and edges, and GNNs are flexible enough to allow the appliance of any ML task (e.g. classification) either to any of its entities or to the graph itself.

This work consists on the prediction of amyloid-positive subjects based on MRI information using graph structures and Graph Neural Networks. The data used in this project is provided by *Fundació Pascual Maragall* and includes two independent datasets, which are addressed as two independent classification tasks. Both datasets, which we call *AAL Dataset* and *DK Dataset*, are basically based on (i) the parcellation of MRI images into regions of interest (ROI) by means of an atlas and (ii) structural features such as the volume or the grey matter of these ROIs. In this project we study the ROIs as interconnected objects, that is, nodes of graphs. The methodology to obtain the relationships between vertices varies depending on the dataset: the *AAL Dataset* includes the structural connectivity matrices between ROIs from DTI which are directly used as graph adjacency matrices whereas the *DK Dataset* does not contain that information and the graphs connectivity between nodes is inferred from the regions adjacency in the atlas.

Classic GNN-based models are used to solve both tasks. Nevertheless, Casamitjana et al. 2018 [3] concluded that the most relevant information is contained by the structural connectivity matrices, used as graph edges in the *AAL Dataset*. That is why we also define, implement and test another DL model that operates on graphs but that emphasizes the edge features as an alternative to classic GNN. Moreover, since we do not have the structural connectivity matrices of the *DK Dataset* and the output must be computed from the nodes signal, we also propose a custom pooling layer that learns which are the more relevant vertices by itself.

Finally, in Machine Learning tasks and especially in neuroimaging and medical applications, not only is it important to solve a clinical problem, but it is also necessary to find interpretable solutions independently of the application and the algorithm used. The results of the second task are satisfactory enough to carry out this kind of analysis and that is why we compute a class activation map, which determines the regions that influence the most on the task and allows us to observe if there is any locality or special adjacency between them.

1.2 *Objective of the project*

The main goal of this project is the study and analysis of a preclinical stage of the Alzheimer's disease (AD). Following the work of Casamitjana et al. 2018 [3], which uses classic machine learning algorithms for the classification of this earlier stage based on Magnetic Resonance Imaging (MRI), we aim to extend that work by giving a graph-structure meaning to the processed neuroimages. Our purpose is to approach the preclinical AD detection from MRI as a graph classification task.

To do so, the first step of this project consist on getting a deep understanding of this data structures, which have been object of study for years and there is a lot of literature about them, and gain basic insights on Graph Signal Processing (GSP) field. Then, we want to learn the state-of-the-art of Deep Learning techniques applied to graphs, Graph Neural Networks (GNN), from its initial intuition to more recent definitions and implementations, as well as its applications.

In a second stage, we aim to address two classification tasks, corresponding to two independent, distinct datasets, which in this work we call *AAL Dataset* and *DK Dataset*. We want to approach both tasks using classic GNN models and, from our learnings and insights, implement custom layers that may be helpful for these concrete tasks.

Our final objective is to interpret the results achieved in order to be able to conclude which are the most relevant regions or features for the detection of amyloid positive individuals from MRI.

1.3 Structure of the thesis file

This report is structured as follows:

Part 1. Introduction

The context and the objectives of the project are detailed in this section. The two main topics of this thesis are presented: (i) Alzheimer's disease and (ii) Graph Neural Networks.

Part 2. Graphs

A technical background on graphs is provided. This section ranges from basic definitions and entities of Graph Theory to more complex Graph Signal Processing (GSP) concepts such as graph convolution. Finally, the first intuition and most known Graph Neural Network layers are explained, as well as the extension of graph convolutions to NN.

Part 3. Alzheimer's disease

This sections consists on the basics of the Alzheimer's disease and its definition, with emphasis on the preclinical stage of this degenerative disease. Finally, an intuition to the neuroimaging techniques is given in order to understand the origin of the data used in this project.

Part 4. Experiments

Overview of the methodologies followed and the results obtained in both of the tasks. Extra definitions and extensions of previously seen methodologies are included in the context they have been applied.

Part 5. Conclusions

Final conclusions, discussion of the results of the project and proposals for future development on the field are included in this last section.

2 Graphs

This chapter contains the technical background about graphs needed to understand this project. It ranges from basic Graph Theory definitions and an introduction to Graph Signal Processing to the application of Machine Learning on graphs and a more in-depth analysis of the idea behind the Graph Neural Networks and current existing layers that belong to that family of models.

2.1 Graph Theory

Graph Theory is the branch of mathematics dedicated to the study of graphs. We are only adding the definition of basic concepts but it is a huge field with many approaches, e.g. Algebraic and Spectral Graph Theory. The following definitions have been taken from *Diestel, R (2020). Graph Theory.*[4]

2.1.1 Definitions

Graph, node and edge
A graph G is a tuple G=(V,E) composed by the *vertex set* V, whose elements are the *nodes* or *vertices*, and the *edge set* E, whose elements are the *edges*. This edges set satisfies$E \subseteq [V]^2$. That is, the elements of E are pairs $e = \{u, v\} = uv,\ u, v \in V$.

Being a graph G=(V,E), it is defined

- the *order* of G, $|G|$, as the number of vertices. $|G| = |V|$
- the *size* of G, $||G||$, as the number of edges. $||G|| = |E|$

Note that if the order of G is n, then the size of G is between 0 and $\binom{n}{2}$.

Undirected graph
A *undirected graph* G=(V,E) is a graph whose edges are unordered pairs, that is, $\{u, v\} = \{v, u\}$. Two vertices $u \neq v$ in a *undirected graph* G=(V,E) are said to be *adjacent* or *neighbours* if $e = \{u, v\} \in E$. In this case, it is also said that given an edge $e \in E$, a node $u \in V$ is *incident* on e if $u \in e$. The two incident nodes of an edge are its *ends*.

Directed graph
A *directed graph* G=(V,E), also called *digraph*, is a graph whose edges are ordered pairs, that is, $(u, v) \neq (v, u)$. Given two vertices $u \neq v, u, v \in V$ in a *directed graph* G=(V,E) so that are connected by a directed edge $e = (u, v) \in E$, u is the *tail vertex* of the edge $e = (u, v)$, while $v$is its *tip vertex*. Moreover, we will say that e is an *input edge* at v and an *output edge* ad u.

Degree of a vertex
For a vertex $v \in V$ of a *undirected graph* G=(V,E), being $E(v) \subseteq E$ the subset of edges v is incident, the *degree* of v, $d(v)$, is defined as the number of edges at v: $d(v) = |E(v)|$. A vertex of degree 0 is *isolated*.
For a vertex $v \in V$ of a *directed graph* G=(V,E), three different degree measures are defined. The *in-degree* of v, $d_{in}(v)$, is the number of input edges at v. The *out-degree* of v, $d_{out}(v)$, is the number of output edges at v. Finally the *degree* of v, $d(v)$,is the sum of the input and output edges at v: $d(v) = |E(v)| = d_{in}(v) + d_{out}(v)$.

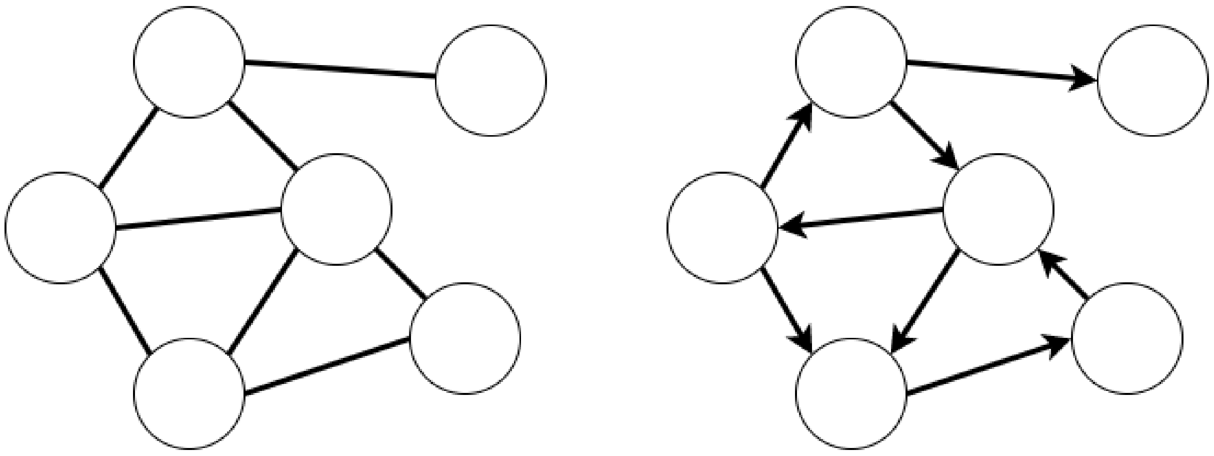
Figure 1 - (a) Undirected graph (b) Directed graph

Isomorphism
Two graphs $G_1=(V_1,E_1)$ and $G_2=(V_2,E_2)$ are *isomorphic* if there exist a bijection $\phi: V_1 \rightarrow V_2$such that $\{\phi(u)\phi(v)\} \in E_2 \Leftrightarrow \{u,v\} \in E$. The mapping ϕ is called an *isomorphism*.

Walk, path and cycle
A *walk* in G is a non-empty graph $W=(V_w,E_w)$ of vertices set $V_w = \{v_1, v_2, \dots, v_k\}$ and edges $E_w = \{(v_o, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)\}$. If $(v_o, v_k) \in E$, that is, the walk contains an edge between the first and the last node, then the W is a *closed walk*. The *length* of a walk is the number of edges in the sequence.

A *trail* is a walk in which all the edges e_j are distinct and a *closed trail* or *circuit* is a closed walk that is also a trail.

A *path* $P=(V_p,E_p)$ is a trail in which all the vertices in the $V_p = \{v_0 \dots v_k\}$ set are distinct. The vertices v_0 and v_k are the *ends* of P and $v_1 \dots v_{k-1}$ are its *inner* vertices. The path of length $k$is denoted by P^k.

Distance between nodes and diameter
For two vertices $u, v \in V$of a graph G=(V,E), the *distance* between u and v, $d_G(G)$, is the length of the shortest $u - v$ path. Moreover, the *diameter* of G, $diam(G)$, is the greatest distance between any two vertices of G.

Connectivity
In a graph G=(V,E), two vertices u to v are said to be *connected* if there is a walk from u to v. A non-empty graph G is a *connected graph* if for any pair of vertices $u, v \in G$ they are connected vertices.

Weighted graph
Let G=(V,E) be a graph, either directed or undirected, on which a mapping $w: E \rightarrow \Re$ is defined. Then the new tuple G=(V,E,w) is a *weighted graph*. Being $e \in E$an edge of G, the *length* of this edge is $w(e)$. For a vertex $v \in V$ of a *weighted undirected graph* G=(V,E), the *degree* of $v, d(v)$, is defined as the sum of the weights of the edges at v: $d(v) = \sum_{u \mid (u,v) \in E} w(e_{uv})$. Note that this definition is extended to *weighted directed graphs* and the *in-degree* and *out-degree*. Finally, the *length* of a walk in a weighted graph is the sum of the weights of the edges of the walk: $\sum_{j=1}^{l} \quad w(e_j)$.

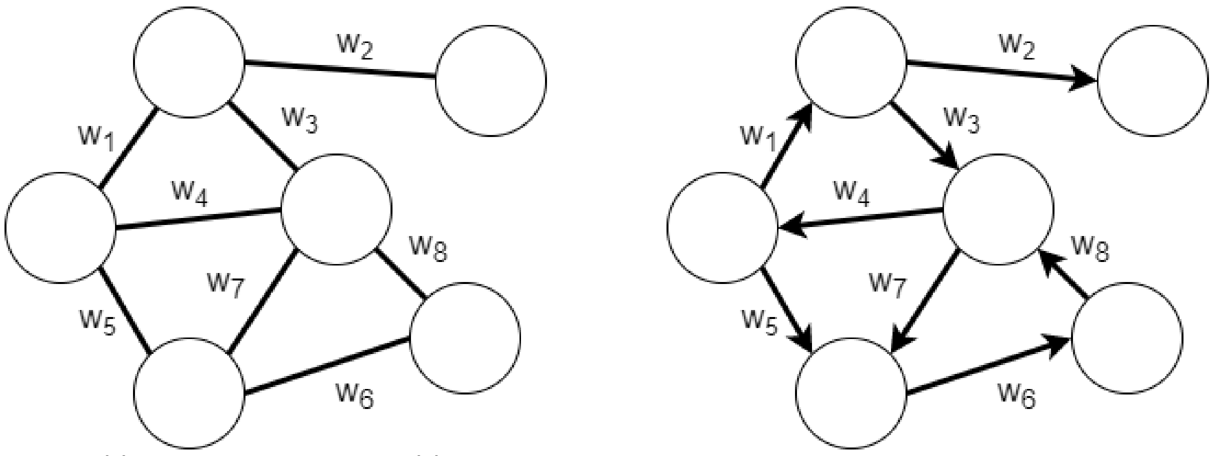

Figure 2 - (a) Weighted undirected graph (b) Weighted directed graph

2.1.2 Matrix Representation

Adjacency matrix

Let G=(V,E) be a *(un)directed graph* with $V = \{v_1 \dots v_n\}$ and $E = \{e_1 \dots e_m\}$. Then the *Adjacency matrix* A of G is an $n\ x\ n$ matrix whose entries are given by

$$A_{ij} = \begin{cases} 1, & (v_i, v_j) \in E \\ 0, & otherwise \end{cases}$$

Note that:

- the adjacency matrix is not unique because it depends on a numbering scheme for the vertices (isomorphisms).
- If G=(V,E) is undirected then its adjacency matrix A is symmetric $A = A^T$.
- If the graph has no loops, then $A_{ii} = 0\ for\ 1 \le j \le n$.

