Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Reproducible Knowledge Distillation for Graph Neural Networks

Author:
Lorenzo G. Stigliano

Supervisor:
Dr. Islem Rekik

Submitted in partial fulfillment of the requirements for the MSc degree in Computing Science / Artificial Intelligence & Machine Learning of Imperial College London

September 2023

Abstract

Graph Neural Networks (GNNs) have emerged as a powerful tool for modelling graph-structured data in various domains, such as network neuroscience. However, the scalability of GNNs becomes a challenge when dealing with large graphs, hindering real-time inference and deployment on resource-limited devices. Knowledge distillation on graphs (KDG) offers a solution by compressing large GNN models while maintaining performance. However, existing KDG methods often focus on performance (e.g., accuracy) while neglecting reproducibility, which is crucial for the trustworthiness of these models. Reproducibility is the identification of consistent biomarkers or features under different perturbation methodologies. Although several studies have investigated biomarker reproducibility, reproducibility in the context of KDG has not been explored.

Thus, in this project, we aim to bridge the three domains; GNNs, knowledge distillation, and reproducibility into a novel domain that we term *reproducible offline knowledge distillation for GNN* concerned with the reproducibility of distilled models. We begin by motivating research into this domain by showing that with several different knowledge distillation (KD) and KDG techniques, there is generally a degradation in the reproducibility of the distilled models.

In light of this, we propose a novel algorithm *Reproducibility aware Knowledge Distillation on Graphs* (RepKD) which aims to retain and increase reproducibility while conserving the performance of distilled models. RepKD uses a two-step process. Firstly, a one-to-many teacher-student framework is used to train the student models collectively, and secondly, the selection process selects the best student from the ensemble. We tested our method under a range of different datasets and GNN architectures while comparing it to the state-of-the-art (SOTA) KD and KDG methods, and we saw that our method managed to successfully increase the self-reproducibility score while preserving the performance of the student models. Furthermore, we showed that during the training process, our method exhibited comparable training times and incurred negligible memory overhead in comparison to other SOTA methods. More impressively, these results were obtained while decreasing the number of parameters by at least 95.63%. Finally, we also delved into the interpretability of the distilled models, attempting to explain and understand them.

We hope that our proposed method and research can act as a pivotal starting point for future exploration into reproducible knowledge distillation for GNNs.

Acknowledgements

Firstly, I would like to thank my supervisor, Dr. Islem Rekik, for her help throughout this project. Their advice during weekly meetings guided me through the expansive field of graph neural networks and reproducibility. Secondly, I would like to thank my personal tutor, Prof. Abbas Edalat, for his unwavering support throughout the project.

Lastly, I would like to thank my friends and family for their unconditional support throughout this year. To my loving father, brother, and Sarah, thank you for always being there for me. To my mother: thank you for everything; I would not be here without you.

Contents

1 Introduction **3**
- 1.1 Contributions . . . 4
- 1.2 Ethical Considerations . . . 5

2 Background **6**
- 2.1 Graph Neural Networks . . . 6
  - 2.1.1 Graphs . . . 6
  - 2.1.2 Graph Neural Networks . . . 7
  - 2.1.3 Neighbour Explosion Problem . . . 8
- 2.2 Knowledge Distillation on Graphs . . . 9
  - 2.2.1 General Framework . . . 9
  - 2.2.2 Knowledge . . . 10
  - 2.2.3 Distillation Schemes . . . 12
  - 2.2.4 Teacher-Student Frameworks . . . 12
- 2.3 Reproducibility . . . 13
- 2.4 Summary . . . 13

3 Related Work **14**
- 3.1 Knowledge Distillation . . . 14
- 3.2 Interpretable Artificial Intelligence . . . 16
- 3.3 Biomarker Reproducibility Methods . . . 17
- 3.4 Stable Feature Selection . . . 18
- 3.5 Summary . . . 19

4 Motivation **20**
- 4.1 Problem . . . 20
- 4.2 Quantifying Reproducibility . . . 21
- 4.3 Knowledge Distillation & Reproducibility . . . 22
  - 4.3.1 Experimental Setup . . . 22
  - 4.3.2 Analysis . . . 24
- 4.4 Summary . . . 25

5 Method **26**
- 5.1 Problem Formulation . . . 26
- 5.2 Proposed Method . . . 26
  - 5.2.1 Reproducibility aware Knowledge Distillation . . . 27

5.2.2 Selection Process . . . 31
5.3 Implementation . . . 31
5.4 Summary . . . 31

6 Results & Discussion **33**
6.1 Experimental Setup . . . 33
6.1.1 Datasets . . . 33
6.1.2 Baselines . . . 34
6.1.3 Teacher and Student Backbones . . . 35
6.1.4 Shared Hyperparameters . . . 35
6.1.5 Training . . . 36
6.2 Performance . . . 36
6.2.1 Intra-Model Performance . . . 36
6.2.2 Cross-Model Performance . . . 39
6.3 Ablation Study . . . 41
6.3.1 Teacher-Student Loss Parameter . . . 41
6.3.2 Intra-Student Loss Parameter . . . 42
6.3.3 Ensemble Losses . . . 42
6.3.4 Selection Process . . . 43
6.4 Parameter Sensitivity Study . . . 45
6.4.1 Teacher-Student Loss Parameter . . . 45
6.4.2 Intra-Student Loss Parameter . . . 45
6.4.3 Soft Target Temperature . . . 46
6.4.4 Ensemble Size . . . 47
6.5 Model Efficiency . . . 47
6.5.1 Training Efficiency . . . 48
6.5.2 Inference Efficiency . . . 49
6.6 Clinical Interpretability . . . 50
6.7 Summary . . . 51

7 Conclusion **52**
7.1 Future Work . . . 53
7.2 Final Remarks . . . 53

A Parameter Configuration **62**

B Supplementary Results **64**

C Additional Information **71**

Chapter 1

Introduction

Graph-structured data is highly prevalent across many domains, owing its wide-ranging presence to its ability to model complex systems. For instance, in network neuroscience, brain structures are modelled as graphs where nodes represent the anatomical brain regions and edges quantify their interaction [1]. Similarly, graphs are widespread in the natural sciences, as exemplified by their use in representing protein structures [2] in biology and molecular structures [3] in chemistry.

Motivated to understand and leverage graph structures, one would naively look towards deep learning (DL) methods. However, DL techniques fail to generalise well to non-Euclidean data (e.g., graphs) [4]. As a result, a new branch of DL was developed to handle non-Euclidean data, Graph Neural Networks (GNNs) [5, 6, 7, 8, 9, 10, 11]. GNNs have had great success in domains such as recommendation systems [12], traffic networks [13], natural language processing [14] and network neuroscience, where they have enabled significant advances in the study of brain connectivity [15, 16].

However, despite the successes of GNNs, they fail to scale when dealing with large graphs for both real-time inference [17] and deployment to resource-limited devices [18]. GNNs make use of a *message passing scheme*, which requires to aggregate all neighbouring node embeddings at each layer of the GNN. In turn, scalability and deployment becomes a challenge with this scheme, known as the *neighbour explosion problem* [11], which increases both the inference time and size of the deployed GNN model.

As a result, different methods have been proposed to scale GNNs. For example, sampling-based methods [7, 9, 19], are used to speed up training and inference by only considering a subset of the nodes of the graph. Despite solving the neighbour explosion problem, these models still remain large, which poses a problem for deployment with limited resources. Algorithmic-based solutions focus on reducing the size of the GNN model, such as pruning [20], quantization [21, 22] or knowledge distillation (KD).

In particular, KD aims to distil the performance of a large model (teacher) into a small, parameter-efficient model (student) [18, 23], in turn allows for quicker inference and resource-friendly deployability. This idea has been naturally applied to graphs and is known as Knowledge Distillation on Graphs (KDG) [18]. These methods have had large success in compressing models while preserving performance [24, 25, 26, 27]. However, KD and KDG methods prioritise performance, while the reproducibility of the distilled models is often ignored. However, this is a crucial factor to consider during knowledge transfer. In fact, as we shall see in Chapter 3, there are no KD or KDG

methods directly concerned with the reproducibility of the distilled models, raising the question, *can distilled models be trusted?*

Reproducibility is concerned with the identification of the same distinctive biomarkers or features when the data undergoes different perturbation methodologies (e.g. pertubing the distribution of training and testing data) [1]. This is of particular importance in the clinical setting, since a highly reproducible model leads to an increase in its trustworthiness and reliability [28]. Furthermore, reproducibility is closely linked to interpretability. In clinical assessments, interpreting which set of consistent biomarkers cause a specific brain disorder is vital for neurological disorder diagnosis [29]. Despite this, in the context of GNN, there have only been two studies considering feature reproducibility [1, 30]. These works focus on GNN and federated GNN model selection methods based on reproducibility, but not in the context of KDG.

Considering this, it becomes evident that reproducibility within the context of KD and KDG has received insufficient attention and remains unexplored. Hence, this project bridges the three areas; GNNs, knowledge distillation and reproducibility into a novel domain that we term *reproducible offline knowledge distillation for GNN* concerned with the reproducibility of distilled models.

1.1 Contributions

This project explores reproducibility for offline KD and KDG methods. The primary contributions of this project are outlined as follows:

- In Chapter 4 we motivate research into the novel domain of *reproducible offline knowledge distillation for GNN*. In particular, we set out to answer the following question: *What happens to the reproducibility of student models after knowledge distillation takes place?*

- In Section 4.2 we propose a novel score that quantifies the reproducibility of a GNN model. The score is designed to indicate how well a model identifies the same set of biomarkers or features across different data perturbation strategies.

- We propose a novel KDG algorithm, *Reproducibility aware Knowledge Distillation on Graphs* (RepKD), in Chapter 5, which aims to retain and increase the reproducibility of the student model while conserving their performance. A one-to-many teacher-student framework is used, which is the first within KDG methods. In Chapter 6 extensive experiments are carried out to assess the performance of our algorithm compared to current state-of-the-art KD and KDG methods.

- In Section 5.2, we define a novel intra-student loss function used in RepKD that pushes students away from each other in order to promote diversity and reproducibility within the ensemble.

1.2 Ethical Considerations

We aim to uphold principles of transparency, fairness, and inclusivity in our research. In this project, we explore reproducible knowledge distillation on graphs, with a particular focus in the field of network neuroscience. As such, the datasets we are working with involve human participants as we delve into the intricacies of neuroscience. To respect the rights and privacy of human participants involved, we use two open-source and widely available datasets: Brain Genomics Superstruct Project dataset [31] and the BreastMNIST dataset from MedMNIST v2 [32]. Furthermore, the code used for this dissertation is publicly available, allowing for the replication and verification of our findings.

Chapter 2

Background

In this chapter, we will introduce key concepts to the reader to help them understand and give context to our project. We will cover graph neural networks, knowledge distillation on graphs and reproducibility.

2.1 Graph Neural Networks

In this section, we give a concise introduction to the main ideas and terminology used in graph neural networks (GNNs), these will equip the reader with the necessary background knowledge to understand GNNs.

2.1.1 Graphs

Graphs are the cornerstone of GNNs, and as such, we introduce them formally.

