Imperial College London

Department of Computing

Curriculum Learning and Alzheimer's Disease Classification with Graph Neural Networks

Author:
Wulfstan Bain

Supervisor:
Dr. Ben Glocker

Second Marker:
Dr. Wenjia Bai

Submitted in partial fulfillment of the requirements for the MSc degree in Computing Science of Imperial College London

September 2021

Abstract

Graphs provide a flexible way to structure data, representing individual elements (nodes) and the relations between them (edges). While current mainstream machine learning techniques operate on Euclidean domains, the emerging field of Geometric Deep Learning aims to extend them to more general (non-Euclidean) domains, such as graphs.

In this paper we provide two contributions related to graph-based learning. In both contributions, we use subcortical structures represented as meshes, extracted from brain MRI scans.

First, we propose Curriculum By OverSmoothing, a broadly applicable method to improve graph-based learning without additional parameters. Inspired by Curriculum by Smoothing (CBS), a Curriculum Learning method where images are smoothed to reduce noise in early training stages, we propose a novel adaptation of CBS for graph-based tasks. Our evaluation shows mixed results, with classification accuracy improvements on the Cora & Amazon datasets, but no clear advantage for datsets of MNIST Superpixels or subcortical meshes extracted from the UK BioBank. We posit that the benefits of CBOS are confined to graphs encoding strong homophily.

Second, we investigate the ability of Graph Neural Networks (GNNs) to classify Alzheimer's Disease, using subcortical meshes extracted from the OASIS-3 dataset. We propose a GNN architecture capable of taking multiple substructure graphs as input, before comparing our models to a baseline that uses Principal Component Analysis to provide input to a linear classifier. One variant of our GNN significantly outperforms the baseline across a variety of training settings. Moreover, we place our model within the current state of the art, demonstrating that our architecture is competitive despite our small model size and use of simple features.

Acknowledgments

When reading other dissertations in the past, I was always slightly amused by the outpour of thanks to everyone and their cat[1]. However, having now endured the ordeal myself, I finally understand: it is a rare prompt to consider all the people that have put one in a position to pursue a project such as this. I take this opportunity to thank but a few.

I owe a debt of gratitude to my supervisor, Dr Ben Glocker. For your endless support, ideas, and pushing me beyond the topic originally proposed to go into the world of GNNs, I am grateful. Thanks also to Dr Wenjia Bai, not only for supporting this project, but for sparking my interest in Computer Vision & CNNs in the first place. Finally, to Nairouz: painfully decrypting PyTorch errors was better together.

To my friends, for enduring the many mood swings that accompanied my obtaining a screen tan, and encouraging me to do the MSc in the first place. Football was always an escape. Special thanks to those on the MSc, for suffering through COVID learning together[2].

To my family, for nurturing my intellectual curiosity & putting up with my habit of asking *"Why?"*. Special thanks to "Big P" in relation to this project, for your endless time explaining fascinating neurology tangents, totally unrelated to my questions.

To Grace, for saying "Oh really, that's fascinating!" regardless of what I'm talking about (and regardless of whether you're actually listening. . .).

Finally, to K - I learnt to do more than calculate the area of a circle.

Thank you.

[1]Google "F.D.C. Willard" for a wonderful exception. To Bunky, Consty, Hula, Lani & La Tortue: thank you.

[2]I'm especially proud of Book Club reaching a maximum attendance of five (5)! & I smile whenever I take Flappy C out of its (rightfully deep) hiding place.

Contents

1 Introduction 1
1.1 Motivation . . . 1
1.2 Contributions . . . 1
1.3 Report outline . . . 2

2 Background 3
2.1 Machine Learning settings . . . 3
2.2 Deep Learning & artificial Neural Networks . . . 3
2.2.1 The building blocks of Neural Networks . . . 4
2.2.2 Common Neural Network architectures . . . 5
2.3 Graphs & graphical representation of data . . . 9
2.3.1 What is a graph? . . . 9
2.3.2 Traditional graph description . . . 10
2.3.3 Graphs as a representation of relations . . . 11
2.4 Graph-based learning . . . 12
2.4.1 Types of task . . . 12
2.4.2 Graph learning as an extension of traditional ML . . . 13
2.4.3 Theory behind Graph Neural Networks . . . 14
2.4.4 Existing work with GNNs . . . 17
2.4.5 Benefits of learning on graphs . . . 19
2.5 Medical imaging and graph extraction . . . 21
2.5.1 MRI basics . . . 21
2.5.2 Methods for extracting graphs . . . 22
2.5.3 The hierarchy of medical graphs . . . 24

3 Data & Models 26
3.1 Data . . . 26
3.1.1 The UK Biobank dataset . . . 26
3.1.2 The OASIS-3 dataset . . . 27
3.2 Models . . . 28
3.2.1 GNN design . . . 28
3.2.2 Generalisation to multiple substructures . . . 28
3.3 Data transformations . . . 30

4 Contribution 1
Curriculum by Oversmoothing 31
4.1 Introduction to Curriculum Learning . . . 31
4.2 Curriculum by Smoothing . . . 32

4.3 Contribution:
Curriculum By OverSmoothing 32
4.4 Results 33
4.5 Discussion & future work 35

5 Contribution 2
Alzheimer's Disease Classification 37
5.1 Alzheimer's Disease (AD) 37
5.2 Related work on AD classification 37
5.3 Contribution:
AD classification using GNNs 38
5.3.1 Dataset preparation 38
5.3.2 Model design and baseline 39
5.4 Results 39
5.4.1 Metrics reported 39
5.4.2 Comparison of models 40
5.4.3 The impact of decision threshold 45
5.5 Discussion & future work 45
5.6 Pre-training models 48
5.6.1 Benefits of pre-training 48
5.6.2 Pre-trained models: frozen and unfrozen 49
5.6.3 Results & discussion 49

6 Conclusion 52
6.1 Ethical considerations for graph-based learning 52
6.1.1 Existing issues 52
6.1.2 Potential exacerbation by graphs 53

Appendices 67

Chapter 1

Introduction

1.1 Motivation

Graphs provide a flexible way to structure data, representing individual elements (as nodes) and the relations between them (as edges). While current mainstream machine learning (ML) techniques operate on Euclidean domains, such as in image classification, the emerging field of Geometric Deep Learning [32] aims to extend them to more general (non-Euclidean) domains, such as graphs. Graph Neural Networks (GNNs) are one such graph-based learning model, using graph convolutional layers, linear layers, and non-linear activation functions in an analogous way to Convolutional Neural Networks (CNNs).

Our first motivation relates to the general improvement of GNNs (and graph-based learning more generally). One criticism of Neural Networks is their growing size, as seen by the 175bn parameters of GPT-3 [11], with the resulting time [10], cost [11] and environmental impact [144] of training. Therefore, the improvement of models *without* adding parameters is a worthy cause, and Curriculum Learning [27] offers an opportunity for this. Curriculum Learning alters the training 'curriculum' for a model by providing only 'easier' examples in the early stages of training, before gradually providing the harder examples until all examples are used, with the aim of producing an improved model.

Our second motivation is real-world disease classification. Alzheimer's Disease is a neurodegenrative disease that accounts for 60-80% of all dementia cases in the U.S. [105]. Alongside the personal toll that this disease can take, there is a large economic cost, with total U.S. spending on dementia care expected to reach $1.1trn in 2050 [8]. As there is no cure, early diagnosis to prevent disease progression is key, and ML can play a role in this.

1.2 Contributions

Inspired by these motivations, in this paper we present two contributions.

First, we propose Curriculum by OverSmoothing, a broadly applicable method to improve graph-based learning. One proposed Curriculum Learning method for images is Curriculum by Smoothing [140] where images are filtered/smoothed to reduce noise in early stages, with this smoothing being reduced over the course of training to provide 'harder' examples. Inspired by this work, we propose a novel adaptation that aims to improve classification accuracy on graph-based tasks using oversmoothing. Our evaluation of the technique shows mixed results, with accuracy improvements on the Cora citation [136] and Amazon related-products [2] datasets of 4.3 and 1.8 percentage points respectively, but no clear advantage for sex classification on meshes representing brain substructures (extracted from UK BioBank [147] brain MRI scans) or the MNIST Superpixels [6] dataset. We posit that the benefits of our approach are confined to graphs encoding strong homophily.

Second, we investigate the ability of GNNs to classify Alzheimer's Disease, using meshes representing brain substructures (extracted from brain MRI scans in the OASIS-3 dataset [91]). For this, we propose a GNN architecture capable of taking multiple substructures as input, and test two variants that use different types of convolutional layer. We then compare our models to a baseline that uses Principal Component Analysis (PCA) to provide input to a linear classifier. One of our proposed variants significantly outperforms the baseline, achieving ROC-AUC of 0.88 when using *only* the left and right hippocampi as input. Moreover, we place our model within the current state of the art, demonstrating that our architecture is competitive despite our small model size and use of simple features.

1.3 Report outline

The report is organised follows:

- **Chapter 2** provides the reader with the relevant background to our contributions. We start with a gentle introduction to the current mainstream in ML, before refreshing the reader on graph theory. We then review the graph-based learning literature, including GNNs. Finally, we give an introduction to medical imaging and the extraction of graphs from these images.

- **Chapter 3** details two medical imaging datasets used throughout this report, and the processes used to extract the graphs (meshes) of subcortical brain structures on which we perform our work. We then outline the models we have designed to operate on these meshes.

- **Chapter 4** provides our first contribution, giving an introduction to Curriculum Learning and Curriculum By Smoothing, before detailing our proposed Curriculum By Over-Smoothing method. We present the results obtained on four datasets, alongside assessment of the limitations of our proposed technique and suggestions for further work.

- **Chapter 5** provides our second contribution, giving background on Alzheimer's Disease and the current state of the art in its classification. We then present the results of our proposed GNN models on a Alzheimer's Disease vs cognitively normal binary classification task, and compare them to both our own baseline (a linear model using PCA) and existing work in the field. After discussing the limitations of our approach and suggested further work, we explore one extension in the form pre-training models.

- **Chapter 6** concludes this report, considering ethical issues that are particularly relevant to graph-based learning.

Chapter 2

Background

While terms such as ML and CNN are by now ubiquitous, they provide important context for the recent rise in ‘graph representation learning’ [67]. After refreshing the reader on various ML settings (Section 2.1), we outline the use of CNNs in image processing, and their advantages over MultiLayer Perceptrons (MLPs) (2.2). We then provide a brief introduction to graph theory (2.3), before introducing graph-based learning (2.4). Finally, we relate this to the medical imaging setting (2.5).

2.1 Machine Learning settings

Defined as early as 1959 [134], ML can be succinctly described as *“the study of computer algorithms that allow computer programs to automatically improve through experience”* [111]. The approach can be split into three settings: supervised, semi-supervised and unsupervised.

- In **supervised** settings, the task is to learn a function that maps an input observation to its true output value. More formally, given a training set of N input-output pairs $\{< x_1; y_1 >, \ldots, < x_N; y_N >\}$, where x_n is an input feature vector for the n^{th} observation and each output y_n was generated by an unknown function $y_n = f(x_n)$, a supervised learning task aims to discover an approximation of the true function f. During training, feedback is generated by comparing the predicted output for a given example with the ground truth output. A common supervised task that arises throughout this report is classification, where each input is assigned a (discrete) label or ‘class’.
- **Semi-supervised** learning is similar to supervised, except that the training set may contain few labelled examples. A common task is to classify the unlabelled examples.
- By contrast, **unsupervised** learning seeks to find patterns in the input, even though no explicit feedback (or labelling) is supplied [133]. A common task in this setting is clustering, which aims to discover natural groupings in the data.

As seen in these descriptions, the availability of labels to gather feedback from, and the purpose of the task, varies in each. This paper will focus on supervised learning settings.

2.2 Deep Learning & artificial Neural Networks

Artificial Neural Networks (NNs) are a class of ML algorithm, inspired (as their name suggests) by the human brain. Here we outline current mainstream NN architectures both as background and as a reference for graph-based approaches.

2.2.1 The building blocks of Neural Networks

Neurons

The basic building block of a NN is the artificial neuron. Each neuron receives a number of inputs and produces a scalar value called its 'activation'. This activation is typically a weighted sum of its inputs plus a bias term (often implemented as an implicit input to the weighted sum). The Rosenblatt perceptron [132], seen as the predecessor of modern artificial neurons, also computes a weighted sum of inputs. However, the activation of a neuron is passed through an *activation function* to determine the value of the output transmitted to subsequent neurons, akin to synapses 'firing' between biological neurons in the brain. It is this activation function that differentiates modern artificial neurons from perceptrons.

Activation functions

Activation functions in NNs are inspired by action potentials in neuroscience. In biological settings, a neuron will only transmit a signal to subsequent neurons if the electrical potential between a neuron's interior and exterior exceeds a threshold. Activation functions seek to provide an analogue of this behaviour.

The Rosenblatt perceptron uses a heaviside step function [52], whereby the value emitted is set to one if the weighted sum of inputs exceeds a threshold, and zero otherwise. In contrast, modern neural networks use non-linear activation functions which are differentiable, to allow end-to-end gradient based optimisation. Common activation functions include Sigmoid, Softmax and ReLU, which are outlined below.

The Sigmoid function has a characteristic 'S' shape, with output ranging from zero to one for a scalar input. Due to this bounded nature, the Sigmoid function is often used as the output for binary classification tasks.

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

The Softmax is a generalisation of the Sigmoid function to multiple inputs. The Softmax function takes an N-dimensional vector of real-valued inputs and outputs an N-dimensional vector with elements between zero and one, and a sum equal to one. These outputs are often used as probabilities: for example, in a multi-class classification setting N would represent the number of classes, and each element of the vector represents the probability that the input data belongs to a given class.

$$Softmax(\boldsymbol{x})_i = \frac{e^{x_i}}{\Sigma_{j=1}^{N} e^{x_j}} \text{ for i = 1,...,N} \tag{2.2}$$

The ReLU (rectified linear unit) has become increasingly popular as a general purpose activation function [29] due to its simplicity, which leads to faster training times in deep NNs [86]. The ReLU function introduces non-linearity by returning the value of the input if the input is positive, and zero otherwise.

$$ReLU(x) = max(0, x) \tag{2.3}$$

There are alternative formulations of ReLU which are growing in popularity due to the dead ReLU problem, whereby the unit always outputs zero for any input [102]. These formulations typically involve an adjustment to the case where the input is negative, outputting a function of the input rather than zero. For example, the Leaky ReLU [103] outputs a small fraction α of a negative input.

$$LeakyReLU = \begin{cases} x, & \text{if } x < 0 \\ \alpha x, & \text{otherwise} \end{cases} \tag{2.4}$$

2.2.2 Common Neural Network architectures

Having understood the building blocks of NN, we now examine two common forms of NN in today's ML landscape, the MLP and the CNN.

Multi Layer Perceptron

The MLP is a NN consisting of multiple 'layers' of neurons, with neurons in a given layer taking input from the previous layer and transmitting their output to the subsequent layer. Each layer will consist of neurons, and an activation function for their output. Modern 'deep learning' networks contain numerous 'hidden' layers, which take input from previous layers and output to subsequent layers rather than taking the input or producing the output of the overall model (input and output layers respectively). We can view deeper layers as extracting higher-level information from the input. The early layers in the network extract simple or low-level features, but these are combined by later layers allowing more abstract representations of the input information.

Neurons are typically connected to neurons in the previous layer (to receive input), and neurons in the subsequent layer (which receive their output as input). The most basic form of this is a fully connected layer, in which all neurons in layer L are connected to all neurons in layers L-1 and L+1. A network in which all layers are fully connected is called a fully connected network.

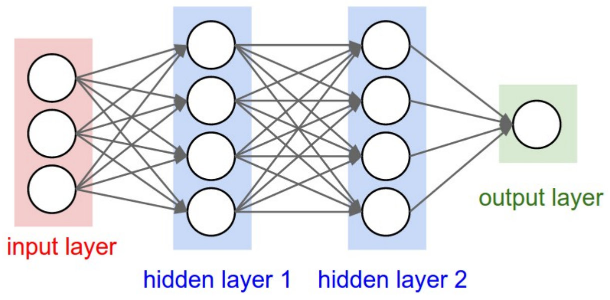


Figure 2.1: A fully connected MLP with two hidden layers and a single output neuron

MLPs, non-linearity and universal approximation

The non-linearity of the activation functions given above, and used in the MLP, is no coincidence. If *all* the layers in an MLP were linear, the network itself would also act in a linear manner, effectively collapsing to a single linear layer. Once non-linearity is introduced, and we combine a sufficient number of neurons into a neural network, we are able to approximate any continuous function. While this universal approximation result was visible in research as early as 1943 [108], more recent results have demonstrated that any continuous function can be accurately approximated by a shallow NN with a single-hidden-layer [47] [71] and for a broad class of non-linear activation functions [94].

This is a powerful result given that, as discussed in Section 2.1, the ML setting can be viewed as learning to approximate a hidden function. However, it is worth noting that the number of neurons required in this hidden layer may be infeasibly large (although bounds exist for a broad class of functions [23]) and that the function learnt may not generalise (i.e. the neural network overfits: the function learnt may be specific to that training set) [65]. Therefore it is common to include some form of regularisation to attempt to avoid functions that overfit, a point to which we will return in Section 2.4.5.

Convolutional Neural Network

While the MLP is a powerful model, it is limited by the fact it takes in a vector of inputs. This leads to a number of problems when attempting to learn on images.

- First, images include a large number of pixels. Applying a fully connected first layer to these inputs would lead to a large number of parameters even for small images [93].
- Second, MLPs have no in-built invariance for translations or local distortions. However, we would wish a picture of a cat, for example, to be classified as a cat regardless of where in the image it appears.
- Finally MLPs ignore the topology of the image. By flattening the image we lose information on the strong two dimensional local structure whereby pixels that are spatially close to each other are highly correlated.

CNN's avoid these issues by implementing a convolutional layer [61] [92].

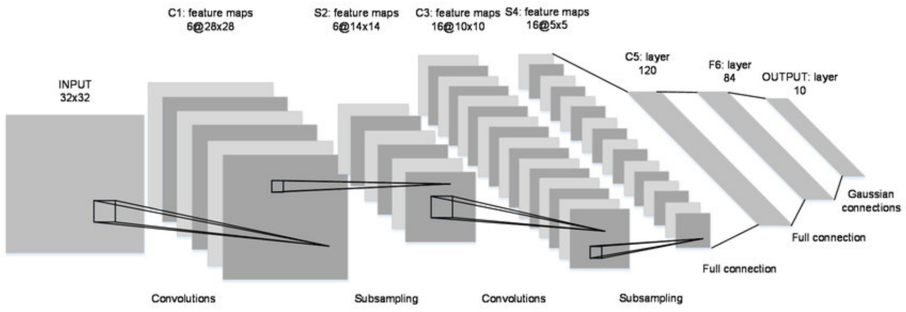


Figure 2.2: The LeNet-5 Architecture, an archetypal CNN

The parameters of a convolutional layer are formed by learnable filters that are convolved with the input to produce output that is passed to the next layer. The input to the convolutional layer is a tensor with a height, width and number of channels (or depth). For example, the input layer would likely take the height and width of an input image, with the number of colour channels forming the third dimension. Each filter is a small spatial kernel that partially covers the height and width of the input, but extends through the full depth. Neurons in a convolutional layer are organized into planes within which all the neurons share the same set of weights. The outputs of the neurons in such a plane is called a feature map.

The process of convolution can be seen as sliding a filter over the width and height of the input, with each output being connected to a local area of input. With local 'receptive fields', neurons can extract basic visual features such as oriented edges, endpoints or corners. While these features are combined into higher order features in later layers, even these basic features are powerful, as shown by oriented edges playing a key role in how animals view the world and react to stimuli [73].

More formally, convolution is a classical signal processing operation, where the rows and columns (m and n respectively) of the output matrix g are given by convolving a signal f with the kernel h. For a two dimensional signal, (discrete) convolution is given by:

$$g[m,n] = f[m,n] * h[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i,j]h[m-i,n-j] \tag{2.5}$$

It is worth noting that in practice convolution may be implemented using cross-correlation, as cross-correlation & convolution are essentially similar, but with the kernel being 'flipped' before the sliding window is applied in convolution. Given that the kernel weights are randomly initialised and are learnt in an ML setting, flipping the kernel has little effect on the output of the convolution [98].

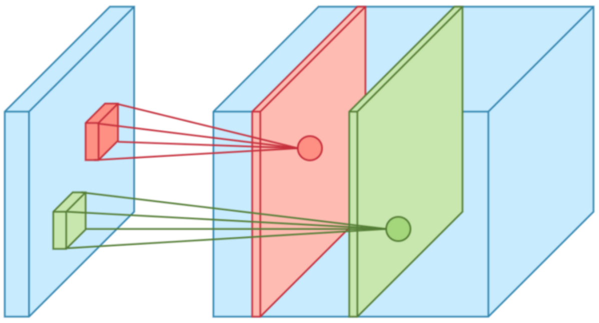


Figure 2.3: Visualisation of a convolutional layer: each output in a feature map obtains input from a small area of input

The convolutional layer therefore has a number of interesting properties that help counteract the aforementioned issues MLPs face when processing images:

- **Weight sharing:** The kernel of a convolutional layer is slid over every position in the input, with the same set of parameters, dramatically reducing the total number of parameters. Moreover, we can view a particular set of parameters as extracting a type of feature from an input (neurons in a feature map are all constrained to perform the same operation on different parts of the input). As stated earlier, although shifting an input image can change the position of salient features, these features will still be important: certain features are useful regardless of their position in the image. Due to the convolution operation using shared weights, if the input image is shifted the feature map output will be shifted by the same amount, but it will be left unchanged otherwise [93] (i.e. it is *equivarient*, a point to which we will return later). This parameter sharing means that the convolutional layer is equivariant to translation and the number of parameters is independent of the input size, counteracting the first two issues with MLPs highlighted above.

- **Local information:** As seen in Figure 2.3 each neuron only receives information from a small, local group of inputs in the image, in contrast to a fully connected layer which would receive input from every part. This allows CNNs to retain local information, counteracting the third issue highlighted above.