A more general adjacency matrix can be defined for *weighted graphs*:

$$A_{ij} = \begin{cases} w_{ij}, & (v_i, v_j) \in E \\ 0, & otherwise \end{cases}$$

Degree matrix

Let G=(V,E) be a *graph* with $V = \{v_1 \dots v_n\}$ and $E = \{e_1 \dots e_m\}$. Then the *Degree matrix* D of G is an $n\ x\ n$ diagonal matrix whose entries are given by

$$D_{ij} = \begin{cases} d(v_i), & i = j \\ 0, & otherwise \end{cases}$$

Laplacian matrix

Let G=(V,E) be a *undirected graph* with $V = \{v_1 \dots v_n\}$ and $E = \{e_1 \dots e_m\}$. Then the *Laplacian matrix* L of G is an $n\ x\ n$matrix whose entries are given by

$$L_{ij} = \begin{cases} d(v_i), & i = j \\ -1, & i \neq j\ and\ (v_i, v_j) \in E \\ 0 & otherwise \end{cases}$$

Note that if G is undirected then L is symmetric

Normalized Laplacian matrix
Let G=(V,E) be a *undirected graph* with $V = \{v_1 \dots v_n\}$ and $E = \{e_1 \dots e_m\}$. Then the *normalized Laplacian matrix* L of G is an $n \ x \ n$matrix whose entries are given by

$$L_{ij} = \begin{cases} 1, & i = j \ and \ d(v_i) \neq 0 \\ -\dfrac{1}{\sqrt{d(v_i)d(v_j)}}, & i \neq j \ and \ (v_i, v_j) \in E \\ 0 & otherwise \end{cases}$$

2.1.2.1 Understanding the matrices

Given a *weighted undirected graph* G=(V,E,w) and a function f indexed and defined on the vertices of G: $f : V \rightarrow \Re^N, f = (f(v_1), \dots, f(v_N))$

- The adjacency matrix can be viewed as an operator:
  $$(A f)_{v(i)} = \sum_{v_j \in N(v_i)} w_{ij} f(v_j)$$
- The adjacency matrix can be viewed as a quadratic form
  $$f^T A f = \sum_{(v_i, v_j) \in E} w_{ij} f(v_i) f(v_j)$$
- The Laplacian can be viewed as an operator
  $$(L f)_{v_i} = \sum_{v_j \in N(v_i)} w_{ij} (f(v_i) - f(v_j))$$
- The Laplacian can be viewed as a quadratic form
  $$f^T L f = 1/2 \sum_{(v_i, v_j) \in E} w_{ij} (f(v_i) - f(v_j))^2$$

2.2 Graph Signal Processing

In classical signal processing, the successive samples of the signals are equally spaced, either in time or space. This can be expressed using graphs whose nodes correspond to each one of the sampled instances of the signal and the fact that they are equally spaced can be modelled by assigning the same weight to each edge of the graph. For example, the graph used to represent the domain of a time signal will be a line graph or a cycle graph if the signal is considered as periodic and a regular grid graph can be used for images. The definition of this section have been taken from *Stanković, L., & Sejdić, E. (2019). Vertex-Frequency Analysis of Graph Signals.* [5]

The aim of Graph Signal Processing (GSP) is to deal with signals defined on an irregular domain represented by a graph and extend the existing common signal processing analysis techniques to that domains. A signal f in a graph G=(V,E) is defined as a function whose values are indexed by the nodes of the structure: $f : V \rightarrow \Re$.

Generally, a linear processing in a graph can be expressed as a linear combination of the signal in each node and its neighbours. A system in a graph can be expressed in terms of the adjacency matrix:

$$y = h_0 A^0 x + h_1 A^1 x + \ldots + h_{|V|-1} A^{|V|-1} x = (h_0 A^0 + \ldots + h_{|V|-1} A^{|V|-1})\, x = H(A)\, x$$

If the matrix A can be decomposed as $A = U\Lambda U^{-1}$,

$y = U(h_0 \Lambda^0 + \ldots + h_{|V|-1} \Lambda^{|V|-1}) U^{-1} x = UH(\Lambda) U^{-1} x.$

Generally speaking, for any diagonalizable matrix $M = U\Lambda U^{-1}$and any matrix function $f(M)$ that can be expressed as a polynomial form, applying f to the matrix is equivalent to applying it directly to the eigenvalues $f(M) = Uf(\Lambda)U^{-1}$

2.2.1 Graph Fourier Transform

The spectral analysis of signals can be extended to graph domains as well. The Graph Fourier Transform (GFT) of a graph signal is defined as:

$$X_F = U^{-1} x$$

where U is the matrix whose columns are the eigenvectors of the Laplacian matrix. Moreover, this spectral processing can also be approached based on the Adjacency matrix, which implies that the matrix U contains the eigenvectors of the Adjacency matrix instead of the Laplacian ones.

Note that if $U^{-1} = U^T$, the definition above can be expressed, for each position or node, as:

$$X_F(k) = x \cdot u_k$$

So each $X(k)$is the dot product between the signal and the *k-th* eigenvector and the GFT can be understood as the decomposition of a signal onto the set of eigenvectors. Finally, the Inverse Graph Fourier Transform can be computed as:

$$x = UX_F$$

2.2.2 Spectral Graph Convolution

One of the main properties of the Fourier Transform is that the convolution operation between two signals in time or spatial domains corresponds to a product in the frequency domain. This idea has been used to define the convolution on graphs as the IGFT of the product of the GFT of the signals:

$$x * y = U(\, U^{-1}(x) \odot U^{-1}(y))$$

being $\odot$ the element-wise product operation. It turns out that a system in a graph can be defined in terms of its Laplacian as well:

$$y = h_0 L^0 x + h_1 L^1 x + \ldots + h_{|V|-1} L^{|V|-1} x = (h_0 L^0 + \ldots + h_{|V|-1} L^{|V|-1})\, x = H(L)\, x$$

Decomposing L as $L = U\Lambda U^{-1}$

$$y = U(h_0 \Lambda^0 + \ldots + h_{|V|-1} \Lambda^{|V|-1}) U^{-1} x = UH(\Lambda) U^{-1} x = UH(\Lambda) X_F$$

Hence, the GFT of the filtered signal y is

$$Y = U^{-1} y = U^{-1} UH(\Lambda) X_F = H(\Lambda)\, X_F$$

2.4 Machine Learning on Graphs

Machine Learning algorithms are commonly designed to work with data represented as vectors or points in an Euclidean space. However, there is an increasing number of applications where data are generated from non-Euclidean domains and are represented as graphs with complex relationships and interdependency between objects.

2.4.1 Applications

Given a graph G=(V,E), features and labels can be assigned either to the entire graph, its nodes or its edges. Hence, learning and prediction tasks can be applied to any of these entities:

Node-level tasks:
Compute an output for each node of a given input graph. There is a wide range of node-level tasks:

i. Community detection: cluster the nodes of the graph.
ii. Centrality: determine the importance of the nodes. The importance can be defined in terms of its features, its connectivity, etc.
iii. Similarity: measure how alike nodes are in terms of shared neighbours.
iv. Embeddings: learn higher-level representations of the nodes that preserve the topology of the graph in the new space.

Edge-level tasks
Predict the likelihood of two nodes to be connected by an edge.

Graph-level tasks:
Compute an output for each graph of a given dataset $D = \{G_i \mid G_i = (V_i, E_i)\}, i = 1...N$. It can consist on either a classic classification/regression task or a graph embedding.

2.4.2 Classic algorithms

Weisfeiler-Lehmann, isomorphism test
1-WL algorithm is used to test whether two graphs G and H are isomorphic or not. A representation for each graph is produced, if the representations of G and H are different, they are not isomorphic. The algorithm is run parallelly in both graphs, the steps are the following:

i. Initialize the label of each node v at $t = 0$, $c^{(0)}(v) = l$
ii. At each time step, update the colour of each node taking previous colour and the multiset of the labels of their neighbours and applying a bijection such as a HASH function:
$$c^{(t)}(v) = \psi((c^{(t-1)}(v), \{\{c^{(t-1)}(u) | u \in N(v)\}\})$$
iii. Repeat step (ii) until there is no change in the partition of the nodes, with a maximum number of iterations of |V|, being |V| the number of nodes. [6]

Graph Kernels for graph classification
Graph Kernels have been, for years, the most known approach to solve graph classification challenges. They consist on a kernel or user-defined mapping that measures the similarity between a pair of graphs to be the input of a kernel method such as a support vector machine (SVM). For example, a popular graph kernel is made by comparing the histograms of the colours at the end of the *Weisfeiler-Lehmann* algorithm.

2.5 *Graph Neural Networks*

Geometric Deep Learning (GDL) is an emerging field within the Machine Learning community that extends the existing Deep Learning (DL) techniques to non-Euclidean domains such as graphs and manifolds.

Concretely, Graph Neural Networks (GNN) are a family of GDL that deals with graph structured data. Its motivation roots in both diffusion-processes as an extension of Recurrent Neural Networks (RNN) and as a generalization of the convolution operation of Convolutional Neural Networks (CNN), which have ended up converging to message passing processes.

GNN do not impose any constraint on the size of its input and the same model can be trained and used with graphs that do not have the same number of nodes. The majority of layers ensures an isomorphism invariance, that is, the output of two isomorphic graphs will be the same. Furthermore, they can face many ML applications on graphs explained above.

Semi-supervised Node Classification
Given one single graph G=(V,E) and a label assigned to part of the nodes of G, a Graph Neural Network can be trained to predict the label of a node. This is achieved by learning higher-level representations of the nodes and applying a projection network, e.g. a multilayer perceptron followed by a softmax, to the representation of each node, predicting and output per node.

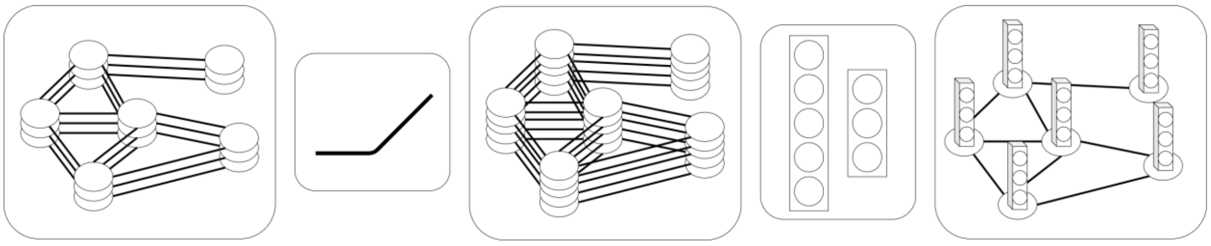
Figure 3 - Semi-supervised node classification with GNNs

Supervised Graph Classification
Given a set of graphs $D = \{G_i \mid G_i = (V_i, E_i)\}, i = 1...N$ and a label assigned to each one of the entire graphs, Graph Neural Network can be trained to predict the label for a given input graph. The approach consist on learning higher-level representation of the nodes of each graph and then perform a pooling or readout operation that will output the representation of the entire graph. This representation is then classified using a multilayer perceptron followed by a softmax, for example.

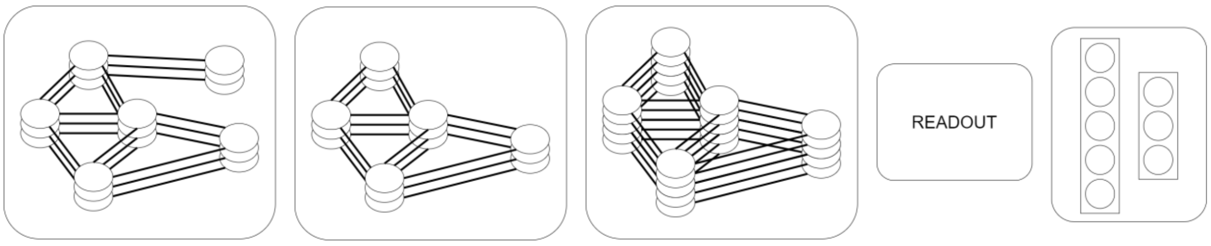

Figure 4 - Supervised graph classification with GNNs

Semi-supervised Link Prediction
Given one single graph G=(V,E), a Graph Neural Network can be trained to predict if there is a link between two input nodes. This is achieved by learning higher-level representations of the nodes and

applying a projection network, e.g. a multilayer perceptron that takes as input the representation of the two endpoints of the hypothetical edge.

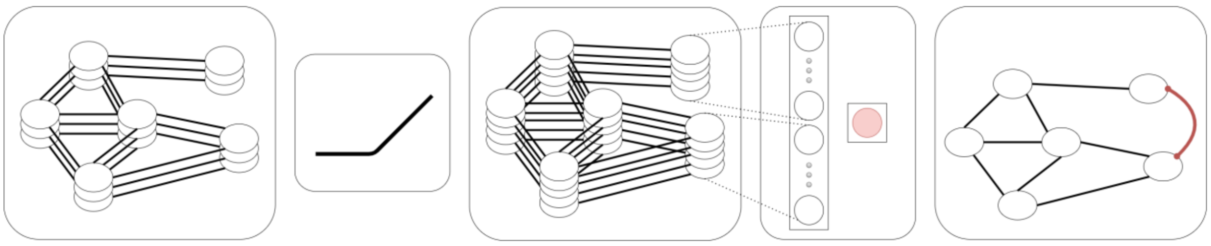
Figure 5 - Semi-supervised link prediction with GNNs

2.5.1 *First Recurrent Intuition*

Graph Neural Network

The concept of Graph Neural Network (GNN) by Scarselli et al.2009 [7] in order to be able to process either cyclic, directed, undirected or acyclic graphs, as an extension of RNN's which only is able to work with direct, acyclic, line graphs. The model is based on a information diffusion mechanism, a hidden state h_v is assigned to each node v of the graph, the information is transferred between nodes according to the connectivity of the graph until an equilibrium point. Then, each node updates its state according to the parametric *transition function*:

$$h_v{}^{t+1} = \sum_{u \in N(v)} f(l_v, l_{(v,u)}, l_u, h_u{}^t)$$

Where $l_v, l_{(v,u)}, l_u, h_u{}^t$ are the labels of the central node, the edge connecting the node and the neighbour, the neighbour and the previous state of the neighbour, respectively. Note that the addition operation allows that this updating function can be applied to each node, independently of the size of its neighbourhood. In the model, another node-shared function is defined, the *output function*, whose value is computed per each node at each time step.

$$o_v{}^t = g(h_v{}^t, l_v)$$

Back propagation through time is required to train this network and the methodology proposed follows the Almeida-Pineda algorithm. That means that it is not necessary to store the hidden states at each time step since not until an equilibrium has been achieved does the forward pass stop, hence, it can be considered that $h_v{}^t = h_v \forall v \in V \ \forall t \geq t_o$, so only h_v is required. However, in order to ensure that the equilibrium points always exist, a constraint must be applied to the transition function. This function f must be a contraction mapping, that is:$||f(x) - f(y)|| \leq \mu||x - y|| \ \forall x, y$

Gated Graph Neural Network

Li et al. 2015 [8] presented a variation of the GNN model, called Gated Graph Neural Network (GGNN) and whose extension consists on using Gated Recurrent Units [9] instead of the basic recurrent function. This improvement avoids having to add a constraint for convergence in the parameters of the model since it is not necessary to iterate until equilibrium and a fixed number of time-steps can be tuned as hyperparameter. However, this implementation requires storing the hidden states of each time-step in order to unfold them to run the backpropagation through time during training, so more memory is required.

The propagation model of GGNNs is:

$h_v{}^t = GRU(h_v{}^{t-1}, \sum_{u \in N(v)} W h_u{}^{t-1})$ where $h_v{}^0 = x_v$

2.5.2 *Graph Convolution*

Parallelly, due to the successful performance of convolution operations and CNNs on images, which can be understood as a regular grid or a graph where each node corresponds to a pixel and the graph connectivity is inferred from the pixels adjacency, the aim to extend the convolutional operation to non-regular graphs appeared.

There are two different approaches to define the convolution operation on graphs. The first one, spectral-based convolutions, are based on mathematical foundations of Graph Signal Processing field whereas spatial-based convolutions are more related with the previous recurrent intuition and the neighbourhood aggregation as in CNNs.