Definition 1 (Graph) *A graph $\mathcal{G}$ is defined by $\{V, A, X, Y\}$ where $V = \{v_1, \ldots, v_n\}$ is the set of vertices or nodes. The degree $deg(v_i)$ of a node is the number of edges attached to it. $A \in \mathbb{R}^{n \times n}$ is a symmetric adjacency matrix which defines the edge relation between the nodes of the graph. If $a_{ij} \neq 0$ then there exists an edge connecting nodes v_i and v_j. $X \in \mathbb{R}^{n \times d}$ is the feature matrix, each node $v_i \in \mathbb{R}^{n \times 1}$ are stacked row-wise. Finally $Y \in \mathbb{R}^{C \times 1}$ is the one-hot encoded class for $\mathcal{G}$.*

We will be dealing with undirected graphs throughout this project, and as such, A is symmetric. However, graphs can be directed, and in turn, A is not symmetric. Furthermore, graphs can be weighted; each edge has a value associated with it; unless stated otherwise, the graphs we are dealing with are weighted.

2.1.2 Graph Neural Networks

GNNs are a branch of deep learning developed to handle non-Euclidean data. In this subsection, we cover the main objectives of GNNs and how the general message passing scheme works.

GNN Objectives

GNNs are used for several different objectives, given the same graph structure. As a result, it is imperative that we make a clear distinction between the objectives since the underlying GNN architecture, training technique, and inference mechanism will vary depending on the end goal.

Node level - In this case, the node embeddings in the final layer of the GNN are used for classification or regression tasks for each node individually. For example, consider a scenario where we have a graph representation of webpage interactions, with webpages represented as nodes. In this case, the objective of a node classification task would involve determining the category or class of unlabeled nodes within the graph.

Edge level - Here, the objective can be either classification or regression of an edge between two nodes. For example, to predict if two nodes are linked in a graph.

Graph level - Again, the objective can be either classification or regression, but in this case on the whole input graph. This is done with the use of pooling, usually a permutation-invariant function, and readout layers [33] which aggregate the node embeddings to find a compact representation of the whole graph. In this project we focus on graph level tasks, in particular, graph classification.

Message Passing

The main goal of a GNN is to generate informative node embeddings of the input graph. A GNN takes an adjacency matrix A and a node feature matrix X which represent the graph $\mathcal{G}$. For a given node, its neighbourhood of nodes is used to define its embedding. To evaluate this node embedding, it involves a three-step process [34]: first, all the neighbour embeddings (messages) are gathered. Secondly, they are aggregated via a pooling function, such as averaging. Finally, the aggregated embedding is passed through an update function, which is normally a learned function such as an MLP [34]. This is then used as the new node embedding. This process is known as *message passing* [35] and is a key cornerstone for many of the GNN models used [5, 6, 7, 8, 9, 11]. In fact, what distinguishes these models from one another is that they use different techniques for aggregation and the choice of update function.
Naturally, one would like to add more layers to increase the expressivity of the node embeddings and capture higher-order relations between the nodes. To do so, we look at the *receptive field* of the node of interest, which is determined by its *N-hop neighbourhood*. The *N-hop neighbourhood* is the set of nodes that can be reached within N edge traversals from the node of interest. When using N layers we begin at the *N-hop* distance from the node of interest and progressively use the *message passing* scheme to

calculate node embeddings at the *(N-1)-hop* distance. This process is repeated until we reach the node of interest, the *0-hop* distance. This is then used as the final embedding for the node of interest.
For example, when using Graph Convolutional Networks (GCN) [5] the node embeddings $H^{(l+1)}$ at layer $l+1$ are given by:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{2.1}$$

with $\tilde{A} = A + I$ where I is the identity matrix and $\tilde{D}$ is the diagonal node degree matrix of $\tilde{A}$, $W^{(l)}$ is the weight matrix for layer l and $\sigma(\cdot)$ is an activation function. Notice that when $l = 0$ then $H^{(0)} = X$, the node features.
Another prevalent GNN is the Graph Attention Network (GAT) [8], which combines notions of attention layers with GNNs. We define $h_i^{(l+1)}$ as the embedding of node i at layer $l+1$ by:

$$h_i^{(l+1)} = \Big\Vert_{k=1}^{K} \sigma\Big(\sum_{j \in N_i} \alpha_{ij} W^k h_j^{(l)}\Big) \tag{2.2}$$

Where K are the number attention heads, $\Vert$ refers to concatenation, α_{ij} are learnable attention weights which quantify the importance of the interaction between node i and j. N_i is the neighbourhood of node i and W^k is a weight matrix for attention head k. Note that if multi-head attention is used for the final layer, averaging over the attention heads is used instead of concatenation [8].

2.1.3 Neighbour Explosion Problem

Despite the success of GNNs, they have some limitations. One large limitation is the *over-smoothing issue*, where node embeddings become the same for all nodes in the graph as the number of layers increases due to the receptive fields of each node being the same [36]. The *neighbour explosion problem* [11] is more critical for GNNs deployment. Many GNNs make use of the *message passing scheme* which requires aggregating all neighbouring node features at each layer of the GNN [33]. With the use of larger graphs and deeper GNNs, scalability and deployment become harder since, as the number of GNN layers increases, the number of nodes aggregated increases exponentially [17].
To overcome this issue, several solutions have been proposed. Separated into two lines of thought: model-based and algorithmic-based solutions [17]. Model-based solutions are broadly classified into two classes: sampling-based [7, 9, 19] or linear propagation-based [6, 37] methods [17].
Linear propagation-based methods focus on reducing the computation required for message-passing in GNNs by leveraging efficient precomputation [17]. For example, SGC [6] simplifies GCN [5] by removing the non-linear activation function of GCN; in turn, neighbourhood aggregation matrices can be precomputed, saving computation. Although inference time is saved, they still struggle with inference due to the removal of the non-linear activation function [17].

Sampling-based methods carefully select nodes or graphs during training or inference. GNNs that use layer-wise sampling, such as [9] are another way to sample nodes. In contrast to node sampling methods such as GraphSAGE [7] where nodes are sampled at the start of training or inference. Layer-wise sampling respects the hierarchical structure of graphs. GNNs that use graph sampling methods, such as Cluster-GCN [19], on the other hand, partition the original graph into smaller clusters. Although these methods manage to mitigate the *neighbour explosion problem*, they suffer from high variance inherited by the sampling process involved [17]. Furthermore, the size of the GNN models in both sampling-based methods is not reduced. As a result, this poses a problem for deployment on resource-limited devices.
Algorithmic-based solutions focus on reducing the size of the GNN models, addressing the limitations of sampling-based methods. For example, pruning [20] and quantization [21, 22] are commonly used. Pruning techniques are concerned with reducing the dimension of the feature embeddings in all the layers of the GNN, while low-precision integer arithmetic [21] is used in quantization to reduce inference time and model size [17]. However, in this project, we focus on offline knowledge distillation (KD) in order to reduce the size of the GNN models.

2.2 Knowledge Distillation on Graphs

Knowledge distillation (KD) is a wide domain aimed at distilling the performance (e.g., accuracy) from a large model, the teacher, into a small one, the student. In turn, light, deployable, and generalizable models are created [23]. Knowledge distillation on graphs (KDG) extended KD to graph-structured data, such that these methods can be tailored for GNNs.

2.2.1 General Framework

To begin, we will introduce the general KD objective. We are interested in transferring knowledge from a teacher to a student. To do so, the general loss function used to train the student model is given by:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{CE} + \beta \cdot \mathcal{L}_{KD} \tag{2.3}$$

Here, the learning procedure is guided by a teacher model through $\mathcal{L}_{KD}$ which transfers the *knowledge* from the teacher to the student. In particular, it acts as a regularisation term that guides the training procedure. In this project, we will focus on classification tasks, and as such, we use the cross-entropy loss $\mathcal{L}_{CE}$ between the predicted class and the ground truth labels. However, any loss can be used, depending on the task at hand. Here α and β are parameters that balance the terms.
In fact, many KDG methods are composed of three ingredients: the *knowledge* used, the *distillation scheme* used, and the *teacher-student framework* used. These will be covered in the subsequent sections.

2.2.2 Knowledge

In this section, we define what constitutes as *knowledge*, in the context of graphs. Knowledge, in the context of KDG, can take many forms, such as logits, node embeddings, or topological structures [18]. In general, knowledge is distilled through $\mathcal{L}_{KD}$ [18]:

$$\mathcal{L}_{KD} = DIV(k^t, k^s) \tag{2.4}$$

DIV is a function that measures the distance between the knowledge of a teacher k^t and the knowledge of a student k^s. There are many functions that can be used as DIV, such as Kullback-Leibler divergence [38], cross entropy, or the L_2 norm; this varies depending on the KDG method and type of knowledge used.

Output Logits were the first form of knowledge used in KD [24] (vanilla KD) , known as response-based knowledge. The output logits are simply the outputs of the final layer of the model before they are passed through a SoftMax activation function. Here, $\mathcal{L}_{KD}$ is the Kullback-Leibler divergence with a soft target-distribution between the logits of the student and the teacher. To get the soft-target distribution, SoftMax with temperature is used, which is defined as:

$$\sigma_\tau(z_i, \tau) = \frac{exp(z_i/\tau)}{\sum_j exp(z_j/\tau)} \tag{2.5}$$

Where z_i, the logit for class i. The result is a probability distribution over the class elements. τ, temperature, is a hyperparameter used to change the shape of the probability distribution. If $\tau > 1$, makes the distribution becomes softer, where the probabilities of different classes are closer. By using this, we can calculate the KD loss, $\mathcal{L}_{KD}$:

$$\mathcal{L}_{KD}(z^t, z^s) = \tau^2 \cdot D_{KL}(\sigma_\tau(z^t, \tau), \sigma_\tau(z^s, \tau)) \tag{2.6}$$

Where z^t and z^s are the logits of the teacher and student. D_{KL} is the Kullback–Leibler divergence. τ^2 is used to stabilise the learning procedure, we use this throughout the project as suggested in [24]. Notice that knowledge transferred through logits is not graph dependant.

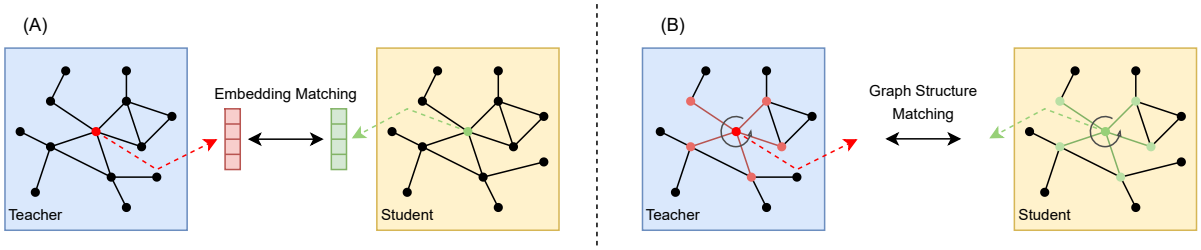

Figure 2.1: Graph-specific knowledge distillation techniques (A) For all nodes, we match the teacher and students learned graph embeddings. (B) For each node, we distil graph structure such that the student learns a similar graph representation as that of the teacher. Blue and yellow denote the teacher's and student's learned representations of the given graph. Red and green indicate the nodes of interest through which we are distilling knowledge, in the teacher and student, respectively.

Node Embeddings can be used in order to leverage graph structure. Knowledge can be distilled by matching the teacher's learned node embeddings at different layers with the students [18]. As a result, the distillation process is guided by the embeddings such that students learn the same node representations as the teacher, as depicted in Figure 2.1 (A). For example, in [39] where the goal is to increase model performance, they directly match the node embedding of the final layer. It is worth noting that this technique is generally used to increase model performance [18].