Many CNNs also employ *pooling layers* on top of their convolutional layers. A pooling layer uses an aggregation operation to produce a summary statistic of local input. Common pooling methods include *max-pooling* and *average-pooling* where the maximum and average values of the input window are taken respectively. These methods help achieve *invariance* to image transformations, more compact representations, and better robustness to noise [31]. Pooling methods do this because, once we know that a feature is present, the precise position of the feature is likely irrelevant for identifying a pattern (in fact, it is potentially harmful because the exact position is likely to vary for different images of the object [93]). Reducing the spatial resolution of the feature map via pooling reduces the sensitivity of the output to shifts and distortions, helping the CNN become theoretically invariant to image transformations.

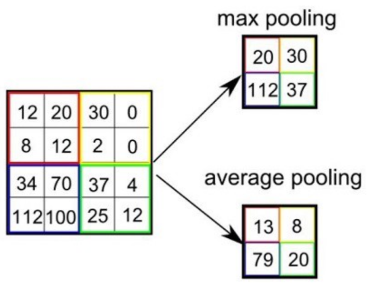


Figure 2.4: Max and average pooling operations

Combining all of this together, CNNs include both convolutional and pooling layers, and may integrate a number of fully-connected final layers for classification tasks (Figure 2.2). Due to the advantages explained above, CNNs are ideally suited to image classification tasks, and were shown to obtain half the error rate of MLPs in a character recognition task [53].

End-to-end learning

Prior to the advent of deep, 'end-to-end' learning, ML algorithms often relied on carefully constructed features as input. This manual feature-engineering required domain expertise, and time, to craft features that were task specific, leading to poor generalisation and transferability across domains [65]. This was, in part, due to the cost of data and computing resources in that era. Therefore, adding additional structure to input features was seen as a way to reduce complexity, and push algorithms to learn off specific features, thus restricting the set of possible functions that could be learnt [24]. In the domain of images, this gave rise to popular descriptors such as HOG [48], SIFT [101] and SURF [25].

However, modern deep learning methods tend to follow an end-to-end design philosophy that tries to *avoid* explicit structure, manual-engineering of features, or other a-priori representational and computational assumptions [24]. In this paradigm, the input to a model tends to be raw (e.g. pixel intensities for an image) and the process of feature extraction is seen as part of the learning process. Indeed, in image analysis the aforementioned handcrafted descriptors have been abandoned in favour of learning approaches [106]. This has been enabled, as have NNs more generally, by an increasing abundance of data and cheap computing resources [68].

The end-to-end approach is embodied in CNNs, where raw images are fed in and convolutional layers learn to extract task-specific salient features from examples. This has led to large performance improvements in many image related tasks, such as classification [87] and segmentation [42].

To re-emphasise a key point to which we will return: a trend in ML which has led to performance improvement, with CNN's being a prime example, was the removal of a-priori structure in favour of end-to-end learning.

2.3 Graphs & graphical representation of data

Across the natural and social sciences, graphs are used to represent data. Their flexibility, encoding information about objects and the relationships between those objects, makes them useful for representing information in diverse settings, from social networks to organic molecules. The aim of this section is to refresh the reader on graph theory, notation, and the graphic-representation of data.

2.3.1 What is a graph?

A graph G = (V, E) is defined by a set of nodes $V = \{1, \ldots, N\}$ and edges E between those nodes such that E is a subset of the cartesian product of the nodes (that is, $E \subseteq V \times V$). Throughout this section we use a simple example of a social network to give demonstrable examples of graph theory. In our example, a node might represent a person, whilst an edge might represent the relationship between two people. Even from this simple example a number of interesting characteristics of graphs emerge:

Edge direction: An edge going from a node $u \in V$ to node $v \in V$ is denoted as $(u, v) \in E$. We already see the expressive power of graphs, as we can model complex relationships. For example, a bi-directional relationship (e.g. friendship on a social media site) can be modelled as an *undirected* edge, where the edge exists from u to v and vice versa. An undirected edge therefore satisfies $(u, v) \in E \iff (v, u) \in E$. However, in a different setting we might model unrequited love as a *directed* edge, with an edge only existing from u to v (but, sadly, not vice versa).

Edge Type: Beyond the distinction between directed and undirected edges, we might have different edge **types**. For example, in our social model above we have already described an edge type for friendship and one for love. In the setting where multiple types of edge may exist, edge notation can be extended to include the relation type τ, i.e. $(u, \tau, v) \in E$.

A **simple graph** is one where there is at most one edge between each pair of nodes, no edges between a node and itself, and where all edges are undirected. In contrast, a **multi-relational graph** can have different types of edges.

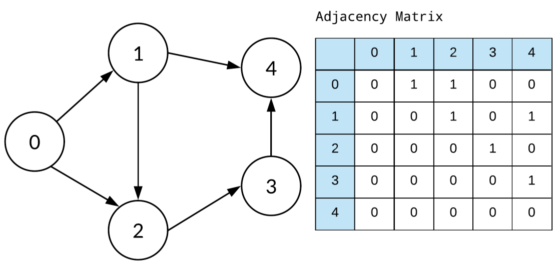
Node Type: We may also have **heterogenous** graphs, a form of multi-relational graph, where nodes are imbued with types. These types allow us to partition nodes into disjoint sets such that $V = V_1 \cup V_2 \cdots \cup V_k$ where $V_i \cap V_j = \emptyset, \forall (i \neq j)$. For example, in our example we might have nodes representing people and nodes repsenting photos posted by those people, with edges between the different types of node representing different things. Indeed, in most heterogenous graphs edges must satisfy constraints restricting which types of edges may connect which types of nodes. *Multipartite* graphs are a well-known case, where edges can only connect nodes that have different types.

Representation in the adjacency matrix

We can represent graphs through an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{|V| \times |V|}$, in which each node indexes one particular row and column. We represent edges as entries in the matrix, such that $\boldsymbol{A}[u, v] = 1$ if $(u, v) \in E$ and $\boldsymbol{A}[u, v] = 0$ otherwise. In the case of a *simple graph*, $\boldsymbol{A}$ will be a symmetric matrix. The adjacency may also encode information about the edges. For example, some graphs might have *weighted* edges, and the elements in A are then real values. For example, in our social network, this could encode the strength of friendship.

In the case of a multi-relational graph, each edge type has its own adjacency matrix. Therefore, we extend the graph representation to an *adjacency tensor* $\boldsymbol{A} \in \mathbb{R}^{|V| \times |V| \times |R|}$ where R is the set of relation types.

One key point regarding the structural properties of graphs is that the nodes in V are usually not assumed to be provided in any particular order (as best seen by the case of a graph with no edges, a **set**). However, by representing the graph in $\boldsymbol{A}$ we imbue an ordering.



	0	1	2	3	4
0	0	1	1	0	0
1	0	0	1	0	1
2	0	0	0	1	0
3	0	0	0	0	1
4	0	0	0	0	0

Figure 2.5: A directed, unweighted graph and its corresponding adjacency matrix

Node, edge and graph features

Our graph may also have feature information, which describe its constituent nodes and edges.

Most often [67] this is done by node-level attributes, whereby each node has an m-dimensional real-valued feature vector $\boldsymbol{x} \in \mathbb{R}^m$, which we represent in a feature matrix $\boldsymbol{X} \in \mathbb{R}^{|V| \times m}$. In our running social network example, these features might include a person's age or sex. Note that the ordering of nodes in $\boldsymbol{X}$ is assumed to be the same as in $\boldsymbol{A}$, which leads to convenient mathematics (see Section 2.3.2).

We may also have real-valued edge features, in addition to differing edge types. In some cases, we may even use features to label entire graphs for classification tasks.

2.3.2 Traditional graph description

Prior to the introduction of deep learning, graphs (and their constituent components, i.e. nodes) were described by manually extracted features. This section will give a brief overview of some common properties that reappear in the graph-based learning literature.

Node Degree

An obvious node-level feature to compute is the node *degree*, represented as d_u for node $u \in V$, which counts the number of edges incident to a node (we might differentiate in directed graphs between 'inbound' or 'outbound' edges).

$$d_u = \sum_{v \in V} \boldsymbol{A}[u, v] \tag{2.6}$$

Node degree is often one of the most informative features in traditional ML applied to node-classification [67]. Node degrees can be represented in a degree matrix, $\boldsymbol{D}$, which is a diagonal matrix with element $[i, i]$ holding d_i .

Bag of nodes & Weisfeiler-Lehman

Moving to graph-level statistics, the bag of nodes is a simple method that aggregates node-level statistics, for example node degrees. This information, often represented in the form of histograms, can be used to describe entire graphs. However, this only utilises local node-level information, and thus might miss important global properties.

An improvement is to perform iterative neighbourhood aggregation, with the Weisfeiler-Lehman (WL) algorithm [157] being a well-known example. The algorithm has three key steps. First, we assign an initial label to each node (e.g. its degree). Next, we iteratively assign each node a new label, computed as the hash of the multiset of labels in the node's neighbourhood. Having run K iterations, the label of each node contains information from its K-hop neighbourhood, and we can compute histograms or other graph summary statistics over these labels.

Graph Laplacians

Graphs can be represented as adjacency matrices ($\boldsymbol{A}$) without any loss of information, however these may not always be the most mathematically convenient representations and so more convenient matrices known as graph *Laplacians* are formed by transformations of $\boldsymbol{A}$.

The most simple of these is the unnormalised Laplacian:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A} \tag{2.7}$$

The graph Laplacian is used extensively in Spectral GNN methods, and so we defer their discussion until we cover these methods in Section 2.4.4.

Calculating neighbourhood information

So far we are yet to understand why these matrix representations might be useful, so here we include an illustrative example. Let us say we want to give each node an attribute that is the sum of the degrees of its neighbouring nodes. Furthermore, let us define a simple setup with three nodes in an undirected graph, where the first node is connected to the other two, and there are no other connections. Finally, for simplicity we will represent the degree matrix as a vector $\overrightarrow{\boldsymbol{d}}$, which contains the information from the diagonal of $\boldsymbol{D}$.

$$\boldsymbol{A}\overrightarrow{\boldsymbol{d}} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} \tag{2.8}$$

As we can see, this simple calculation demonstrates the convenience of the adjacency matrix representation for calculating local neighbourhood features. This is because the zeros in $\boldsymbol{A}$ nullify values from unconnected nodes, while connected nodes have their information aggregated. In reality, instead of $\overrightarrow{\boldsymbol{d}}$, we might use a more complex feature matrix $\boldsymbol{X}$ to obtain more complex information.

2.3.3 Graphs as a representation of relations

As shown throughout this brief introduction, graphs are ideal for representing structured knowledge and representing relationships as they can support arbitrary (pairwise) relational structure within data [43]. Indeed, the act of constructing a graph imposes relationships onto the components (the vertices): edges connecting vertices suggest a relationship between those vertices [24]. Of course, the nature of the relationship need not be pre-defined. For example, a weighted version of our social network graph could encode either the strength of friendship or of dislike; two identical graphs but with inverted weights for the different situations. Therefore, graphs include some aspect of manual feature engineering, whether that is deciding which edges to include, or the interpretation of those edges (e.g. similarity vs dissimilarity).

This additional structure provided by relational biases, and the a-priori nature of assumptions regarding what the relationships (edges) represent, is a first hint at how learning algorithms on graphs may differ from traditional settings.

2.4 Graph-based learning

Having outlined the ML context and an introduction to graphs, we next explore graph-based learning and its differences & similarities to traditional ML. The common theme across graph-based approaches is computation over discrete entities *and* the relations between them [43].

2.4.1 Types of task

As outlined in Section 2.3 we can view a graph as a set of nodes and edges. Here we outline common tasks within graph-based learning that can be applied to these components.

Node classification

In node classification, the goal is to predict the node label y_u associated with all nodes $u \in V$, when we are only given the true labels on a subset of these nodes. This is therefore seen as a (semi-)supervised learning setting [160]. For example, this might be classifying documents (nodes) into classes of 'topic' in a citation network (edges encoding citations) [80].

It is worth noting that although this appears to be the analogue of traditional ML classification, there are a number of important differences. A standard assumption in traditional ML is that observations are *independently and identically distributed* (i.i.d) to avoid the need to model interdependencies. However, in graphs this is clearly not the case. In fact, most graph-based learning models leverage the connections between nodes [67]. For example, the model might try to exploit the tendency for nodes to be similar to their neighbours (*homophily*) [109]. Using homophily, models might try to assign similar labels to neighbouring nodes on a graph [162]. Alternatively, and perhaps less intuitively, we might presume *heterophily*, whereby nodes are preferentially connected to nodes with different labels (a good example of this is gender in heteronormative romantic social networks). Finally, we might assume that nodes in different parts of a graph with similar neighbourhood structures might have similar roles within their local topology, and therefore assign similar labels based on this *structural equivalence* [51]. Regardless of the details, all of these techniques leverage the interdependencies of nodes, which diverges from classic ML settings.

Relation Prediction

While node classification uses information about a node and its relationships to infer information about a node, relation prediction[1] can be used to fill in missing edges (hence why it

[1]a.k.a Edge or Link Prediction

is sometimes called graph completion [153]). Alongside node classification this is a popular task for graph-based ML, and has been used on tasks as diverse as predicting drug side-effects [163] and content recommendation for Pinterest [161].

Graph classification and regression

The final group of tasks work over entire graphs, and are the most analogous to classic ML. Each graph is assumed to be an i.i.d observation with an associated label and, as in supervised learning outlined in Section 2.1, the task is to learn the mapping from the graphs to their labels. Similarly, graph clustering is the extension of unsupervised learning to graphs.

A good example of this type of task is the prediction of the attributes of a particular molecule, for example its toxicity or solubility [64]. The molecule is represented as a graph, and it is an attribute of this entire graph that must be predicted.

2.4.2 Graph learning as an extension of traditional ML

Examining the types of task in Section 2.4.1, we can see similarities and differences from the traditional ML setting. This section explains that graph-based learning can be seen as adjusting traditional ML operations for a different domain, with different geometric priors.

Geometric deep learning

By the late 19th century, multiple types of geometry had emerged, from traditional Euclidean geometry to Projective geometry. These geometries were studied as their own independent fields [34]. However, a young professor of mathematics named Felix Klein proposed that these distinct geometries could be unified by viewing geometry as the study of invariants, or structures, that are preserved under certain types of transformations. The set of transformations that preserve these properties form a *symmetry group*.

This concept is best viewed by example. Euclidean geometry concerns lengths and angles, as these are the properties preserved by the group of Euclidean transformations (rotations and translations). In contrast, Affine geometry studies parallelism, and as such the set of Affine transformations preserve collinearity and ratios of distances. Viewed in this way, geometries are united: for example, it is clear that Euclidean transformations are a subgroup of the Affine group.

Geometric deep learning aims to apply this same logic to the field of deep learning, viewing different types of data as domains with their own invariants that must be respected, and so their own symmetry groups [32]. For example, in Section 2.2.2 we demonstrated that in the image domain we wished a classifier to act in a translation invariant manner, and so convolutional units are used. Given that images are 2D grids, it makes sense that CNNs operate on the Euclidean domain.

Geometric deep learning can be seen as the search for default deep learning 'components', that may be part of any learning algorithm operator on an arbitrary structure [24] having been adjusted to respect the invariants of that structure. As such, the field looks to generalise the deep neural model from the Euclidean domain to non-Euclidean domains.

This begs the question of what invariances a graph-based learner should respect, given graphs are a common example of non-Euclidean structures.

Graph learning by combining permutation invariant and equivariant functions

As described in Section 2.3.1, graphs are given an artificial ordering when we represent them, for example in an adjacency matrix $\boldsymbol{A}$. Given this, a natural invariant we wish to respect is that of permutation invariance. For example, we would wish an algorithm classifying graphs to give the same result for two isomorphic graphs.

To demonstrate these concepts we will first examine **sets**, which are graphs with no edges, so $G = V$. Once again we represent node features in a feature matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)^T$, where row i of $\boldsymbol{X}$ corresponds to the feature vector $\boldsymbol{x}_i$ of node i.

Permutation invariance requires that functions $f()$ over $\boldsymbol{X}$ should give the same output regardless of the permutation of nodes in $\boldsymbol{X}$. More formally, if we were to apply a permutation matrix $\boldsymbol{P}$ to $\boldsymbol{X}$, the output should be the same.

$$f(\boldsymbol{X}) = f(\boldsymbol{P}\boldsymbol{X}) \tag{2.9}$$

This set of functions tends to correspond to *aggregations*. For example, the functions *sum*, *max* and *average* all give the same output regardless of the node ordering of inputs.

In contrast, **permutation equivariance** requires that the set of functions we use allows us to identify what part of the output belongs to which input node (that is, the function does not change node ordering or changes it in a consistent manner). With this set of functions it therefore does not matter if we permute the nodes before or after the function is applied.

$$\boldsymbol{P}f(\boldsymbol{X}) = f(\boldsymbol{P}\boldsymbol{X}) \tag{2.10}$$

Whilst these two sets of functions are used together as shown below, we can also see that they produce information on different levels. Permutation equivariant functions produce output that is attributable to specific nodes, and is therefore suited to node level classification. In contrast, permutation invariant functions tend to aggregate information, losing node specific information but giving information at the level of the entire node set (e.g. for graph classification).

Generalising this to graphs, we must ensure that the rows and columns of the adjacency matrix $\boldsymbol{A}$ are also permuted when we apply node permutations (which amounts to applying $\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^T$). We therefore arrive at the analogous formulas for permutation invariance and equivariance of functions:

$$\text{Invariance: } f(\boldsymbol{P}\boldsymbol{X}, \boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^T) = f(\boldsymbol{X}, \boldsymbol{A}) \tag{2.11}$$

$$\text{Equivariance: } f(\boldsymbol{P}\boldsymbol{X}, \boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^T) = \boldsymbol{P}f(\boldsymbol{X}, \boldsymbol{A}) \tag{2.12}$$

Geometric deep learning therefore has a 'blueprint' [32] for learning arbitrary functions: (stacking) *equivariant* function(s), potentially with an *invariant* aggregation.

2.4.3 Theory behind Graph Neural Networks

Graph Neural Networks (GNNs) are the realisation of this Geometric Deep Learning blueprint on graphs [32], by utilising the symmetry group of graphs (i.e. the permutation group). As before, we represent a graph G with an adjacency matrix $\boldsymbol{A}$ and node feature matrix $\boldsymbol{X}$. GNN architectures are permutation equivarient functions $F(\boldsymbol{X}, \boldsymbol{A})$ that are constructed by applying shared permutation invariant function $\phi(x_u, \boldsymbol{X}_{N_u})$ over local neighbourhoods.

Unlike sets, graphs can have the notion of locality, taking into account the nodes that share edges. We can define the 1-hop neighbourhood of a node i as its adjacent nodes, that is

$$\boldsymbol{N}_i = \{j : (i,j) \in E \text{ or } (j,i) \in E\} \tag{2.13}$$

and this notion can continue to an arbitrary k hop neighbourhood, containing nodes that are at most k edges away. This notion of locality therefore gives us a multi-set of features of the neighbourhood (a set of feature vectors for the nodes in the neighbourhood)

$$\boldsymbol{X}_{N_i} = \{x_j : j \in N_i\} \tag{2.14}$$

Graph neural networks leverage this notion of locality in the function ϕ that acts as a shared local permutation-invariant function, that is applied to every node and its neighbourhood in isolation, as described above.

The defining feature of a GNN is that it uses a form of neural *message passing* in which vector messages are exchanged between nodes within a neighbourhood, leveraging this notion of locality, and node features are then updated using NNs [64]. It is our local shared function ϕ that performs this message passing (also known as diffusion or propagation).

We now have the key components to form a general equation for a GNN layer. Let h_i be the updated (output) feature vector, and x_i be the input feature vector, for node i (note the input feature vector might be an embedding from a previous GNN layer). Our GNN layer will work in two stages.

1. *Aggregating* features from node i's neighbourhood $\boldsymbol{N_i}$ (potentially transformed by a function ψ) with a permutaiton invariant aggregator $\bigoplus$ (such as sum, average or max).

2. *Updating* the features of node i with some function ϕ, potentially taking its own input features as well as the aggregated neighbourhood information.

Combining these two steps, we get a general equation for a GNN node update, which we reproduce from [155]. Typically, ψ and ϕ are learnable, while $\bigoplus$ is a nonparametric operation, though it can also be constructed using recurrent NNs [116].

$$h_i = \phi\left(x_i, \bigoplus_{j \in N_i} g(x_i, x_j)\psi(x_j)\right) \tag{2.15}$$

This leaves the function $g(x_i, x_j)$, which takes the input features of nodes i and j and whose ouptut acts as a weight in the aggregation step, which is a key variation across GNN types.

Veličković [155] describes GNNs as falling into three 'flavours' that share the blueprint just explained, with the variation appearing in how messages are formulated in ϕ. More specifically, each variation alters the action of $g(x_i, x_j)$ in our general formulation above.

Convolutional GNNs

Convolutional GNNs use constant values for g. This can be seen as an interaction constant c_{ij} which determines how much node i values the features from node j.

$$h_i = \phi\left(x_i, \bigoplus_{j \in N_i} c_{ij}\psi(x_j)\right) \tag{2.16}$$

Let us assume that ψ is the identity function for simplicity, x_j to be j's input feature vector, and $\bigoplus$ to be average. With these assumptions, the node embedding for i will be some permutation invariant function applied to i's current feature vector (x_i) and the weighted average of its neighbours' features.

Convolutional GNNs are especially useful for graphs that are *homophilous* (where edges encode similarity) and, due to their simplicity, they tend to be the most scalable flavour. These formulations of GNNs are known as convolutional as the weighting parameters are typically shared over all locations in the graph, or a subset thereof [54].

Attentional GNNs

Attentional GNNs replace fixed weights with learnable attention weights a_{ij}. The weight is computed as $a(x_i, x_j)$, where $a()$ is a transformer function that takes node features of a *sender* and *receiver* node, returning a coefficient which the receiver uses to weight any contribution by the sender.

$$h_i = \phi\left(x_i, \bigoplus_{j \in N_i} a_{ij}\psi(x_j)\right) \tag{2.17}$$

Attentional GNNs are more powerful and general than Convolutional GNNs, without requiring the computation or storage of substantially more information. They therefore form the 'middle ground' [32], having larger capacity than convolutional GNNs but still being somewhat scalable as they compute only a scalar value for each edge. The flexible transformer function means that they can be applied to graphs whose edges need not encode homophily.