2.5.2.1 *Spectral-Based Convolution*

Spectral-based convolutional layers base their implementation on the idea that a convolution in spatial domain corresponds to a product on the frequency domain. As it has been shown above, if $L = U\Lambda U^{-1}$, the Graph Fourier Transform and the convolution of two signals on a graph are defined as $X_F = U^{-1}x$ and $x * g = U(X_F \odot G_F) = U(U^{-1}x \odot U^{-1}g)$, respectively. If the filter is redefined as $g_w = diag(U^{-1}g)$, then the convolution is simplified to $Ug_wU^{-1}x$. The difference between spectral-based convolutional layers remains on the definition of that filter [10].

Spectral Convolutional Neural Networks
Spectral CNN [11] transforms, at each layer l, a multi-channel input $h_{l-1} \in \Re^{|V|\,x\,f_{l-1}}$ vector to a $h_l \in \Re^{|V|\,x\,f_k}$. The filter used corresponds to learnable diagonal matrices ${W_{i,j}}^k$.

$$ {h_j}^l = \sigma(U \sum_{i=0}^{f_{l-1}} {W_{i,j}}^l U^T {h_j}^{l-1}) \qquad j = 1,2,\ldots,f_l $$

being U the matrix of eigenvectors ordered by eigenvalue. Furthermore, it is mentioned that not all eigenvectors are always needed and that this layer can be implemented using the d first eigenvectors, using U_d instead of U. Note that, for each layer k, the number of parameters is $f_{l-1} \cdot f_l \cdot d$.

However, this layer presents some limitations such as that it is not possible to use the learned parameters with other graphs since any edge or edge-weight modification leads to a different eigenbasis and that the computational costs of obtaining the eigenvectors of a matrix is $O(|V|^3)$.

Chebyshev Spectral Convolutional Neural Networks
ChebNet [12] aims to reduce the computational cost of the previous layer as well as being able to be applied to graphs with different structures. The solution proposes the usage of space-localized filters using polynomial parameterization showing that obtaining local features reduces the complexity.

The layer defines the filter g_w as the Chebyshev polynomial approximation of the diagonal matrix of eigenvalues $g_w = \sum_{i=0}^{K-1} \; W_i T_i(\sim\Lambda)$ where $W_i \in \Re^M$ is a vector of Chebyshev coefficients and $T_m(\sim\Lambda) \in \Re^{|V|x|V|}$ is the Chebyshev polynomial of order m evaluated at $\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I$. Being the Chebyshev polynomial $T_m(x)$ of order m, it can be computed recursively as $T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x)$ with $T_0 = 1$ and $T_1 = x$. Since $\tilde{L} = 2L/\lambda_{max} - 1$and $T(\tilde{L}) = UT(\tilde{\Lambda})U^{-1}$, the convolution operation is approximated as

$$x * g_w \simeq \sum_{i=0}^{K-1} W_i T_i(\tilde{L})x$$

the filter operation at each layer l can be computed as
$h^l = \sigma(\sum_{i=0}^{K-1} W_i{}^l T_i{}^l(\tilde{L})h^{l-1})$.
so that the parameterization is done directly to the Laplacian matrix. Finally, the fact that the features are extracted locally implies that the input graph size can be arbitrary since the weights do not depend on the eigenbasis.

Graph Convolution Network
GCN [13] is a first-order approximation of the previous ChebConv that assumes that $K = 1$ and $\lambda_{max} = 2$. Hence, $x * g_w \simeq \sum_{i=0}^{K-1} W_i T_i(\tilde{L})x$ is simplified to

$$x * g_w = w_0 x + w_1(L - I) = w_0 x - w_1 D^{-1/2} A D^{-1/2} x$$

Note that the operation contains two parameters w_o and w_1 and stacking k filters of this form will convolve the *kth*-order neighbourhood of each node. However, this expression is simplified in order to avoid overfitting by reducing the number of parameters to one:

$$x * g_w = w\,(I + \, D^{-1/2} A D^{-1/2})\, x$$

and a regularization trick is used in order to avoid numerical issues, using $\tilde{A} = A + I$ and $\widetilde{D_{ij}} = \sum_j \tilde{A}_{ij}$ instead of A and D. Hence, taking into account that the signal defined in a graph can be $d - dim$, the convolution can be defined as

$$x * g_w = \widetilde{D}^{-1/2} \tilde{A} \widetilde{D}^{-1/2} x W$$

the filter operation at each layer l can be computed as

$$h^l = \sigma(\, \widetilde{D}^{-1/2} \tilde{A} \widetilde{D}^{-1/2} h^{l-1} W)$$

2.5.2.2 Spatial-Based Convolution

The spatial-based graph convolution operation is a generalization of the spatial convolution performed by classic CNNs on grids or images but using the graph adjacency between nodes. They follow the same approach than GNN and GGNN, that is, the propagation of information between adjacent nodes. Nevertheless, instead of propagating the information iteratively, it is propagated once per layer and higher order neighbourhood information is achieved by stacking multiple convolutional layers.

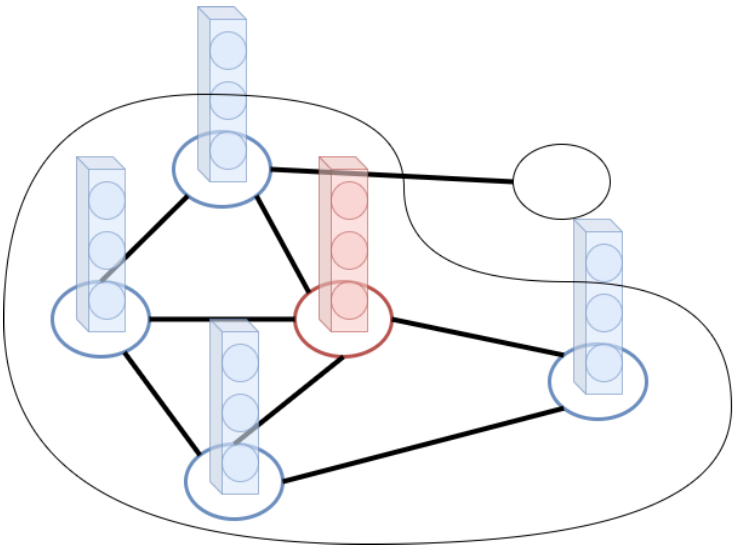

Figure 6 - Spatial graph convolution

Diffusion-Convolutional Neural Network
DCNN [14] implements the convolution on a graph as a diffusion process of the features or states of the nodes. The diffusion is modelled according to the propagation matrix $P = D^{-1/2}A$. Each element P_{ij} of the matrix is understood as the transition probability from node v_i to node v_j. Hence, the output at each layer l is defined as:

$$h^l = \sigma(W^l \odot P^l h^{l-1})$$

GraphConv
GraphConv [6] is presented as an extension of the Weisfeiler-Leman algorithm in a higher order manner. The layer performs a linear transformation to the node $v \in V$representation and another independent to the aggregation of the representations of the neighbourhood. Then, the output at each layer l, for each node $v \in V$ is defined as:

$$h_v{}^l = \sigma(W_1 h_v{}^{l-1} + \sum_{u \in N(v)} W_2 h_u{}^{l-1})$$

GraphSAGE
GraphSAGE [15] aggregates the information from local neighbours using a shift-invariant function such as *mean*, concatenates the result with the state of the node and apply a linear transformation using a trainable matrix W. Then, the output at each layer l, for each node $v \in V$, is defined as:

$$h_v{}^l = \sigma(W^l (h_v^l \mid\mid \sum_{u \in N(v)} h_u{}^l))$$

Graph Attention Network
GAT [16] uses attention mechanisms to decide the contribution of each node of the neighbourhood to the final state of the central node. Given a node v, a parameter α_{uv} that models the connectivity strength between nodes u and v is computed $\forall u \in N(v)$using a trainable vector a. Hence, the output at each label l, for each node $v \in V$, is defined as:

$$h_v{}^l = \sigma(\alpha_{vv} W h_v{}^{l-1} + \sum_{u \in N(v)} \alpha_{uv} W h_u{}^{l-1})$$

2.5.3 Message Passing Neural Networks

Message Passing Neural Networks (MPNNs) was introduced in Glimer et al. 2017 [17] as an umbrella that covered the majority of the graph-convolutional layers published until the moment, including both spatial and spectral approaches. They presented MPNN as a framework for supervised learning on graphs, which generalizes the commonalities between different models published.

The framework defines the forward pass as two different phases: the message passing and the read out phases. During the message passing phase, which runs T times, the hidden state of each one of the nodes is updated according to the messages received by each one of its neighbours M_t and the update function U_t. Hence, at each time step, the message received by a node is defined as aggregation of message of the neighbors

$$m_v{}^{t+1} = \sum_{u \in N(v)} M_t(h_v{}^t, h_u{}^t, e_{vu})$$

The aggregation function $\sum_{u \in N(v)}$ could be generalized to any other permutation invariant function such as *mean* or *max*. Then, the hidden state is updated according to:

$$h_v{}^{t+1} = U_t(h_v{}^t, m_v{}^{t+1})$$

The readout phase computes a one single feature vector for the whole graph given the node features computed during the forward phase. This global pooling operation has the form:

$$h_G = R(\{h_v{}^T \mid v \in G\})$$

This readout function is used for graph-classification tasks to be able to apply a projection layer or any other common ML algorithm that works with Euclidean data. Note that, in order to ensure that two isomorphic graphs will have the same representation, this global pooling operation must be swift invariant, such as *sum*, *avg*, *max*, etc of the hidden states of the nodes.

2.5.4 Hierarchical Pooling

The pooling layers refers to the set of operators that aims to reduce the graph dimensionality in order to decrease the number of parameters and avoid overfitting. The downsampling can be achieved, for example, by removing nodes of the graph. This is analogous to pooling operations performed in classic CNNs but the fact that the data is not structured requires more complex implementations.

Top-K Pooling
Top-K Pooling [18] computes a pooled graph G'=(V',E') by dropping $|V| - ceil(k\,|V|\,)$ nodes of the input graph G=(V,E) by computing a score for each node $v\ \in V$using a learnable projection vector $p$and selecting the top-k nodes.

Differentiable Pooling
DiffPool [19] computes a new adjacency matrix and node representations by learning a matrix S which softly assigns each node of the input graph G=(V,E) to a cluster of the output graph G'=(V',E').

2.5.5 Global Pooling

Graph-Level tasks requires a representation of the entire graph to compute the final output. This is commonly approached by computing one single vector per graph using a Global Pooling operator that aggregates the states of the nodes and edges of the graph. Note that this can be achieved by using any shift-invariant operation such as *mean, sum* or *max*. However, more complex operators are present in the literature.

Global Attention
GlobalAttention [8] is presented jointly with GGNNs that using a soft-attention mechanism decides which nodes are relevant for the final graph-level task and computes the output by a weighted sum of the states of the nodes. The output is computed as

$$h_G = \sum_{v \in V} softmax(\, nn_1(h_v)\,) \odot nn_2(h_v)$$

where nn_1 maps the node embeddings to 1-dimensional scores and nn_2 maps them to an output size vector.

Global Sort Pooling
GlobalSortPool [20] sorts the nodes of the graph in a descending order of the last feature channel and selects the k first nodes, where k is an hyperparameter of the layer, that is, it is not learnable. The output consist on the full embeddings of those k selected nodes.

2.5.6 *PyTorch Geometric*

PyTorch Geometric [21] is an extension of the well-known Python library PyTorch [22] which implements a large variety of Deep Learning techniques that work with non-Euclidean data such as graphs, known as Geometric Deep Learning. The library also provides the implementation of the object required to model graphs, as well as an advanced easy-to-use mini-batch loader, which allows scaling the training process of the models to large amount of data.

The library is mainly focused on (but not limited to) Graph Neural Networks. Concretely, an implementation of the base-class MessagePassing is provided jointly with different Graph Convolutional layers which extend this base-class.

2.5.6.1 *MessagePassing Base-Class*

The convolution operation in graphs can be generalized, as explained above, using the form:

$$h_v{}^{t+1} = U_t(h_v{}^t, A_{u\in N(v)}\{M_t(h_v{}^t, h_u{}^t, e_{uv})\})$$

MessagePassing class provides the framework to implement this kind of operations using PyTorch Modules and Tensors. The main methods of the class are:

- *MessagePassing(aggr="add", flow="source_to_target")*: init method. The *aggr* parameter refers to the neighbourhood aggregation function, $A_{u\in N(v)}$ in the formula, whose values can be either *"add", "mean"* or *"max"*. The flow defines the direction the messages between nodes will be passes: either *"source_to_target"* or *"target_to_source"*.
- *message()*: build messages to be sent for each edge of the graph, that is, implementation of M_t.
- *update()*: update nodes hidden states given the output of the aggregation A, that is, implementation of U_t.
- *propagate(edge_index, size=None, dim=0, **kwargs)*: the initial call to propagate the messages. This method will call *message()* and *update()* explained above.

This class can be extended to implement any custom graph convolutional layer and only the implementation of *message()* and *update()* will be needed. As well as the *forward()* if some previous steps are needed. In fact, this library provided the implementation of multiple known and published layers, such as GCN and GraphSAGE.

2.5.7 *Class Activation Mapping*

Zhou et al. 2015 [23] introduced the *Class Activation Mapping (CAM)*, a technique that uses global average pooling (GAP) on CNNs that identify the discriminative image regions used by the model to classify a given category, also known as saliency maps.

This technique consists on performing an average global pooling operator after the convolutional layers of the network and then directly applies a linear layer and a softmax to classify the samples. Then, the saliency maps are computed using the weights that projects the averaged pooled feature maps to the class activations. The idea is that, for a given image, let $f_i(x,y)$the activation of unit i in the last convolutional layer at location (x, y). For this unit i, F_i is defined as the result of performing the global average pooling across (x, y), that is, $F_i = \frac{1}{H*W}\sum_{(x,y)} f_i(x, y)$. Hence, for a class c, the input of the softmax is $\sum_i \ w_i{}^c F_i$ where $w_i{}^c$ is the weight associated to F_i for the class c. Broadly speaking, $w_i{}^c$denotes the importance of F_i for class c. Therefore, given an image, this weights can be used to define the class activation map for class c, denoted by$M_c(x, y)$, where the value at each spatial location is computed by $M_c(x, y) = \sum_i \ w_i{}^c f_i(x, y)$.

Arslan et al. 2018 [24] proposed a class activation mapping implementation in the context of GNNs. Moreover, they also apply this technique to graphs generated by a brain-parcellation in ROIS for graph classification. In this solution, the CAM are defined in the vertex domain of the graph. Note that is is straightforward to extend the previous definition for a graph domain instead of a spatial one. Given a graph, $f_i(v)$ is defined as the activation of unit i in the last convolutional layer for node v and F_i as the result of performing the global average pooling across the nodes of the graph $F_i = \frac{1}{|V|}\sum_v \ f_i(v)$. Similarly, the input to the softmax for class $c$is $\sum_i \ w_i{}^c F_i$. Therefore, the class activation map for each node v is defined as $M_c(v) = \sum_i \ w_i{}^c f_i(v)$.