Graph Structure is another way of distilling knowledge. This is done by capturing how graph structure is described by the teacher and distilling this to the student model [18]. The main idea here is to find ways to preserve the embedded topological structure while considering node embeddings and the relative structure of the graph found through the teacher [26]. In turn, allowing for richer and more informative student models. For example, LSP [26] and MSKD [27] make use of local structures, as depicted in Figure 2.1 (B), but there are other techniques concerned with the conservation of global structures, such as [40].
For example, the local structures used in **LSP** [26] defined in the following way: For a graph $\mathcal{G} = \{V, A\}$ where $V \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$ the local structures learned by the teacher model t or a student model s can be expressed as a set of vectors for each node:

$$\mathbb{LS} = \{LS_1, LS_2, ..., LS_n\} \text{ where } LS_i \in \mathbb{R}^d \tag{2.7}$$

where d is the degree of the centre node i of the local structure. Where each element of the vector LS_{ij} is given by:

$$LS_{ij} = \text{SoftMax}(exp(-\frac{1}{2}||h_i - h_j||^2)) \tag{2.8}$$

This measures the similarity of the given pair of nodes i and j, where j is the neighbourhood of i. h_i and h_j represent their node embeddings. As a result, LS_i can be calculated for all the nodes in the graph in turn $\mathbb{LS}^s$ and $\mathbb{LS}^t$ can be calculated for the student and teacher, respectively. Once these are calculated, we can transfer the knowledge from the teacher to the student by using the following loss function:

$$\mathcal{L}_{KD}(\mathbb{LS}^t, \mathbb{LS}^s) = \frac{1}{n} \sum_{i=1}^{n} D_{KL}(\mathbb{LS}_i^t, \mathbb{LS}_i^s) \tag{2.9}$$

Here, the loss is given by the average Kullback-Leibler divergence loss D_{KL}, for all nodes n in graph $\mathcal{G}$. Finally, $\mathcal{L}_{KD}$ defined in equation 2.9 can be used in equation 2.3 to train a GNN by using graph structure as knowledge.

2.2.3 Distillation Schemes

Here we discuss the three main distillation schemes for KD and KDG. In particular, these can be divided into offline, online, and self-distillation [18, 23].

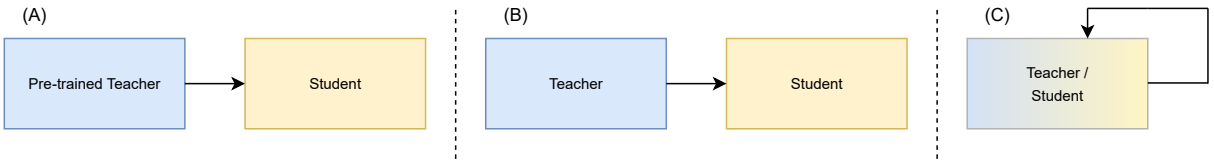

Figure 2.2: Different distillation schemes: (A) offline, (B) online, and (C) self-distillation The arrow indicates the direction of knowledge transfer. Figure inspired by [23].

Offline distillation - Figure 2.2 (A), makes use of a two step process. The first step, involves pre-training a teacher model. Subsequently, the student network is trained while keeping the teacher fixed [23]. It is worth noting that the majority of the KDG methods follow this paradigm [23]. In fact, during our project, we will focus on these methods since we want to reduce the size of a pre-trained model.

Online distillation - Figure 2.2 (B), trains both a teacher and student together during the distillation process in an end-to-end fashion [18]. This saves the time and memory overhead needed to pre-train and store a teacher model. These methods are particularly useful when we do not have access to massive teacher models [23].

Self-distillation - Figure 2.2 (C), or teacher-free distillation is when one model is both the teacher and student [18]. In this paradigm, the model learns to improve itself by transferring knowledge through the layers of the model [41].

2.2.4 Teacher-Student Frameworks

In this section, we will cover the main teacher-student frameworks used. The schemes discussed in Subsection 2.2.3 are often used with one of these frameworks.

One to One - (one teacher to one student) framework is mostly used in KDG [18]. In this case, one teacher exclusively distils knowledge into one student. In most cases, both the teacher and the students are GNNs; however, there is a separate line of work that aims to distil GNNs into a multilayer perceptron (MLP) [42, 43, 44].

Many to One (many teachers to one student) framework is also used in KDG. In this case, an ensemble of diverse teachers is used to train one student [18]. Despite the increase in knowledge attained by the use of multiple teachers, the robustness of students trained with this paradigm is reduced due to the introduction of noise in the knowledge by the use of multiple teachers [18].

The **Many to Many** and **One to Many** frameworks have not been extensively studied in KD. KD aims to compress models, and the use of ensembles conflicts with this goal. In turn, only a handful of papers make use of these paradigms. For example, in [45], an ensemble of teachers is used to train an ensemble of students, many to many. In [46] they use the one-to-many framework for text sentiment classification.

2.3 Reproducibility

In this section, we introduce the notion of reproducibility in the context of network neuroscience. Network neuroscience is a domain that focuses on studying and understanding the patterns of neural activity that arise within the brain [47]. Here, brains are modelled as graphs. Nodes represent regions of interest (ROIs), ROIs are different anatomical regions of the brain.
The identification of biomarkers, which, in the context of network neuroscience, translate to ROIs capable of conveying significant information about underlying neurological conditions. Reproducibility, in this context, is concerned with the identification of the same distinctive biomarkers or features when the data undergoes different perturbation methodologies. More precisely:

Definition 2 (Self-Repoducibility) *Given a GNN model trained with different data perturbation strategies (e.g., cross-validation), self-reproducibility quantifies the ability of the model to find consistent biomarkers or features across these perturbations.*

Self-reproducibility is of great importance, especially in the clinical setting. If a model is highly reproducible, this means that it can find consistent biomarkers or features, which, in effect, increases its trustworthiness. In Section 4.2 we formally introduce the self-reproducibility score used throughout this project. It is worth mentioning that this definition of reproducibility, despite being introduced in the context of network neuroscience and the clinical setting, can be applied to any domain.
Furthermore, it is important to understand that this definition of reproducibility is different from [1, 30] where they investigate the reproducibility **between different GNN architectures** that find the same discriminate biomarkers or features; we, on the other hand, look at the **same GNN architecture** under different perturbation strategies.
An important question arises: *How does one identify biomarkers or features when using GNNs for graph classification tasks?* [1] first proposes a solution to this by using a one-layer MLP as an output layer applied over the concatenated node embeddings (in order to preserve the graph structure) of the final layer of the GNN. As a result, the importance of a biomaker or features is given by the **magnitude of the weights in the output layer of the MLP**, since this layer is used for final graph classification. This notion will be used throughout this project.

2.4 Summary

In this chapter, we introduced the preliminary information and background needed to understand this project. In particular, we explored GNNs, understood how they work and their limitations, in particular the neighbour explosion problem, and motivated KD and KDG. For KDG, we explained what constitutes as knowledge, what distillation schemes can be used, in this project we focus on offline distillation since we are interested in compressing pre-trained models, and which teacher-student frameworks can be used. Finally, we introduce reproducibility in the context of network neuroscience. All of which are fundamental cornerstones of our project. In the next section, we will cover the related work and literature relevant to our project.

Chapter 3

Related Work

Reproducible knowledge distillation on graphs is a domain that has yet to be explored, and to the best of our knowledge, there are no KDG methods concerned with reproducibility. However, in this section, we explore closely related areas to our project and proposed method. In particular, we will look at state-of-the art methods in KD and KDG. We will also look at interpretable artificial intelligence, biomarker reproducibility methods, and stable feature selection methods, areas which are closely related.

3.1 Knowledge Distillation

Knowledge distillation (KD) aims to transfer the performance of a large model into a smaller one. KD was first proposed in [24], where knowledge is transferred through the output logits of the teacher, as described in Section 2.2. **FitNets** [25] propose knowledge transfer through intermediate features (learned representations, e.g., the output of intermediate layers) of the models. The L_2 distance loss is used to match the intermediate features between the teacher and student. Here, the loss function is given by:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{CE} + \beta \cdot \mathcal{L}_{KD} + \gamma \cdot \mathcal{L}_F \tag{3.1}$$

where $\mathcal{L}_F = D_{L_2}(\psi_s(f_s(x)), \psi_t(f_t(x)))$, $f_s(x)$ and $f_t(x)$ denote intermediate feature maps for the student and teacher, respectively. ψ_s and ψ_t denote feature transformations for consistent feature shapes for the student and teacher, respectively. D_{L_2} denotes the L_2 distance, and $\mathcal{L}_{KD}$ is the response-based knowledge loss defined in equation 2.6. In fact, many other KD methods have been proposed throughout the literature [23], all of them with a unique way to transfer knowledge, distillation scheme used, or the use of different teacher-student frameworks. However, despite their great successes and usage in many applications such as NLP or computer vision, these methods are not tailored for GNNs. As a result, methods were developed to take into account graph-structured data, as discussed in Section 2.2.

LSP [26] was the first method that considered the graph structure learned by GNNs and used this as knowledge during distillation. This is done with the use of the local structure-preserving module [26] which takes into account the topological structure of the graph learned by the teacher and then matches it with the students learned graph

structure during the distillation process, as explained in Subsection 2.2.2. **Multi-scale knowledge distillation** (MSKD) [27], takes LSP one step further by using multiple teachers to transfer knowledge to the student model. Again, knowledge is transferred through the teacher's graph structures. However, an attention mechanism is used to determine the importance of each teacher during knowledge distillation. The attention weight α_l for each teacher l is given by:

$$\alpha_l = \frac{MLP(z^s)^T MLP(z^{t_l})}{\sum_{l=1}^{L} MLP(z^s)^T MLP(z^{t_l})} \tag{3.2}$$

z^s and z^t are the logits of the student and teacher models, respectively. L is the total number of teachers. MLP refers to a multi-layer perception and is used to project the logits to a lower-dimensional space [27]. The loss $\mathcal{L}_{MSKD}$ is given by:

$$\mathcal{L}_{MSKD} = \frac{1}{n} \sum_{l=1}^{L} \sum_{i=1}^{n} \alpha_l D_{KL}(\mathbb{LS}_i^t, \mathbb{LS}_i^s) \tag{3.3}$$

Furthermore, MSKD also transfers knowledge through the output logits from all the teachers to the student. The final loss is given by:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{CE} + \beta \cdot \sum_{l=1}^{L} \mathcal{L}_{KD}(z^{t_l}, z^s)) + \gamma \cdot \mathcal{L}_{MSKD} \tag{3.4}$$

where $\mathcal{L}_{KD}$ has been defined in equation 2.6, $\mathcal{L}_{CE}$ is the cross entropy loss, and α, β and γ are parameters to balance the terms.

Many other KDG methods have been proposed; for example, in [42, 43, 44] they distil GNNs to MLPs. The motivation behind this is to minimise the complexity and inference time of GNNs while also reducing the dependency on the graph structure [18]. Other methods, such as [48], are concerned with the **fairness** of the distilled models. In particular, they aim to tackle inheriting biases distilled from the teacher into the student model, which is different from reproducibility.

Despite the fact that KDG methods are designed for GNNs and graph-structured data, they fail to consider the distilled model's reproducibility.