Message-passing GNNs

Message-passing GNNs are the most general form of GNN layer. Here, arbitrary vectors form messages that are sent across edges. Messages m_{ij} are computed as a function of sender and receiver features, via a learnable message function $m_{ij} = \psi(x_i, x_j)$. We can therefore view the sender and receiver as collaborating to produce the content of the vector message.

$$h_i = \phi\left(x_i, \bigoplus_{j \in N_i} m_{ij}\right) \tag{2.18}$$

Given that a vector must be computed for every edge in the graph, and graphs typically have more edges than nodes, this type of GNN layer requires a lot more information and so is less scalable.

Combination into GNN layers & classification tasks

The equations described so far return h_i, the updated node features for node i, rather than values for the node set. The GNN layer, defined above as $F(\boldsymbol{X}, \boldsymbol{A})$ applies the function ϕ over all neighbourhoods and simply stacks the resulting updated node features (a.k.a latent features) as rows of a matrix. Given that ϕ must be permutation *invariant* in the above blueprint, the layer $F(\boldsymbol{X}, \boldsymbol{A})$ is permutation *equivariant*

$$F(\boldsymbol{X}, \boldsymbol{A}) = \begin{bmatrix} \phi(x_1, \boldsymbol{X}_{N_1}) \\ \cdots \\ \phi(x_n, \boldsymbol{X}_{N_n}) \end{bmatrix} \tag{2.19}$$

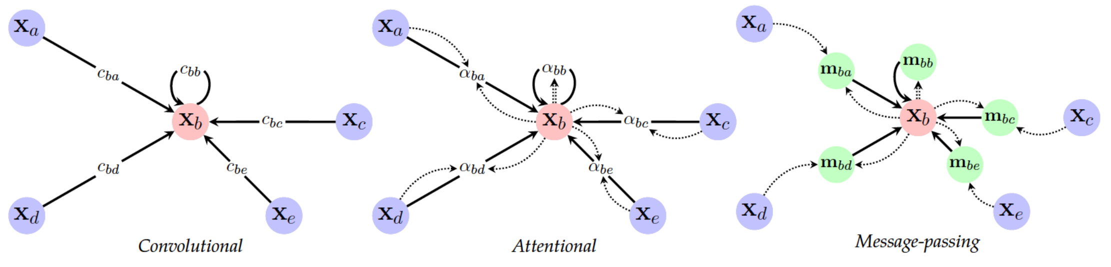


Figure 2.6: Visualisation of the dataflow for the three flavours of GNN layers, reproduced from [32]

Looking at equation 2.15 for a general GNN update step, one may also notice that the function ϕ takes the neighbourhood of a given node, and if this is (for example) a one-hop neighbourhood this ignores longer distance relationships. The relational power can be improved by stacking these GNN layers, with the input to each layer being the output of the previous layer. After the first layer, the node features of i's neighbours will have been updated to include information on their own neighbours. Therefore, after a second iteration node i will be receiving messages containing information about its two-hop neighbourhood. More generally, after k iterations a node will receive information about its k-hop neighbourhood.

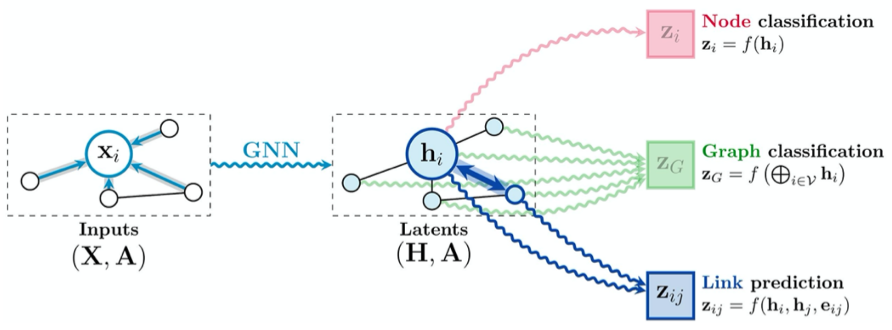

$\mathbf{x}_i$
Inputs
$(\mathbf{X}, \mathbf{A})$
GNN
$\mathbf{h}_i$
Latents
$(\mathbf{H}, \mathbf{A})$
$\mathbf{z}_i$
Node classification
$\mathbf{z}_i = f(\mathbf{h}_i)$
$\mathbf{z}_G$
Graph classification
$\mathbf{z}_G = f\left(\bigoplus_{i \in \mathcal{V}} \mathbf{h}_i\right)$
$\mathbf{z}_{ij}$
Link prediction
$\mathbf{z}_{ij} = f(\mathbf{h}_i, \mathbf{h}_j, \mathbf{e}_{ij})$

Figure 2.7: The use of GNN output for different learning tasks, reproduced from [155]

Having produced the node level features h_i from these convolutional layers, we can now see how various graph-based classification tasks might proceed.

- **Node classification** uses these features directly, for example by applying an MLP to classify a node based on latent features.
- **Graph classification** aggregates this information using a pooling layer, applying classification techniques to this aggregated information.
- **Link (edge) prediction** utilises pairwise combinations of latent features and existing edge information, to predict the existence of an edge between those two points.

2.4.4 Existing work with GNNs

Before the unifying view of geometric deep learning existed, neural networks operating on graphs were typically divided into *spectral* and *spatial* methods. Here we outline these methods, before explaining that they are in fact not dichotomous.

Spectral methods

Spectral methods are based upon spectral graph theory [41], where eigenvalues of the graph's Laplacian matrix are interpreted as frequencies of node signals [139]. A spectral convolution is therefore defined in terms of Fourier transforms of this signal and a filter (see [32] for a full explanation, including derivation using the Convolution Theorem). However, there are two key criticisms of these methods.

First, initial implementations of a Spectral CNN [35] explicitly compute the graph Fourier transform, requiring computationally expensive matrix operations. However, subsequent descendent approaches approximate this computation, for example by repeated application of the graph Laplacian in ChebNet [49]. The seminal Graph Convolutional Network (GCN) [80] is a simplification of ChebNet, considering the one-hop neighbourhood only.

Second, it is commonly claimed that spectral methods are limited in their transferability. As the spectral definition of convolution depends on the Laplacian eigenbasis, which is domain dependent, it is claimed that a spectral CNN learned on one graph cannot be transferred to a different graph [112]. It is worth noting however that many descendent methods, such as Chebnet, diverge from the original spectral convolution implementation and as such do not suffer from this shortcoming [81]. Moreover, despite its prominence in the literature, this notion of non-generalisability is challenged [95].

Spatial methods

Spatial approaches, in contrast, perform more traditional convolution by defining local Euclidean neighbourhoods on which they perform template matching with a filter. This is performed by creating local positional relations between points, using different coordinate systems (e.g. spherical coordinates). They therefore do not suffer the inability of spectral methods to generalise across domains.

The Geodesic CNN (GCNN) [106] is one such GNN that operates in the spatial domain on manifolds (a form of polygonal mesh, see Section 2.5.2, that locally resembles Euclidean space near each point). At each point, a "patch operator" is used to map the values in a local neighbourhood into local polar coordinates [83], and a template is then matched to the extracted 'patch' at each point (taking the maximum over all possible rotations of the template due to the origin ambiguity in angular coordinates [112]).

Another spatial approach of note is the Anisotropic GNN (AGNN) [30], which uses anisotropic diffusion equations on a manifold to similarly extract patches, which are subsequently used for convolution.

A former divide: spectral vs spatial

Although the distinction appears throughout the literature, the dichotomy between spectral and spatial methods is somewhat artificial [32]. As demonstrated in [112], several geometric deep learning methods from both groups can be viewed as specific instances of a broader framework, by defining the pseudoccordinates and the weight function (kernel) to be used. For example, the spatial methods GCNN [106] and ACNN [30] use local polar coordinates, our beloved Euclidean CNN uses local Euclidean, while the spectrally inspired GCN [80] uses vertex degree.

Case Studies of layers used in this report

- **Graph Convolutional Network**: A seminal architecture in the GNN literature is the Graph Convolutional Network [80], which also gives a simple application of the mathematical underpinning explained in Section 2.3.2. With $\boldsymbol{A}$ representing the adjacency matrix and $\boldsymbol{I}$ the identity matrix, $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ gives the adjacency matrix with self loops inserted (so a node update step also depends on itself). This new adjacency matrix is scaled by the degree matrix, $\widehat{\boldsymbol{A}} = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$, and the resulting normalised adjacency matrix is then used in the neighbourhood aggregation of the update steps. Using weight matrix $\boldsymbol{W}$, node embeddings $\boldsymbol{H}$ during message passing are given by:

  $$\boldsymbol{H}^{l+1} = \sigma(\boldsymbol{W}\widehat{\boldsymbol{A}}\boldsymbol{H}^{l}) \tag{2.20}$$

  Relating this back to our blueprint in Equation 2.15, we see a simple weighted sum (using node degrees for weighting) as the aggregation method, and a sigmoid function adding non-linearity to the resulting node embedding. This is an example of the Convolutional GNN flavour explained in Section 2.4.3.

- **GraphConv**: The GraphConv layer formulated in [113] omits the normalisation step in the Graph Convolutional Network, as applying neighbourhood normalisation can reduce the ability of GNNs to distinguish certain graph structures [158] [113]. Alongside this, the GraphConv layer introduces a simple skip-connection to give each node an increased self weight.

- **Spline CNN**: The Spline CNN [58] aggregates neighbourhood features in the spatial domain [20], leveraging the local support properties of B-spline basis functions (that is, for all inputs outside a known interval they evaluate to zero [125]). The continuous kernel function, using B-spline bases, is parameterised by a fixed number of trainable weights, and therefore the computation time is independent of kernel size. This leads to speed improvements over comparable methods. A core reason for this report utilising this layer is its ability to incorporate edge features in feature aggregation: the kernel function maps pseudo coordinates (e.g. spherical or polar node coordinates) to a scalar used as a weight for neighbourhood aggregation [58].

2.4.5 Benefits of learning on graphs

The architectural assumptions embedded in graphs and GNNs impose a-priori structure on a problem. This goes against the trend of fully end-to-end learning (Section 2.2.2), and re-integrates some of the ML techniques from when resources were more limited [43]. Instead of choosing between end-to-end learning and hand-engineering, graph-based learning seeks to use them jointly [24]. We now outline three key benefits of learning on graphs.

The curse of dimensionality

Here we return to the notion of relations, introduced in Section 2.3.3, and more specifically relational inductive bias. A relational inductive bias [24] imposes constraints on relationships and interactions among entities in a learning process.

Since we can view many ML settings as attempting to approximate a true function, we are effectively searching the function space to find the best fitting solution. Ideally, this search also looks for the most regular functions within the class of hypothesis functions [32], as a counter to overfitting to training data. However, as the dimensions of the problem rise, the number of samples required to ensure our estimate is close to the true function grows exponentially

A common form of regularity in the MLP settings is the addition of a regularisation term to avoid overfitting [63], for example by adding a complexity function to promote weight sparsity [21]. However, this makes strong assumptions about the nature of the target function, and therefore may not be suited to complex functions encountered in complex data. We therefore need another form of regularity.

Inductive biases allow an algorithm to prioritise one solution over another, with the aim of improving/restricting the search for solutions without substantially diminishing the accuracy of the approximation. The introduction of these biases can therefore be seen as having a regularising effect, improving the bias-variance trade-off and helping to find a solution that generalises. For example, an image's natural spatial structure, and the geometric priors therefore imbued to a target function, define a source of regularity. In this way, leveraging inductive bias and exploiting symmetries in graph-based learning can be seen as a re-discovery of a classic remedy for the curse of dimensionality [32]. We have seen this idea before, in our discussion of CNNs using weight sharing which mitigates the curse [93].

Component	Entites	Relations	Rel. Inductive bias	Invariance
Fully Connected	Units	All-to-all	Weak	-
CNN	Grid Elements	Local	Locality	Spatial translation
Recurrent NN	Timesteps	Sequential	Sequentiality	Time translation
GNN	Nodes	Edges	Arbitrary	Node, Edge permutations

Table 2.1: Various relational inductive biases in common deep learning components, adapted from [24]

By uncovering what the entities and relations are in a specific learning process, we can see that this concept exists in the MLP and CNN. For example, since in a fully connected layer all the entities (i.e. neurons) in the layer are connected, the relations are all to all, and so there is minimal inductive bias as any inputs can interact to determine any output. In contrast, in a CNN the notion of locality is the relation (introduced by a shared local kernel across all inputs, so only a local set of inputs can interact to determine one output). As is made clear from Table 2.1, the notion of relational inductive bias is related to the invariances discussed above, as relations dictate the invariances that must be respected. Therefore, a key difference between a fully connected and convolutional layer is the imposition of additional inductive biases, and these biases are present in the relational structure of graphs.

Improved generalisation

It is also argued that the compositional structure of graphs (compositions of entities and their relations) is a key step towards generalisable models.

A key aspect of human intelligence is our ability to make new inferences and predictions from known building blocks (known as combinatorial generalisation [24]). An example of this is generative grammar, where a finite set of words can be composed in limitless ways into new sentences [40]. Once humans understand rules of grammar around verbs, they can instantly apply them to new words: even young humans are capable of this, while modern NNs are not [89]. One explanation of this is human mental models [45], with which we represent the world compositionally [66].

Despite the success of deep learning in many domains, a continual struggle is generalisation beyond training conditions and the ability to learn from small amounts of experience. These steps demand combinatorial generalisation, and so some [24] suggest that it is not surprising that existing models (that attempt to avoid explicit structure and compositionality) fail to meet them. Indeed, current NNs have been attacked as poor models of the mind as they are purely associative devices that cannot capture systematic compositionality [59], and this has been a key argument for proponents of symbolic forms of AI [60].

As a result of these shortcomings, graph networks that harness compositionality may provide an improved path to generalisable AI [90], offering redemption for NNs. Since GNN functions applied per-edge and per-node are reused across all edges and nodes respectively, they automatically support a limited form of combinatorial generalisation as they can operate on graphs of different sizes and shapes. For example, across COVID-19 imaging datasets where image size and shape differ, traditional CNNs require image cropping (and other preprocessing) while GNNs are naturally transferable [148].

Additional information provided by graphs

A final benefit of graph-based learning is a simple one: additional information is available. For example, the Cora [136] dataset is a citation network, in which each node represents a scientific paper, and each link (edge) shows that one article cites another. A common task on this dataset is to classify the class (topic) of each paper. Traditional approaches attempt to classify each document separately (e.g. using a paper's content). In contrast, a graph-based algorithm can take into account the relations between papers: a paper is more likely to cite another paper in the same field. By using this additional information, a fusion of citation and text information improves the classification accuracy significantly [104].

2.5 Medical imaging and graph extraction

Having laid out the background of ML, graphs, and the advantages of learning on graphs, we now turn to the application of this theory to medical imaging and the brain scans on which we perform our work.

2.5.1 MRI basics

There are multiple methods for obtaining images of the brain, including Computerised Tomography (CT), Positron Emission Tomography (PET) and Magnetic Resonance Imaging (MRI), the latter being the focus of this project. In contrast to CT, MRI scans do not use damaging ionising radiations (X-rays). This property, combined with its comparative suitability for detailed soft tissue imaging, makes MRI ideal for neurological imaging.

MRI works by placing the subject inside a strong magnetic field which forces the spin-alignment of protons within the human body (e.g. within hydrogen nuclei). An MRI machine then emits radio-wave pulses disrupting this equilibrium. When this second signal is removed, protons return to their equilibrium states. The time to return to equilibrium, and the resulting signal emitted during this 'relaxation', can be measured to build a 3D image of the body (it is worth noting that this is typically multiple 2D 'slices' that are combined).

2.5.2 Methods for extracting graphs

Having obtained a 3D image, the task of transforming this into a graph-based representation remains. In this section we will focus on two well known methods for extracting a graph from 3D images.

Marching cubes

A popular algorithm to generate a polygon mesh from 3D images is the Marching Cubes algorithm [100]. Mathematically, a polygon mesh is akin to an undirected graph, with additional properties such as faces (a closed set of edges). Its vertices, edges and faces define the shape of the object of interest. Triangles are a popular choice of polygon, to simplify rendering.

The algorithm works by extracting a triangle mesh of an 'isosurface' from the 3D image (which typically consists of voxels, representing values on this 3D regular grid). An isosurface is a surface that represents points of a constant value within a volume of space; given that Marching Cubes operates in 3D space, let us assume the isosurface defined by an implicit function $f(x, y, z) = 0$ as an example in this section.

The algorithm works by iterating ("marching") over a uniform grid of cubes that has been superimposed over a region of the function, and comparing the values of the vertices of the cube to the value of the isosurface. In our example, if all the vertices are positive, or all are negative, we know that the cube is entirely above or below the isosurface. However, if neither of these are the case, then the cube must intersect the surface.

The different ways that the isosurface may pass through the cube are characterised by the number of vertices that have values above or below the isosurface. For example, if one vertex is above the isosurface, and an adjacent vertex is below it, then we know the isosurface cuts the edge joining the two vertices (and the position that it cuts along the edge can be linearly interpolated). Since each vertex may be above or below the line, there are technically $2^8 = 256$ configurations of how the isosurface might cut the cube. However, many of these are topologically equivalent to each other due to two symmetries of the cube: they are related by rotation and/or by switching the states (above/below the isosurface) of the nodes. In the original paper, the authors conclude that this leads to 15 topologically unique cases [100], although subsequent work has demonstrated that this set is incomplete and there are 33 different configurations [38].

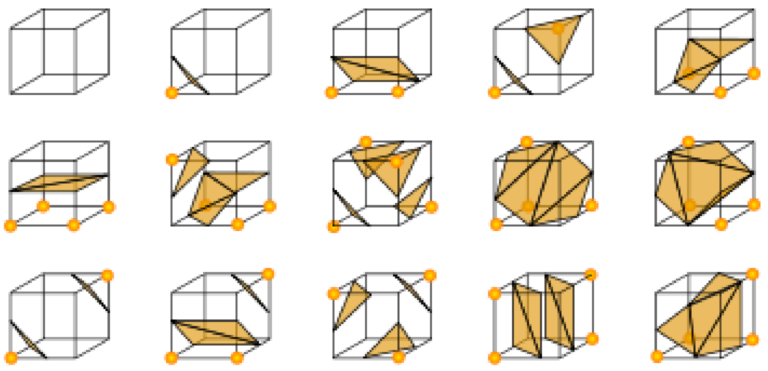


Figure 2.8: An example of 15 topologically unique cases of triangulated cubes, replicated from [100], included to aid understanding despite further work demonstrating that there are actually 33 unique cases [38]

The pattern of node values is then used as an index into a lookup table for these different configurations, and the resulting configuration is saved. Having iterated over all cubes, the union of the triangles emitted from each cube is the final mesh.

Whilst the original algorithm extracted meshes that presented discontinuities and topological issues, its widespread adoption has lead to numerous theoretical improvements to overcome these [38] and increasingly efficient implementations have been proposed [96]. As an example in the medical imaging field, [56] apply marching cubes to brain MRI, extending the algorithm to include the ability to correct holes that appear in the surface of the generated visualisation.

Superpixels

An alternative way to extract a graph from an image is by the use of oversegmentation, also known as *superpixels*. Superpixels (introduced in [130]) group pixels that share common characteristics, such as similar colour and other low-level properties, like location, into perceptually meaningful representation regions/segments [145]. These simplified images can then be applied in a number of common tasks.

SLIC (Simple Linear Iterative Clustering) is an algorithm to generate superpixels from an image [13]. The algorithm can be seen as a special case of k-means clustering, adapted to this task. The user dictates the number of superpixels they wish to use to segment the input image, K, and the algorithm starts by sampling K regularly spaced cluster centres. These centres are moved to seed locations at the lowest gradient position in a small neighbourhood, reducing the chance that they correspond to a noisy pixel and or are placed on an edge. As in k-means, pixels are then associated with the nearest cluster centre, and a new centre is computed using the average positional vector of all the pixels belonging to the cluster. This process is iteratively repeated until convergence.

One key point to note is the notion of 'nearest' in this algorithm uses an adjusted distance measure. In [13], superpixel cluster centres are choices in a five-dimensional space $C = [l, a, b, x, y]$, where $[l, a, b]$ is the pixel colour vector in the CIELAB colour space and $[x, y]$ are the pixel position in the image. Above a certain threshold, Euclidean distances in the spatial pixel dimensions begin to exceed and outweigh pixel colour similarities (leading to superpixels that only respect $[x, y]$ proximity, and not region boundaries). Therefore, [13] normalise the $[x, y]$ plane distance by the approximate interval between superpixels.

The popularity of SLIC is due to its simplicity: while it respects boundaries as well as, or better than, previous superpixel generating algorithms, it is faster and more memory efficient. Moreover, it is simple to extend to three dimensional images (*supervoxel generation*) [14]. While there exist multiple techniques to generate superpixels from an image (e.g. SNIC [15] or SEEDS [154]), SLIC is still recommended among state-of-the-art algorithms due to its simplicity and stability [145].

The use of superpixels for graph-based learning therefore comprises the following high-level steps:

1. Generate a superpixel representation of the input image (oversegmentation).
2. Transform the set of extracted superpixels into graph form, including the definition of locality/edges.
3. Supplement the graph with (node/edge) features.
4. Feed this representation to a graph-based learner, such as a GNN.