3 *Alzheimer's Disease*

In this section we review the basics of the Alzheimer's disease (AD) as a biological continuum and we give further information related to the preclinical stage, the main object of study of this project. Finally, we introduce Magnetic Resonance Imaging (MRI) and a brief description of the processes needed to acquire the data we are using in this project.

3.1 Description

Dementia is syndrome caused by brain damages, probably due to a disease, that affects the part of the brain functionalities such as memory, learning abilities and orientation. Alzheimer's disease (AD) is the most common form of dementia and it is thought that can contribute to 60-70% of the cases [25].

Alzheimer's disease is a neurological disease that affects the brain and is characterized by a progressive deterioration of the cognitive function and intellectual abilities which lead the patients to dementia. It is known that during the disease, there are neuropathological changes that usually involve the presence of plaques of Amyloid-β (Aβ) and tangles of tau proteins and lead to the degeneration of the brain tissue and the damage, and even death, of the brain nerve cells and neurons. AD is more popular among people older than 65 years, but it must be differentiated from the decline of cognitive functionalities due to age factors, which are slower and less aggressive. [1], [25], [26]

3.1.1.1 Biomarkers

Last decade studies have observed a paradigm shift on which AD is generally understood as a biological continuum that ranges all the way from healthy patients to dementia and that is characterized by the evolution of a set of biomarkers that are known to have influence on the degeneration of the brain tissue and neurons. A biomarker is a biological signature that can be sued as an indicator of a pathological situation. The main biomarkers used to track the evolution of the disease are

I. Aβ positron emission tomography (PET)
II. Cerebrospinal fluid (CSF)
III. Magnetic resonance imaging (MRI)

Nevertheless, there are discrepancies among Alzheimer's disease research institutions and groups when defining a criteria for the AD diagnosis and what biomarkers must be use to consider the illness.

3.1.1.2 Stages

AD is a progressive and degenerative disease and the first stage of the disease (Preclinical AD - PreAD) can start decades before the diagnosis of the illness. However, being in the PreAD stage does not imply that the patient will necessarily develop the disease to later stages. The stages of the biological continuum considered in this work are the following:

I. Preclinical AD (PreAD)
II. Mild Cognitive Impairment AD (MCI-AD)
III. Alzheimer's disease (AD)

Sperling et al. 2011 [27] showed an hypothetical evolution of the biomarkers mentioned along the different stages defined above. As it is observable, amyloid-β accumulation, obtained from CSF and PET analysis, is supposed to be the earliest biomarker to experiment a variation on first preclinical stage.

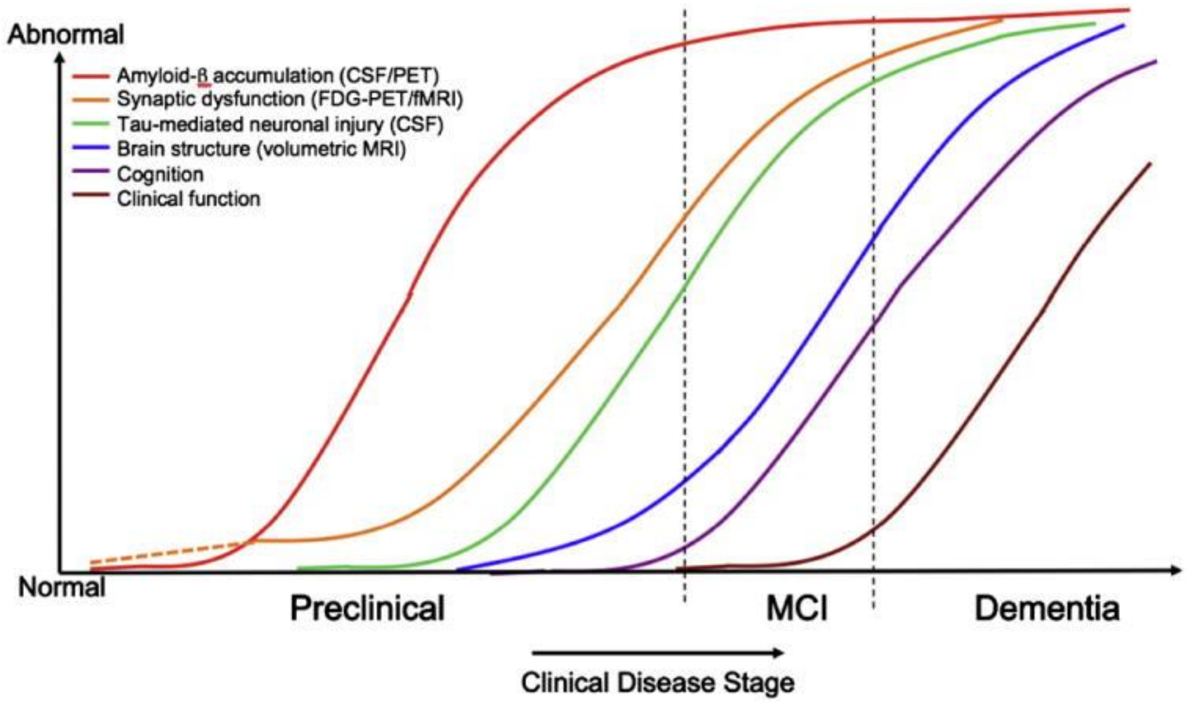

Figure 7 - Hypothetical evolution of the biomarkers due to AD (Sperling et al. 2011) [27]

3.1.1.3 *The role of genetics*

Although AD is generally not inherited, there are some genes and gene mutations that can be risk factors and can increase the likelihood of development of the disease. The APOE@4 is nowadays the strongest known genetic risk factor for sporadic AD. Individuals carrying either one or two copies of the @4 allele have three to fifteen time increase risk of developing AD. Moreover, the same allele also significantly reduces the mean age for AD onset [28]–[31]. However, they do not guarantee that it will happen [25]. Finally, it has been reported that *APOE* ε4 directly affects MRI, CSF, and cognitive biomarkers in AD [32].

3.2 *Preclinical AD (PreAD)*

The preclinical Alzheimer's disease (PreAD) was initially defined as cognitively unimpaired individuals who displayed AD brain lesions on postmortem examination [33]. Nowadays, due to the incorporation of the biomarkers for the diagnosis of AD, PreAD ranges from the moment that these biomarkers start varying in cognitive normal subjects until the first appearance of cognitive symptoms. [1], [28]. The preclinical stage can start decades before the detection of the AD because of cognitive symptoms and that is why its study is crucial for the disease prevention.

Once again, there are discrepancies defining the lower and upper boundaries of the stage but the principal studies agree on the pathophysiological biomarker for AD: the amyloid-β accumulation in the brain, which has been observed to be the first biomarker that starts varying due to the pathology..

3.3 MRI Analysis

3.3.1 MRI

Magnetic resonance imaging (MRI) is a spectroscopic imaging technique used to capture brain anatomy-in-vivo. It is based on the Nuclear Magnetic Resonance (NMR) effect that non-zero spin particles produce in the brain when excited by a strong constant magnetic field (B0) and an oscillating, weak and orthogonal magnetic field (B1). Different tissue configurations will appear to have different signal intensities in the resulting image. More diffuse mediums, such as intracranial cerebrospinal fluid (CSF) will produce different signal than white matter (WM) or grey matter (GM) tissues, which are less diffuse and contain higher proportion of other particles. Brain lesions (e.g. tumors) would also appear differently than their surroundings. MRI can produce several image modalities adapting the acquisition parameters to produce different contrasts between brain tissues and thus providing complementary information [1]. MRI is used as a biomarker for brain degeneration but is not constrained to Alzheimer analysis.

There are different MRI techniques that provide different information of the brain structure: (i) diffusion MRI (DWI), (ii) structural MRI (DWI) and (iii) functional MRI (fMRI).

3.3.1.1 Diffusion MRI

Diffusion weighted imaging (DWI) provides information about the medium diffusivity. It uses the rate of diffusion of molecules, mainly water, to generate a contrast in MRI. Diffusion tensor imaging (DTI) provides, for each voxel of the image, the probability distribution that describes the direction of the water molecules. Due to the underlying structure of white matter tissue where the architecture of axons in myelinated parallel bundles facilitate the diffusion to a certain direction, DTI can help to reconstruct in-vivo the main white matter pathways as well as to assess the integrity of white matter bundles [1]. This is achieved by radiating a magnetic field gradient in different directions. For each direction measured, the gradient intensity will be increased or decreased depending on the direction that the molecules take. Concretely, if the molecules take the same direction than the gradient the intensity will be reduced whereas it will be increased if they propagate perpendicularly [34].

3.3.1.2 Structural MRI

Structural MRI provides static anatomical information about the brain. It describes qualitatively and quantitatively the shape, size and texture of grey and white matter structures in the brain. It can be used either in clinical practice for radiological reporting or for detailed analysis [1].

The most widely used structural MRI technique is T1 weighted imaging, which is sensible to different brain tissues and can provide information of the proportion, shape and texture of grey and white matter in the regions of the brain.

3.3.2 Brain Parcellation

A brain parcellation is a neuroimaging methodology that divides the brain in regions of interest (ROI). The regions and their boundaries are defined by a brain atlas, which consist on a volume representing the brain whose pixels are labelled according to the regions they belong. Given a raw MRI, a projection can be applied so that the atlas and the image have the same size and dimensionality in order to be able to define the regions in the original image and extract information of each ROI such as its volume or the connectivity between them for further analysis.

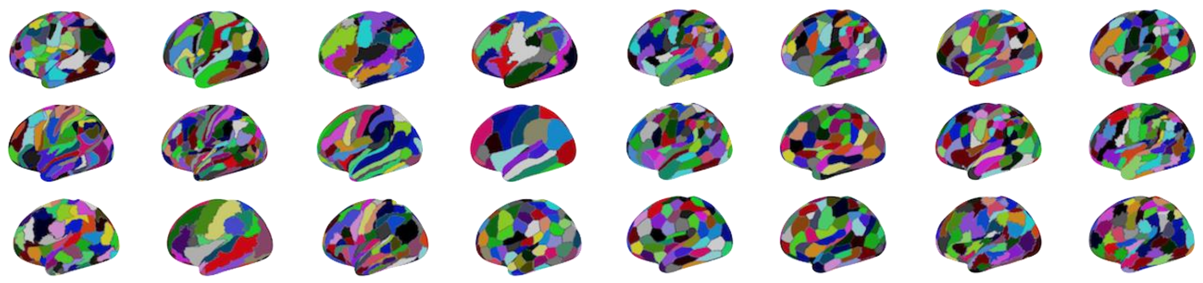
Figure 8 - Brain parcellations (Brain Parcellation Survey, BioMedIA) [35]

3.3.3 MRI to Matrices

Structural Connectivity Matrix

Structural Connectivity Matrix (SCM) is a square *NxN* matrix, being *N* the number of ROIs defined in the atlas used to carry out the parcellation, where each element SCM_{ij} is the estimation of the number of fibers that connects ROI_i to ROI_j. This matrix is obtained by applying a brain parcellation to a DWI image and a tractography algorithm that estimates the number of fibers from one ROI to another by allocating a set of seeds in each ROI and counting the portion of that seed that goes to each other ROI.

Grey Matter Volume Vector

Grey Matter Volume Vector is a *N*-dimensional vector, being *N* the number of ROIs defined in the atlas used to carry out the parcellation, where each element i of the vector corresponds to the amount of cubic units of grey matter in that ROI. This vector is obtained from a T1-weighted image. The first step is to perform a segmentation to separate the grey matter and white matter, between others, and apply a parcellation to the MRI.

MRI images and brain atlases can be used to compute more features related with the ROIs of the brain of each subject. For example, another useful and widely used feature is the volume of the entire ROI.

4 Experiments

This section consists on an explanation and an exploration of the datasets used in this project. Then, we detail the experiments carried out, from the methodologies and implementations needed to the results obtained. As we have explained along this report, the two datasets have been addressed as two independent tasks so we have splitted the explanation into two separate subsections.

4.1 Context and Previous Work

This classification challenge is based on Casamitjana et al. 2018 [3], which faced the problem using classic Machine Learning (ML) algorithms. What is new in this work is that, due to the nature of the data, we aim to approach this classification task by giving a graph-structure meaning to the MRI information. Hence, the DL techniques used are under the umbrella of Geometric Deep Learning (GDL) and Graph Neural Networks (GNN).

Following Casamitjana et al. 2018 [3] proposal, we have studied the viability of the models, a part from the standard metrics such as the precision, recall, AUC ROC and AUC PR, using the savings in clinical recruitment, calculated as the percentage difference in resources between the standard recruitment protocol and the alternative methodology proposed in Casamitjana et al. 2018 [3] (Figure 9). Concretely, the savings have been computed in terms of economic costs and participant burden:

$$Savings_{COST} = \frac{1}{C_{avg}}(\rho\frac{C_{PET}}{P} + \frac{C_{MRI}}{R})$$

$$Savings_{BURDEN} = 1 - \frac{\rho}{P}$$

where P is the *precision*, R is the *recall*, $\rho$is the expected percentage of amyloid positive in general population $\rho = 20\%$, C_{PET} is the cost, in euros, of the gold test $C_{PET} = 3000€$, C_{MRI} is the MRI acquisition cost $C_{MRI} = 700€$ and C_{avg} is the average cost among the screening tests.

The protocol proposed in Casamitjana et al. 2018 [3] , in contrast with the standard protocol, adds another exclusion layer in which only the subset of preAD subjects that are more likely to develop abnormal amyloid-levels are subjected to PET/CSF clinical trials, which implies a minimization of these expensive techniques.