Furthermore, as we shall see, our proposed method uses a **one-to-many teacher-student framework**. As such, we explore works that use methods with similar framework designs. The use of many students in KD and KDG has not been extensively studied; in fact, only a handful of papers explore these ideas. In [45], an ensemble of teachers is used to train an ensemble of students. What makes this particularly powerful is that the students are connected via branches, which encourages collaboration and reduces the variance of the student model [45]. In [46] the one-to-many framework is used for sentiment classification, where each student model is trained on a different angle of the data. For example, in the task of *cross-lingual sentiment classification* [46] each student is trained on a different language of the same text. The drawback of these methods is that, again, they do not take into account the reproducibility of the distilled models and are not designed to be used with graph-structured data.

More importantly, to the best of our knowledge, the one-to-many teacher-student framework in the context of **KDG** has not been explored nor implemented. This leaves an opportunity to leverage the ideas from these papers into our proposed KDG method.
As we have discussed, KD and KDG methods do not take into account the reproducibility of the distilled models. However, there are two methods that are concerned with the *stability* of distilled models. Here, stability refers to the ability of a model to find consistent discriminative features, similar to our definition of reproducibility.
In [49] they use decision tree models as the student, due to their human interpretability. However, decision trees are sensitive to the data used to train them. As a result, in order to stabilise the student, they generate a large pool of pseudo-samples from the teacher. Then the stability of a split is assessed by running an hypothesis test to ensure that sufficient samples are used such that the split is consistent [49]. In [50] they generalise the notion of stability such that the student can be any intelligible model such as decision trees or symbolic regression, allowing for more flexibility of the student model. Furthermore, they produce more stable models when compared to the previous method. In contrast to [49] where they only use one student, here they generate a large number of candidate models used for comparison during training.
Despite the success of these methods, they have limitations when applied to GNNs. Firstly, these methods require a large number of pseudo-samples to be generated in order to be effective. In the context of *deep graph generation*, which captures the underlying distribution from which the graphs are generated, sampling becomes infeasible since complexity scales quadratically $O(n^2)$ as a function of n nodes [51]. Another limitation is that these methods restrict the student's model architecture to simple models such as decision trees, which fail to capture the nuances of graphs. In fact, the complexity and depth of these trees are, in many cases, restricted to a few levels, further reducing their expressive power. The reason for this is that these methods are mainly used for model interpretability and to provide interpretations for black-box models. Furthermore, [50] requires a large number of candidate models to be compared in the case of decision trees, which can easily be generated; however, in the context of GNNs, it is not clear how one would go about this.
[49] and [50] have a particular focus on interpretability. This motivated us to explore model-agnostic post-hoc explanation methods, a branch of interpretable artificial intelligence focused on explaining models once they have been trained. We found that these methods have close parallels with KD. Here the teacher is a black-box model, which we wish to explain, and the student is an interpretable model.

3.2 Interpretable Artificial Intelligence

Despite the remarkable accomplishments of machine learning and deep learning models across various fields, these models often lack the ability to provide users with a clear understanding of their outputs. As such, there has been a significant surge in the adoption of explainable artificial intelligence (XAI). XAI lies in addressing the interpretability and explainability issues associated with these black-box models.
In particular, post-hoc model explanation, a branch of XAI where models are explained after they have been trained [52]. They can be separated into two classes: model-

specific [53] or model-agnostic [54, 55]. Model-specific explanations attempt to directly explain the model, for example, with the use of saliency maps [56].
On the other hand, model-agnostic post-hoc explanation methods do not require access to the internal structure of the black box model; they only need the predictions. As a result, model-agnostic methods explain the outputs of black-box models by distilling this knowledge into an interpretable model. However, these methods have been criticised for their instability [57] and exploitability [58]. As a result, researchers have investigated ways of stabilising them such that these methods give consistent interpretations, in turn increasing their reproducibility.
For example, S-LIME [59] attempts to stabilise a well-known model explanation technique LIME [55]. LIME is a post hoc model explanation technique based on perturbations of the data, which are then used to train a sparse linear explainer, such as LASSO regression [55]. LIME has been shown to give unstable interpretations due to sampling variance or the sensitivity of LIME to hyperparameters [57]. S-LIME attempts to stabilise these interpretations by using a hypothesis testing framework similar to [49] ensuring that sufficient samples are used to guarantee stability.
However, these methods have similar limitations to those of [49] and [50] mentioned in the previous section. Furthermore, post-hoc models only have access to the output of the model, which means they are restricted in how one can distil knowledge for the student. As a result, the nuances of graph structure cannot be taken into account during the distillation process, which limits their usability with GNNs.

3.3 Biomarker Reproducibility Methods

As we have mentioned, reproducibility in the clinical setting is of great importance due to the increased trust and interpretability of these models. Despite this, in biomedical domains such as network neuroscience, prediction accuracy is the main measure to assess how well a model is able to distinguish between different neurological states often overlooking reproducibility [1].
Despite the importance of biomarker reproducibility, only a handful of papers have focused on this. In [60] they create a novel architecture by fusing the attention mechanism [61] and residual network (ResNet) [62] to capture the most discriminate ROIs while also optimising for classification performance. Furthermore, in [63] reproducible biomarkers across different datasets are used to identify critical brain changes responsible for neurodegenerative disorders, such as schizophrenia and autism. In [64] they improve the reliability of the identified biomarkers in [63] by using a large and diverse pool of training samples. In a similar vein, [65] exhaustively aims to identify the most reproducible biomarkers for autism diagnosis by using two different imaging modalities: functional and structural MRI.
Despite the successes of these works, they work directly with brain images and fail to make use of GNNs to leverage the topological structures of brain graphs. In particular, [60] make use of 3D-ResNets, in [63, 64] independent component analysis [66] a decomposition-based technique. Notably, in [65] they use a variety of classical machine learning and deep learning models.

However, in [67] they leverage graph structures in order to find the most distinctive feature selection algorithm that produces the most reproducible features for multi-graph datasets. The issue with this method is that it makes use of traditional feature selection methods and, as such, cannot be directly used with GNNs. In fact, [1, 30] address this limitation and investigate the reproducibility of GNNs. In [1] they create a novel framework used to find the most reproducible GNN model given a pool of different GNN architectures such as GCNs or GATs. Inspired by this, [30] uses many of the same notions and techniques but applies them to federated learning. Nevertheless, these papers investigated intra-model reproducibility, "if two models have consensus over the most important features or biomarkers, this shows that those features are reproducible across models. Hence, since different models end up finding the same top discriminate features, this shows that such models are reproducible." [1]. In contrast, we are concerned with the self-reproducibility of a model; that is, we want to increase the reproducibility of a single GNN architecture such that it is robust to perturbations to the data. As a result, the methods set out in these papers are not directly applicable to our problem. In light of this, we can see that the reproducibility of KDG methods has not been investigated, which leaves an opportunity to explore this domain.

3.4 Stable Feature Selection

Another line of work which is related but orthogonal to our work is stable feature selection. This domain is concerned with stabilising feature selection (FS) algorithms such that they give consistent and robust features [68]. In the context of bioinformatics, stable biomarker discovery is of great importance. This is because clinicians can be confident that the biomarkers identified by the FS algorithm and subsequently used as features for the machine learning algorithms can consistently separate controls from cases [68]. However, finding stable biomarkers is a difficult challenge due to various sources of instability. Firstly, the limited high-dimensional data, which in many cases is incomplete or noisy [28]. Secondly, underlying disorders can have multiple informative biomarkers that are difficult to identify [68]. Finally, many of the FS algorithms do not take stability into account [68]. This has motivated researchers to investigate stable FS algorithms. In fact, there has been a growing number of papers that are concerned with stable FS algorithms since the robustness of these methods to perturbation in the data is of the utmost importance for reliable biomarker discovery [69]. For example, in the context of biomarker discovery, [70, 71, 72] makes use of ensemble and data perturbation methods to find robust features by training a diverse set of FS algorithms. Then ensembles the results to get the final features, such that the features are more robust to perturbation in the data.
Reproducibility is closely related with stable FS. If we discover features that are consistent and stable, in turn, the FS algorithms are highly reproducible [68]. However, fusing FS algorithms together with GNNs has become a strenuous challenge [1]. The difficulty arises in the extraction of informative biomarkers or features of GNN models. This is because, unlike traditional machine learning models which directly make use of the features discovered by the FS algorithms, GNNs use node feature vectors and adjacency matrices as inputs. To address these limitations, [1] proposes the use of a

one-layer MLP as an output layer, for graph classification, as explained in Section 2.3. In turn, the reproducibility of GNNs relies on the absolute values within this layer, which indicate the importance of each feature or biomarker. As a result, applying stable FS algorithms directly to GNNs is not clear.

3.5 Summary

In this chapter, we covered the literature related to our project and the proposed method. We saw that many of the KD and KDG methods do not take into account the reproducibility of the distilled models. We also explored XAI methods that distil black-box models into an interpretable one due to the close parallels with KD. Finally, we looked at biomarker reproducibility methods and stable feature selection; these areas aim to tackle the same problem, but not in the context of KD. In light of the literature, we can see that there is an opportunity to explore a new research domain: *reproducible offline knowledge distillation for GNN*. In the next section, we aim to motivate this domain.

Chapter 4

Motivation

Having introduced the reader to the background and related work for our project, we have seen that there is a lack of research concerned with the reproducibility of KD and KDG methods. As a result, we aim to motivate a novel research area: *reproducible offline knowledge distillation for GNN*. To do so, we first formally introduce the problem. Then, we propose a method to quantify the reproducibility of a model. Subsequently, we show empirically that if reproducibility is not taken into account, in many cases, it leads to a degradation of the distilled model's reproducibility.

4.1 Problem

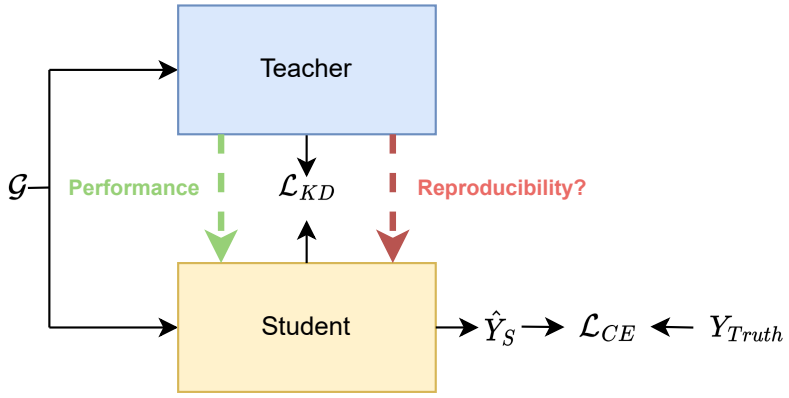

Figure 4.1: Illustration of the fundamental problem with KD and KDG methods. The performance of the teacher is being distilled to the student, but reproducibility is not being taken into account. Losses are explained in Equation 2.3. $\hat{Y}_S$ and Y_{Truth} are the student predictions and ground truth labels.

As we have discussed extensively in Chapters 2 and 3, the main goal of knowledge distillation is to transfer performance, generally accuracy, from the teacher to the student. Different forms of knowledge, distillation schemes, and teacher-student frameworks can be used. However, we found that no method takes into account the reproducibility of the student model during the distillation process, as illustrated in Figure 4.1.