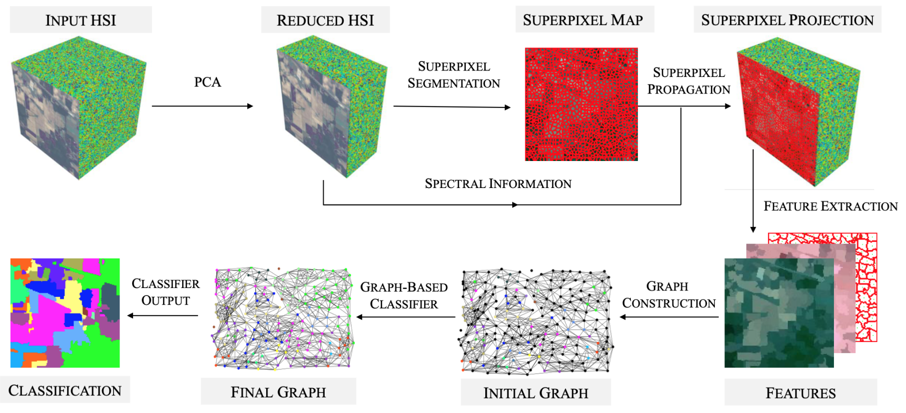


Figure 2.9: The superpixel based learning framework, implemented on hyperspectral images. The key steps are: superpixel extraction (oversegmentation), feature extraction, graph creation and classification with a graph-based learner. Reproduced from [135]

The oversegmentation step may be done by any of the methods mentioned in this section, although SLIC (or a variation of SLIC) is a common choice. For example, [135] applies these techniques to hyperspectral images (Figure 2.9), adjusting SLIC to work on high-dimensional data and introducing adjusted distance metrics more suited to this setting.

The transformation into a graph offers more opportunity for novel approaches. For example, [20] generate a “Region Adjacency Graph” by treating each superpixel as a node and adding edges between all directly adjacent superpixels (effectively creating one-hop neighbourhood connections). In contrast, in their implementation of Geometric Mixture Model CNNs, [112] include edges with K-nearest-neighbours, effectively increasing the span of the neighbourhood. These decisions therefore affect the connectivity of the graph, with neighbourhood size impacting information flow.

The features selected for nodes also offer an opportunity for diversity. Nodes (superpixels) may be imbued with features regarding their position in the image, or summary information regarding the pixels they encompass such as the mean (as in [135]), standard deviation or correlation matrices of pixel intensity. For example, when using the FashionMNIST dataset, which contains only grayscale images, [20] use as node features the concatenation of the average luminosity of the pixels in a superpixel and the superpixel’s geometric centroid.

Superpixels are therefore useful for graph-based classification. Pixels are highly redundant (neighbouring pixels are highly correlated) and numerous (making optimisation difficult). In contrast, superpixels are perceptually meaningful while vastly reducing the size of the graph classification required.

2.5.3 The hierarchy of medical graphs

Finally, having considered how medical images might be obtained and how graphs can be created from those images, we outline different graph types that appear in the medical setting. A key advantage of graph-based learning is its transferability across graphs of different sizes. In the medical imaging domain, there are three possible ‘levels’ of graph-based learning, which form a graph hierarchy.

- **Structure/object level** graphs might be meshes of a specific anatomical structure, where vertices define the shape of this mesh. Examples of tasks at this level could be disease classification using a specific subcortical brain structure.

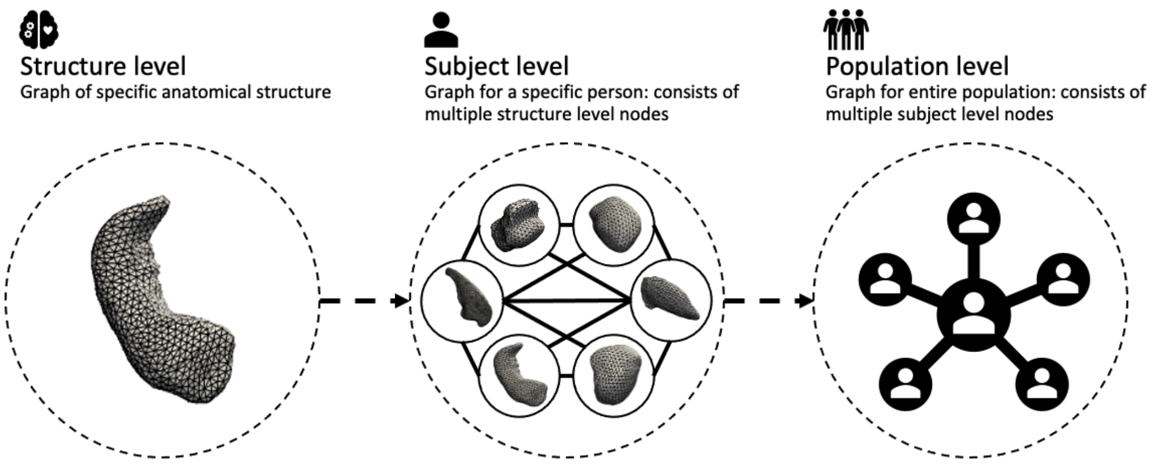

Structure level
Graph of specific anatomical structure
Subject level
Graph for a specific person: consists of multiple structure level nodes
Population level
Graph for entire population: consists of multiple subject level nodes

Figure 2.10: Visualisation of the three levels of the medical graph hierarchy, with a graph (or some summary statistic for the graph) at each level being a possible node at the level above.

- **Subject level** graphs represent a patient, with nodes representing different organs or structures that comprise the whole (e.g. a graph of the patient's brain, with nodes representing substructures). For example, [78] aim to predict the neurodevelopment of preterm infants, creating graphs with 90 nodes representing different brain regions based on the neo-natal atlas in [137].

- **Population level graphs** represent entire populations, with each node representing an individual. For example, for the classification of subjects with Autism Spectrum Disorders, [121] utilise a graph where nodes represent subjects. The nodes' feature vectors are extracted from brain imaging data, and edge weights are based on auxiliary phenotypic data to encode the similarity between subjects. Indeed, the follow up work [120] hypothesises that their state of the art performance is in part due to *relational bias* discussed above: by integrating expert knowledge (non-imaging information that is known to be linked to pathologies) into the graph structure in the form of edges encoding similarity, the learned representations can be improved. This structure, and the interaction between subjects that it captures, is a powerful advantage over traditional learning methods.

From this brief description, it becomes apparent that each level could contain the previous levels as subgraphs: a population of subject nodes, with each structure level graph as a subgraph. Aside from the ability to incorporate information at different levels and of different types, the hierarchical graph approach makes it possible to use a single graph-based learning algorithm that is transferable across each level of the graph.

Chapter 3

Data & Models

Having discussed the foundations of graph-based learning and medical imaging in Chapter 2, we now outline the data sources used in this report and the models we have constructed for our experiments.

3.1 Data

Alongside pre-prepared datasets available from the Pytorch Geometric library (see Section 4.4), both contributions in this paper use brain substructure meshes extracted from MRI scans in the UK Biobank Imaging Study [18]. Furthermore, our contribution on Alzheimer's Disease classification uses brain substructure meshes extracted from MRI scans in the OASIS-3 dataset. We outline both datasets here.

3.1.1 The UK Biobank dataset

The UK Biobank (UKBB) is a large-scale cohort study, containing over 500,00 individuals [147]. Of these original volunteers, 100,000 were invited back for brain, heart and body imaging. A key advantage of this dataset is that participants agreed to have their UKBB data linked to their NHS records, providing non-imaging data to complement the images.

The UKBB includes multiple brain imaging modalities, allowing studies of anatomical and neuropathological features (structural MRI), brain activity (functional MRI) and local tissue microstructure (diffusion MRI) [110]. This paper focuses on structural MRI, and more specifically the T1 modality. T1-weighted images are produced by using short times between successive radio-wave pulses, and are characterised by cerebrospinal fluid (CSF) showing as a dark region [5]. This modality is most informative about the gross structure of the brain (i.e. main anatomical landmarks) and the depiction of main tissue types (grey and white matter). The MRI scans must then undergo pre-processing before they are ready to use. These steps include field of view reduction to reduce empty space in the image, skull stripping, and registration to a reference shape (see [18] for further details).

Having obtained MRI scans, meshes of different subcortical structures need to be extracted. The method used on our data is a Bayesian Appearance Model, which uses both shape and intensity information from the MRI in a probabilistic manner. This method has been shown to be accurate for the representation and segmentation of 15 subcortical structures [122], which will be used in this report[1]. The method starts by fitting a 3D mesh to

[1]The 15 substructures are the brainstem and left & right: Hippocampus, Amygdala, Pallidum, Caudate, Putamen, Accumbens and Thalamus

the most typical shape across (training) subjects, before deforming this mesh to fit the individual images for each structure and subject. The method also relates to the marching cubes algorithm (Section 2.5.2), as to create a volumetric (voxelated) mask from the mesh, voxels intercepted by the mesh are identified as boundary voxels. All voxels within the interior of this boundary are subsequently filled. The application of the Bayesian Appearance Model is commonly done using the FSL FIRST software [123].

UKBB data is used in both contributions in this report. In all cases, the inputs are meshes of subcortical structures extracted from UKBB T1-weighted MRI scans. The task performed is biological sex classification as a simple binary graph-classification task, providing a proof of concept, using the Male/Female binary categorisation in the UKBB data. The dataset is well balanced for this task, with a total of 14,502 samples (subjects), of which 47.7% are female and 52.3% are male.

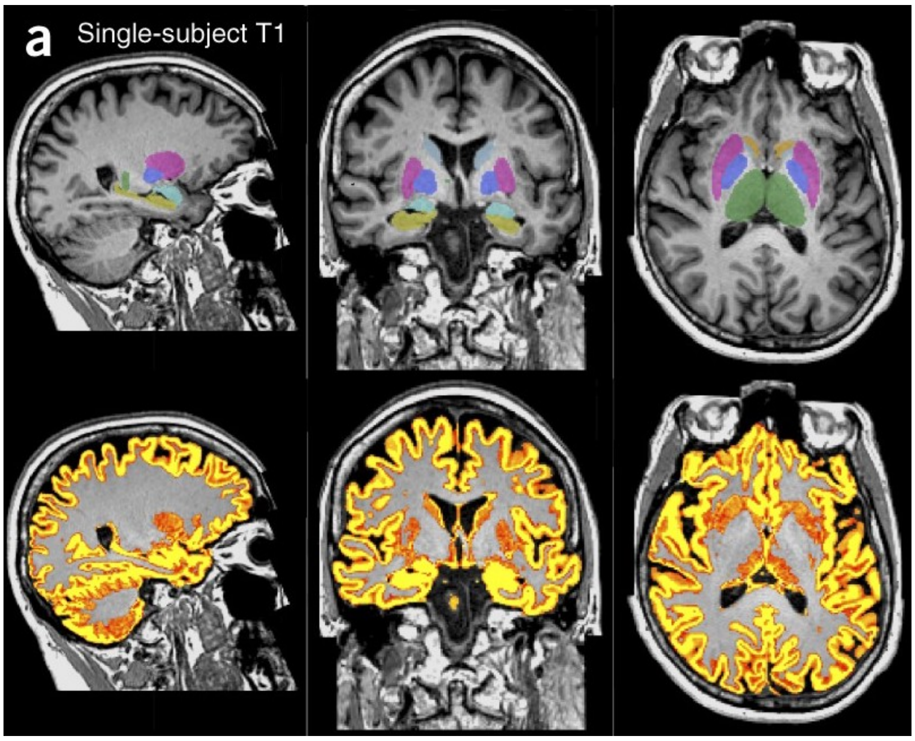


Figure 3.1: UKBB T1-weighted structural image for a single subject. Colour overlays show automated segmentation of subcortical structures (top row) and grey matter (bottom row). Reproduced from [110]

3.1.2 The OASIS-3 dataset

The OASIS-3 [91] (*OASIS*) dataset consists of MRI and PET imaging data, alongside related clinical data, for 1,098 participants across several studies over a 15 year period. Within this, there are 2,000 MRI sessions with multiple structural and functional sequences. From the T1-weighted MRI scans in this dataset, meshes of 15 substructures are extracted in a similar process to that used on the UKBB dataset, explained in Section 3.1.1. OASIS pre-processing (Appendix B) was designed to closely parallel the UKBB procedure (e.g. skull stripping and alignment), although it is not identical. Meshes from UKBB and OASIS were provided to us ready to use, with the extraction process already complete.

While the UKBB dataset contains multiple types of information on subjects alongside the scans (e.g. sex), pathological data is sparse. By contrast, the OASIS dataset contains 605 cognitively normal participants and 493 participants who suffer various stages of cognitive decline during the study, allowing Alzheimer's Disease related tasks. This is available in the Clinical Dementia Rating (CDR) accompanying the imaging dataset, with subjects receiving a score between 0 and 3. Whilst there are alternative dementia rating scales available (e.g. the NIA [7]), OASIS uses the CDR proposed in [114]: 0 = Normal, 0.5 = Very Mild Dementia, 1 = Mild Dementia, 2 = Moderate Dementia, 3 = Severe Dementia.

The CDR score is collected in clinical sessions that are separate to the imaging sessions, meaning sessions must be 'matched' to get an {image, CDR score} pair of input and target for classification. To avoid using a label that might be incorrect (e.g. a scan, with clinical data from many years before), we first find the closest available clinical diagnosis for each scan, before filtering out pairs where the absolute time difference between scan and clinical assessment is greater than 365 days. It is worth noting that this means multiple entries may occur per *subject* increasing the amount of data available (for example, if a subject has four scans over the 15 year period, each of these can be used as a separate sample). Finally, this dataset is filtered to only include scans available to us.

The result of this process was a dataset of 1,740 unique patient-imaging sessions (samples), each with a paired CDR score. This is comprised of 1,601, 120, 18 and 1 samples, for CDR of 0, 1, 2 and 3 respectively. We therefore work on the mesh representations of substructures as inputs, and their paired CDR scores as targets.

3.2 Models

3.2.1 GNN design

The models used in this paper are GNNs of different varieties, and therefore follow closely the GNN blueprint described in Section 2.4.3. The basic structure of one of our GNNs is several sequential modules, each consisting of a convolutional layer, batch normalisation layer, and non-linear (ReLU) activation. For graph classification tasks (in contrast to node classification tasks), a global pooling layer is added to aggregate the resulting node embeddings, and a final linear layer is used to form a prediction. Where applicable, a sigmoid layer converts this prediction into a probability. Unless otherwise stated, the convolutional layer used is the *GraphConv* layer outlined in Section 2.4.4 due to its increased expressivity.

Batch normalisation [3] layers were added as the regularisation effect of batch normalisation allows the use of higher learning rates, leading to faster training, and reduces the care required in parameter initialisation [74]. Before the final linear layer we also add a dropout layer to prevent overfitting due to neurons developing complex co-adaptation [142][69]. While the need for dropout might be eliminated by the introduction of batch normalisation [74], we do not know whether that applies in this specific case and so include both.

3.2.2 Generalisation to multiple substructures

The models described so far are capable of taking a single graph as input (e.g. a single substructure) for classification. However, our datasets provide 15 substructures, and we thus sought to design a GNN capable of taking multiple substructures as input.

Our multi-substructure GNNs had two design options: we could either use a submodel for each substructure, or we could train a single convolutional network on all substructures. These represent learning substructure specific features or learning generally useful features r

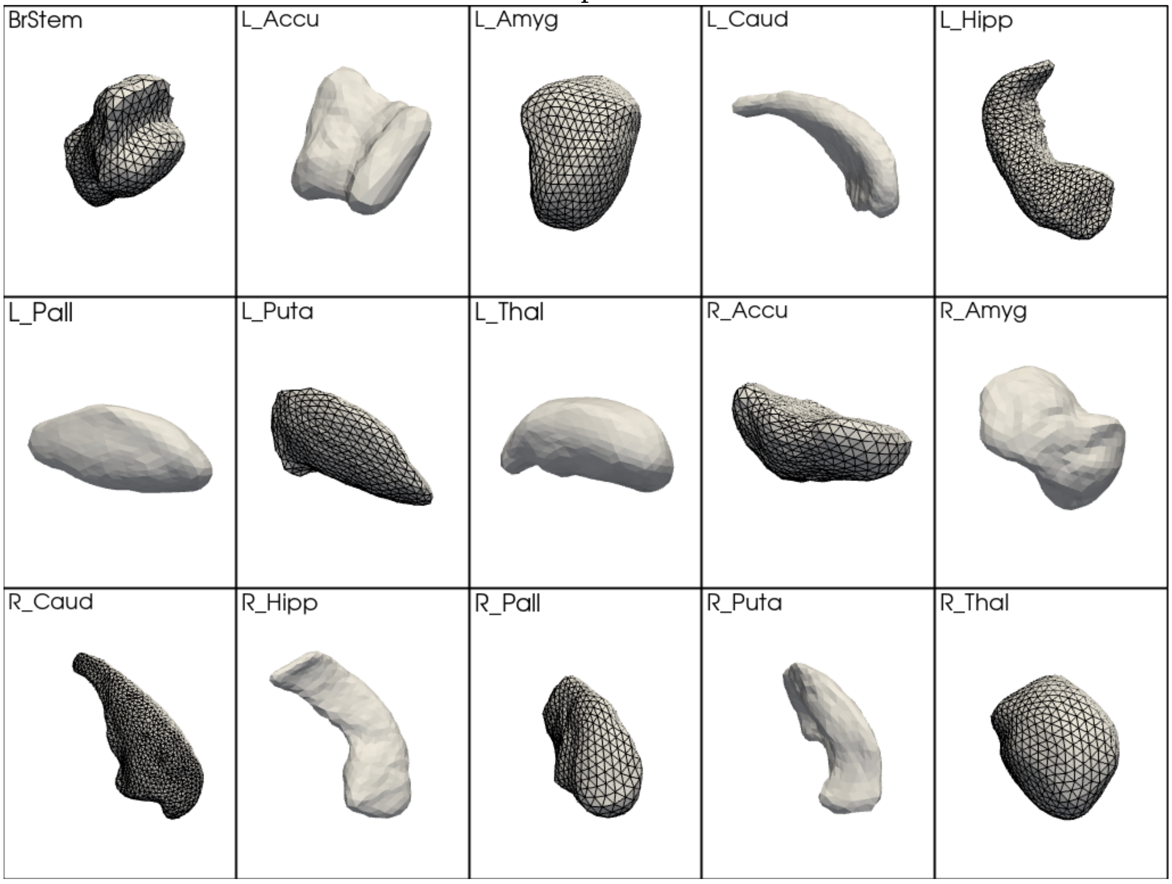


Figure 3.2: The 15 substructures used in learning tasks on UKBB & OASIS. This OASIS sample is for a single patient. Meshes rendered with and without edges to demonstrate the (triangular) polygonal meshing. Translation of abbreviation: (L : left, R : right), (BrStem : Brainstem, Hipp : Hippocampus, Amyg : Amygdala, Pall : Pallidum, Caud : Caudate, Puta : Putamen, Accu : Accumbens, Thal : Thalamus)

across all substructures together, respectively. Having no prior expectation, we decided to test the difference empirically by building models for the task of UKBB binary sex classification explained in Section 3.1.1.

The result of this testing was that using multiple (structure specific) submodels led to superior performance, and we therefore build off this approach. Each submodel returns an embedding for its specific input substructure, and the concatenation of these embeddings is then fed to a final linear layer for classification, in a method akin to ensemble learning. This model, including its submodels, is then trained end-to-end like a single GNN. Models are trained with the Adam optimizer [79], using (Binary where applicable) Cross Entropy Loss.

It is worth noting that this multi-structure model represents the hierarchy of medical graphs outlined in Section 2.5.3. Each submodel (GNN) takes as input a single *(sub)structure* level graph (e.g. the left hippocampus), and the resulting embeddings are fed into a fully connected layer. As highlighted in Table 2.1, this fully connected layer can be seen as an all-to-all connected *subject* level graph, with substructures representing nodes at this level.

3.3 Data transformations

Having obtained the brain substructure meshes described above, and designed the models to utilise them, we used the UKBB binary sex classification task to test whether further data transformations were required. We do not report the results from this dummy task, instead focussing on two key learnings that have shaped our approach. While additional graph features are explained in the following Chapters, the most basic features to use were the nodes' raw (3D) Cartesian coordinates.

Having built models that performed relatively well in binary sex classification on UKBB using Cartesian coordinates, our initial attempts to apply these (pre-trained) models to the same task on other datasets showed poor transferability, suggesting that the models had learnt UKBB specific features. One possibility was that the raw Cartesian coordinates contained information that, whilst discriminative in UKBB, was not the type of feature we wished to learn (for example, if all male subjects were higher up in the scanner). We therefore added a data transformation to align each substructure to a reference shape (specific to that substructure) that was calculated as a de-meaned average of 100 samples.

There are traditionally two components to alignment problems: point-to-point (here, node) correspondence and subsequently using these correspondences for alignment [16]. However, due to the data extraction process the point-to-point correspondence was known. Therefore, we used the Umeyama rigid transformation [152] which uses singular value decomposition to transform a point cloud to a reference shape, whilst retaining volumetric information.

Having implemented this transformation the overall accuracy of models on UKBB decreased, but the transferability improved, showing that we had been capturing unwanted information from the raw Cartesian coordinates. This reinforced our decision to use the Umeyama transformation: raw Cartesian coordinates, while discriminative on UKBB, were not the type of features we wished to capture. We therefore implement this transform for all tasks relating to brain substructure meshes.

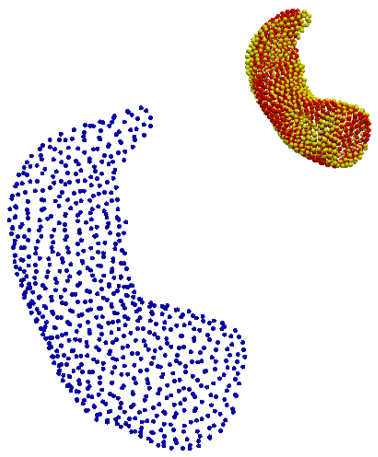


Figure 3.3: The Umeyama transform: a left hippocampus sample from UKBB is transformed to the average shape (blue = original sample; red = average demeaned shape; yellow = transformed sample). Note the spatial transformation (demeaning and alignment), whilst retaining shape variation (hence the lack of exact overlap between transformed sample and mean). Meshes here shown as point clouds for ease of depiction.

Chapter 4

Contribution 1 Curriculum by Oversmoothing

Having previously outlined the relevant theory on GNNs, and the advantages of graph-based learning, we now turn to our first contribution. In this chapter we briefly introduce the concept of Curriculum Learning, and one possible approach known as Curriculum By Smoothing. We then propose a novel idea in adapting this theory to a graph-based setting, and display experimental results to support our proposition. Finally, we highlight possible limits of our proposed approach and discuss areas for further research.