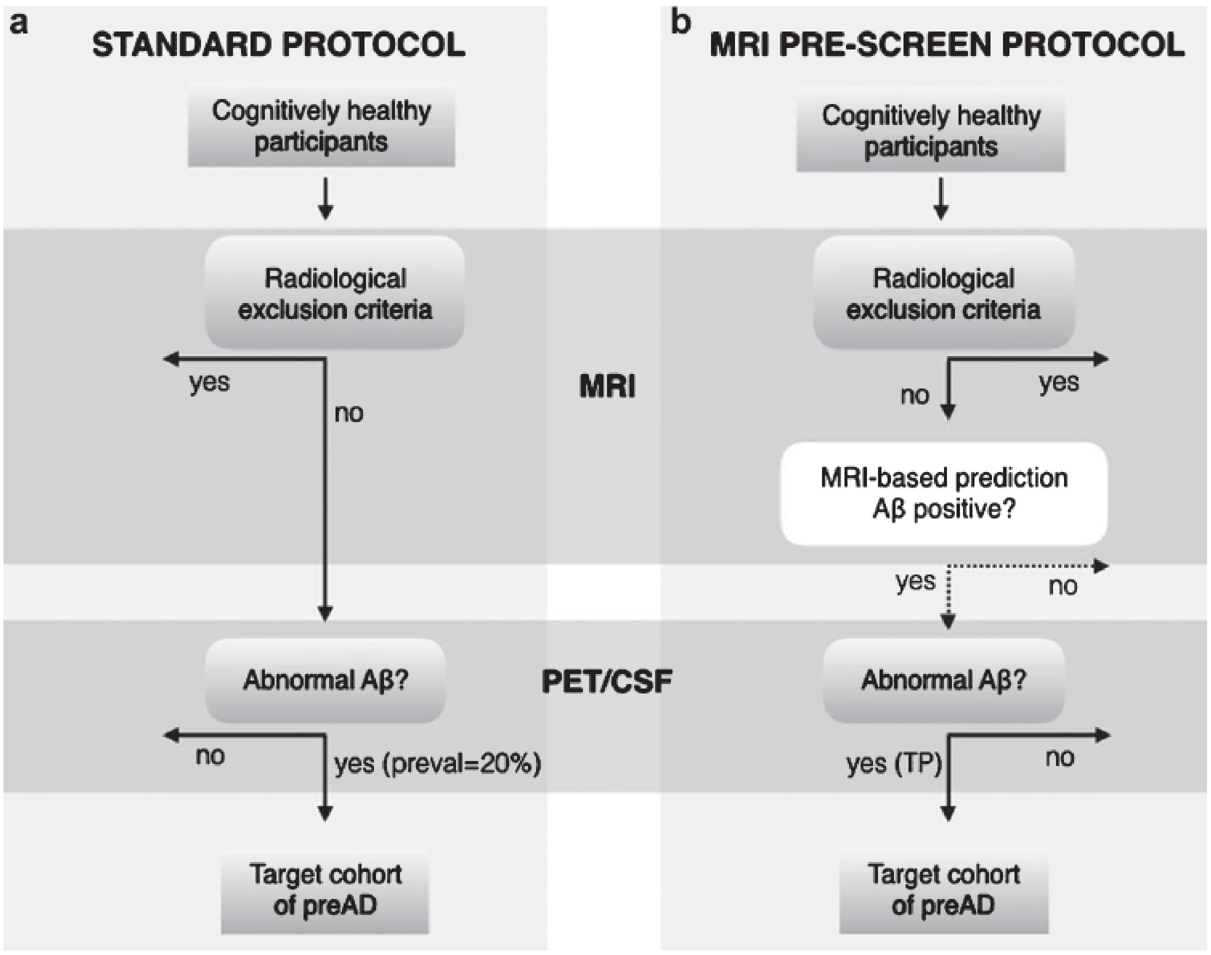

Figure 9 - Outline of clinical recuitment protocol of subjects with PreAD pathophysiology [36]
(a) Standard recruitment protocol used in clinical trials. (b) Our proposed protocol with MRI-based machine learning. In the standard protocol, cognitively healthy participants undergo an MRI scan to identify radiological exclusion criteria (e.g., cerebrovascular disease). All cognitively healthy subjects with no brain injuries then participate in a PET/CSF test, and the fraction of A positives is determined only by the disease prevalence. The proposed protocol adds an exclusion layer in which automated MRI-based classification predicts a subset of PreAD subjects that are later subjected to PET/CSF acquisition. In this case, the target cohort is defined by the true positive rate (TP%) of the classification algorithm. The initial participant pool needs to be larger for the proposed protocol due to classifier false negatives that may be excluded.

4.2 Data

The *Fundació Pasqual Maragall* has provided us with 2 datasets of pre-processed MRI images. However, the techniques used to process the data and the atlases used for the brain parcellation are different between them, that is why we have approached it as two independent tasks. The first dataset is parcelled using the *AAL Atlas*, which divides the brain into 90 regions, whereas the second dataset images have been parcelled using the *Desikan-Killiany Atlas*, which defines 42 subcortical and 68 cortical parcellations. In this work, for simplicity when referring to them, we will call them *AAL Dataset* and *DK Dataset*, respectively.

4.2.1 AAL Dataset

The *AAL Dataset* is the one used in Casamitjana et al. 2018 [3] and it is composed by two independent cohorts. The first one has been obtained from the ADNI public database and the second one has been provided by the *Hospital Clínic de Barcelona (HBC)*. The *Fundació Pasqual*

Maragall has labelled each subject as either *Normal, PreAD, MCI or AD* depending on the amyloid levels. However, we are only interested in the first two classes. The number of subjects is summarized in the following table:

	TOTAL	ND		PreAD	
		individuals	%	individuals	%
ADNI	52	32	62	20	38
AETIONOMY	88	69	78	19	22

Table 1 - AAL Dataset description

The AAL Atlas defines 90 regions of interest. The data of this first dataset consists on the *volume* and the *grey matter volume* of each one of the ROIs and the structural connectivity matrices (SCM). Moreover, extra subject-level features such as the APOE and the gender or the age are informed too.

First of all, we will analyse the subject-level features such as age and gender which can give us an intuition of the population average of the dataset. Furthermore, we are going to group the data by *database*: ADNI and AETIONOMY and by *label*: normal (N) and preAD (P). We can show a table summarizing the main information subject-level features. Note that we show the mean and standard deviation for real valued features and the counts for the categorical ones.

		Age		School Years		Intracranial Volume		Gender		APOE	
		mean	std	mean	std	mean	std	M	F	0	1
ADNI	**N**	72.56	5.56	17.19	2.71	1.39e+06	108e+03	15	17	24	8
	P	75.57	4.81	16.40	2.19	1.37e+06	88e+03	7	13	10	10
AETIONOMY	**N**	62.04	7.69	12.22	4.43	1.33e+06	122e+03	44	25	56	13
	P	70.05	7.39	11.16	4.57	1.31e+06	120e+03	13	6	13	6

Table 2 - AAL Dataset subject-level features summary

It is observable a bias between databases in the majority of the features. The average *age* of the ADNI databases is 10 years higher than the one in AETIONOMY. Regarding the *gender*, whereas ADNI has more or less the same number of individuals, AETIONOMY has two times more samples of males (M) than females (F). These two features may have influence on the *intracranial volume* average of the dataset.

Nevertheless, what we mainly want to study in this work are the features of the regions of interest defined by the atlas obtained from T1 MRI, and the structural connectivity matrix (SCM) between ROIs. The regions of interest (ROI) contain two features: (i) the *size* and (ii) the *grey matter* volume.

These features are normalized by the subject's *intracranial volume*. The histograms of the resulting features, grouped by class (Normal and PreAD) and by cohort (ADNI and AETIONOMY) have the following shape:

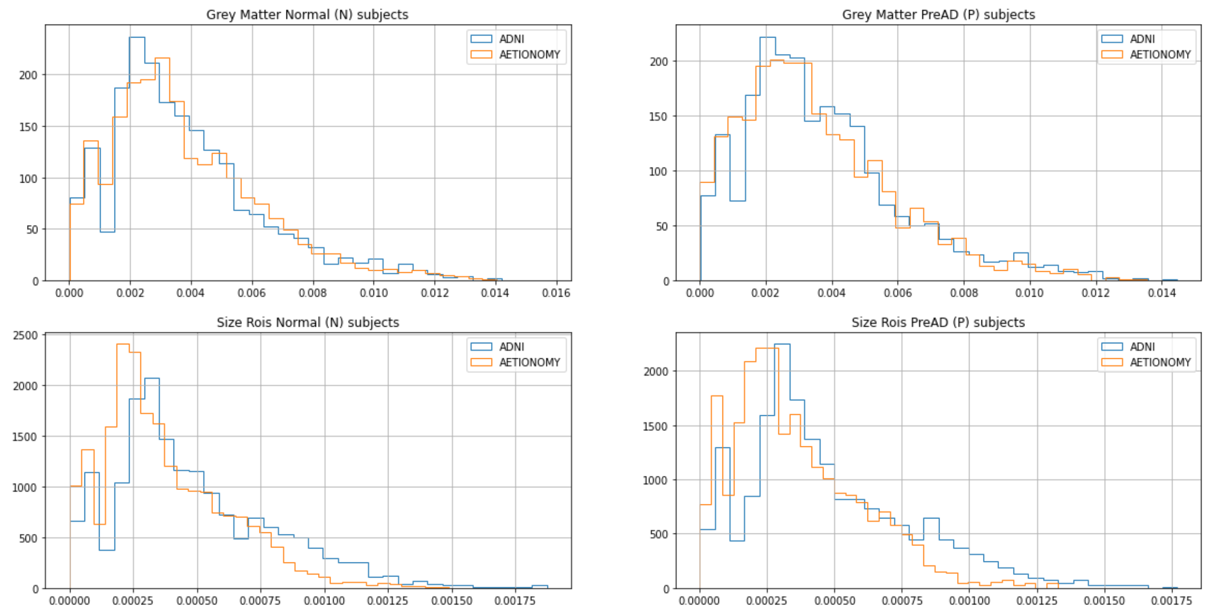

Figure 10 - AAL Dataset node-level features histograms per class and cohort

The histograms clearly look quite distinct, even more for the *size*. We have run the *Kolmogorov-Smirnov* test with a significance level $\alpha = 0.05$to check whether the features, grouped by cohort and label, can be considered of the same distribution (null hypotheses H0) or not (H1). Unfortunately, the *p-value* obtained is below 5% for both *grey matter* and *size rois* of preAD individuals.

Hence, in order to increase the similarity between databases, standardization must be independently applied to the cohorts. In this work, we have used the Z-Score standardization$x^* = (x - \mu_x)/\sigma_x$. We have computed the Z-Score globally per cohort but the output of the KS test has been the same than before the normalization.

However, in this example we are aggregating the information of each ROI as one unique feature whereas there are 90 defined ROIs whose *grey matter* volume and total *size* could be treated as independent features. That is, *grey matter* of *ROI i* can be seen as a different feature than *grey matter* of *ROI j* ($i \neq j$). Hence, the same KS test can be run independently per ROI.

The *grey matter* and *size rois* of 90 ROIs sum up a total of 180 features to run the test with. We present the results using a table that counts the number of nodes on which we must reject the null hypothesis (H0) and which not:

	Grey matter		**Size**	
	Do not reject H0	**Reject H0**	**Do not reject H0**	**Reject H0**
Normal	52	38	1	89
PreAD	57	33	8	82

Table 3 - AAL Dataset node features KS test by ROI results

Apparently, the *grey matter* statistics are more similar between nodes than joining the ROIs. Hence, Z-Score by node instead of globally seems more appropriate for the task. In fact, if a Z-Score by ROI is applied, the test detects almost all of them as matching distributions for both *grey matter* and *size* features and for both classes.

Finally, the edges of the graphs must be analysed as well. The first step has consisted on symmetrizing the adjacency matrix in order to impose that the number of fibres from ROI *i* to ROI *j* is the same than the one that goes contrary wise: $SCM_{ij} = SCM_{ji}$.

We can analyse either the *weights*, or the *node degree*:

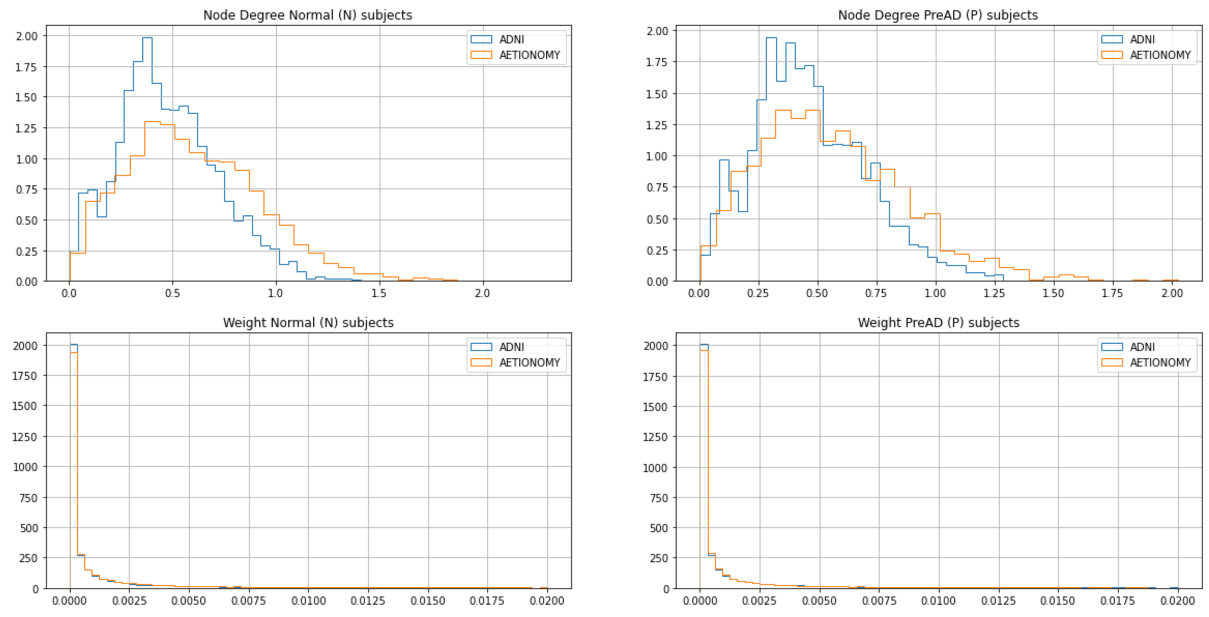

Figure 11 - AAL Dataset node degree and weights histograms per class and cohort

From the *weight* histograms we can deduce that the weights are equally distributed, near to 0, which also means that the range of values of the structural connectivity matrices is the same between cohorts. However, the *node degree* histograms are not similar, which implies that the entries of the SCM are distributed dissimilarly.

The *node degree* has been useful to gain the insight explained, but it is a higher-level feature and focusing on the standardization of the weights is more appropriate. Once again, the *weights* can be treated either (i) globally, (ii) by ROI, or even (iii) by pair of ROIs, that is, processing each entry SCM_{ij}as a feature. Taking into account that the dimensionality of the SCM matrices is 90x90 but that we have symmetrized them and that the diagonal must not be considered, this last approach compresses 4005 features for the SCM matrices.

The normalization that matches more distribution is (iii) Z-Score by pair of ROIs. We have run the tests on samples corresponding to normal and preAD subjects separately, and another extra test merging the two classes. The results are summarized in the following table:

	Weight	
	Do not reject H0	Reject H0
Normal	2591	1414
PreAD	3276	729
All	2275	1730

Table 4 - AAL Dataset weight KS test by pair of ROIs results

Finally, it turns out that the number of edges or node pairs that pass the three tests is 2130.

In summary, the two cohorts are independent. There are population level differences such as the *age* and the *gender* that may have influence on the ROI *volume* and *grey matter*, which we have proved to be not equally distributed but that we have been able to find a mapping that matches the distributions. However, the process has not been that effective with the structural connectivity matrices since only a 53% of the total features have passed the KS tests, which suggests us that using only those edges could help the classification models to generalize between cohorts.