As discussed in Section 2.3, it is important in the clinical setting since it puts into question whether we can trust distilled models. As a result, we explore the following question: *What happens to the reproducibility of student models after knowledge distillation takes place?* Before answering this question, we need to quantify reproducibility.

4.2 Quantifying Reproducibility

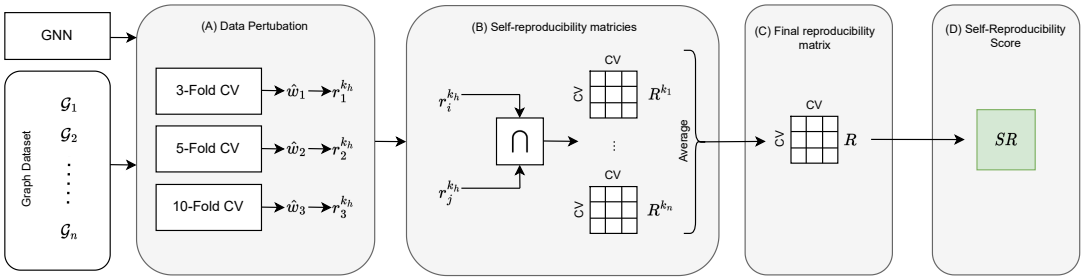

Figure 4.2: Framework used to derive the *self-reproducibility score* of a fixed GNN model. **(A)** For a fixed model, we train it using different data distribution perturbation strategies. Once the model has been trained, we extract the final weight vectors for each cross-validation method $\hat{w}_i$. Using $\hat{w}_i$ and for n different threshold values, we extract the rank vector $r_i^{k_h}$ for each cross validation method ($i \in \{1, 2, 3\}$). **(B)** Using $r_i^{k_h}$, we evaluate the *self-reproducibility matrix* R^{k_h} for different threshold values. **(C)** Next, we use them to calculate the final *self-reproducibility matrix* R **(D)** By using R, we calculate the self-reproducibility score SR.

In this section, we introduce a novel score, the self-reproducibility (SR) score, to measure the reproducibility of a model (Definition 2). If a model has a large SR score, then this means that it identifies the **same** set of biomarkers or features across different data perturbation strategies, in turn increasing its **trustworthiness**. It is important to understand that the self-reproducibility score proposed is different from the ones in [1, 30]. In [1, 30] the score aims to quantify reproducibility **between different GNN architectures**; we, on the other hand, quantify reproducibility **same GNN architecture** under different perturbation strategies.
We define a *fixed model* as a model for which the architecture, hyperparameters, and initial weights are the same before it undergoes a different data perturbation training strategy, in this case cross-validation. Notice that during cross-validation, any training strategy can be used, such as offline knowledge distillation or simply gradient descent. To understand how we derive the SR score, we will walk through Figure 4.2.

(A) We begin by training a model using 3, 5, and 10-fold cross-validation as the data perturbation strategies, resulting in 18 trained models in total (one for each type of cross-validation, 3, 5, and 10-fold). We look at the weights of the final layer of each model, as explained in Section 2.3, and take the average of these weights for each cross-validation scheme used (3, 5, and 10-fold) as the final weight vector. As a result, we end up with **three final weight vectors**, one for each cross-validation scheme; we call them $\hat{w}_i$ where $i \in \{1, 2, 3\}$ indicates the cross-validation scheme used.

Once all the $\hat{w}_i$ have been obtained, we can look at the top K ROIs or features for each of these vectors. To do so, we take the **absolute value of each of the weights** $\hat{w}_i$, rank them in descending order, and select the top K *indices*. We call this the rank vector $r_i^{k_h}$ of the top k_hth ROIs or features. Where i is the index of $\hat{w}_i$ used. We do this for n different K threshold values.
(B) For a given threshold K we can define a **self-reproducibility matrix** R^{k_h} which measures the consistency between different perturbation strategies for a given threshold value. Each entry of the matrix R^{k_h} is given by $R_{i,j}^{k_h} = \frac{|r_i^{k_h} \cap r_j^k|}{|r_i^{k_h}|}$, **the intersection of indices** between for r_i^k and r_j^k. This done for all unique combinations of r_i^k, resulting in a three-by-three matrix. One can evaluate different n *self-reproducibility matricies* R^{k_h} for different threshold values.
(C) To generate the **final self-reproducibility matrix** R, where each entry is given by $R_{i,j} = \frac{\sum_{h=1}^{n} R_{i,j}^{k_h}}{n}$, here n is the total number of thresholds K. This is simply **the average of the** n ***self-reproducibility matrices*** R^{k_h}.
(D) The **final self-reproducibility score** SR is given by **averaging the off-diagonal elements of the upper triangular part of the matrix** R, since it is symmetric. Each entry of R is a score for each data perturbation strategy across all thresholds; thus, averaging over these values gives a score that incorporates all perturbation strategies and different threshold values for a model.
The SR score varies from 0 to 1. A model is perfectly reproducible if its SR score is equal to 1. This means that the model always finds the same set of top biomarkers under various perturbations of the training data. Note that the SR score can be scaled by any factor. Using a scaling factor of 100, the score varies from 0 to 100 instead.
It is important to highlight that despite variations of the metric above being extensively used in biomarker discovery [73], it has some drawbacks. Firstly, the values of K and n are often chosen empirically. Secondly, the score simply looks at the intersection of the ranks; in turn, this does not take into account the weights for each biomarker. Thirdly, when looking at larger values of K if a biomarker appears in a vastly different location in both vectors, it will get the same score as if it appeared in similar locations since the intersection does not consider relative ranking.

4.3 Knowledge Distillation & Reproducibility

In this section, we explore the reproducibility of knowledge distillation on graphs methods. In particular, aim to answer the following research question: *What happens to the reproducibility of student models after knowledge distillation takes place?*

4.3.1 Experimental Setup

In this section, we give a brief overview of the experimental setup used for motivating the problem of *reproducible offline knowledge distillation for GNN*. Note that the experimental setup here is the **same** as the one in the evaluation and discussion section of the project. As such, in-depth explanations can be found in Section 6.1.

Datasets: We conduct experiments on graph classification tasks. In particular, we look at four datasets derived from the **Brain Genomics Superstruct Project (GSP)** dataset [31]. The datasets are constructed from the structural and functional MRI scans to produce cortical morphological networks (CMNs) for each individual. In particular, for each individual, the maximum principal curvature network C_1, cortical thickness network C_2, sulcal depth network C_3 and average curvature network C_4 are derived. Each dataset is comprised of 698 graphs, split into two classes: male and female. The number of nodes per graph is 35, the average number of edges is 298, and the average node degree is 17. We also use the **BreastMNIST** dataset from MedMNIST v2 [32]. The dataset is comprised of 780 samples; the number of nodes per graph is 28, the average number of edges is 135.58, and the average node degree is 10.11. We treat the images as graphs, as is done in [30].

Knowledge Distillation Methods: To understand what happens to the reproducibility of distilled models, we investigate four different KD methods. In particular, **Vanilla** KD [24] which is the first proposed KD technique, introduced in Subsection 2.2.2. The second method we explore is **FitNet** [25], explained in Section 3.1 where the loss is defined in equation 3.1. Finally, we investigate two KDG methods: **LSP** [26] and **MSKD** [27]. LSP transfers knowledge through graph structure, detials of losses can be found in Subsection 2.2.2. MSKD does too, but makes use of multiple teachers as explained in Section 3.1, loss is defined in equation 3.4. Due to the use of multiple teachers of varying layers, in these experiments a one- and two-layer GNN were used as the teachers.

Teacher and Student Backbones: Once the KD methods have been decided, we need to define the teacher and student architectures to use. We explore two widespread and common GNNs: Graph Convolution Networks (GCN) [5] and Graph Attention Networks (GAT) [8]. These were introduced in Subsection 2.1.2. The GCN teacher is a 2-layer GCN. The node embedding dimensions per layer are 64 and 2, respectively. The GCN student is a 1-layer GCN, and the node embedding dimension is 2. The GAT teacher is a 2-layer GAT. The first layer has 8 attention heads, each with a node embedding dimension of 8. The second layer has one attention head with node embedding dimension 2. The GAT student is a 1-layer GAT. In this case, there is only one attention head with node embedding dimension 2. In the final layer of the GNN, a topology-preserving aggregation function is used on the node embeddings, which is then passed into a single-layer MLP for final classification. This is done such that each weight that is extracted is related to a biomarker or feature [1]. However, this means that the hidden dimension final layer must be equal to the number of classes, which in our case is two.

Training & Parameter Settings: All models need to be trained using 3, 5, and 10 fold cross-validation, due to the definition of self-reproducibility, Section 4.2. For each cross-validation, we report the performance of the model on each of the test splits and then evaluate the self-reproducibility score, using four different threshold values: $K = \{5, 10, 15, 20\}$. **We ran all experiments across 10 different seeds**; to mitigate biases in the weight initialization or data splits, results are reported over the different seeds. Furthermore, to ensure fair tests, we used many shared parameters across all models, all defined in Table 6.1. Pre-trained teacher and student-specific hyperparameters can be found in the table A.1. KD model-specific hyperparameters are defined in Table A.2.

4.3.2 Analysis

Before we look at the reproducibility of the distilled models and analyse the results, we need to ensure that successful knowledge distillation has in fact taken place. From Table 4.1 we can see that the accuracy of the student when trained with the teacher, across all datasets and architectures, increases with respect to the original student.
We now turn our attention to the self-reproducibilty scores of the distilled models for different datasets, KD methods, and GNN architectures used. From Figure 4.3 we can immediately see clear trends.
Firstly, we can see that for **all different knowledge distillation methods, the students reproducibility decreases**. As seen by the percentage decrease of the student model when trained with a KD method with respect to the student trained without any KD method (orange bars). FitNet sees the largest fall in reproducibility for both GCN and GAT architectures on average across all datasets, -21.43% and -11.92% respectively. LSP and MSKD see similar degradations in student reproducibility since they use the same knowledge transfer technique (graph structures). Finally, vanilla KD sees the least degradation of reproducibility for both GCN and GAT architectures on average across all datasets, -1.32% and -1.38%. This clearly shows that KD methods that do not take reproducibility into account lead to a degradation in the distilled model's reproducibility, in the worst case, a decrease of 31.42%.
Secondly, **on average, student models are more reproducible than the teachers for both architectures**. To explain this, we look at the stability of GNNs, [74] showed that deeper models have large fluctuations in their weights. In turn, this could explain why the teachers are less reproducible. Reproducibility depends on the weights of the final layer, since weights are less stable for larger models. In turn, this degrades the self-reproducibility score.