4.1 Introduction to Curriculum Learning

Trying to teach children maths is often not an easy task. However, it is made easier by being structured: children start on high-level, easy concepts, and are gradually introduced to more specialised, difficult examples. Just as an academic curriculum follows this easy-to-hard progression, so too does Curriculum Learning in AI.

Within the classification setting for NNs, instead of using all training examples equally, a Curriculum Learning strategy starts training with 'easy' examples and gradually introduces harder examples until the full training set is used. Curriculum strategies have two key advantages, with explanations offered in [27]:

- *Faster training*, as the learner does not waste time with noisy examples while it is not ready for them. Faster training with a curriculum has been demonstrated across a number of tasks [88] [27].

- *An improved parameter space*, as avoiding noisy examples moves the learner into better bases of attraction, and ultimately improved local minima, with better generalisation.

Due to this, curriculum strategies act like a regulariser [27], with test-set results obtaining the most benefit. Curriculum Learning appears to work by improving early training stages, which are critical for the training of NNs [75], setting the path for later stages.

The use of Curriculum Learning is far more general than the brief explanation here. For example, in a generative setting, [77] use a curriculum strategy to produce high-quality images using Generative Adversarial Networks. This is done by progressively growing the generator and discriminator, adding new layers over time to effectively increase the resolution of the images produced.

4.2 Curriculum by Smoothing

So far we have left the concept of what constitutes an 'easy' example untouched. A very human-like example is demonstrated in [27], whereby a NN classifying shapes obtains better results when it is first shown shapes with less variability (e.g. only equilateral triangles, circles not eclipses) before moving onto the full shape dataset. However we might also see other, more subtle, forms of noise. Indeed, this concept is not new. A 1994 paper [50] uses a Teacher-Learner Perceptron pair, with the teacher deliberately excluding the examples most different from its (known) decision rule, resulting in improved generalisation for the learner.

Curriculum By smoothing (CBS) [140] takes this notion of noise reduction most literally, proposing a widely applicable filter-based curriculum. Taking the simple example of CNN based image classification, CBS proposes to smooth the feature embeddings produced by convolutional layers by convolving them with a Gaussian kernel, a type of low-pass filter, reducing the amount of high-frequency information. The amount of smoothing is determined by the standard deviation, σ, of the Gaussian kernel. A curriculum is therefore formed by annealing σ during training, allowing more high-frequency information to be propagated within the network over time.

With this approach, [140] demonstrate substantial improvements in image classification. For example, using the CIFAR100 dataset they achieve an average 3.23 percentage point increase in classification accuracy over four different architectures when simply adding CBS.

The attractiveness of this approach is two fold. First, it appears to be widely applicable in many settings, requiring no a-priori knowledge of what constitutes an 'easy' example. Second, it promises to improve the performance of CNNs without adding additional trainable parameters, leading to time and computational efficiencies.

However, the CBS results are yet to be reproduced, and the presence and/or magnitude of accuracy improvements should be treated as provisional.

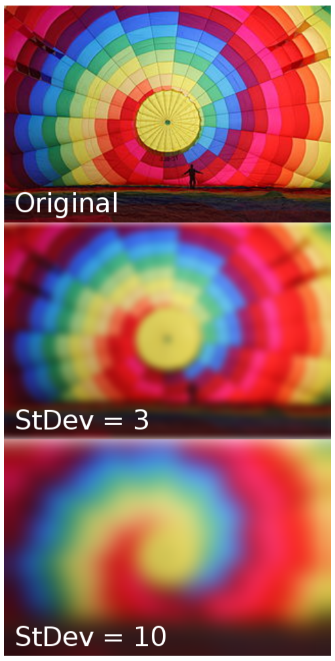


Figure 4.1: An image with Gaussian smoothing applied, for different values of σ. High frequency information (details) are removed, leading to blurring

4.3 Contribution: Curriculum By OverSmoothing

A commonly referenced problem in GNN literature is 'oversmoothing', whereby all node latents converge to homogeneity [159]. Referring to the GNN blueprint discussed in Section 2.4.3, we can view this as an issue in the Aggregation and Update steps. As the size of the neighbourhood over which we aggregate tends to infinity, node-specific information becomes "washed out" after several iterations of message passing [67]. Thus, oversmoothing acts like a low-pass filter [20].

Inspired by this 'problem', we propose to use oversmoothing for graph-based CBS in Curriculum By OverSmoothing (CBOS). The annealing of σ in image-classification CBS [140] has the effect of gradually increasing the amount of information retained from an individual pixel in its feature representation, and we attempt to parallel this for graph nodes.

Let x_i be the feature vector of node i, and $0 \leq w_e \leq 1$ be a fraction representing the 'self-weight' for a node. The subscript e denotes that this weight is a function of the epoch in training. We propose a simple formula for our 'smoothed' graph. First, an aggregation step: we calculate the average neighbourhood feature vector for a node

$$avg_i = \frac{1}{N_i} \sum_{j \in N_i} x_j \tag{4.1}$$

Second, an update step consisting of a simple weighted sum to determine the output latent for the node (in that epoch)

$$h_{i,e} = w_e x_i + (1 - w_e) avg_i \tag{4.2}$$

This simple setup can be used to create a graph-based approximation of Gaussian smoothing. For example, the neighbourhood size may be varied to obtain information from a larger area, akin to increasing σ in CBS. However, this leads to an oversimplification, as nodes are treated equally despite different distances from the source node i. We therefore propose instead implementing the average step in a 1-hop neighbourhood loop: by repeatedly running the average step, before the weighted update, information is diffused across the graph. In the first loop, node i will only obtain information on its neighbours. However, on the next loop, the neighbours themselves will have obtained information from their neighbours, and so i will get 2-hop information, but with the 2-hop neighbours obtaining less weight.

As in CBS, we propose to perform this smoothing on the output of convolutional layers in our GNN, with the self-weight w_e annealing to a value of 1 over time, to gradually decrease the smoothing and allow more high-frequency signal to propagate through the network.

4.4 Results

We apply CBOS to four datasets and report their results here, using variations of the GNN described in Section 3.2.

- **UK BioBank (UKBB)**: using the brain meshes extracted from UKBB we perform binary sex graph classification (see Section 3.1.1). For simplicity, a single substructure (the brain stem) is used, and the 642 nodes in each graph are imbued with their Cartesian coordinates as features, with the Umeyama transform applied.

- **MNIST-Superpixels (MNIST-S)** [6]: converts the well known MNIST dataset into superpixel representation (see Section 2.5.2), as in [112]. The dataset contains 70,000 graphs, each with 75 nodes. Each graph is labelled by one of 10 classes (digits 0 to 9), with the dataset being well balanced (all classes are between 9.0% and 11.2% of the total). The classification task is multi-class graph classification, aiming to classify each superpixel graph as one of the 10 classes.

- **Amazon Computers** [2]: represents goods purchased from Amazon. Nodes represent goods, with bag-of-words representations of product reviews forming node features. Edges represent two goods frequently bought together. This is a node classification task, and so the dataset consists of a single graph with 13,752 nodes, with each node belonging to one of 10 categories. The dataset is somewhat imbalanced, with the largest class (4) representing 37% of samples. The remaining classes range between 2.2% and 15.7%, with a mean of 6.9%.

- **Cora**: this citation network dataset (introduced in Section 2.4.5) represents scientific papers (nodes) and their citations (edges). Bag-of-words representations of papers' content are used as node features. Due to memory limitations we use a reduced version of the dataset. The task is node classification for a single graph containing 2,995 nodes, with each node belonging to one of seven categories. The dataset is somewhat imbalanced, with the largest class (4) representing 28.6% of samples. The remaining classes range between 6.5% and 15.1%, with a mean of 11.9%.

For both graph-classification tasks (UKBB, MNIST-S) the data is split into graphs belonging to test, train and validation sets. However, as node-classification tasks, the Amazon and Cora graphs are instead given test, train and validation masks to allow for held-out testing of individual nodes.

CBOS is performed by setting the initial self-weight for nodes to zero, annealing to one over the first 25 epochs, with a single loop to obtain the neighbourhood average (so only 1-hop neighbours are considered). The control (no CBOS) is implemented by setting the initial self-weight to one, so no smoothing is performed.

The results can be seen in Table 4.1, which reports the accuracies obtained over five random seeds.

	UKBB		**MNIST-S**		**Amazon**		**Cora**	
Seed	Control	CBOS	Control	CBOS	Control	CBOS	Control	CBOS
1	0.7417	0.7383	0.6275	0.6215	0.9030	0.9170	0.7140	0.9220
2	0.7386	0.7383	0.6079	0.6266	0.9070	0.9300	0.8380	0.8300
3	0.7151	0.7154	0.6339	0.6080	0.9160	0.9300	0.7840	0.7890
4	0.7400	0.7390	0.6264	0.6113	0.9190	0.9200	0.8390	0.8450
5	0.7241	0.7297	0.6156	0.6147	0.9050	0.9440	0.8310	0.8330
Avg	**0.7319**	**0.7321**	**0.6220**	**0.6164**	**0.9100**	**0.9282**	**0.8012**	**0.8438**
std	±0.012	±0.010	±0.010	±0.008	±0.007	±0.011	±0.054	±0.049

Table 4.1: Accuracies over datasets with and without CBOS. Results shown over 5 random seeds, with corresponding average and standard deviation

As is visible in Table 4.1, it appears that CBOS has negligible effect on UKBB and MNIST. In neither case is accuracy significantly different across control and CBOS, with average accuracies for UKBB of 73% and MNIST-S of 62%, both with and without CBOS. Moreover, in both cases there are seeds in which CBOS outperforms the control and vice versa. We therefore view these as null results, which are noted for completeness and future work.

However, it appears that CBOS has a beneficial effect on the Amazon & Cora tasks. For Amazon, mean accuracy is improved, with the CBOS enhanced model obtaining an average 93% accuracy in comparison to 91% for the control. Moreover, the CBOS enhanced model outperforms the control on every random seed, strengthening the case that CBOS has materially improved the discriminative power of the model. For Cora, there is one seed (seed 1) for which CBOS has a large positive impact. However, on only one seed does the control outperform CBOS, reinforcing the conclusion that CBOS is beneficial.

4.5 Discussion & future work

To our knowledge, CBOS is the first proposal of CBS on graphs and so marks a novel contribution to the literature. As with CBS, if these results are representative, CBOS offers an easy and highly transferable approach to improve GNNs. By implementing our approach in a simple and highly explainable manner, CBOS offers increased classification accuracy without additional parameters being required. However, based on the results above, it is also worth reflecting on its limitations.

One query is over why Cora contains one seed with a very large benefit from CBOS. Our proposed hypothesis goes back to the basics of Curriculum Learning: if a curriculum can be seen as pushing the model into preferable bases of attraction, then you would expect small improvements when the model finds good local minima naturally, but large improvements when CBOS prevents the model from otherwise going into poor local minima.

However, the key query hangs on the inconsistency of results: why was the effect pronounced on Amazon, and to an extent Cora, but not on the other tasks? Our proposed explanation goes back to the core of graph-based learning and representational bias: *CBOS should only be implemented on graphs where edges encode strong homophily.*

CBOS, like CBS, smooths high-frequency information with information from surrounding nodes, (hoping) to reduce noise towards an 'average' feature vector for a node of that class. In graphs where edges encode homophily this makes sense, as can be seen in a trivial example. Consider a (Hollywood movie) high-school dance, with shy teen boys on one side of the hall and girls on the other. Now imagine these students are nodes, with edges encoding friendship. If we were performing sex classification based on, for example, their Cartesian position in the hall, the 'difficult' examples would be the very few individuals who had started to move towards the other side. *If* edges encode sex-based homophily (i.e. boys are friends with boys), CBOS works perfectly: the outliers are 'smoothed' back towards their side of the hall, as that is where their graph neighbours (i.e. friends) are. However, if instead the node was a hetero-normative romantic graph, those outliers (and everyone else) would be smoothed towards partners on the other side, mixing the two groups, and making separation/classification impossible.

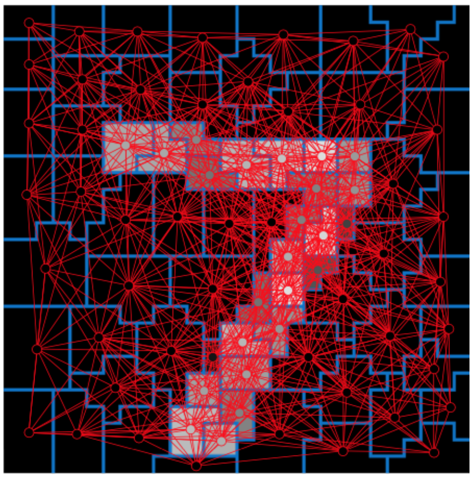


Figure 4.2: Superpixel-graph representation of MNIST digit; these graphs differ for each sample. Vertices are shown as red circles, edges as red lines. Reproduced from [112]

This hypothesis might explain why our method did not work on MNIST-S image classification, despite the original CBS paper demonstrating a beneficial effect for the original MNIST dataset. In [140], smoothing is done at a pixel (representation) level. Pixels are highly correlated, include a lot of redundant/duplicative information, and so are strongly related. In other words, their spatial closeness reflects homophily. However, a superpixel representation breaks this homophily. Superpixels intend to collate similar pixels, while respecting edges. Therefore, superpixels that are spatially close in fact represent *heterogeneity*, for example a sharp boundary or change in colour. In this situation, we would not want neighbours to obtain more similar representations by smoothing. We note, however, that this explanation relies on the superpixel methodology encouraging heterogeneity across superpixels which, whilst broadly true, is dependent on hyperparameters as in SLIC (Section 2.5.2).

For UKBB, this explanation also carries weight: assuming that any sex specific differences in Brain stem *shape* are minor, smoothing would reduce the difference between the classes. Equally, if *volume* is the key discriminative feature, then smoothing will have minimal effect as the two classes would retain their volume differences when local neighbourhoods are used for CBOS.

This brings us back to the heart of what differentiates graph-based learning from most current ML: we must *manually* consider the relation types within the graphs we create, and consider whether CBOS applies in *specific cases*. Further work might investigate our proposed explanation by implementing CBOS on both homophilic and heterophilic graphs to test our hypothesis.

Additional work might also be done to understand the optimum weighting method: What size of neighbourhood should be used? How many rounds of message passing? And how quickly should the self-weight be annealed? Perhaps the most exciting topic might be how to structure graphs to *enable* CBOS, and whether this additional imposition of relational bias improves overall performance. These questions pose exciting topics for future research.

Chapter 5

Contribution 2
Alzheimer's Disease Classification

Our second contribution applies GNNs to Alzheimer's Disease (AD) classification. After examining the AD literature, we propose two variants of multi-structure GNNs for this task, and compare them to a non-graph-based baseline. Finally, we examine these results and propose areas for further research.

5.1 Alzheimer's Disease (AD)

Alzheimer's Disease (AD) is a neurodegenerative disease mainly affecting older people, accounting for 60-80% of all dementia cases in the U.S. [105]. Typical symptoms of AD include memory loss and tremors, but more severe cases can include delusions and psychotic symptoms [57]. Despite the prevalence of AD, with an estimated 1/14 people over the age of 65 affected [9] and AD related death rates increasing every year [85], there is no known cure [146]. However, disease progression may be slowed with appropriate treatment.

There is evidence to suggest that AD could begin more than twenty years before symptoms become material [129]. Disease progression involves increasing loss of brain neuron connectivity and function, or even neuron death. The result is a significant reduction of life expectancy [131] and quality of life [99]. Combining this with the lack of a cure, the early identification and treatment of AD is paramount in reducing its impact on patient well-being. While this affects individuals, the economic cost in the U.S. is an estimated $370,000 lifetime care cost per patient, with total spending on dementias expected to rise to more than $1.1trn in 2050 [8]. This human and economic cost makes AD a worthy research topic.

5.2 Related work on AD classification

The ability of deep learning to identify complex features in high-dimensional data has led to its widespread use in medical imaging and disease classification.

Much of the existing work on AD classification is based on the use of traditional CNNs and linear classifiers, using imaging features only. [84] and [37] deploy 3D CNNs for binary AD vs cognitively normal classification, using MRI scans from the ADNI[1] dataset [1]. [46] compare 10 methods for classification tasks on ADNI, focussing on the type of features extracted from the MRI which are used as input for Support Vector Machines. These features

[1]Alzheimer's Disease Neuroimaging Initiative

fall into three key groups: voxel analysis (e.g. segmentation maps); cortical thickness; and hippocampus only (e.g. the volume, or an engineered feature describing the shape, of the hippocampus). The OASIS-3 dataset is used by [19] to predict preclinical AD (i.e. patients diagnosed as cognitively normal who subsequently develop AD). They propose a 3D Recurrent Visual Attention Model that takes as input all MRI images for that subject (in practice 96 images due to memory limitations). OASIS is also used by [97], who combine multiple MRI modalities and non-imaging data to predict cognitive decline from mild cognitive impairment (MCI) to AD. Finally, [150] utilise MRI image intensities to form a novel biomarker, which they subsequently combine with age and other cognitive measures to predict the progression of MCI to AD.

The use of graphs in AD prediction is currently limited, and predominantly focussed on population level graphs. [151] use non-linear graph fusion to combine multiple modes of subject information (e.g. MRI volumes, CSF biomarker measures, and genetic information). Each mode is used to make a population level graph, with edge weight representing similarity between subjects, and these graphs are unified into the final graph for classification. Meanwhile, [120] introduced in Section 2.5.3 for Autism Spectrum Disorder apply a similar population graph method to predict whether MCI will progress to AD, using the ADNI dataset. This graph leverages imaging feature vectors for node features and phenotypic information for edge weights. Finally, [141] propose a novel population level graph-construction method, performing various AD classification tasks on the ADNI dataset. An initial GCN operates on a graph that uses traditional similarity metrics (e.g. imaging information) to determine node connectivity. The resulting node embeddings are then used to encode similarity on an updated graph, and this 'similarity-aware' graph is used for classification.

5.3 Contribution: AD classification using GNNs

5.3.1 Dataset preparation

We selected the task of classification of AD using GNNs. One initial issue was significant class imbalance: as described in Section 3.1.2, healthy samples (CDR of 0) made up 92.0% of the dataset, whilst there was only one sample with a CDR of 3. We therefore decided to use a binary classification task (i.e. healthy vs AD) as opposed to classifying each level of CDR. For this, we binarised the labels, labelling samples with a CDR of 0 as class 0 (healthy), and all other samples as class 1 (unhealthy).

Whilst this binarisation helped reduce the class imbalance, significant imbalance still persisted. To mitigate this, we implemented a Weighted Random Sampler to artificially balance the training and validation sets [12] which works by drawing samples according to a specified distribution. By using the inverse class label prevalence as a sample's probability, we achieve approximately equal class representation. This weighted sampling was implemented during training and validation, and after separation of test/train/validation sets to avoid sample contamination across sets. We report results for models trained on both the original (imbalanced) training dataset and this artificially balanced training dataset. All tests use the original dataset with no oversampling, maintaining its original class (im)balance.

5.3.2 Model design and baseline

To understand whether GNNs provide an advantage in this classification task, we first establish a baseline model using Principle Component Analysis (PCA) [72]. PCA is commonly used as a dimensionality reduction technique that aims to extract key modes of variation in data to reduce its size whilst minimizing information loss. This is achieved by projecting the data into lower dimensional space through singular value decomposition. In the context of shape analysis, given a set of shapes, PCA can be used to describe the major modes of shape variation [143]. Moreover, the modes can be sorted in order of importance (i.e. those that contribute most to variance), as evidenced by the derivation of PCA using eigendecomposition of the covariance matrix, whereby the matrix of eigenvectors obtains this ordering.

PCA, and its derivative methods, have been used for shape analysis extensively [128]. Aside from this, PCA provides a useful baseline for two main reasons: first, it is powerful but simple to implement, thanks to its implementation in popular Python libraries. Second, it only uses the information contained in node features (here Cartesian coordinates), treating brain substructures like a point cloud (no edges) rather than a mesh; it therefore provides a benchmark for whether graph-based learning is truly beneficial.

Having established this baseline, we test three types of models. The output of each is a probability score, using the sigmoid function from Section 2.2.1:

1. A **PCA + Linear** (Baseline) first takes the training data and, for each individual substructure, uses PCA to extract N features, where N is the minimum number of features to retain 90% of explained variance. Then, during training, samples undergo this PCA transformation, with the resulting embeddings being input to a linear classifier.

2. A **GCN based** multi-structure model, outlined in Section 3.2, using GraphConv layers.

3. A **Spline GNN based** model, identical to the GCN except that Spline Convolutional layers [58] (see Section 2.4.4) are used in the substructure specific GNNs. The Spline layer additionally takes edge attributes as inputs to generate node embeddings. For this purpose, we use the spherical coordinates of linked nodes to obtain their relative positions (and thus increase the geometric information available).

For each of these three model types, we use two different sets of substructures as input. The first utilises all (15) substructures available in the extracted brain meshes. However, while AD can have broad ranging impacts on the brain, it first affects the hippocampus [164] making the left and right hippocampi especially vulnerable to damage in AD's early stages [115]. We therefore also report findings on models trained and tested using only the left and right hippocampi as input.