4.2.2 DK Dataset

The *DK Dataset* is composed by two independent cohorts as well. The first one is obtained from the ADNI public database whereas the second is from the ALFA project from Barcelona Beta Brain Research Center. The MRI images have been parcelled using the Desikan-Killiany Atlas and preprocessed using the FreeSurfer software. The *Fundació Pasqual Maragall* has labeled each subject according to their amyloid biomarker following the same criteria above. The numbers of subjects is summarized in the table:

	TOTAL	ND		PreAD	
		Individuals	%	Individuals	%
ADNI	250	161	64	89	36
ALFA	349	193	55	156	45

Table 5 - DK Dataset description

The *Desikan-Killiany* dataset defines 68 cortical and 42 subcortical regions. Two features are added for cortical ROIs, its *volume* and its *thickness*. However, the subcortical regions only contain one feature, the *volume*. This last dataset contains no neither structural nor functional connectivity matrices and the graph connectivity has been created using the adjacency of the regions defined by the atlas in the brain. Finally, as AAL Dataset does, this dataset contains subject-level features such as the APOE, the gender and the age. The following table shows a summary of these subject-level features to get an intuition of the population:

		Age		Gender		APOE		
		mean	**std**	**M**	**F**	**0**	**1**	**2**
ADNI	**N**	71.50	6.68	74	87	130	31	0
	P	72.05	6.12	36	53	34	46	9
ALFA	**N**	60.85	4.47	68	125	111	72	10
	P	61.72	4.88	69	87	53	84	19

Table 6 - DK Dataset subject-level features summary

Once again, the average *age* of the individuals included in the ADNI cohort is 10 years higher than the ones in ALFA. However, we can observe a similar distribution of the counts *gender* between cohorts and a high correlation between the preAD (P) subjects and both the APOE 1 and 2 categories, from this we may infer that the APOE will have a relevant role in this dataset.

As we have mentioned, the node-level features are the *volume* and the *thickness* for cortical regions and the *volume* for the ones from the subcortex, which are normalized by the *intracranial volume* of the subjects. The resulting histograms are the following:

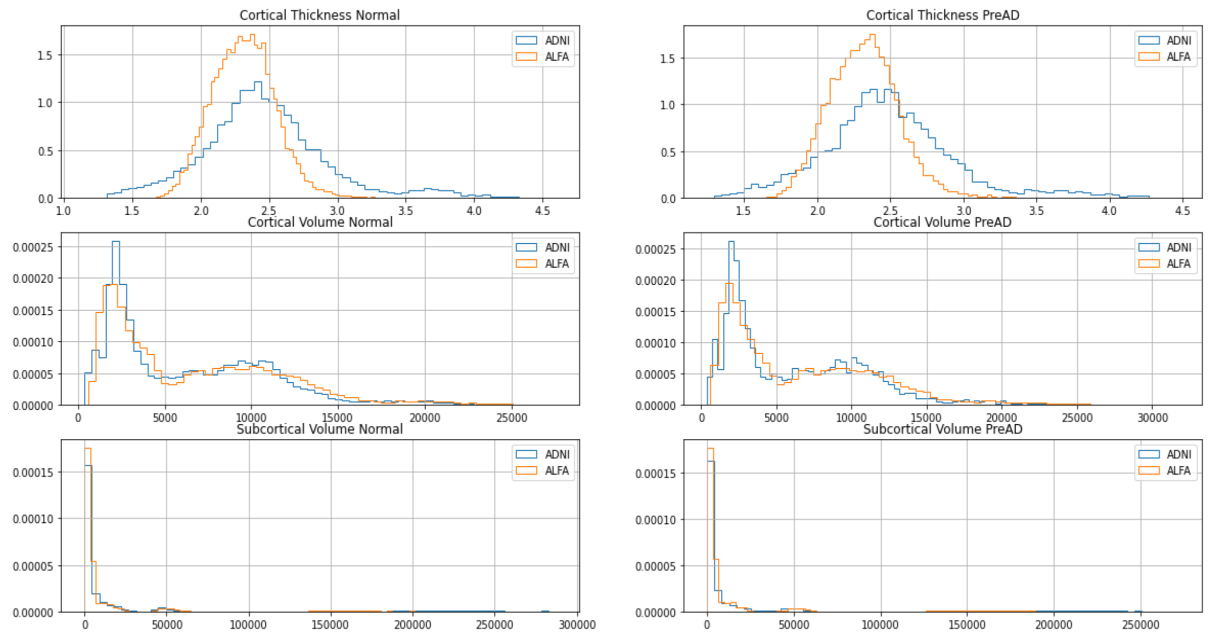

Figure 12 - DK Dataset node-level features histograms per class and cohort

Following the same procedure than with the *AAL Atlas*, we have run the KS test, whose result has only been successful using the subcortical volumes. After applying a Z-Score normalization to (i) thickness of cortical regions, (ii) volume of cortical regions and (iii) volume of subcortical regions does not imply significant improvements on the results of the test.

The analysis can be conducted using the *volume* and *thickness* of the defined ROI as independent features as well. Unfortunately, the outcomes are not as satisfactory as the results of this approach on the *AAL Dataset*.

Taking everything into consideration, we can conclude that the *volume* of the subcortical regions is the feature with more similar distributions between cohorts and it may be considered if it is desired to use both of them. However, it does not mean that the subcortical *volumes* are the most discriminative features between classes. Finally, we have seen that the APOE may be a relevant feature to distinguish between classes.

4.3 Procedures

4.3.1 AAL Dataset

4.3.1.1 Graph Classification using GNNs

A set of graphs with a label assigned to each one of them is needed to train a model for a graph classification task in a supervised manner. That is, we need a dataset of the form $D = \{ (G_i, y_i) \}\ i = 1 \ldots N$ where $G_i = (V_i, E_i)$ is a graph and V_i, E_i are its nodes and edges sets, respectively. Moreover, a graph G_i can have a d-dimensional signal x_i indexed by its vertices $x_i: V_i \to \Re^d$ and a weight mapping on its edges if the graph is weighted $w_i: E \to \Re$.

The set of graphs built from the AAL Dataset assigns a graph to each one of the subjects of the dataset. The method considers each ROI defined by the atlas as a node of the graph and edges are obtained using the structural connectivity matrix. Since the structural connectivity matrix connects each ROI with almost all the other ones, that imply that each resulting graph is almost complete, depending on the subject, there are some NULL or zero entries and that is why not all graphs can be considered as complete. Due to the nature of the structural connectivity matrix, a weight, representing the number of fibres that connect two ROIs, is assigned to each edge so the graphs are weighted. Furthermore, the *grey matter* of each ROI has been used to define a 1-dimensional signal on the nodes of each graph.

This graph construction methodology used implies that every network of the dataset will have the same number of nodes, 90, but both the edges weights and the node signals will be different. Intuitively, the number of fibres that goes from ROI i with another ROI j may be the same than the ones that goes the other way around. That is why, in every experiment runned, apart from other scalings and standardizations, the structural connectivity matrix has been symmetrized. Hence, each graph of the dataset is formed by a set of 90 nodes and a 90x90 weighted and symmetric adjacency matrix.

In this task, the main difference between experiments is the model we have used, that is, its convolutional layers as well as the global pooling function.

4.3.1.1.1 Experiment 1: Classic GNN approach

Because of the fact that the dataset is small, using too deep models with a lot of learnable parameters would lead us to overfit the training set and not being able to generalize. The hypothesis of this first experiment has been that we should use a few convolutional layers that do not increase the number of channels too much.

Convolution
The convolutional part of the model consists on two spatial convolutional layers that increase the number of channels of the signal to 3. The non-linearity applied after each layer is a ReLU function. The output of this block is directly the output of the last activation function.

The usage of hierarchical pooling operators has been experimented in this block but they have not improved the performance. For simplicity of the model, we have removed them.

Global Pooling
A more sophisticated global pooling operator has been used in this model since it notably increased the results compared to more generic operators such as *mean* or *max*. Concretely, the global sort pooling [20], outputting the feature channels of the *K* (10) selected nodes.

Projection
The previous pooling function returns a tensor of size *K* (10) by the number of feature channels (3), which can be considered a large tensor if we want to fetch them in a fully connected layer. The original paper proposes 1D convolutions after the layer, but it did not improve out results. In order to increase the number of parameters the minimum, we have taken the naivest approach, which consists on using one single linear layer that maps the 30-dimensional vector to two output units, that are the inputs for the final softmax function.

Data	TRAIN set	AETIONOMY
	TEST set	ADNI
	Subject-level features (Graph features)	*Gender*: One-hot + Global Z-Score *APOE*: One-hot + Global Z-Score
	ROI-level features (Node features)	*Grey Matter Volume* Intracranial normalization + Z-Score
	Edge features (Shared)	*Structural Connectivity* Symmetrize + Intracranial normalization + Z-Score
	Edge threshold	0.5
Hyperparameters	Learning rate	5e-4
	Weight Decay	5e-3
	Loss weights	Inverse of the frequency *Normal*: 1.27 *PreAD*: 4.63

Table 7 - AAL Dataset experiment 1 configuration

We show results on both the training set and the test set after training 1000 epochs to show the model is able to learn but it fails fitting the other cohort.

	precision	recall	f1-score	accuracy	PR AUC	ROC AUC	savings cost	savings burden
mean	0.610371	0.907895	0.728622	0.852273	0.744133	0.923150	0.522605	0.669281
std	0.064247	0.026316	0.051059	0.038256	0.084500	0.029294	0.033828	0.038645

Table 8 - AAL Dataset train performance (1000 epochs)

	precision	recall	f1-score	accuracy	PR AUC	ROC AUC	savings cost	savings burden
mean	0.527817	0.6875	0.596633	0.644231	0.572516	0.686133	0.413685	0.620513
std	0.023570	0.0750	0.042877	0.024827	0.059685	0.049462	0.044331	0.016977

Table 9 - AAL Dataset test performance (1000 epochs)

After getting acceptable results for such a complex task with this configuration, we aimed to train the model using the ADNI database and test it on AETIONOMY in order to compare the results with the ones presented in Casamitjana et al. 2018 [3]. However, we have not been able to get comparable results, the model is able to learn the training data but is not able to generalize to the other cohort. As we will discuss in next chapter, it is due to the differences between cohorts, the prevalence of the classes and the amount of data.

Finally, because of the advanced global pooling operation, it is more difficult to interpret the results and learnings of the model. In addition, the results of Casamitjana et al. 2018 [3] suggest that the more relevant features for this task correspond to entries of the structural connectivity matrix, that is, the edges.

4.3.1.1.2 Experiment 2: Focus on the edges

A weighted graph is a graph G=(V,E) on which a mapping $w: E \rightarrow \Re$ is defined, the idea is that the main information may be contained by w instead of by f and that w can be seen as another signal, this time defined on the edges $w : E \rightarrow \Re$. Classic GNNs operate on signals defined on the nodes of the graph $f : V \rightarrow \Re^D$ and perform a Message Passing operation between neighbor nodes linked by and edge which can be either weighted or not. Hence, edge information is sometimes not correctly captured by GNN layers if the network is not deep enough and the performance can be even worse when the final pooling operation is not adequate.

In this section we have developed and used a layer that operates directly on the edges of the graph. Each block takes as input the features of the edge, the signal of the incident nodes of the edge and the aggregation of the features of the other edges the nodes are incident. Similar to Kearnes et al. 2016 [37], the edge states are updated at the output of each layer, concretely, the function that updates the hidden state h_{uv} of an edge uvat layer l is:

$$h_{uv}{}^{l} = \sigma(W_0{}^{l} h_{uv}{}^{l-1} \,||\, W_1{}^{l} A_x\{x_u{}^{l-1}, x_v{}^{l-1}\} \,||\, W_2{}^{l}\, A_e\{e \in E(u) \cup E(v)\})$$

where ${W_0}^l, {W_1}^l, {W_2}^l$ are learnable weights and bias of the layer, ${h_{uv}}^{l-1}$ is the state of the edge uv at the previous layer, x_u, x_v are the signals at nodes u and v,respectively, and A_x is any shift invariant aggregator function such as *sum*, *max* or *mean*. Note that the output of the layer is a graph of the same size than the input one, but with updated edge feature vectors, which could be of higher dimensionality.

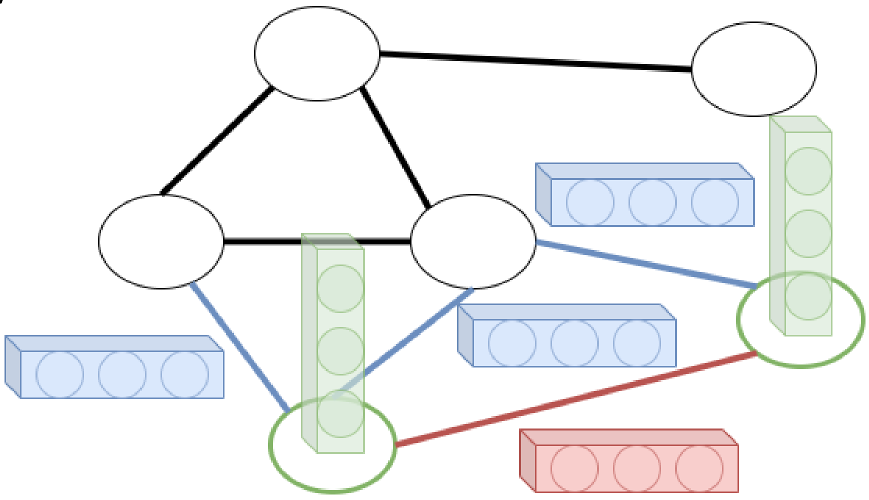
Figure 13 - Proposed graph convolution layer that updates edge states

Thanks to the advanced mini-batch strategy used in PyTorch Geometric it is straightforward to implement the layer explained. Having the following tensors representing the graphs:

- $edge\ index$: $|E|\ x$ 2 LongTensor that contains the index of the source and destination nodes of the edges.
- $edge\ weight$: $|E|\ x\ \Re^{d_w}$ Float tensor containing the weight of each edge.
- x : $|V|\ x\ \Re^{d_x}$ FloatTensor that contains the d-dimensional graph signal at each node of V.

The information of the nodes incident to the edge can be easily obtained as:

$$m = A_{uv}\{\ x[edge\ index, 0]\ , x[edge\ index, 1]\ \}$$

Then, m can be concatenated to $edge\ weight$ to compute the $(d_x + d_w)$-dimensional input vector to the layer.

Convolution

The convolutional part of the models stacks three blocks that operates on both edges and nodes. The block is composed by (i) a layer like the one described above that updates the states of the edges to a higher dimensionality, (ii) a neural network that maps the obtained edge states to a 1-dimensional weight and (iii) a spatial graph convolutional layer that uses the new weights. In order to do not increase too much the number of parameters of the model the node signal outputted by the spatial graph convolutional layers is always 1-dimensional.

The final output is composed by both the concatenation of the edges hidden states along the three blocks and the concatenation of the nodes.