Model	C_1	C_2	C_3	C_4	BreastMNIST	**Average**
GCN T.	61.01±0.30	66.16±0.09	66.91±0.09	65.44±0.12	72.48±0.31	66.40±0.18
GCN S.	57.78±0.28	63.75±0.20	63.25±0.11	63.62±0.09	69.21±0.66	63.52±0.27
Vanilla	58.16±0.24	63.98±0.14	63.75±0.15	63.77±0.23	70.34±0.60	64.00±0.27
FitNet	60.73±0.57	64.21±0.39	65.20±0.31	64.45±0.20	73.32±0.08	65.58±0.31
LSP	61.21±0.27	65.58±0.31	66.43±0.08	65.78±0.13	73.41±0.17	66.48±0.19
MSKD	**62.18±0.14**	**65.72±0.38**	**66.51±0.26**	**65.79±0.09**	**73.99±0.04**	**66.84±0.18**
GAT T.	62.03±0.31	67.77±0.09	65.29±0.23	66.40±0.55	72.92±0.09	66.88±0.26
GAT S.	57.52±0.49	61.17±0.07	60.29±0.19	59.96±0.08	55.59±0.08	58.91±0.18
Vanilla	58.05±0.38	63.12±0.19	61.13±0.03	60.65±0.15	56.91±0.08	59.97±0.16
FitNet	62.52±0.16	66.37±0.26	65.85±0.26	66.46±0.23	73.54±0.16	66.95±0.21
LSP	63.28±0.20	66.40±0.35	66.67±0.22	66.83±0.05	73.42±0.07	67.32±0.18
MSKD	**63.45±0.35**	**66.92±0.51**	**66.93±0.11**	**66.48±0.13**	**73.60±0.18**	**67.47±0.25**

Table 4.1: Average test-set accuracy for 3, 5 and 10-fold cross validation. GCN (top) and GAT (bottom) architectures. Teacher (T.) student (S.). **Bold** and Underline indicate the best and second best performance among the KD methods. Average column is the averaged performance of each model across all datasets.

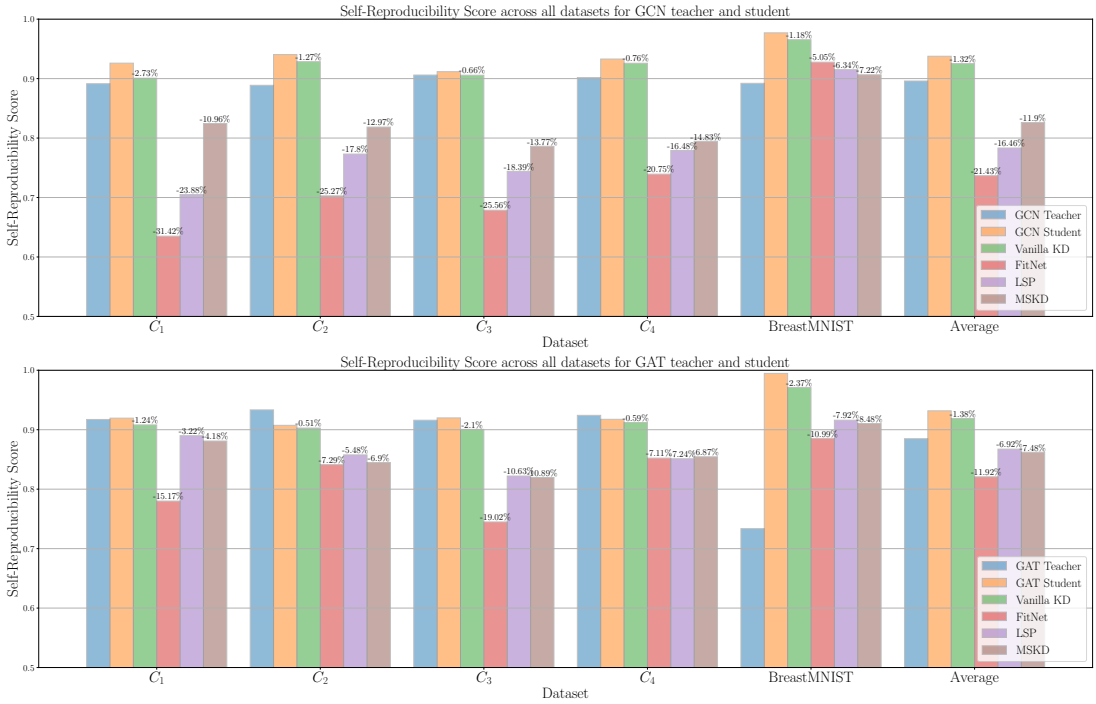

Figure 4.3: Average self-reproducibility scores for GCN (top) and GAT (bottom) architectures. Value above the KD method bar indicates the percentage decrease of the self-reproducibility with respect to the student trained without KD (orange). The last set of bars shows the averaged performance of each model across all datasets.

We suspect that the **inherit uncertainty of the teacher model** causes a degradation in students reproducibility. Teachers are models that try to best solve a task; however, they are not perfect and, as such, have uncertainty. In turn, this means that teachers have variability, which, if not taken into account, is distilled into the student models, thus reducing the self-reproducibility score. Furthermore, we attempt to explain the differences in the students degradation between different KD techniques. We suspect this is due to the **way knowledge is transferred from the teacher to the student**. Vanilla KD uses output logits, while FitNet, LSP, and MSKD match the internal model structure directly. As such, matching graph structures and internal representations influence the weights in the final layer more than output logits, which results in a larger degradation.

4.4 Summary

To conclude, in this chapter, we formally introduced the problem of reproducibility and knowledge distillation. We formalised the notion of reproducibility and put forward a score that can be used for GNNs. More importantly, we answered the following question: *What happens to the reproducibility of student models after knowledge distillation takes place?* We empirically showed that for a wide range of GNN architectures and KD and KDG baselines, there is a degradation in the distributed models reproducibility. In light of these findings, in the next section, we propose a novel KDG method that aims to solve this issue.

Chapter 5

Method

In the previous chapters, we explored KD and KDG methods and saw that the reproducibility of distilled models decreased. As a result, in this section, we propose a novel offline knowledge distillation method for GNNs designed to retain the reproducibility of student models while also conserving their performance.

5.1 Problem Formulation

In this section, we formalise the problem we are trying to solve. In particular, *we want to retain and increase reproducibility while conserving the performance of distilled models (students) during offline knowledge distillation for GNN-based teacher-student frameworks.* In order to achieve this, the designed model must satisfy the following:

Performance - The proposed method should prioritise conserving and increasing the students reproducibility without degrading their performance.

Model agnostic - KD methods should work independently of what GNN architectures are used. As a result, the proposed method should be robust to the choice of GNN used for both the teacher and student architectures.

Efficiency - When comparing the proposed method to other KD and KDG methods, we should not see a large overhead in the training time, number of parameters or inference time of the distilled model.

5.2 Proposed Method

We intorduce our novel proposed method, *Reproducibility aware Knowledge Distillation on Graphs* (RepKD), which aims to transfer the knowledge from a large pre-trained teacher GNN to a smaller GNN student while preserving reproducibility for graph classification. As seen in Figure 5.1 the overall framework is a two-step process which consists of **(A)** reproducibility aware knowledge distillation process and **(B)** student selection process, which decides the final GNN student used for inference.

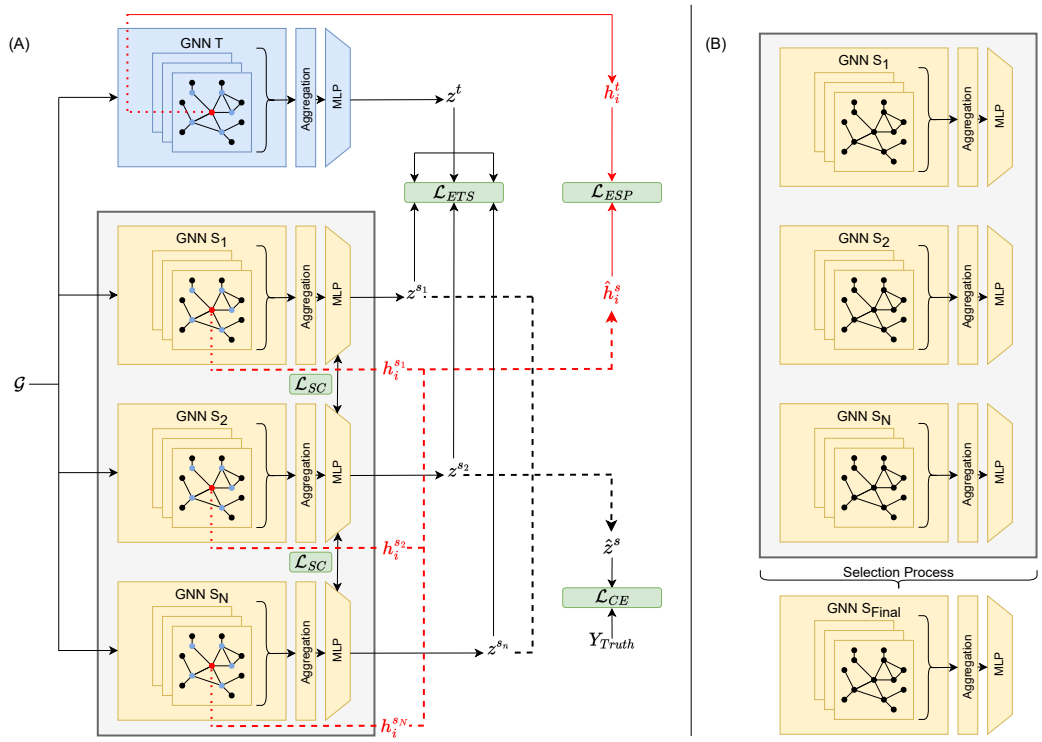

Figure 5.1: An overview of the proposed Reproducibility aware Knowledge Distillation on Graphs (RepKD) framework. A two-step process, **(A)** reproducibility aware knowledge distillation **(B)** student selection process, is used to decide the final student GNN used. GNN T, GNN S_1, GNN S_2, GNN S_N represent graph neural networks for the teacher and the 1st, 2nd, and Nth student. z^t, z^{s_1}, z^{s_2}, z^{s_N}, $\hat{z}^s$ represent the teachers and 1st, 2nd, Nth, and average students logits and Y_{Truth} are the ground truth labels for graph $\mathcal{G}$. h_i^t, $h_i^{s_1}$, $h_i^{s_2}$, $h_i^{s_N}$, $\hat{h}_i^s$ represent the teachers and 1st, 2nd, Nth, and average students embedding for node i. Black and red lines indicate response-based and graph-based knowledge transfer, respectively. Dashed lines indicate averaging. The green boxes indicate loss functions used during training.

5.2.1 Reproducibility aware Knowledge Distillation

We begin by exploring **(A)** the reproducibility aware knowledge distillation method used to train the student models. To understand the framework, we will begin by explaining why an ensemble of students is used and then walk through the different loss functions that form the method. It is worth noting that our method is different from current KD, which leverages ensembles [45, 46] In particular, [46] is exclusively designed for text data, and [45] does not leverage graph structures. In fact, our method, to the best of our knowledge, is the first KD on **graphs** method to leverage the one-to-many teacher-student framework.

Student Ensemble

Ensembles are a well-known technique used in machine learning to increase model performance [75]. Recently, ensembles have gained a lot of interest in deep learning,

known as *deep ensembles*, due to their simplicity and ability to quantify uncertainty [76]. They have also had great success in stabilising feature selection algorithms [70, 71, 72], such that consistent features are discovered. Furthermore, ensembles have been used in KD. In, [45] showed that by joining the output logits of the students and incorporating a loss at the ensemble level, it helped reduce the variance of the predictions of each student model and encouraged within-model collaboration in the ensemble, further increasing performance.
As such, we leverage these ideas and incorporate them into our method. In particular, we use ensemble of N students, a hyperparameter that needs to be tuned, where each model has the **same GNN architecture** but is initialised with **different weights**. This is done in order to embrace the variability incurred by model initialization, which is leveraged to stabilise student reproducibility.
In particular, we incorporate two ensemble-level losses for collaborative learning: an intra-student loss to promote diversity within the ensemble and a teacher-student loss within the ensemble to retain performance at the individual level.
Furthermore, it is important to highlight that all the models pass their node embeddings through a topological persevering aggregation function before passing them through the MLP for classification [1]. This is because it is vital to preserve the topological structure such that the weights are permutation variants since the ordering of the weights is important when evaluating the reproducibility score and for model interpretability. In particular, instead of averaging the node embeddings in the final layer of the GNN, we simply **concatenate** the embeddings row-wise. Then we apply the MLP to each column. However, this means that the hidden dimension final layer is restricted to the number of classes in order to be able to get the output logits for each class needed for graph classification.