5.4 Results

5.4.1 Metrics reported

The results reported here are for our binary classification task of AD diagnosis. While many metrics have been proposed for measuring the performance of binary classifiers, our specific setting is nuanced due to its imbalanced test set, reflecting the distribution of the original dataset. This imbalanced test set is the case for *all* models, whether trained on the original (imbalanced) training dataset or the artificially balanced training dataset. Thus, a traditional accuracy measure provides an estimation of classifier ability that is balanced in favour of the majority class, and so is inappropriate. This is especially the case when the minority class (here, unhealthy individuals) are of interest [26][17]. We therefore also report alternative metrics that are more suitable to this situation. The reported metrics are:

- **Normalised Confusion Matrix**: reports true positives, false positives, true negatives, and false negatives as a percentage of the overall number of samples. In this report, Class 0 is healthy, and Class 1 is unhealthy (CDR>0). Confusion Matrices are reported with rows representing true class, and columns representing predicted class, in class order (i.e. 0 then 1). In our setting, a true positive represents an individual with CDR>0 being predicted as Class 1.
- **Accuracy**: correct predictions, that is true positives + true negatives, divided by the total number of samples.
- **Balanced Accuracy**: a simple average of the accuracy score for each class, to adjust for the large class imbalance.
- **Precision**: true positives divided by total Class 1 predictions. In our setting, this relates to the proportion of AD diagnoses that are correct (and also therefore indicates false positives/unnecessary scares).
- **Recall**: true positives divided by total Class 1 samples. In our setting, this relates to the proportion of subjects who have AD that are diagnosed (and also therefore indicates false negatives/missed AD cases).
- **F1**: the harmonic mean of precision and recall, the F1 score[2] is popular in ML as a single metric to summarise performance.
- **Specificity**: true negatives divided by total Class 0 samples. While Precision, Recall and F1 relate to Class 1 (i.e. AD), specificity reflects the classification of Class 0 (healthy) samples.
- **Matthews Correlation Coefficient (MCC)**: calculates the Pearson product-moment correlation coefficient between actual and predicted values [126], giving a worst value of -1 and best value of +1. In an imbalanced setting, the MCC is highly suitable as it only generates a high score if the classifier is able to correctly predict the majority of instances for both classes [39][3].
- **ROC (AUC)**: Receiver Operator Characteristic (ROC) curves show the trade-off between false positives and false negatives for different prediction thresholds. The area under the ROC curve (AUC) therefore represents the degree of separability between classes, or the classifiers capability to distinguish classes, indicating its performance regardless of class imbalance.

All metrics reported are the average results from five random seeds, with standard deviation displayed where appropriate.

5.4.2 Comparison of models

There are three core angles for comparison of models: first, the imbalanced vs balanced training sets; second, models using all (15) substructures vs those using hippocampi only;

[2]https://deepai.org/machine-learning-glossary-and-terms/f-score

[3]The MCC is becoming more popular as a metric: since its original proposal in 1975 [107] and subsequent re-purposing for ML [22], the MCC has been employed by the FDA as a key evaluation measure for models built off gene expression data from microarrays [138]

third, the different model types (PCA, GCN, Spline). Throughout these comparisons, we use a 0.5 threshold to decide if the model output indicates Class 0 or Class 1, a sensible prior. The impact of this decision threshold is examined in the following section (5.4.3).

	All substructures (15), original training dataset		
Metrics	**PCA**	**GCN**	**Spline**
Normalised	77.9 ¦ 14.8	91.3 ¦ 1.4	90.6 ¦ 2.1
Conf. Matrix	2.2 ¦ 5.1	5.5 ¦ 1.8	5.3 ¦ 2
Accuracy	0.8293 ± 0.0170	0.9316 ± 0.0114	0.9264 ± 0.0107
Balanced acc.	0.7680 ± 0.0350	0.6237 ± 0.0597	0.6264 ± 0.0533
Precision	0.2565 ± 0.0306	0.6309 ± 0.2032	0.5299 ± 0.1155
Recall	0.6958 ± 0.0891	0.2623 ± 0.1274	0.2751 ± 0.1157
F1 Score	0.3718 ± 0.0313	0.3448 ± 0.1003	0.3388 ± 0.1120
Specificity	0.8401 ± 0.0245	0.9851 ± 0.0112	0.9777 ± 0.0151
MCC	0.3489 ± 0.0330	0.3628 ± 0.0993	0.3340 ± 0.0906
ROC-AUC	0.8341 ± 0.0325	0.8747 ± 0.0369	0.8514 ± 0.0553

Table 5.1: Metrics for different model types trained using all substructures, with the original (imbalanced) dataset

	Left & Right Hippocampi, original training dataset		
Metrics	**PCA**	**GCN**	**Spline**
Normalised	82.3 ¦ 10.4	92.0 ¦ 0.7	91.2 ¦ 1.5
Conf. Matrix	2.9 ¦ 4.4	5.7 ¦ 1.6	5.6 ¦ 1.7
Accuracy	0.8667 ± 0.0238	0.9356 ± 0.0107	0.9287 ± 0.0053
Balanced acc.	0.7435 ± 0.0299	0.6091 ± 0.0405	0.6083 ± 0.0597
Precision	0.3031 ± 0.0371	0.7222 ± 0.1648	0.4989 ± 0.1113
Recall	0.5993 ± 0.0843	0.2262 ± 0.0828	0.2326 ± 0.1264
F1 Score	0.3968 ± 0.0224	0.3327 ± 0.0906	0.3019 ± 0.1364
Specificity	0.8878 ± 0.0305	0.9919 ± 0.0054	0.9839 ± 0.0081
MCC	0.3597 ± 0.0216	0.3715 ± 0.0787	0.3004 ± 0.1223
ROC-AUC	0.8390 ± 0.0397	0.8500 ± 0.0274	0.8244 ± 0.0407

Table 5.2: Metrics for different model types trained using left and right hippocampi only, with the original (imbalanced) dataset

Imbalanced vs Balanced training datasets

As described above, some models were trained using the original dataset and others with the artificially balanced dataset. The differences for these models can be found by comparing Table 5.3 with Table 5.1 (both containing models using all substructures) and Table 5.2 with Table 5.4 (both containing hippocampi only models).

As expected, the models trained on the original data obtain a higher accuracy score. As evidenced by their lower recall, and the confusion matrix, these models generally predict the majority class and as such obtain accuracies in line with the class balance (approximately 92% Class 0). As a result, their balanced accuracies are low. The models trained on balanced data, by contrast, have improved balanced accuracies, with recall increasing substantially at the cost of low precision. Finally looking at ROC-AUC and MCC, two class-balance agnostic metrics, we cannot claim that balanced or imbalanced training is universally better across all model types: ROC-AUC is higher when using balanced datasets for three model types of the six (model and substructure variants), while MCC is higher for two. It therefore remains subjective whether increased false positives (balanced training) or false negatives (imbalanced) is preferable, a point to which we shall return in our discussion.

	All substructures (15), balanced training dataset					
Metrics	**PCA**		**GCN**		**Spline**	
Normalised	67.9	24.8	78.6	14.1	90.3	2.4
Conf. Matrix	1.6	5.7	1.6	5.7	5.6	1.7
Accuracy	0.7362 ± 0.0134		0.8437 ± 0.0314		0.9201 ± 0.0162	
Balanced acc.	0.7615 ± 0.0327		0.8187 ± 0.0331		0.6032 ± 0.0448	
Precision	0.1879 ± 0.0141		0.2987 ± 0.0682		0.4704 ± 0.1555	
Recall	0.7908 ± 0.0831		0.7892 ± 0.0703		0.2325 ± 0.0971	
F1 Score	0.3028 ± 0.0198		0.4286 ± 0.0669		0.2896 ± 0.0995	
Specificity	0.7322 ± 0.0202		0.8482 ± 0.0357		0.9739 ± 0.0179	
MCC	0.2941 ± 0.0284		0.4201 ± 0.0632		0.2799 ± 0.0921	
ROC-AUC	0.8425 ± 0.0363		0.8640 ± 0.0380		0.8428 ± 0.0487	

Table 5.3: Metrics for different model types trained using all substructures, with an artificially balanced dataset

	Left & Right Hippocampi, balanced training dataset					
Metrics	**PCA**		**GCN**		**Spline**	
Normalised	65.0	27.7	74.5	18.2	88.1	4.6
Conf. Matrix	1.2	6.1	1.9	5.4	4.3	3.0
Accuracy	0.7109 ± 0.0268		0.7994 ± 0.0632		0.9109 ± 0.0115	
Balanced acc.	0.7690 ± 0.0288		0.7746 ± 0.0380		0.6802 ± 0.0223	
Precision	0.1812 ± 0.0211		0.2427 ± 0.0514		0.4013 ± 0.0849	
Recall	0.8367 ± 0.0738		0.7448 ± 0.1324		0.4100 ± 0.0389	
F1 Score	0.2969 ± 0.0279		0.3583 ± 0.0529		0.4025 ± 0.0528	
Specificity	0.7012 ± 0.0321		0.8043 ± 0.0773		0.9504 ± 0.0115	
MCC	0.2959 ± 0.0311		0.3443 ± 0.0449		0.3564 ± 0.0595	
ROC-AUC	0.8438 ± 0.0436		0.8612 ± 0.0262		0.8191 ± 0.0432	

Table 5.4: Metrics for different model types trained using left and right hippocampi only, with an artificially balanced dataset

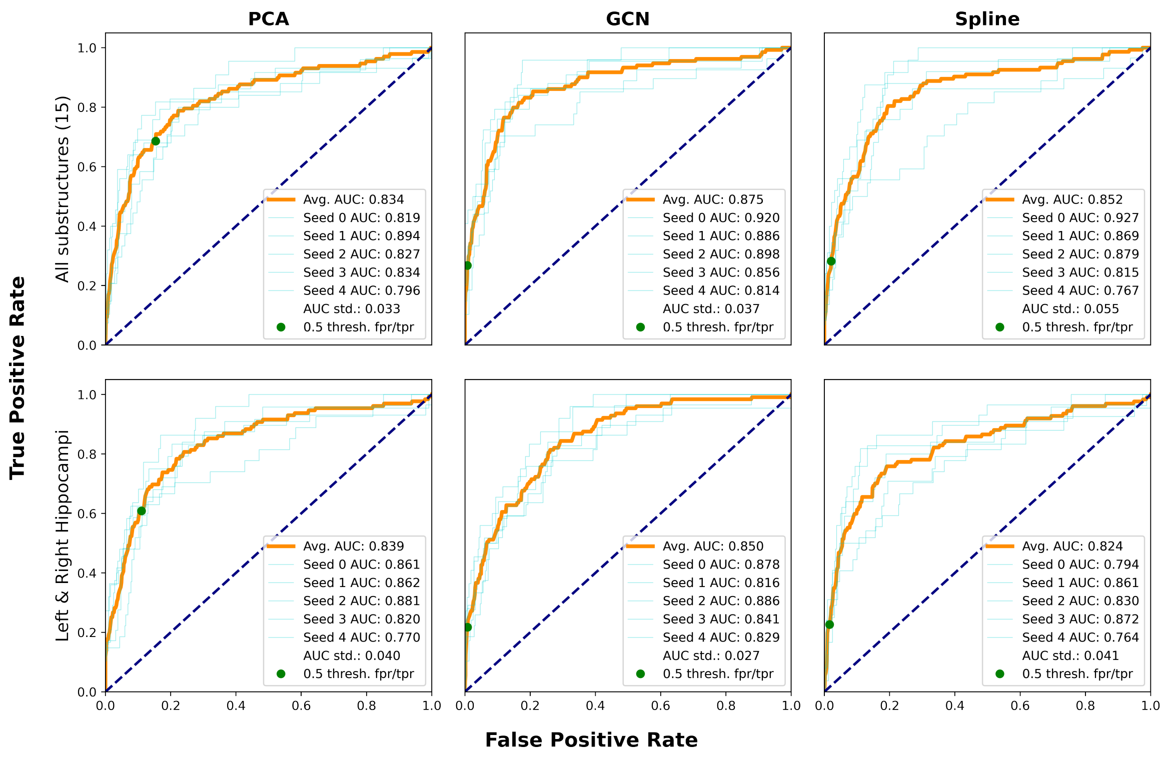


Figure 5.1: ROC & AUC for each model type, trained with the original **imbalanced** dataset

All substructures (ALL-SUBS) vs hippocampi only (HO) models

Reducing the number of substructures from fifteen to two (left and right hippocampi) substantially reduces the size of the model, with benefits in model storage and speed of training, but reduces the information input. The differences for these models can be found by comparing the models in Table 5.3 with Table 5.4 (both containing models trained on balanced datasets) and Table 5.1 with Table 5.2 (both with models trained on imbalanced datasets).

While we might expect this reduced information to impair performance, the evidence presented in Section 5.1 suggests that, because the hippocampi are a severely impacted substructure, they might be (one of) the key discriminative substructures. Indeed, our findings support this hypothesis. HO models obtain higher accuracies than ALL-SUBS models when trained on the original data, though ALL-SUBS models obtain higher accuracies with balanced training. Balanced accuracies are comparable in both training settings.

Turning to ROC-AUC, we see that ALL-SUBS models are indeed superior classifiers (comparing the top row with the bottom row in Figures 5.1 and 5.2) as we would expect. Comparing models trained on the original dataset, ALL-SUBS GCN and Spline models obtain higher ROC-AUCs vs their HO model counterparts, while PCA ROC-AUC is marginally better for HO. This pattern is repeated for models trained on a balanced dataset.

One final comparison is the highest scores obtained by any ALL-SUBS model compared to those for any HO model. The maximum ROC-AUC for ALL-SUBS is obtained by the GCN model trained on the original dataset, which obtains ROC-AUC of 0.8747 and MCC of 0.3628. Both of these metrics are greater than those of the maximum ROC-AUC HO model, the GCN trained on the balanced dataset, which obtains ROC-AUC of 0.8612 and MCC of 0.3443.

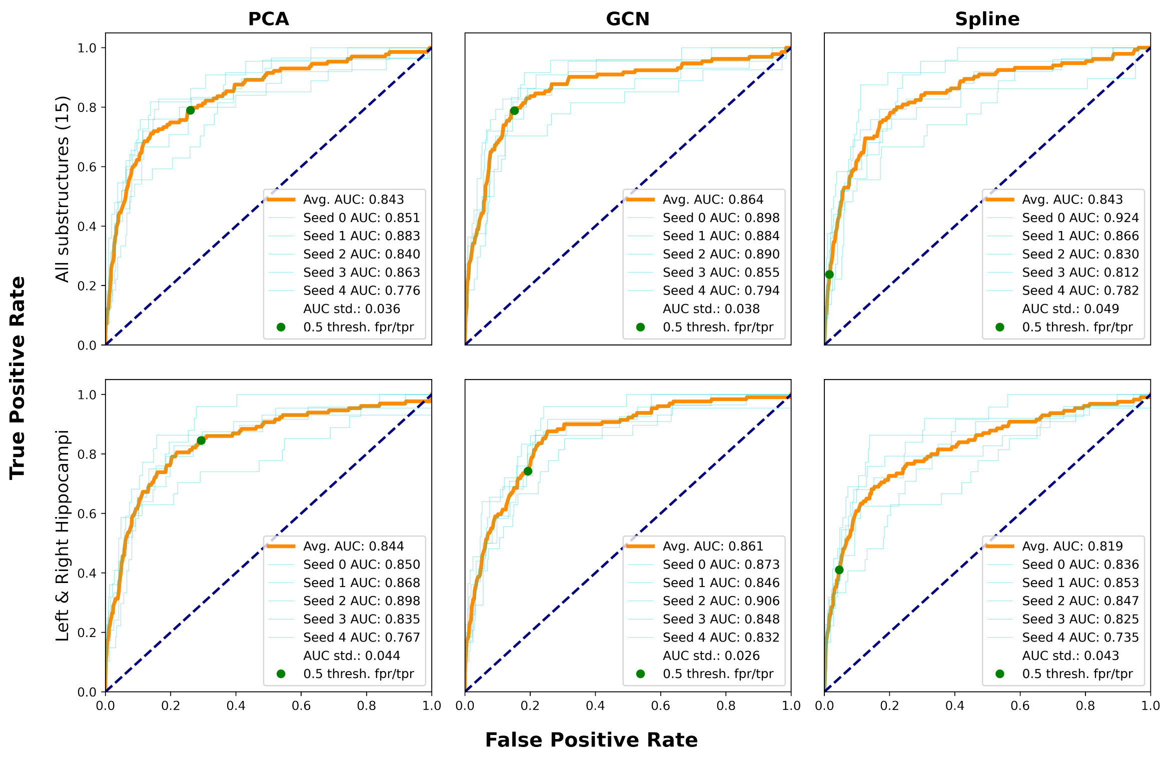


Figure 5.2: ROC & AUC for each model type, trained with an artificially **balanced** dataset

PCA vs GCN vs Spline models

Finally, we turn to a comparison of the different model types, and whether our proposed graph based learning outperforms the PCA baseline. The differences can be found by comparing the columns within each of Tables 5.1, 5.2, 5.3 and 5.4.

Looking at models trained with the original dataset, PCA performs reasonably well, obtaining the highest balanced accuracy. However, GCN based variants outperform the baseline on accuracy, MCC and ROC-AUC on both ALL-SUBS and HO models. In contrast, despite Spline obtaining a high accuracy score, it fails to outperform the baseline. All models obtain reasonable accuracy, in line with the class imbalance. One point of interest is the Precision-Recall balance: while PCA has low precision and reasonable recall, GCN and Spline models invert this. In other words, PCA is over-predicting AD, whilst GCN and Spline miss it.

Turning to the balanced setting, the outperformance of GCN compared to the baseline is even more evident. In both ALL-SUBS and HO models, GCN outperforms in seven of eight metrics, including clear outperformance in MCC and ROC-AUC. Spline, on the other hand, obtains strong accuracy, but fails to outperform in a significant manner (if anything, underperforming).

This underperformance by Spline merits examination, as we would typically expect this larger capacity model to have stronger performance than simpler variants. We can gain insight by looking at the ROC-AUC curves depicted in Figures 5.1 and 5.2. Alongside the average ROC curves shown in orange, which reconfirm that our GCN model outperforms the baseline, we show the ROC curve from each seed in light blue. As shown by the spread of these curves, and the AUC standard deviation, the Spline based model displays much more variable performance across seeds. Indeed, the highest ROC-AUC in both the balanced and imbalanced training settings are obtained by the Spline based model using Seed 0 (ROC-AUC above 0.92 in both cases). Combining this insight with Spline's relative underperformance, our results suggest that the Spline based model is both sensitive to initial conditions and using a strongly suboptimal decision threshold. While the latter point is explored in Section 5.4.3, the underperformance and ROC curve variance might also suggest overfitting and overparameterisation, which is more likely for Spline due to its larger capacity.

Finally, comparing the maximum ROC-AUC for each model type across all settings we find PCA = 0.8438, GCN = 0.8747 and Spline = 0.8514. From the performance presented in this section, and this final comparison for reference, we conclude that our GCN model does indeed outperform the baseline. We cannot draw this conclusion for Spline.

5.4.3 The impact of decision threshold

As explained, all models in Section 5.4.2 use a decision threshold of 0.5. However, whilst a sensible prior, this may be suboptimal. In Figures 5.2 and 5.1, the FPR and TPR for the current 0.5 decision boundary are plotted (as a green dot) on the average ROC curve. The most optimal point for each classifier is the top left corner of the graph (i.e. TPR = 1, FPR = 0), and so the best attainable point can be seen as a point on the ROC curve closest to this optimum. The optimal decision point can be a learnable parameter, chosen using a validation set to determine what threshold should be set. However, due to time limitations preventing us from performing this optimisation, we instead qualitatively discuss the discrepancy between the current threshold and the optimal threshold.

Examining the location of the plotted points for the current decision boundary, two key themes emerge:

1. **Balanced Training**: Models trained using the artificially balanced training dataset appear to be closer to their optimum point than those trained on the original dataset. This might be seen as a benefit of balanced training, but it also likely results in the metrics reported in Section 5.4.2 appearing disproportionately in favour of balanced training. Models trained on imbalanced training would benefit most from decision point optimisation.

2. **Discrepancy across models**: The PCA baseline appears to be closer to its optimum point than GCN and Spline, with this effect being especially pronounced when training on the original dataset. By contrast, the GCN appears to be close to its optimal point when trained on balanced data. These results suggest that the GCN would further outperform the baseline PCA with decision threshold optimisation, especially in the imbalanced setting where the 0.5 threshold is substantially suboptimal. Meanwhile, Spline's decision threshold is severely suboptimal in both training settings, and this provides one explanation of its surprisingly poor performance.

Seeing these large discrepancies both in settings and in models, further work (and models put into production) should treat decision threshold optimisation as a key aspect of training.

5.5 Discussion & future work

To our knowledge this marks the first use of brain substructure meshes in the classification of AD. There appear to be clear advantages to using GNNs in this setting, as shown by the performance of the GCN model compared to our baseline PCA model. Before further evaluating our model, there are two broad caveats:

First is the accuracy of labelling, which applies equally to our work and others. The OASIS dataset provides a CDR from clinical evaluation, without histopathological verification, meaning that samples labelled as having AD may in fact not. Moreover, the appearance of AD prior to symptoms showing means that some samples labelled as healthy may in fact have preclinical AD. Whilst these factors are unlikely to lead to large scale mislabelling, a model might be penalised for *correctly* classifying an *incorrectly* labelled sample.

Second, it is difficult to evaluate our models' *absolute* performance due to limited directly comparable studies. For example, in the task of AD vs cognitively normal binary classification, the definition of AD (e.g. what the CDR threshold is, or what CDR scale is used) may vary. Moreover, different class balances across datasets reduce the comparability of class balance sensitive metrics (e.g. accuracy).

With these caveats noted, it appears that our GCN model is competitive with existing state of the art models for this AD vs cognitively normal binary classification. All the examples found for this specific task use the ADNI dataset, but we briefly cover their results here to give a directional benchmark. [84] obtain accuracies of 0.79 and 0.80 for two different 3D CNN models, with ROC-AUC of 0.88 and 0.87. Our ALL-SUB GCN obtains a comparable ROC-AUC of 0.86, and *balanced* accuracy of 0.82 (which likely overpenalises our model by negating class imbalance). [37] use both a single 3D CNN and a multi-3D-CNN ensemble. Using a single CNN, they report accuracy of 0.85 and ROC-AUC of 0.90; using multi-CNN ensemble they achieve accuracy of 0.87 and ROC-AUC of 0.92. Once again, these figures are comparable. Finally, [46] report sensitivities and specificities (Appendix A) for Support Vector Machines trained using different imaging features. Of the 28 variant models reported there, our ALL-SUBS GCN trained on a balanced dataset would rank 4th for sensitivity/recall and 24th for specificity.