Global Pooling

The global pooling operator implemented performs a global mean on both the nodes and edges vectors outputted by the convolutional part and concatenates them.

Projection
Once again, since the representation of the whole graph after the global pooling is a high dimensional vector, only one Linear layer and a softmax is applied to do not increase the number of parameters and avoid overfitting.

The results obtained are the following:

	precision	recall	f1-score	accurac y	PR AUC	ROC AUC	savings cost	savings burden
mean	0.489120	0.802632	0.605241	0.772727	0.713289	0.871472	0.428979	0.588423
std	0.047627	0.050391	0.021756	0.030773	0.015006	0.009975	0.017268	0.036710

Table 10 - AAL Dataset experiment 2(a) train performance

	precision	recall	f1-score	accuracy	PR AUC	ROC AUC	savings cost	savings burden
mean	0.466866	0.6875	0.555665	0.576923	0.583202	0.651953	0.375858	0.571154
std	0.017621	0.0250	0.010692	0.022206	0.059083	0.031025	0.011375	0.016185

Table 11 - AAL Dataset experiment 2(a) test performance

However, if we train on ADNI and test on AETIONOMY using the same model and the same hyperparameter configuration, in except of the loss weights which are set to `(1.625, 2.6)` since the ADNI database is more balanced than AETIONOMY, the performance on test set is worst:

	precision	recall	f1-score	accuracy	PR AUC	ROC AUC	savings cost	savings burden
mean	0.637969	0.725000	0.678483	0.735577	0.795721	0.858203	0.483310	0.685952
std	0.031361	0.028868	0.026950	0.024198	0.030409	0.018149	0.020031	0.015006

Table 12 - AAL Dataset experiment 2(b) train performance

	precision	recall	f1-score	accuracy	PR AUC	ROC AUC	savings cost	savings burden
mean	0.453448	0.592105	0.512318	0.755682	0.423243	0.696606	0.316126	0.554167
std	0.054047	0.050391	0.045380	0.032804	0.093628	0.057299	0.061430	0.053359

Table 13 - AAL Dataset experiment 2(b) test performance

4.3.2 DK Dataset

4.3.2.1 Graph Classification Using GNNs

The DK Dataset uses a different brain parcellation than the AAL Dataset. This dataset includes the volume of the 68 cortical and 42 sub-cortical regions and the thickness of the regions of the cortex. This can be used as a 2-dimensional node signal if we use only cortical regions, and 1-dimensional signal, corresponding to the volume, if we use both cortical and subcortical regions. However, no structural neither functional connectivity are included so it is not possible to directly assign a a connectivity between nodes.

Graph Connectivity Definition
We have defined one single connectivity matrix using the Desikan Killiany Atlas itself and it is shared by each graph of the dataset. Hence, the only difference between networks will be the node signals. The idea behind the fact that the graph structure is shared across subjects is that we are changing the domain of the signal from an Euclidean space to the new domain defined by the graph. For example, imagine we are only considering the volume of both cortical and subcortical parcellations, which sum up a total of 110 regions. In a classical ML approach, each subject would be represented by a vector in a 110-dimensional Euclidean space whose entries are the volume of each one of the ROIS. In fact, parallely to this work, *Fundació Pasqual Maragall* is carrying out experiments in this dataset with that approach. However, defining this one single graph of 110 nodes and indexing the signal on them, we are creating a new domain that modifies the interaction between ROIs. GNNs allows us to take into account those interactions and to learn from this new domain by classifying signal defined on it.

We have built the graph connectivity by analysing the labelled volume of the atlas that determines the regions and their boundaries in the brain and assigning an edge between those ROIs whose boundaries are adjacent. Knowing that each voxel of the volume is labelled accordingly to the region they correspond to or if they correspond to none, we have scanned the neighbourhood of each voxel and if two voxels of different regions are adjacent, an edge between those two ROIs is added. We have only taken into account the adjacent voxels in either *x*, *y* or *z* directions, the diagonal adjacent voxels has not been considered as part of the neighbourhood. In order to add extra information to the graph, we have weighted the edges by counting the number of neighbour voxels between ROIs. Consequently, the resulting adjacency matrix of the graph is symmetric and weighted.

Model
As well as any classic model based on CNNs for image classification tasks, models for graph classification usually consists on a set of convolutional and pooling layers, a global pooling operator and a projection multi-layer perceptron, followed by a final non-linear activation such as the softmax function.

Convolution
The convolutional part of the model used in this task is based on the Chebyshev Convolution, stacking 3 ChebConv layers that maps the node signal channels from 1 to 5 and uses ReLU as non-linearity. Due to the fact that we want to carry out a class activation mapping analysis, no hierarchical pooling layers has been added. Moreover, a batch normalization after each convolutional layer has been used for training purposes.

After the 3 convolutional layers, the output of each layer and the initial signal are concatenated for each example, obtaining a higher dimensional signal indexed by the nodes of each graph.

Global Pooling

We have implemented a global pooling layer that takes advantage of the fact that it is not a general network classification task, such as molecules classification, and every subject or input graph will have the same number of nodes.

The layer performs a weighted sum of the signal defined on the nodes, obtaining a vector of the same dimension that represents the embedding of the whole network. This summation is achieved by a learnable |V|-dimensional vector, w. Being $f(v)$the d-dimensional signal defined at node v at the output of the convolutional part of the model and w_v the weight assigned to that model, the output of this global pooling operator f_G is defined as:

$$f_G = \sum_{v \in V} w_v \, f(v)$$

Note that this global pooling operation will not output the same result for two isomorphic graphs with the nodes reordered. Hence, this is only applicable to this task since we can ensure that the nodes of each input sample will be in the same order, that is, the *i-th* node of each graph corresponds to the same ROI.

This layer can be seen as a generalization of the *global average pooling* since if we fixed$w_v = 1/|V|$ per each node v, f_G would be the average across the nodes. However, the results obtained using this approach are better, since we are letting the network to learn which are the nodes that are more relevant.

Projection

The projection layer only consists on the concatenation of the subject-level features such as the APOE or the gender with the f_G obtained from the previous hierarchical pooling layer and a linear layer that outputs two units followed by a log-softmax to determine the log-probability to belong to each one of the classes.

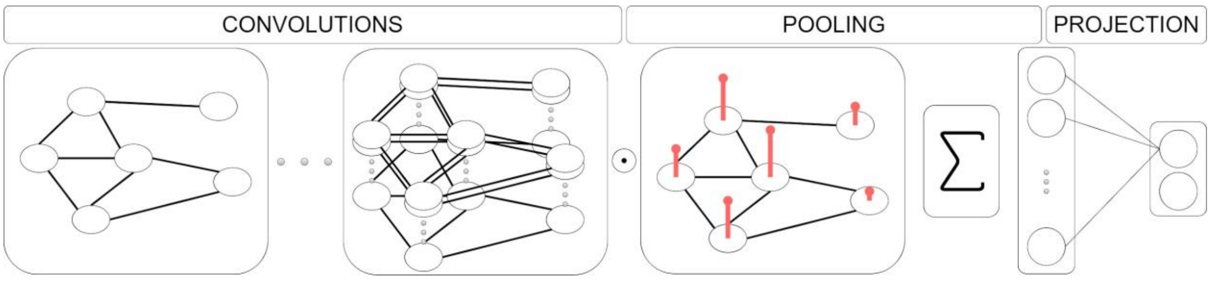

Figure 14 - Model architecture used in DK Dataset experiments

4.3.2.1.1 Experiment 1: Volume of subcortical ROIs

This first experiment consists on building the graphs by only using the subcortical regions of the atlas. Hence, the signal defined on the nodes is 1-dimensional corresponding to the volume of each ROI.

The data is normalized by subject and standardized. The two steps data pipeline consist on (i) normalize the volume of each ROI by the intracranial volume of the subject, (ii) global z-score standardization, that is, instead of considering the volume of the ROI i and the one from ROI j, $i \neq j$, as two independent features, compute one single mean and standardization using together every ROI. Moreover, the weight assigned to the edges have been used as well, all of them have been normalized by a factor of $2/\sum_{(u,v)\in E} w_{uv}$. Finally, the subject-level features used are the age and the APOE, which have been one-hot encoded and standardized using z-score.

Only the ADNI database has been used, splitting it in TRAIN, DEVEL and TEST sets. The data and the training configuration is summarized in the following table:

Data	TRAIN set	70% ADNI
	DEVEL set	15% ADNI
	TEST set	15% ADNI
	Subject-level features (Graph features)	*Age*: Global Z-Score *APOE*: One-hot + Global Z-Score
	Nodes	42 subcortical ROIs
	ROI-level features (Node features)	*ROI volume*: Intracranial normalization +Global Z-Score
	Edge features (Shared)	*Count adjacent voxels:* Sum normalization
Hyperparameters	Learning rate	5e-4
	Weight Decay	5e-3
	Loss weights	Inverse of the frequency *Normal*: 1.56 *PreAD*: 2.76

Table 14 - DK Dataset experiment 1 configuration

The outcome achieved with this configuration are shown in the following table:

	precision	**recall**	**f1-score**	**accuracy**	**PR AUC**	**ROC AUC**	**savings cost**	**savings burden**
mean	0.641689	0.775559	0.698026	0.736842	0.639186	0.765895	0.490001	0.676788
std	0.131280	0.100225	0.110057	0.135469	0.130197	0.123848	0.077222	0.070583

Table 15 - DK Dataset experiment 1 test results (ADNI)

Moreover, we decided to use the same models obtained from different runnings to directly test them on ALFA database, obtaining the following results:

	precision	recall	f1-score	accuracy	PR AUC	ROC AUC	savings cost	savings burden
mean	0.545862	0.637179	0.587132	0.600573	0.570887	0.627860	0.403499	0.633518
std	0.009464	0.050189	0.020880	0.009196	0.018739	0.016847	0.023121	0.006431

Table 16 - DK Dataset experiment 1 test results (ALFA)

4.3.2.1.2 Experiment 2: Volume of both cortical and subcortical ROIs

This second experiment uses the same configuration than the previous one but takes into account the whole set of regions defined by the Desikan-Killiany, building graphs of 110 nodes.

Data	TRAIN set	70% ADNI
	DEVEL set	15% ADNI
	TEST set	15% ADNI
	Subject-level features (Graph features)	*Age*: Global Z-Score *APOE*: One-hot + Global Z-Score
	Nodes	42 subcortical ROIs 68 cortical ROIs
	ROI-level features (Node features)	*ROI volume*: Intracranial normalization +Global Z-Score
	Edge features (Shared)	*Count adjacent voxels:* Sum normalization
Hyperparameters	Learning rate	5e-4
	Weight Decay	5e-3
	Loss weights	Inverse of the frequency *Normal*: 1.56 *PreAD*: 2.76

Table 17 - DK Dataset experiment 2 configuration

The achievements on the TEST sets of ADNI are slightly better than the ones obtained in the first experiment.

	precision	recall	f1-score	accuracy	PR AUC	ROC AUC	savings cost	savings burden
mean	0.662187	0.829483	0.735692	0.763158	0.784549	0.834036	0.522386	0.693993
std	0.082279	0.052060	0.071128	0.107434	0.147228	0.125225	0.048400	0.042738

Table 18 - DK Dataset experiment 2 test results (ADNI)

However, if we test the trained models on ALFA, the majority of the trained models classify whole dataset as the same class (either 0 or 1, depending on the execution), getting then poor results:

	precision	recall	f1-score	accuracy	PR AUC	ROC AUC	savings cost	savings burden
mean	0.186514	0.329487	0.234553	0.528367	0.541958	0.562216	0.164008	0.228137
std	0.255758	0.468072	0.321944	0.046095	0.022091	0.283462	0.226108	0.312642

Table 19 - DK Dataset experiment 2 results (ALFA)

Due to the hierarchical pooling operation implemented the number of parameters of the model increases according to the number of nodes of the input graph. Moreover, as we have showed in previous sections, the two databases are independent and the volume of the cortical regions were the most statistically independent features between databases so adding them may have a big influence in this poor performance.

4.3.2.2 Interpretability

In this subsection we interpret the learnings of the models. We consider the Experiment 1 the most robust and successful and that is why we base this section on it. Concretely, we are have studied the role of the APOE in the decisions of the model and a class activation mapping to detect which are the more relevant nodes.

Class Activation Mapping

Class activation mapping is classically approached by using a global mean pooling operation. Nevertheless, the model used for the classification task of *DK Dataset* does not use a global average pooling operation after the convolutional layers. Instead, the network performs a learning pooling operation which directly assigns a weight w_v to each one of the nodes of the graph. This weight is applied to each unit i of the node. Hence, being $f_i(v)$ the activation of unit i in the last convolutional layer for node v, the result of the pooling operation for that unit is $F_i = \sum_v \quad w_v\, f_i(v)$. Note that if each w_v were fixed as $w_v = 1/|V|$, a global pooling operation would be performed. Once again, the input to the final classification softmax for class c is $\sum_i \quad {w_i}^c F_i$where ${w_i}^c$is the importance of unit i for class c. Finally, the class activation mapping for each node v using this pooling operation has been defined as $M_c\ (v) \ = w_v \sum_i \ {w_i}^c f_i(v)$.

As well as they did in Arslan et al. 2018 [24], to get the population-level more relevant ROIs we have taken the *top-k* nodes with a higher class activation mapping (*argmax*). Then, we have averaged the number of times a node appears as one of the most k relevant across subjects and executions and normalized between 0 and 1 to compute the population relevance of each node. In our experiments we have used $k = 3$.