Ensemble Losses

As mentioned previously, we incorporate two losses for collaborative learning within the ensemble. In particular, $\mathcal{L}_{CE}$ and $\mathcal{L}_{ESP}$.
$\mathcal{L}_{CE}$ is simply the cross-entropy loss between ensemble prediction:

$$\hat{Y}_i^{Ensemble} = \text{SoftMax}(\hat{z}_i^s) = \frac{exp(\hat{z}_i^s)}{\sum_j exp(\hat{z}_j^s)} \tag{5.1}$$

where $\hat{z}_i^s$ is the average logits for the students in the ensemble for class i, and the ground truth prediction Y^{Truth} which are one hot encoded.

$$\mathcal{L}_{CE}(Y^{Truth}, \hat{Y}^{Ensemble}) = -\sum_c^C Y_c^{Truth} log(\hat{Y}_c^{Ensemble}) \tag{5.2}$$

where C is the total number of classes used for classification.
Since we are dealing with graph-structured data, it was important that this be taken into account during the distillation process. As a result, **ensemble structure preserving loss** $\mathcal{L}_{ESP}$ aims to preserve the topological graph structure learned by the teacher model and distil this into the ensemble. We use the same definition of local structure used in [26, 27].

For a graph $\mathcal{G} = \{V, A\}$ where $V \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$ the local structures learned by the teacher model or an ensemble mode can be expressed as:

$$\mathbb{LS} = \{LS_1, LS_2, ..., LS_n\} \text{ where } LS_i \in \mathbb{R}^d \tag{5.3}$$

Where $\mathbb{LS}$ is a set of vectors for each node n with degree d.
LS_i measures the similarity between node i and all its neighbours. In order to calculate each element of the vector LS_{ij}, we need to pass $\mathcal{G}$ through the model such that we have the node embeddings learned for each node of the graph. LS_{ij} given by:

$$LS_{ij} = \text{SoftMax}(exp(-\frac{1}{2}||h_i - h_j||^2)) \tag{5.4}$$

where i and j is a pair of nodes and j is the neighbourhood of i. h_i and h_j represent their respective node embeddings. In turn, $\widehat{\mathbb{LS}}^s$ and $\mathbb{LS}^t$ can be calculated for the ensemble and teacher, respectively.
It is important to note that this is different to [26] and [27] to order to be able to evaluate $\widehat{\mathbb{LS}}^s$ which is the local structure for the ensemble. We need to first evaluate the node embeddings for the ensemble $\hat{h}_i^s$ for all nodes. To do so, we simply take the average of the node embeddings across the students in the ensemble.

$$\hat{h}_i^s = \frac{1}{N} \sum_{k=1}^{N} h_k^s \tag{5.5}$$

Where N is the total number of students in the ensemble. Having evaluated $\hat{h}_i^s$ for all nodes, we can now use these in equation 5.3 to calculate $\widehat{\mathbb{LS}}^s$. We can now use the local structures we evaluated and use them during KD, through the use of the following loss function:

$$\mathcal{L}_{ESP}(\mathbb{LS}^t, \widehat{\mathbb{LS}}^s) = \frac{1}{n} \sum_{i=1}^{n} D_{KL}(\mathbb{LS}_i^t, \widehat{\mathbb{LS}}_i^s) \tag{5.6}$$

Here, the loss is given by the average Kullback-Leibler divergence loss D_{KL}, for all nodes n in the graph $\mathcal{G}$.

Intra-Student Loss

To promote diversity and stability within the ensemble, we introduce a novel **intra-student loss** $\mathcal{L}_{IS}$. Motivated by distance-aware deep ensembles [77] and anti-distillation techniques [75], where during optimisation models in the ensemble are pushed away from each other. In [75] a loss is used to decorrelate the output logits of the models in the ensemble for each batch. In [77] they incorporate a weight repulsive term in the gradient of the weight parameters to prevent them from having the same value. Putting these two ideas together, we incorporate a loss that pushes the weights of **the final layer** away from each other by using the cosine similarity measure between these weights. This is done to ensure that during the model selection process, there is a wide diversity of students that are orthogonal to each other. As we shall see vital for the students reproducibility.

To calculate $\mathcal{L}_{IS}$, we first need to evaluate the cosine similarity loss $\mathcal{L}_{SC}$ between all students in the ensemble N. This is done by evaluating the *cosine similarity* between the weights w^{s_i} and w^{s_j} of the MLP for all unique pairs of students i and j:

$$\mathcal{L}_{SC} = S_C(w^{s_i}, w^{s_j}) = \frac{w^{s_i} \cdot w^{s_j}}{||w^{s_i}||_2 \times ||w^{s_j}||_2} \tag{5.7}$$

S_C is the cosine similarity measure, $\cdot$ indicates the dot product, and $||\cdot||_2$ is the L2 norm. If $S_C(w^{s_i}, w^{s_j}) = 0$ then w^{s_i} and w^{s_j} are orthogonal or decorrleated to each other; as such, we wish to minimise $\mathcal{L}_{SC}$. Finally, to evaluate $\mathcal{L}_{IS}$, we define $\mathcal{B}$ as the unique set of all pairs of student weights $\mathcal{B} = \{(w^{s_i}, w^{s_j}) \in \binom{N}{2}\}$ then:

$$\mathcal{L}_{IS} = \sum_{k=1}^{|\mathcal{B}|} \mathcal{L}_{SC}(\mathcal{B}_k) \tag{5.8}$$

As a result, when minimising this loss function, it results in decorrelarted weights for all pairs of students, which in turn promotes model diversity within the ensemble and stabilises the weights, which in turn results in reproducible models.

Teacher-Student Loss

We also introduce a novel **ensemble teacher-student** loss $\mathcal{L}_{ETS}$ inspired by [45]. Here a one-to-one knowledge distillation from each teacher to the student in their respective ensembles is incorporated to promote a diverse set of representations. We incorporate this loss for both diversity within the ensemble and to guide the students so that, *individually* they have similar performances to the teacher. This is crucial to preserving individual student performance during the selection process.
$\mathcal{L}_{ETS}$ is a response-based loss, meaning we distil knowledge from the teacher to each student in the ensemble through their output logits:

$$\mathcal{L}_{ETS} = \tau^2 \sum_{i=1}^{N} D_{KL}(\sigma_\tau(z^t, \tau), \sigma_\tau(z^{s_i}, \tau)) \tag{5.9}$$

Where N is the total number of students in the ensemble, z^t and z^{s_i} are the logits of the teacher and student i. D_{KL} is the Kullback-Leibler divergence. τ is the temperature parameter (τ^2 is used to stabilise the learning procedure [24]).

Final Loss Function

Finally, we can define the final loss function used during training by simply joining the above losses together:

$$\mathcal{L}_{Final} = \alpha \cdot \mathcal{L}_{CE} + \beta \cdot \mathcal{L}_{ESP} + \gamma \cdot \mathcal{L}_{ETS} + \lambda \cdot \mathcal{L}_{IS} \tag{5.10}$$

Where α, β, γ and λ are hyperparameters used to balance the loss terms. The pseudo-code for the framework is summarised in Algorithm 1.

5.2.2 Selection Process

After the ensemble has been trained, as explained above, we then pick a final student model from the ensemble, which is then used during inference. There are several reasons why we pick one student instead of using the ensemble. Firstly, this is to reduce the number of model parameters such that the final model used during inference is parameter efficient. Secondly, if parallelization is not possible in the end device, such as CPUs with a single core, then the inference time of the ensemble is greater than that of a single model. Since we need to evaluate multiple models and join their outputs. Finally, this gives us the opportunity to find a model that is both reproducible and retains performance.
To determine the final student model S_{Final}, the selection criteria used are simply selecting the model within the ensemble that has the highest weighted reproducibility score and accuracy. To calculate it, for each model in the ensemble, we evaluate its reproducibility score and accuracy, then take the mean of these two values as the final score. The reason this is used is to find models that find a balance between reproducibility and performance; as we shall see, these are inversely correlated.

5.3 Implementation

Our proposed method was implemented in Python using the PyTorch [78] and PyTorch Geometric [79] libraries. Since we are working with graph-structured data, Pytorch Geometric is specifically intended to interface with GNN, allowing us to directly operate on graphs using built-in methods. Both of these libraries are optimised for GPU utilisation, ensuring high-performance execution. Moreover, a key reason behind our choice of these libraries is their extensive use within the research community. This popularity assures that our results can be easily replicated by others, fostering the credibility of our approach. Appendix C has detailed information about the code repository of our project.

5.4 Summary

In this chapter, we introduce our novel proposed knowledge distillation method, *Reproducibility aware Knowledge Distillation on Graphs* (RepKD), which aims to solve the problem highlighted in Chapter 4, the deterioration of reproducibility in distilled models. RepKD is made up of two key components. Firstly, a one-to-many teacher-student framework is used to train an ensemble of students using distance aware and knowledge distillation losses to ensure both reproducibility and performance. Secondly, once the training is complete, the best student from the ensemble is chosen in order to find the most reproducible and performant student. In the next chapter, we will compare our method to other state-of-the-art methods and carry out extensive experiments to justify the different design choices of our method.