Whilst already positive, this evaluation is likely an underestimate of our model's potential performance relative to these external benchmarks:

- Our **model size** is constrained due to the resources available to us, both in computation and (training) time: for example, our ALL-SUBS GCN has approximately 260,000 parameters. Table 5.5 shows the GCN as reported in Section 5.4.2, compared to an identically structured but larger model. Whilst even this larger model would not be seen as large by modern standards (with 15.8M parameters), it offers a clear improvement on most metrics, with ROC-AUC rising from 0.8612 to 0.8770.
- The **mesh features** we use are simple (Cartesian and Spherical node coordinates). Whilst we had insufficient time to test the use of additional features to find an optimal collection, and so instead provide a proof of concept for a 'baseline' GNN, providing additional features might improve classifier performance further.
- **Suboptimal decision thresholds** are prevalent in our models, as discussed in Section 5.4.3, whilst the models we compare them to are likely optimised in this aspect.

Substantially larger models, with additional features and optimised decision thresholds, might therefore take our proposed approach from competitive to state of the art.

	GCN Hippocampi only, balanced training dataset			
Metrics	**GCN: 260k params.**		**GCN: 15.8M params.**	
Normalised	74.5	18.2	79.6	13.2
Conf. Matrix	1.9	5.4	2.2	5.0
Accuracy	0.7994 ± 0.0632		0.8455 ± 0.0446	
Balanced acc.	0.7746 ± 0.0380		0.7817 ± 0.0506	
Precision	0.2427 ± 0.0514		0.3062 ± 0.0956	
Recall	0.7448 ± 0.1324		0.7055 ± 0.1550	
F1 Score	0.3583 ± 0.0529		0.4005 ± 0.0433	
Specificity	0.8043 ± 0.0773		0.8580 ± 0.0611	
MCC	0.3443 ± 0.0449		0.3872 ± 0.0294	
ROC-AUC	0.8612 ± 0.0262		0.8770 ± 0.0147	

Table 5.5: GCN models differing only in model size, trained on Hippocampi only and using a balanced training data.

Our evaluation must also consider an ethical issue: it is an open question over which type of error (false positive or false negative) are more important in this setting. On one hand, false negatives are clearly serious as they may lead to an early diagnosis being missed, and therefore serious detriment to the patient's future. On the other, false positives may put unnecessary stress on healthy individuals. This has implications for model evaluation: if we consider false positives most important, our GCN model's strong performance on sensitivity, with weaker specificity, may be seen as a beneficial aspect. Whilst the reader is encouraged to form their own opinion, we consider sensitivity the more important metric because further diagnostic work can disprove false positives, whilst false negatives are left to suffer. Moreover, this conclusion is reinforced by Automation Bias [119], where AI recommendations are accepted by clinicians who then stop searching for (dis)confirming evidence. This bias is strongest when a machine advises that a case is normal [119].

With regard to further work, we have outlined simple enhancements (larger capacity, using more features, and optimising decision thresholds). However, we believe that there is also scope for two extensions of our hierarchical graph-based models:

1. **Examining substructure relations:** As outlined in Section 3.2.2, our multi-substructure GNNs can be seen as using information on two levels of the medical graph hierarchy: individual meshes at the structure level, and a fully connected layer at the subject level. Whilst we have used a fully connected subject level graph, further work might explore the imposition of relational bias on the subject level graph. For example, whether only certain substructure embeddings should be connected to each other.
2. **Population graph**: Likely most beneficial is extending our model to a population level graph, akin to [120]. This population graph might connect all subjects with edges encoding phenotypic similarity, and use outputs from our multi-structure GNN as node features. This way, information can be extracted at all levels of the medical hierarchy. Moreover, in this graph edges would encode homophily, and therefore our proposed CBOS might be applicable. An end-to-end trainable GNN capable of working on all levels of this hierarchy would be the fullest expression of our graph-based work.

Finally, our model is trained on a relatively small dataset, and might benefit from *pre-training* on a larger dataset. It is to this extension that we now turn.

5.6 Pre-training models

Pre-trained models are models that, having been trained on a given dataset, are available to use 'off-the-shelf' in a different setting. Pre-trained NNs can be seen as having parameters trained to extract useful features for whatever dataset & task they were pre-trained on. As explained in Section 2.2.2, deeper levels in NNs tend to extract higher level features. In many cases we might want to extract the same low level features for different tasks. For example, in a classic image classification task, we might wish to detect slanted lines regardless of the classification target. These pre-trained networks can be used as 'backbone networks' for other tasks, whereby they are used for low level feature extraction and some final layers are added. It is only these final layers that must be trained, or fine-tuned, to extract *task specific* high-level features[4].

After first discussing the benefits of pre-training, we attempt to use the large UKBB dataset to pre-train models for use in our AD classification task.

5.6.1 Benefits of pre-training

There are multiple benefits of pre-training:

- **Cost:** Training a model has monetary costs. Whilst the models in this report are relatively small, the cost of training GPT-3, OpenAI's state of the art NLP model, is estimated between \$4.6m [10] and \$12m [11]. Using pre-trained models as backbone networks avoids duplicating these costs.
- **Time:** Similarly to cost, the time to train large models can be prohibitive. It is estimated that GPT-3's 175bn parameters [11] would take approximately 355 years to train on a single GPU cloud instance [10].
- **Environmental impact:** This increasingly long training for state of the art models has lead to an increasing environmental impact, with substantial carbon footprint from the computation [144]. Moreover, trial and error to get to the final model can be substantial: [144] estimate that thousands of different versions of a given model might be trained and, in their example case, building a final model can produce up to 78,000lbs of CO_2. For reference, the average American adult produces around half this amount each year [4].
- **Sparse Data & generalisation:** due to the curse of dimensionality (Section 2.4.5), the number of samples required to train a model capable of generalisation grows with the number of parameters. For small datasets, this might limit the size (and therefore expressivity) of the model. However, by pre-training a backbone network on a large and more general dataset, before training the final layers on a (task specific) small dataset, this issue can be avoided. Indeed, pre-training can be seen as a regularizer, especially improving the generalisation of lower levels in deep architectures [55], allowing fine tuning to operate on task specific learning. Due to this, the benefits of using a fine-tuned pre-trained model, as opposed to training a model from scratch, grow as the size of the target dataset shrinks [156].

[4]As an interesting addendum, the reliability of pre-trained models has been questioned in some forums, with some existing pre-trained models available in the Keras library producing inconsistent results. However, this is due to methodological errors in their production (such as weight gradients not being frozen), rather than being intrinsic to pre-training [117].

The use of pre-trained models, of course, depends on the backbone network being transferable: if the model only learns features highly specific to its original training task, it may be useless in other settings.

5.6.2 Pre-trained models: frozen and unfrozen

In this report we test two methods of pre-training. Due to time limitations, we use a single model type for proof of concept: the GCN (Section 5.3.2) trained on hippocampi only. We select this variant due to its strong performance on the AD classification task and its faster training times due to the reduced number of substructures. In both of these methods, we pre-train the model using the UKBB binary sex classification task outlined in Section 3.1.1 before fine-tuning on OASIS for our AD binary classification task. Due to the much larger size of the UKBB dataset, we might hope that pre-training models helps them learn more generalisable or universally discriminative features.

In the first method, referred to as *frozen* in this report, all non-linear layers have their parameters frozen after pre-training, while new linear layers replace the pre-trained ones. This pre-training can be seen as teaching the model to extract features on UKBB, and then use the extraction of these features for OASIS AD classification. The (unfrozen) new linear layers are fine-tuned to use these features for the AD classification task.

The second method, referred to as *unfrozen*, leaves all gradients unfrozen for non-linear layers. This means that the parameters, and the resulting features they extract, can be fine-tuned to be more AD specific. This pre-training can be seen as a complex initialisation step for OASIS AD classification, as the parameters are pushed to a basin of attraction and fine tuned within this confine. Linear layers are replaced and trained as in the *frozen* method.

5.6.3 Results & discussion

Results are presented in Tables 5.6 (frozen backbone) and 5.7 (unfrozen backbone). In this section "balanced"/"imbalanced" training data refers to the OASIS data used for fine tuning: the (balanced) UKBB data is used similarly for all models.

	Hippocampi GCN, pre-trained on UKBB, frozen			
Metrics	**Original (imbal.) training**		**Balanced training**	
Normalised	90.1	2.60	71.90	20.80
Conf. Matrix	6.80	0.50	3.30	4.00
Accuracy	0.9052 ± 0.0275		0.7586 ± 0.0979	
Balanced acc.	0.5188 ± 0.0292		0.6587 ± 0.0469	
Precision	0.0583 ± 0.0726		0.2345 ± 0.1622	
Recall	0.0661 ± 0.0900		0.5419 ± 0.1932	
F1 Score	0.0611 ± 0.0789		0.2569 ± 0.0396	
Specificity	0.9715 ± 0.0351		0.7755 ± 0.1181	
MCC	0.0343 ± 0.0507		0.2218 ± 0.0611	
ROC-AUC	0.6826 ± 0.0683		0.7650 ± 0.0510	

Table 5.6: Metrics for Hippocampi-only GCNs pre-trained on UKBB sex prediction task. All layers were then *frozen*, with new final linear layers added for fine tuning and prediction on OASIS

The **frozen** models (Table 5.6) show reduced performance compared to the models trained only on OASIS (reported in Section 5.4.2). The model trained on imbalanced OASIS data

	Hippocampi GCN, pre-trained on UKBB, unfrozen			
Metrics	**Original (imbal.) training**		**Balanced training**	
Normalised	91.80	0.90	80.10	12.60
Conf. Matrix	5.60	1.70	2.40	4.90
Accuracy	0.9351 ± 0.0092		0.8500 ± 0.0404	
Balanced acc.	0.6108 ± 0.0310		0.7731 ± 0.0265	
Precision	0.6927 ± 0.1274		0.2982 ± 0.0715	
Recall	0.2309 ± 0.0650		0.6824 ± 0.0853	
F1 Score	0.3372 ± 0.0717		0.4055 ± 0.0584	
Specificity	0.9907 ± 0.0065		0.8638 ± 0.0498	
MCC	0.3688 ± 0.0632		0.3806 ± 0.0484	
ROC-AUC	0.8576 ± 0.0248		0.8653 ± 0.0234	

Table 5.7: Metrics for Hippocampi-only GCNs pre-trained on UKBB sex prediction task. All layers left *unfrozen* and fine tuned alongside new final linear layers for prediction on OASIS

obtains an MCC close to 0, balanced accuracy close to 0.5, and a ROC-AUC of 0.68, highlighting the poor performance of this classifier. Meanwhile, the model trained on balanced data obtains an improved balanced accuracy (0.66) and ROC-AUC (0.77). We therefore conclude that the *frozen* method of pre-training is not beneficial. This suggests that the features learnt by the model (when training on UKBB sex classification) are suboptimal for AD classification, though not totally inapplicable given the balanced model's performance.

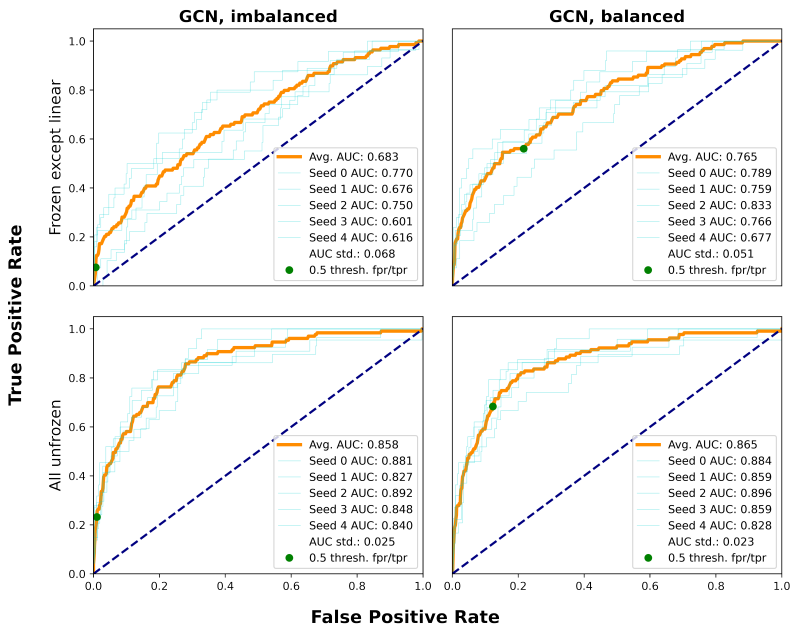


Figure 5.3: ROC & AUC for GCNs pre-trained on UKBB sex prediction task, before fine tuning and prediction on OASIS (original/imbalanced and balanced). This was performed for two settings: first, fine tuning linear layer only; second, leaving all layers to be fine tuned.

However, with the **unfrozen** models (Table 5.7) a different story emerges. Though marginal, the unfrozen pre-trained models ROC-AUC outperform their purely OASIS trained counterparts, and they obtain comparable MCCs, for both balanced and imbalanced OASIS training. Focussing on the pre-trained model with balanced OASIS training, we see outperformance on six of eight metrics compared to its counterpart trained exclusively on OASIS. Here pre-training appears to have pushed the model into an improved basin of attraction (Section 5.6.1) meaning that the fine-tuning process leads to improved classifier performance.

Finally, we must note the impact of the 0.5 decision threshold used by default. As found on the general AD classification task (Section 5.4.2), the models trained on balanced OASIS data are closer to their optimal decision threshold (see green dots in Figure 5.3), meaning that the metrics reported here are a fairer reflection of their ability. In contrast, the models trained on the original OASIS (imbalanced) data diverge substantially from their optimal point, and as such their performance may be an underestimate.

One minor evaluation point relates to the pre-processing pipelines performed on UKBB and OASIS brain meshes. As noted in 3.1.2, whilst the pipelines are similar they are not identical, meaning that the meshes extracted from the two datasets will have minor but systematic differences. Whilst we cannot quantify the impact of this on the transferability of the pre-trained models, further work might attempt to explore this impact or further refine the pipelines to increase similarity.

Further work might also explore the impact of pre-training on all model types outlined in this report. Of particular interest would be the benefits to the Spline model. As noted in this report, the variance of performance for different seeds for Spline is larger than that of other models. Given that pre-training might help to push Spline into more stable bases of attraction, we hypothesise that Spline might benefit.

Chapter 6

Conclusion

This report provides two contributions to the graph-based learning literature. In Chapter 4, we demonstrated that our novel *Curriculum By OverSmoothing* provides a way to improve model performance without additional parameters. The key issue raised for further work was understanding the settings in which this is beneficial, and those in which it is not. In Chapter 5, we use our proposed multi-structure GNNs for Alzheimer's Disease classification, demonstrating that our models outperform a PCA baseline. A key issue proposed for further research was the optimisation of decision thresholds.

Both contributions are self contained: their respective chapters include discussions of their results and recommendations for further work, and we do not repeat those discussions here. Instead we focus on a higher level consideration: ethical issues in graph-based learning.

6.1 Ethical considerations for graph-based learning

After introducing issues within 'AI Ethics' [76] that are relevant to the whole AI community, we outline our argument for why graph-based learning might face these issues more acutely. We aim to provoke thought in this concluding section, with the hope of stirring further research, rather than provide an in depth analysis.

6.1.1 Existing issues

The productivity gains driven by AI [127] have led to large appetite for its widespread adoption, both from businesses and governments. In the medical setting, AI provides clinical support ranging from cardiovascular risk screening [70] to recommending personalised radiation treatment plans for cancer patients [149]. More broadly, AI has impacted content provision on social networks, firms' hiring, and countless other areas [62]. However, with this widespread use has come problems of unfairness, bias and polarisation. We focus on two issues that might impact graph-based learning acutely.

A first issue is the role AI has played in political polarisation. User engagement is key for social networks, and providing content with narrow viewpoints matched to a user's preferences (i.e. an echo chamber) appears to be a successful strategy [28]. This has contributed to increasing polarisation of political views and communities on sites including Twitter [44], Facebook and YouTube [28].

A second issue is that of algorithmic fairness [124], which seeks to prevent AI from acting in an unfair or discriminative way[1]. AI-based discrimination has presented in diverse ways,

[1]See the Oxford handbook on AI ethics for more examples [62], or [36] for an overview in medical settings.

from Amazon's automated hiring tool that was negatively biased against women [82], to a U.S. healthcare algorithm that required black patients to be substantially more sick than white patients before it recommended extra care [118].

6.1.2 Potential exacerbation by graphs

There are two potential reasons these issues are of particular relevance to the graph-based learning community: first, graphs reintroduce human decisions in the *structuring* of data. Second, message-passing and CBOS might suppress minority groups.

A moral responsibility in structuring graph data

Existing biases in AI are often unintentional, and arise from systemic inequalities in the real world that are then reflected in data. For example, the Amazon hiring tool rated candidates based on their resemblance to prior successful candidates [82]. A biased institutional hiring procedure leads to 'successful candidates' being predominantly men, and so the model learns to discriminate *without* inherent sexist design of the model itself. Equally, social networks are promoting content that users enjoy, but creating echo chambers (e.g. a user likes a post from a certain political group, meaning other posts similar to that are recommended). Here, the biases are in the data naturally; they arise from real world processes.

Data engineers structuring graphs, however, can *choose* how the graph is structured and so have moral choices to make. Let us take the example of a recommender system on a social network, that recommends users content that is 'liked' by those close to them in the user graph. If we use edges to encode political similarity, a recommender system will create an echo chamber by providing content that aligns with a user's political views. However, if instead edges encode *dissimilarity*, and so stronger weights represent more different views, we could generate a network which recommends pieces written from alternative view points.

Although this is a simple example, it clearly demonstrates that because humans must structure the graph, and encode the relations, there is more opportunity to shape models that learn off graphs. This introduces morality into the manual design of graphs.

Smoothing out minorities

Both GNN Message passing and CBOS consist of information being shared between nodes. While this leads to more complex aggregations of data, it also smooths out 'unique' signals in the graph (as in *oversmoothing*). When these unique signals are noise, that is a positive. However, when unique signals are the signals of a minority group that differ, this leads to a model that works less well for minorities.

For example, let us imagine trying to classify a skin disease from images, with a population graph containing (subject) nodes with imaging features. Let us also assume the imaging features that differentiate healthy vs unhealthy are expressed differently by race (a current issue in the heavily white focussed U.K. dermatological curriculum [33]). Finally, let us assume edges only encode similarity of age and sex. If the vast majority of subjects are white, message passing pushes all feature vectors to be closer to the average (white) representation. With *CBOS*, this interpretation is even more accurate. This is beneficial for overall accuracy, but while the model is accurate on average it likely fails to learn how to classify for black subjects.

Combining the two concerns (graph structure and smoothing out minorities) we see they interplay. If the population graph is structured so edges encode similarity by race, with distinct components by race, we can maintain distinct features: black subjects will only pass messages with each other, preventing their 'smoothing out'. The model can then learn to classify differently for different representations or clusters.

Many forms of AI face these risks, but they are potentially exacerbated in graph-based learning. Both practitioners and researchers of graph-based learning must keep these ethical considerations top of mind.

Bibliography

[1] (2021). Adni dataset. http://adni.loni.usc.edu. Accessed: 2021-08-09.

[2] (2021). Amazon dataset. https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/datasets/amazon.html#Amazon. Accessed: 2021-07-01.

[3] (2021). Batch normalisation pytorch geometric. https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/nn/norm/batch_norm.html. Accessed: 2021-07-02.

[4] (2021). Deep learning's carbon emissions problem. https://www.forbes.com/sites/robtoews/2020/06/17/deep-learnings-climate-change-problem/?sh=21643c606b43. Accessed: 2021-07-02.

[5] (2021). Magnetic resonance imaging (mri) of the brain and spine: Basics. https://case.edu/med/neurology/NR/MRI%20Basics.html. Accessed: 2021-07-31.

[6] (2021). Mnist superpixel dataset. https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/datasets/mnist_superpixels.html. Accessed: 2021-07-01.

[7] (2021). National institute of aging: Alzheimer's disease diagnostic guidelines. https://www.nia.nih.gov/health/alzheimers-disease-diagnostic-guideline. Accessed: 2021-07-01.

[8] (2021). New alzheimer's association report examines racial and ethnic attitudes on alzheimer's and dementia care. https://www.alz.org/news/2021/new-alzheimers-association-report-examines-racial. Accessed: 2021-07-02.

[9] (2021). Nhs.co.uk, alzheimer's disease. https://www.nhs.uk/conditions/alzheimers-disease/. Accessed: 2021-07-02.

[10] (2021a). Openai's gpt-3 language model: A technical overview. https://lambdalabs.com/blog/demystifying-gpt-3/#1. Accessed: 2021-07-02.

[11] (2021b). Openai's massive gpt-3 model is impressive, but size isn't everything. https://venturebeat.com/2020/06/01/ai-machine-learning-openai-gpt-3-size-isnt-everything/. Accessed: 2021-07-02.

[12] (2021). Weighted random sampler. https://pytorch.org/docs/stable/_modules/torch/utils/data/sampler.html#WeightedRandomSampler. Accessed: 2021-08-01.

[13] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2010). Slic superpixels.

[14] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282.

[15] Achanta, R. and Susstrunk, S. (2017). Superpixels and polygons using simple non-iterative clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4651–4660.

[16] Ahmed, S. M., Das, N. R., and Chaudhury, K. N. (2019). Least-squares registration of point sets over se (d) using closed-form projections. *Computer Vision and Image Understanding*, 183:20–32.

[17] Akosa, J. (2017). Predictive accuracy: A misleading performance measure for highly imbalanced data. In *Proceedings of the SAS Global Forum*, volume 12.

[18] Alfaro-Almagro, F., Jenkinson, M., Bangerter, N. K., Andersson, J. L., Griffanti, L., Douaud, G., Sotiropoulos, S. N., Jbabdi, S., Hernandez-Fernandez, M., Vallee, E., Vidaurre, D., Webster, M., McCarthy, P., Rorden, C., Daducci, A., Alexander, D. C., Zhang, H., Dragonu, I., Matthews, P. M., Miller, K. L., and Smith, S. M. (2018). Image processing and quality control for the first 10,000 brain imaging datasets from uk biobank. *NeuroImage*, 166:400–424.