It turns out that from the 8 most important ROIs obtained, 6 are 3 feature pairs corresponding to the same region but from left and right hemispheres. The features are plotted separately in order to be able to localize them easier:

1. Optic chiasm

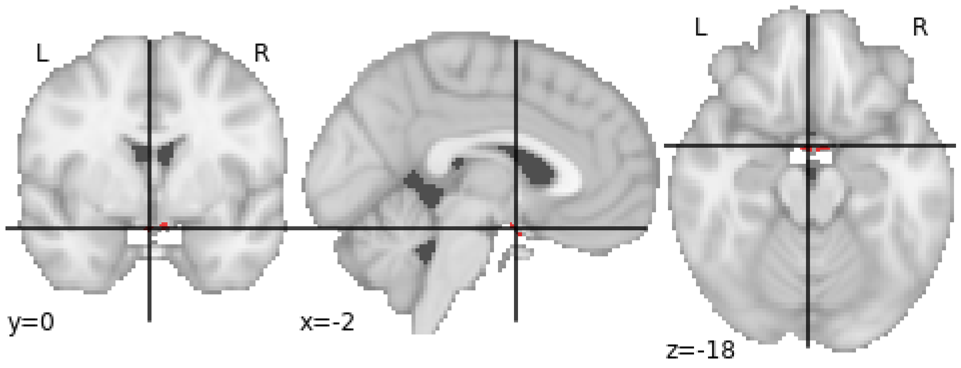

Figure 15 - Optic chiasm

2. Left and right cerebral white matter

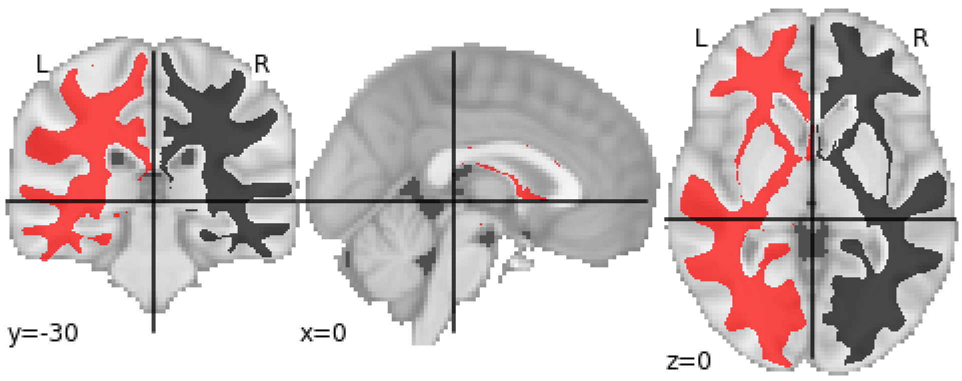

Figure 16 - Left and right cerebral white matter

3. Left and right cerebellum white matter

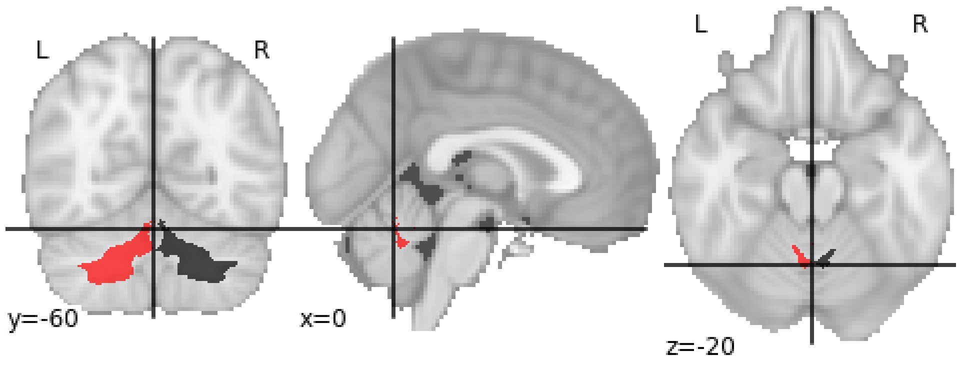

Figure 17 - Left and right cerebellum white matter

4. Left and right inferior lateral ventricle

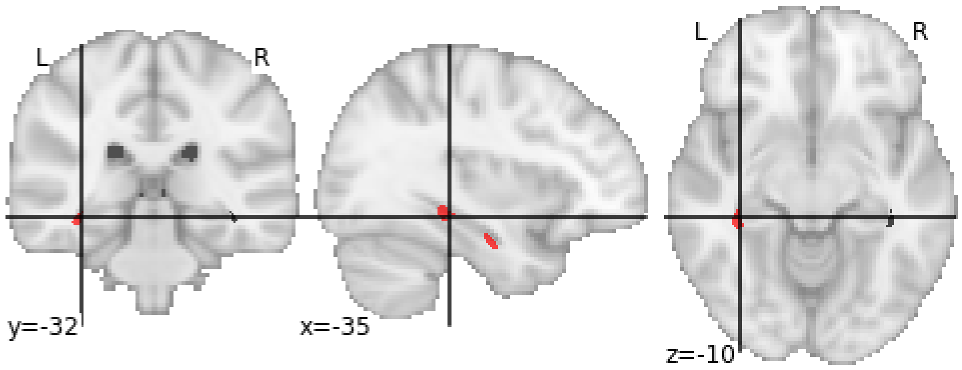

Figure 18 - Left and right inferior lateral ventricle

APOE

The APOE is one-hot encoded, directly concatenated with the output of the pooling operator and fetched to the final linear layer that maps them to the two output units and the softmax. Hence, the same idea behind the class activation mapping can be applied for the APOE. However, in this case, the weight that maps each APOE category A_x to the output units corresponding to either Normal or PreAD class c can be directly seen as the importance of the that category for the class: ${w_{A_x}}^c$. In order to get a meaningful metric of the relevance of the APOE we have measured the contribution of the APOE categories to the output classes and compared it with the total activation, which consist on the contribution of the output of the convolutional layers, the gender and the APOE itself.

The contribution to be classified as preAD for individuals that carry both mutations of the APOE-ε4 is the 25% compared with the convolutional models. However, this can be hidden by the output of the convolutional part if the score to be classified as normal obtained from the convolutional layers is very large. Subjects that carry one mutation of APOE-ε4 have a minor contribution of that feature and highly depend on the output of the convolutions.

5 Conclusions

5.1 Summary

Deep Learning models are known for solving problems in an end-to-end manner, that is, avoiding the usage of hand-engineering features and learning by itself those features that are more relevant for the task from raw data. To do so, the models stack layers which learn new features from the ones learned on the previous layers, that is, the models learn the task by function compositions which are able to detect hidden patterns. This approach requires a large number of parameters that, at the same time, require a large amount of data in order to be trained correctly.

The results obtained in the AAL Task are under our expectations since the performances of the models are not comparable with the ones presented in Casamitjana et al. 2018 [3]. We have been experimenting with the state-of-the-art graph convolutional layers and pooling operations, but we have not been able to obtain better results than the ones shown in this document.

In spite of the results, we strongly believe that GNNs is a set of very powerful algorithms. In fact, they have achieved the state-of-the-art in a lot of tasks, replacing, for example, Graph Kernels. However, as any other Deep Learning model, they require big datasets to be trained.

The *AAL Dataset* is composed by two small, different, independent databases: ADNI (52 subjects) and AETIONOMY (88 subjects). We believe that our results are influenced by two main factors related with the dataset: (i) the size of the databases and (ii) the statistical differences between them.

First of all, overfitting is an effect that occurs when a model perfectly fit a set of data but is unable to fit additional or unseen data. When working with small datasets, the models must not have too many parameters since they will rapidly overfit the training data and they will not generalize accurately. There are regularization techniques to mitigate that effect such as Dropout and weight decay, which actually have been used in our models, but if the dataset is that small, they are not enough.

Moreover, as well as they did in Casamitjana et al. 2018 [3], we aimed to train the model using one database and testing on the other to see how well did the model generalize. In order to compare the databases, standardizations and normalizations have been applied to their data independently but there are still differences between them after those pre-processing steps, which implies that, although a model learns one of the databases and uses regularization techniques to do not overfit, does not mean that it will perform well on the other since their statistics may be different.

A naive solution to the database independence would be to merge the databases and split them randomly. However, the number of subjects is very short and since there is no easy detectable hidden pattern, if the model is too simple is not able to learn from the training data and, if it is too complex, rapidly overfits.

The results on the *DK Dataset* are the proof of our conclusions exposed in this section. Since the ADNI database is large enough to be splitted getting meaningful sets, we have been able to train, validate and test on the same database. This fact ensures us that the statistics between sets are the same and that means that from the problems exposed, we only must worry about the overfitting.

From the results of both first and second experiments we deduce that we have been able to avoid overfitting and the models generalize well to unseen data of the same statistics.

The performance on ALFA is inferior in both experiments. Nonetheless, whereas the first model has achieved acceptable results, the second one has not been able to distinguish between subjects of different classes and for each execution it has assigned the same label to the whole set. This validates our hypothesis that, although we can correctly fit a database, it does not mean that the model will generalize to an independent cohort that has been generated using different machinery and configurations, which implies that their overall statistics will be different. In fact, the outcome of the data analysis section has been that the volumes of the cortical regions are statistically different between cohorts. They have been used in the second experiment and that is the reason why the performance on ALFA is that unsatisfactory in the second experiment although it fits the ADNI database better than the first one.

In the first experiment, where only the subcortical volumes have been used, satisfactory results on both ADNI and ALFA have been achieved. For this reason, we consider the graph-based approach proposed in this work as a promising, interesting line of research in neuroimaging and Alzheimer's disease.

In closing, Graph Neural Network are a powerful, flexible family of models that are able to detect and learn hidden patterns defined on non-Euclidean spaces. They are generic enough to accept inputs of any size and that means that the same problem could even be faced defining each graph in a different manner, not as a fixed number of ROIs as we have done in this project.

5.2 Future Work

First of all, in this work we have seen the problems involved in dealing with independent cohorts that have been generated from different sources and that are statistically distinct although they represent the same information. This is a common situation in real world applications and new standardizations and mappings between cohorts techniques, that could be based on Deep Learning as well, should be studied and considered for this concrete task.

The ALFA project from Barcelona Beta Brain Research Center is a big, ongoing project so it is an opportunity to enhance the achievements of this initial work by, for example, adding extra information such as the structural connectivity matrices from DTI to use as connectivity of the graphs as we have done with the *AAL Dataset*. Having bigger datasets would help to draw more robust conclusions.

Finally, as far as we know, the ALFA project is screening the same individuals periodically, which opens a window to combine the classification of the preclinical stage of AD with a time-forecasting of the brain structure and the likelihood of developing the disease based on last screenings. New graph time-forecasting methodologies would have to be developed to follow with the approach of this work, but we believe that the tracking evolution of amyloid-positive and normal individuals could notably improve the performance of this kind of algorithms on the prediction of the AD.

Bibliography

[1] A. Casamitjana, “Study of early stages of Alzheimer’s disease using magnetic resonance imaging,” Barcelona, 2019.

[2] J. Cummings and N. Fox, “Defining Disease Modifying Therapy for Alzheimer’s Disease,” *J Prev Alzheimers Dis*, vol. 4, no. 2, pp. 109–115, Apr. 2017.

[3] A. Casamitjana *et al.*, “MRI-Based Screening of Preclinical Alzheimer’s Disease for Prevention Clinical Trials,” *J. Alzheimers. Dis.*, vol. 64, no. 4, pp. 1099–1112, 2018.

[4] R. Diestel, *Graph Theory*. New York: Springer-Verlag, 2000.

[5] L. Stanković and E. Sejdić, Eds., *Vertex-Frequency Analysis of Graph Signals*. Springer, Cham, 2019.

[6] C. Morris *et al.*, “Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks,” *arXiv [cs.LG]*. 2018.

[7] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[8] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, “Gated Graph Sequence Neural Networks,” *arXiv [cs.LG]*. 2015.

[9] K. Cho *et al.*, “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[10] Z. Wu, S. Pan, F. Chen, C. Long Guodong and Zhang, and P. S. Yu, “A Comprehensive Survey on Graph Neural Networks,” *IEEE Trans Neural Netw Learn Syst*, Mar. 2020.

[11] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral Networks and Locally Connected Networks on Graphs,” *arXiv [cs.LG]*. 2013.

[12] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering,” *arXiv [cs.LG]*. 2016.

[13] T. N. Kipf and M. Welling, “Semi-Supervised Classification with Graph Convolutional Networks,” *arXiv [cs.LG]*. 2016.

[14] J. Atwood and D. Towsley, “Diffusion-Convolutional Neural Networks,” *arXiv [cs.LG]*. 2015.

[15] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive Representation Learning on Large Graphs,” *arXiv [cs.SI]*. 2017.

[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph Attention Networks,” *arXiv [stat.ML]*. 2017.

[17] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural Message Passing for Quantum Chemistry,” *arXiv [cs.LG]*. 2017.

[18] C. Cangea, P. Veličković, N. Jovanović, T. Kipf, and P. Liò, “Towards Sparse Hierarchical Graph Classifiers,” *arXiv [stat.ML]*. 2018.

[19] R. Ying, J. You, C. Morris, W. L. Ren Xiang and Hamilton, and J. Leskovec, “Hierarchical Graph Representation Learning with Differentiable Pooling,” *arXiv [cs.LG]*. 2018.

[20] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, “An End-to-End Deep Learning Architecture for Graph Classification,” 2018.

[21] M. Fey and J. E. Lenssen, “Fast Graph Representation Learning with PyTorch Geometric,” *arXiv [cs.LG]*. 2019.

[22] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” *arXiv [cs.LG]*. 2019.

[23] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning Deep Features for Discriminative Localization,” *arXiv [cs.CV]*. 2015.

[24] S. Arslan, S. I. Ktena, B. Glocker, and D. Rueckert, "Graph Saliency Maps through Spectral Convolutional Networks: Application to Sex Classification with Brain Connectivity," *arXiv [cs.CV]*. 2018.

[25] B. Duthey and S. Tanna, "Background Paper 6.11 - Alzheimer Disease and other Dementias." .

[26] "What is Alzheimer's disease?," *Alzheimer's Society*. .

[27] R. A. Sperling *et al.*, "Toward defining the preclinical stages of Alzheimer's disease: recommendations from the National Institute on Aging-Alzheimer's Association workgroups on diagnostic guidelines for Alzheimer's disease," *Alzheimers. Dement.*, vol. 7, no. 3, pp. 280–292, May 2011.

[28] B. Dubois *et al.*, "Preclinical Alzheimer's disease: Definition, natural history, and diagnostic criteria," *Alzheimers. Dement.*, vol. 12, no. 3, pp. 292–323, Mar. 2016.

[29] E. H. Corder *et al.*, "Gene dose of apolipoprotein E type 4 allele and the risk of Alzheimer's disease in late onset families," *Science*, vol. 261, no. 5123, pp. 921–923, Aug. 1993.

[30] A. M. Saunders *et al.*, "Association of apolipoprotein E allele epsilon 4 with late-onset familial and sporadic Alzheimer's disease," *Neurology*, vol. 43, no. 8, pp. 1467–1472, Aug. 1993.

[31] L. A. Farrer *et al.*, "Effects of age, sex, and ethnicity on the association between apolipoprotein E genotype and Alzheimer disease. A meta-analysis. APOE and Alzheimer Disease Meta Analysis Consortium," *JAMA*, vol. 278, no. 16, pp. 1349–1356, 1997.

[32] V. Leoni, "The effect of apolipoprotein E (ApoE) genotype on biomarkers of amyloidogenesis, tau pathology and neurodegeneration in Alzheimer's disease," *Clin. Chem. Lab. Med.*, vol. 49, no. 3, pp. 375–383, Mar. 2011.

[33] B. M. Hubbard, G. W. Fenton, and J. M. Anderson, "A quantitative histological study of early clinical and preclinical Alzheimer's disease," *Neuropathol. Appl. Neurobiol.*, vol. 16, no. 2, pp. 111–121, Apr. 1990.

[34] E. P. Fortea, "Classification techniques for Alzheimer's disease early diagnosis," Barcelona, 2015.

[35] "Brain Parcellation Survey," *BioMedIA*. .

[36] A. Casamitjana *et al.*, "Outline of clinical recruitment protocol of subjects with PreAD pathophysiology." 2018.

[37] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: moving beyond fingerprints," *J. Comput. Aided Mol. Des.*, vol. 30, no. 8, pp. 595–608, Aug. 2016.