[19] Altay, F., Sanchez, G. R., James, Y., Faraone, S. V., Velipasalar, S., and Salekin, A. (2020). Preclinical stage alzheimer's disease detection using magnetic resonance image scans. *arXiv preprint arXiv:2011.14139*.

[20] Avelar, P. H. C., Tavares, A. R., da Silveira, T. L. T., Jung, C. R., and Lamb, L. C. (2020). Superpixel image classification with graph attention networks.

[21] Bach, F. (2017). Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681.

[22] Baldi, P., Brunak, S., Chauvin, Y., Andersen, C. A., and Nielsen, H. (2000). Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16(5):412–424.

[23] Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945.

[24] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks.

[25] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer.

[26] Bekkar, M., Djemaa, H. K., and Alitouche, T. A. (2013). Evaluation measures for models assessment over imbalanced data sets. *J Inf Eng Appl*, 3(10).

[27] Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

[28] Bessi, A., Zollo, F., Del Vicario, M., Puliga, M., Scala, A., Caldarelli, G., Uzzi, B., and Quattrociocchi, W. (2016). Users polarization on facebook and youtube. *PloS one*, 11(8):e0159641.

[29] Bircanoğlu, C. and Arıca, N. (2018). A comparison of activation functions in artificial neural networks. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE.

[30] Boscaini, D., Masci, J., Rodolà, E., and Bronstein, M. M. (2016). Learning shape correspondence with anisotropic convolutional neural networks.

[31] Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118.

[32] Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges.

[33] Brooks, K. C. (2015). A silent curriculum. *Jama*, 313(19):1909–1910.

[34] Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric foundations of deep learning. Accessed: 2021-05-19.

[35] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

[36] Challen, R., Denny, J., Pitt, M., Gompels, L., Edwards, T., and Tsaneva-Atanasova, K. (2019). Artificial intelligence, bias and clinical safety. *BMJ Quality & Safety*, 28(3):231–237.

[37] Cheng, D., Liu, M., Fu, J., and Wang, Y. (2017). Classification of mr brain images by combination of multi-cnns for ad diagnosis. In *Ninth international conference on digital image processing (ICDIP 2017)*, volume 10420, page 1042042. International Society for Optics and Photonics.

[38] Chernyaev, E. (1995). Marching cubes 33: Construction of topologically correct isosurfaces. Technical report.

[39] Chicco, D. and Jurman, G. (2020). The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13.

[40] Chomsky, N. (2014). *Aspects of the Theory of Syntax*, volume 11. MIT press.

[41] Chung, F. R. and Graham, F. C. (1997). *Spectral graph theory*. Number 92. American Mathematical Soc.

[42] Ciresan, D., Giusti, A., Gambardella, L., and Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. *Advances in neural information processing systems*, 25:2843–2851.

[43] Colyer, A. (2018). Relational inductive biases, deep learning, and graph networks. Accessed: 2021-05-23.

[44] Conover, M. D., Ratkiewicz, J., Francisco, M., Gonçalves, B., Menczer, F., and Flammini, A. (2011). Political polarization on twitter. In *Fifth international AAAI conference on weblogs and social media*.

[45] Craik, K. J. W. (1943). The nature of explanation.

[46] Cuingnet, R., Gerardin, E., Tessieras, J., Auzias, G., Lehéricy, S., Habert, M.-O., Chupin, M., Benali, H., Colliot, O., Initiative, A. D. N., et al. (2011). Automatic classification of patients with alzheimer's disease from structural mri: a comparison of ten methods using the adni database. *neuroimage*, 56(2):766–781.

[47] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

[48] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee.

[49] Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*.

[50] Derényi, I., Geszti, T., and Györgyi, G. (1994). Generalization in the programed teaching of a perceptron. *Physical Review E*, 50(4):3192.

[51] Donnat, C., Zitnik, M., Hallac, D., and Leskovec, J. (2018). Spectral graph wavelets for structural role similarity in networks.

[52] Downs, T. and Gaynier, R. (1995). The use of random weights for the training of multilayer networks of neurons with heaviside characteristics. *Mathematical and Computer Modelling*, 22(10):53–61.

[53] Driss, S. B., Soua, M., Kachouri, R., and Akil, M. (2017). A comparison study between mlp and convolutional neural network models for character recognition. In *Real-Time Image and Video Processing 2017*, volume 10223, page 1022306. International Society for Optics and Photonics.

[54] Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292*.

[55] Erhan, D., Manzagol, P.-A., Bengio, Y., Bengio, S., and Vincent, P. (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Artificial Intelligence and Statistics*, pages 153–160. PMLR.

[56] Feltell, D. and Bai, L. (2010). A new marching cubes algorithm for interactive level set with application to mr image segmentation. In *International Symposium on Visual Computing*, pages 371–380. Springer.

[57] Fernández, M., Gobartt, A. L., and Balañá, M. (2010). Behavioural symptoms in patients with alzheimer's disease and their association with cognitive impairment. *BMC neurology*, 10(1):1–9.

[58] Fey, M., Lenssen, J. E., Weichert, F., and Müller, H. (2018). Splinecnn: Fast geometric deep learning with continuous b-spline kernels.

[59] Fodor, J. A. and Lepore, E. (2002). *The compositionality papers*. Oxford University Press.

[60] Fodor, J. A. and Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

[61] Fukushima, K. and Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer.

[62] Gebru, T. (2020). Race and gender. *The Oxford handbook of ethics of aI*, pages 251–269.

[63] Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58.

[64] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR.

[65] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[66] Gregory, R. L. (1983). Forty years on: Kenneth craik's the nature of explanation (1943).

[67] Hamilton, W. L. (2020). Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159.

[68] Hardt, M. and Recht, B. (2021). *Patterns, predictions, and actions: A story about machine learning*. `https://mlstory.org`.

[69] Hinton, G. e. a. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*.

[70] Hippisley-Cox, J., Coupland, C., Vinogradova, Y., Robson, J., Minhas, R., Sheikh, A., and Brindle, P. (2008). Predicting cardiovascular risk in england and wales: prospective derivation and validation of qrisk2. *Bmj*, 336(7659):1475–1482.

[71] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

[72] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417.

[73] Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154.

[74] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*.

[75] Jastrzebski, S., Szymczak, M., Fort, S., Arpit, D., Tabor, J., Cho, K., and Geras, K. (2020). The break-even point on optimization trajectories of deep neural networks. *CoRR*, abs/2002.09572.

[76] Jobin, A., Ienca, M., and Vayena, E. (2019). The global landscape of ai ethics guidelines. *Nature Machine Intelligence*, 1(9):389–399.

[77] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196.

[78] Kawahara, J., Brown, C. J., Miller, S. P., Booth, B. G., Chau, V., Grunau, R. E., Zwicker, J. G., and Hamarneh, G. (2017). Brainnetcnn: Convolutional neural networks for brain networks; towards predicting neurodevelopment. *NeuroImage*, 146:1038–1049.

[79] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[80] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[81] Knyazev, B., Lin, X., Amer, M. R., and Taylor, G. W. (2018). Spectral multigraph networks for discovering and fusing relationships in molecules. *arXiv preprint arXiv:1811.09595*.

[82] Kodiyan, A. A. (2019). An overview of ethical issues in using ai systems in hiring with a case study of amazon's ai based hiring tool.

[83] Kokkinos, I., Bronstein, M. M., Litman, R., and Bronstein, A. M. (2012). Intrinsic shape context descriptors for deformable shapes. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 159–166. IEEE.

[84] Korolev, S., Safiullin, A., Belyaev, M., and Dodonova, Y. (2017). Residual and plain convolutional neural networks for 3d brain mri classification. In *2017 IEEE 14th international symposium on biomedical imaging (ISBI 2017)*, pages 835–838. IEEE.

[85] Kramarow, E. A. and Tejada-Vera, B. (2019). Dementia mortality in the united states, 2000-2017. *National Vital Statistics Reports: From the Centers for Disease Control and Prevention, National Center for Health Statistics, National Vital Statistics System*, 68(2):1–29.

[86] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

[87] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012b). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

[88] Krueger, K. A. and Dayan, P. (2009). Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394.

[89] Lake, B. and Baroni, M. (2018). Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks.

[90] Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and brain sciences*, 40.

[91] LaMontagne, P. J., Benzinger, T. L., Morris, J. C., Keefe, S., Hornbeck, R., Xiong, C., Grant, E., Hassenstab, J., Moulder, K., Vlassenko, A. G., Raichle, M. E., Cruchaga, C., and Marcus, D. (2019). Oasis-3: Longitudinal neuroimaging, clinical, and cognitive dataset for normal aging and alzheimer disease. *medRxiv*.

[92] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[93] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[94] Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867.

[95] Levie, R., Huang, W., Bucci, L., Bronstein, M. M., and Kutyniok, G. (2020). Transferability of spectral graph convolutional neural networks.

[96] Lewiner, T., Lopes, H., Vieira, A. W., and Tavares, G. (2003). Efficient implementation of marching cubes' cases with topological guarantees. *Journal of graphics tools*, 8(2):1–15.

[97] Liem, F., Dadi, K., Engemann, D. A., Gramfort, A., Bellec, P., Craddock, R. C., Damoiseaux, J. S., Steele, C. J., Yarkoni, T., Margulies, D. S., et al. (2020). Predicting future cognitive decline from non-brain and multimodal brain imaging data in healthy and pathological aging. *bioRxiv*.

[98] Liew, S. S., Hani, M. K., Radzi, S. A., and Bakhteri, R. (2016). Gender classification: a convolutional neural network approach. *Turkish Journal of Electrical Engineering & Computer Sciences*, 24(3):1248–1264.

[99] Logsdon, R. G., Gibbons, L. E., McCurry, S. M., Teri, L., et al. (1999). Quality of life in alzheimer's disease: patient and caregiver reports. *Journal of Mental health and Aging*, 5:21–32.

[100] Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169.

[101] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[102] Lu, L., Shin, Y., Su, Y., and Karniadakis, G. E. (2019). Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*.

[103] Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer.

[104] Makarov, I., Makarov, M., and Kiselev, D. (2021). Fusion of text and graph information for machine learning problems on networks. *PeerJ Computer Science*.

[105] Manzano, S., González, J., Marcos, A., Payno, M., Villanueva, C., and Matias-Guiu, J. (2009). Experimental models in alzheimer's disease. *Neurologia (Barcelona, Spain)*, 24(4):255–262.

[106] Masci, J., Boscaini, D., Bronstein, M. M., and Vandergheynst, P. (2018). Geodesic convolutional neural networks on riemannian manifolds.

[107] Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

[108] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[109] McPherson, M., Smith-Lovin, L., and Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444.

[110] Miller, K. L., Alfaro-Almagro, F., Bangerter, N. K., Thomas, D. L., Yacoub, E., Xu, J., Bartsch, A. J., Jbabdi, S., Sotiropoulos, S. N., Andersson, J. L., et al. (2016). Multimodal population brain imaging in the uk biobank prospective epidemiological study. *Nature neuroscience*, 19(11):1523–1536.

[111] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

[112] Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. M. (2016). Geometric deep learning on graphs and manifolds using mixture model cnns.

[113] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609.

[114] Morris, J. C. (1991). The clinical dementia rating (cdr): Current version and. *Young*, 41:1588–1592.

[115] Mu, Y. and Gage, F. H. (2011). Adult hippocampal neurogenesis and its role in alzheimer's disease. *Molecular neurodegeneration*, 6(1):1–9.

[116] Murphy, R. L., Srinivasan, B., Rao, V., and Ribeiro, B. (2018). Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *arXiv preprint arXiv:1811.01900*.

[117] Northcutt, C. (2021). Towards reproducibility: Benchmarking keras and pytorch.

[118] Obermeyer, Z., Powers, B., Vogeli, C., and Mullainathan, S. (2019). Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453.

[119] Parasuraman, R. and Manzey, D. H. (2010). Complacency and bias in human use of automation: An attentional integration. *Human factors*, 52(3):381–410.

[120] Parisot, S., Ktena, S. I., Ferrante, E., Lee, M., Guerrero, R., Glocker, B., and Rueckert, D. (2018). Disease prediction using graph convolutional networks: application to autism spectrum disorder and alzheimer's disease. *Medical image analysis*, 48:117–130.

[121] Parisot, S., Ktena, S. I., Ferrante, E., Lee, M., Moreno, R. G., Glocker, B., and Rueckert, D. (2017). Spectral graph convolutions for population-based disease prediction. In *International conference on medical image computing and computer-assisted intervention*, pages 177–185. Springer.

[122] Patenaude, B., Smith, S. M., Kennedy, D. N., and Jenkinson, M. (2011). A bayesian model of shape and appearance for subcortical brain segmentation. *Neuroimage*, 56(3):907–922.

[123] Patenaude, B., Smith, S. M., Kennedy, D. N., and Jenkinson, M. (2021). First model-based segmentation/registration tool. Accessed: 2021-05-26.

[124] Pessach, D. and Shmueli, E. (2020). Algorithmic fairness. *arXiv preprint arXiv:2001.09784*.

[125] Piegl, L. and Tiller, W. (1996). *The NURBS book*. Springer Science & Business Media.

[126] Powers, D. (2011). Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. j mach learn technol. 2011; 2: 37–63.

[127] Rao, A. S. and Verweij, G. (2017). Sizing the prize: What's the real value of ai for your business and how can you capitalise. *PwC Publication, PwC*, pages 1–30.

[128] Rathi, Y., Dambreville, S., and Tannenbaum, A. (2006). Statistical shape analysis using kernel pca. In *Image processing: algorithms and systems, neural networks, and machine learning*, volume 6064, page 60641B. International Society for Optics and Photonics.

[129] Reiman, E. M., Quiroz, Y. T., Fleisher, A. S., Chen, K., Velez-Pardo, C., Jimenez-Del-Rio, M., Fagan, A. M., Shah, A. R., Alvarez, S., Arbelaez, A., et al. (2012). Brain imaging and fluid biomarker analysis in young adults at genetic risk for autosomal dominant alzheimer's disease in the presenilin 1 e280a kindred: a case-control study. *The Lancet Neurology*, 11(12):1048–1056.

[130] Ren, X. and Malik, J. (2003). Learning a classification model for segmentation. In *Computer Vision, IEEE International Conference on*, volume 2, pages 10–10. IEEE Computer Society.

[131] Rice, D. P., Fillit, H. M., Max, W., Knopman, D. S., Lloyd, J. R., Duttagupta, S., et al. (2001). Prevalence, costs, and treatment of alzheimer's disease and related dementia: a managed care perspective. *American Journal of Managed Care*, 7(8):809–820.

[132] Rosenblatt, F. (1957). "the perceptron: A perceiving and recognizing automaton".

[133] Russell, S. J. and Norvig, P. (2009). *Artificial Intelligence: a modern approach*. Pearson, 3 edition.

[134] Samuel, A. L. (2000). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 44(1.2):206–226.

[135] Sellars, P., Aviles-Rivero, A. I., and Schönlieb, C.-B. (2020). Superpixel contracted graph-based learning for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 58(6):4180–4193.

[136] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3):93–93.

[137] Shi, F., Yap, P.-T., Wu, G., Jia, H., Gilmore, J. H., Lin, W., and Shen, D. (2011). Infant brain atlases from neonates to 1-and 2-year-olds. *PloS one*, 6(4):e18746.

[138] Shi, L., Campbell, G., Jones, W., Campagne, F., Wen, Z., Walker, S., Su, Z., Chu, T., Goodsaid, F., Pusztai, L., et al. (2010). The maqc-ii project: a comprehensive study of common practices for the development and validation of microarray-based predictive models.

[139] Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98.

[140] Sinha, S., Garg, A., and Larochelle, H. (2020). Curriculum by smoothing. *Advances in Neural Information Processing Systems*, 33.

[141] Song, X., Zhou, F., Frangi, A. F., Cao, J., Xiao, X., Lei, Y., Wang, T., and Lei, B. (2021). Graph convolution network with similarity awareness and adaptive calibration for disease-induced deterioration prediction. *Medical Image Analysis*, 69:101947.

[142] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and R., S. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*.

[143] Stegmann, M. B. and Gomez, D. D. (2002). A brief introduction to statistical shape analysis. *Informatics and mathematical modelling, Technical University of Denmark, DTU*, 15(11).

[144] Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. *arXiv:1906.02243*.

[145] Stutz, D., Hermans, A., and Leibe, B. (2018). Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27.

[146] Stutzmann, G. E. (2007). The pathogenesis of alzheimers disease—is it a lifelong "calciumopathy"? *The Neuroscientist*, 13(5):546–559.

[147] Sudlow, C., Gallacher, J., Allen, N., Beral, V., Burton, P., Danesh, J., Downey, P., Elliott, P., Green, J., Landray, M., et al. (2015). Uk biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *Plos med*, 12(3):e1001779.

[148] Sun, L., Yu, K., and Batmanghelich, K. (2020). Context matters: Graph-based self-supervised representation learning for medical images. *arXiv preprint arXiv:2012.06457*.

[149] Thompson, R. F., Valdes, G., Fuller, C. D., Carpenter, C. M., Morin, O., Aneja, S., Lindsay, W. D., Aerts, H. J., Agrimson, B., Deville Jr, C., et al. (2018). Artificial intelligence in radiation oncology: a specialty-wide disruptive transformation? *Radiotherapy and Oncology*, 129(3):421–426.

[150] Tong, T., Gao, Q., Guerrero, R., Ledig, C., Chen, L., Rueckert, D., Initiative, A. D. N., et al. (2016). A novel grading biomarker for the prediction of conversion from mild cognitive impairment to alzheimer's disease. *IEEE Transactions on Biomedical Engineering*, 64(1):155–165.

[151] Tong, T., Gray, K., Gao, Q., Chen, L., Rueckert, D., Initiative, A. D. N., et al. (2017). Multi-modal classification of alzheimer's disease using nonlinear graph fusion. *Pattern recognition*, 63:171–181.

[152] Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380.

[153] van den Berg, R., Kipf, T. N., and Welling, M. (2017). Graph convolutional matrix completion.

[154] Van den Bergh, M., Boix, X., Roig, G., de Capitani, B., and Van Gool, L. (2012). Seeds: Superpixels extracted via energy-driven sampling. In *European conference on computer vision*, pages 13–26. Springer.

[155] Veličković, P. (2021). Theoretical foundations of graph neural networks. Accessed: 2021-05-19.

[156] Wang, S., Khabsa, M., and Ma, H. (2020). To pretrain or not to pretrain: Examining the benefits of pretraining on resource rich tasks. *arXiv preprint arXiv:2006.08671*.

[157] Weisfeiler, B. and Leman, A. (1968). The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16.

[158] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018a). How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

[159] Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. (2018b). Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR.

[160] Yang, Z., Cohen, W., and Salakhudinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR.

[161] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983.

[162] Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328.

[163] Zitnik, M., Agrawal, M., and Leskovec, J. (2018). Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466.

[164] Zott, B., Busche, M. A., Sperling, R. A., and Konnerth, A. (2018). What happens with the circuit in alzheimer's disease in mice and humans? *Annual review of neuroscience*, 41:277–297.

Appendices

Appendix A

Method's name	Sensitivity	Specificity
Voxel-Direct-D-gm	81%	95%
Voxel-Direct-D-all	68%	98%
Voxel-Direct-S-gm	72%	89%
Voxel-Direct-S-all	65%	88%
Voxel-Direct_VOI-D-gm	71%	95%
Voxel-Direct_VOI-D-all	65%	95%
Voxel-Direct_VOI-S-gm	65%	91%
Voxel-Direct_VOI-S-all	59%	81%
Voxel-STAND-D-gm	69%	90%
Voxel-STAND-D-all	71%	91%
Voxel-STAND-S-gm	75%	91%
Voxel-STAND-S-all	75%	86%
Voxel-STAND-Sc-gm	72%	91%
Voxel-STAND-Sc-all	71%	91%
Voxel-Atlas-D-gm	78%	93%
Voxel-Atlas-D-all	81%	90%
Voxel-Atlas-S-gm	75%	93%
Voxel-Atlas-S-all	74%	93%
Voxel-COMPARE-D-gm	82%	89%
Voxel-COMPARE-D-all	69%	81%
Voxel-COMPARE-S-gm	66%	86%
Voxel-COMPARE-S-all	72%	91%
Thickness-Direct	74%	90%
Thickness-Atlas	79%	90%
Thickness-ROI	69%	94%
Hippo-Volume-F	63%	80%
Hippo-Volume-S	71%	77%

Table A.1: Results from AD vs Cognitively Normal binary classification tasks on ADNI dataset; reproduced from [46]. The models each use different imaging features for the classification task.

Appendix B

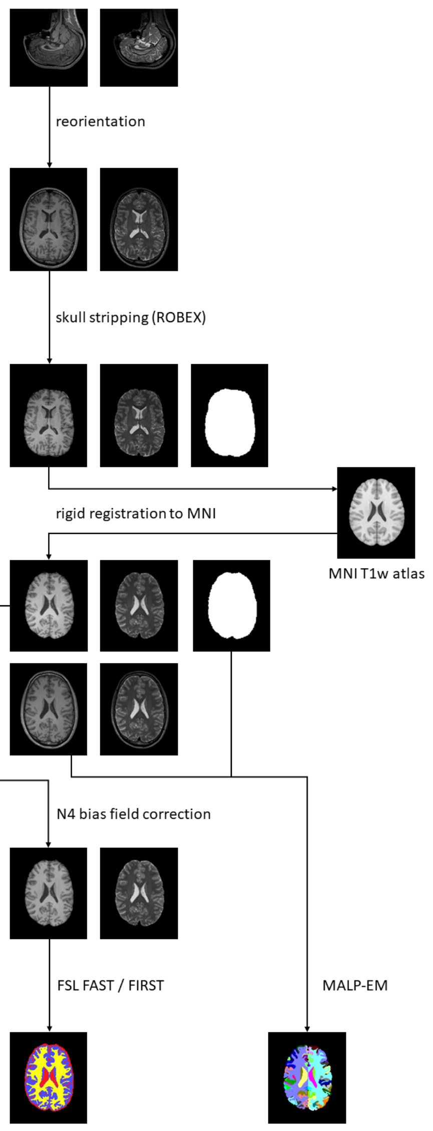


Figure B.1: Pre-processing pipeline for meshes extracted from OASIS dataset. This was designed to closely parallel the process used on UKBB. Provided by Dr. Ben Glocker