Algorithm 1 Reproducibility aware Knowledge Distillation on Graphs

Input: Set of training graphs $G = \{\mathcal{G}\}$; Pre-trained GNN teacher T with model parameters θ^t; Total number of students in ensemble N

Output: A compact student model S_{Final} with model parameters $\theta^{s_{Final}}$

initialize model parameters $\{\theta^{s_1}, \theta^{s_2}, ..., \theta^{s_N}\} \leftarrow S_{Ensemble}$

`//knowledge distillation process`

while $S_{Ensemble}$ has not converged **do**

for $\mathcal{G}$ in G **do**

`//cross entropy loss`

calculate ensemble $\hat{z}^s$ as the average of $\{z^{s_1}, z^{s_2}, ..., z^{s_N}\}$

calculate $\mathcal{L}_{CE}$ using ground truth labels, $\hat{z}^s$ and equation 5.2

`//ensemble structure preserving loss`

evaluate $\mathbb{LS}^t$ using $\mathcal{G}$, θ^t and equation 5.3

extract $\{h^{s_1}, ..., h^{s_N}\}$ using $\mathcal{G}$ and $S_{Ensemble}$

evaluate $\hat{h}^s$ using $\{h^{s_1}, ..., h^{s_N}\}$ and equation 5.5

evaluate $\widehat{\mathbb{LS}}^s$ using $\hat{h}^s$ and equation 5.3

calculate $\mathcal{L}_{ESP}$ using $\widehat{\mathbb{LS}}^s$, $\mathbb{LS}^t$ and equation 5.6

`//intra-student loss`

extract weights $\{w^{s_1}, w^{s_2}, ..., w^{s_N}\}$ from $S_{Ensemble}$

calculate loss $\mathcal{L}_{IS}$ using $\{w^{s_1}, w^{s_2}, ..., w^{s_N}\}$ and equation 5.8

`//teacher-student loss`

extract z^t using $\mathcal{G}$ and θ^t

extract $\{z^{s_1}, z^{s_2}, ..., z^{s_N}\}$ using $\mathcal{G}$ and $S_{Ensemble}$

calculate loss $\mathcal{L}_{ETS}$ using z^t, $\{z^{s_1}, z^{s_2}, ..., z^{s_N}\}$ and equation 5.9

`//final loss and update model parameters`

evaluate final loss $\mathcal{L}_{Final}$ using $\mathcal{L}_{CE}, \mathcal{L}_{ESP}, \mathcal{L}_{IS}, \mathcal{L}_{ETS}$ and equation 5.10

update all $S_{Ensemble}$ parameters $\theta^{s_i} \leftarrow \theta^{s_i} - \theta^{s_i} \nabla \mathcal{L}_{Final}$

end for

end while

`//student selection process`

evaluate self-reproducibility score $\{\text{SR}^{s_1}, \text{SR}^{s_2}, ..., \text{SR}^{s_N}\}$ in $S_{Ensemble}$

evaluate accuracy $\{\text{Acc}^{s_1}, \text{Acc}^{s_2}, ..., \text{Acc}^{s_N}\}$ for students in $S_{Ensemble}$

calculate average SR and accuracy score $\{\frac{1}{2}(\text{SR}^{s_i} + \text{Acc}^{s_i})\}$ for students in $S_{Ensemble}$

return S_{Final}, student with max $\{\frac{1}{2}(\text{SR}^{s_i} + \text{Acc}^{s_i})\}$ score

Chapter 6

Results & Discussion

In this section, we evaluate our proposed framework, **RepKD**, which we introduced in the previous chapter, against the problem formulation (Section 5.1). We carry out ablation studies that support the design choices of our proposed framework. We also explore the efficiency of our method during training and inference. Finally, we look at the interpretability of the distilled models.

6.1 Experimental Setup

6.1.1 Datasets

We chose to test our method by using datasets in the medical domain due to the importance of reproducibility within the clinical setting.
The **Brain Genomics Superstruct Project (GSP)** dataset [31] was used. The GSP dataset is derived from structural and functional MRI scans, which are used to create cortical morphological networks (CMNs). Here, brains are represented as graphs, where nodes represent regions of interest (ROIs). ROIs are different anatomical regions of the brain, defined in Table C.2. Each edge quantifies the interaction between these ROIs. CMNs model the relationship between different ROIs using a specific measurement. We focused on four: maximum principal curvature C_1, cortical thickness network C_2, sulcal depth network C_3 and average curvature network C_4.
Each of the CMNs are comprised of 698 healthy individuals between the ages of 21 and 23 years, separated into two classes: male and female. Each CMNs are treated as an independent dataset since different morphological views have different distributions, which in turn means the graph structures derived from them differ. The number of nodes per graph is 35, the average number of edges is 298, and the average node degree is 17. The node feature matrix used is a one hot-encoded vector for each ROI.
The CMNs where generated in the following way. Firstly, the FreeSurfer software [80] was used to construct left and right cortical hemispheres. Then each cortical hemisphere was divided into 35 regions using the Desikan-Killiany atlas, defined in Table C.2. Subsequently, various measurements were utilized for each subject, for different ROIs, measurements of morphological differences in sulcal and gyral convolutions where taken for each CMNs. For more details refer to [81].

The second dataset we used was the **BreastMNIST** dataset from MedMNIST v2 [32]. MedMNIST contains a large collection of biomedical imaging datasets. In particular, BreastMNIST is based on 780 ultrasound images for breast cancer tumour classification. Images are classified into two classes: normal and benign, the positive class, and malignant, the negative class. Since we are using GNNs, we simply use the images as graphs. In [30], concerned with the reproducibility of GNN methods for federated learning, they do the same. Here, the number of nodes per graph is 28, the average number of edges is 135.58, and the average node degree is 10.11.
It is important to note that median thresholding was applied to all edges of the graphs, a common technique used so that the graphs are not fully connected. Here, an edge between two nodes is maintained if its weight is greater than the graph's median. This is done with CMNs to remove redundant and noisy edges between ROIs [82].

6.1.2 Baselines

To evaluate and demonstrate the performance of our proposed method, we compared it against a range of KD and KDG methods. The first baseline we used was **Vanilla** KD [24] which is the first proposed KD technique, as discussed previously in Subsection 2.2.2. It is a response-based method where knowledge is transferred via output logits. In particular by using SoftMax with temperature as described in equations 2.5 and through the loss function defined in equations 2.3 and 2.6.
The second baseline used is **FitNet** [25]. Here knowledge is transferred through the teacher logits, as before, but also through intermediate features (learned representations, e.g., the output of intermediate layers) of the models. The loss, defined in equation 3.1, is used to match the intermediate features between the teacher and student. Since we are dealing with GNNs, we use the notion that an intermediate feature map is the set of learned node embeddings of a particular layer of the GNN [26, 27]. In this project, we match the node embeddings learned in the final layers of the teacher and student models.
Our proposed method is designed for GNNs, and as such, we explore two KDG techniques. Firstly, local structure preserving KD, **LSP** [26]. LSP is the first method that considers the graph structure learned by GNNs. This is done with the use of the local structure preserving module, which takes into account the topological structure of the graph learned by the teacher and then matches it with the students learned graph structure during the distillation process, as covered in Subsection 2.2.2. The loss function is defined in equation 2.3 where the knowledge-specific loss is defined in equation 2.9.
The second graph-based KD technique used is multi-scale knowledge distillation **MSKD** [27], current state-of-the-art (SOTA) method. MSKD takes LSP one step further by using multiple teachers with different layers to transfer knowledge to the student model. Again, knowledge is transferred through the teacher's graph structures. However, an attention mechanism is used to determine the importance of each teacher during knowledge distillation, defined in equation 3.2, explanation can be found in Section 3.1. The loss function for MSKD is defined in equation 3.4. The MSKD paper suggests to stop incorporating teachers when you see a degradation in performance; in our case, this was at two-layer GNNs, as such, we use two teachers.

6.1.3 Teacher and Student Backbones

Having decided on the KD methods we will compare our method to, we also need to define the teacher and student architectures used with these methods. We used two GNNs to validate the generalizability of our proposed method. In particular, we use Graph Convolution Networks (GCN) [5] and Graph Attention Networks (GAT) [8], introduced and explained in Subsection 2.1.2.
We give a summary of the GNN teacher and student architectures. The **GCN teacher** is a 2-layer GCN. Where the node embedding dimensions per layer are 64 and 2, respectively. The **GCN student** is a 1-layer GCN, and the node embedding dimension is 2. The **GAT teacher** is a 2-layer GAT. The first layer has 8 attention heads, each with a node embedding dimension of 8. The second layer has one attention head with node embedding dimension 2. Finally, the **GAT student** is a 1-layer GAT. In this case, there is only one attention head with node embedding dimension 2.
It is worth noting that due to the small size of the input graph (35 $\times$ 35 for GSP and 28 $\times$ 28 for BreastMNIST) we are restricted in the number of layers used for the GCN and GAT. In fact, due to the over-smoothing problem, we saw a degradation in performance when using more than two layers. Furthermore, as explained in Subsection 5.2.1, the dimension of the final layer of the GNN needs to be equal to the number of classes so that it can be used in the topologically preserving aggregation function.

6.1.4 Shared Hyperparameters

In order to ensure a fair evaluation of the models, we attempted to use the same parameter settings among the models and only tune model-specific parameters for both the pre-trained models and KD baselines. Shared parameters are defined in the table 6.1.

Hyperparameter	Value
Training Epochs	50
Optmiser	Adam [83]
Optmiser Weight Decay	5x10-4
Batch Size	1
Graph Thresholding	Median
Soft Targets Temperature τ (if used)	3
KD α* (if used)	1

Table 6.1: Table of shared hyperparameters. * for GSP datasets, tuned for BreastMNIST.

The baselines we have introduced all have different hyperparameters, which in turn need to be tuned. In order to do this, we simply used the hyperparameter values that were used in the literature for each of the methods and chose others empirically, based on the lowest validation loss. In-depth details of the student and teacher GNN hyperparameters can be found in table A.1. The hyperparameter settings details for each KD baseline can be found in table A.2 for each dataset. Finally, we also provide a RepKD-specific table that defines the number of students used for each GNN architecture combination; this can be found in table A.3. For completeness, the hyperparameter

search space can be found in table A.4. It is worth noting that due to the large search space, limited access to resources, and difficulty of training models for 3, 5, and 10-fold cross validation across 10 seeds, for all KD baseline methods we used the same hyperparameters that were found with a pre-trained GCN teacher and GCN student architecture when exploring intra- and cross-model performance.

6.1.5 Training

It is important to understand how training is undertaken to understand the results. Due to the definition of self-reproducibility, **all** model needs to be trained using 3, 5, and 10-fold cross-validation. For each cross-validation, we report the performance of the model on each of the test splits and then evaluate the self-reproducibility score using four different thresholds values $K = \{5, 10, 15, 20\}$.
Furthermore, in order to ensure fairness in our experiments, we have followed the guidelines set out in [84]. Firstly, we used stratified splits, such that class distributions are conserved. Furthermore, what needs to be carefully considered when using offline KD methods is that when training student models, the pre-trained teacher is trained on the same data to prevent data leakage. More importantly, **all experiments** were carried out across **10 different seeds**, values 0 through 9, in order to mitigate biases in the weight initialization or data splits [84], we report the average performance over these runs. All of the experiments were carried out on an Intel Core i7-10700 @2.90GHz CPU with Ubuntu 22.04 Linux OS.
Moreover, when analysing our results, we will use the following notation: "GCN" or "GAT" followed by "T". or "S.", this simply refers to the GNN architecture, and if it is the teacher or the student, "T." and "S"., respectively. Furthermore, these models are trained **without** knowledge distillation; they are simply trained using an Adam optimiser and with the parameters explained in Subsection 6.1.4.

6.2 Performance

We begin by evaluating our proposed KD method, **RepKD**, under two different scenarios: intra-model distillation, where the teacher and student have the same architecture, and cross-model distillation, where the teacher and student have different architectures. To evaluate our proposed method, we look at two metrics: the self-reproducibility score, which is the main concern of our model, and accuracy (since we ensure balanced splits across all classes) to assess if knowledge distillation has taken place. This metric is extensively used when comparing KD methods, for example in [24, 26, 25].

6.2.1 Intra-Model Performance

We begin by comparing the intra-model performance of our model against the different baseline KD methods. In particular, we are distilling from a teacher to a student with the same GNN architecture; GCN to GCN and GAT to GAT.
Table 6.2 shows the intra-model self-reproducibility score for all models and datasets. We can immediately see clear results. Firstly, we can see that **across all architectures**

Model	C_1	C_2	C_3	C_4	BreastMNIST	Average
GCN T.	89.14±3.74	88.88±3.39	90.60±4.47	90.17±5.46	89.21±5.45	89.60±4.50
GCN S.	92.60±4.64	94.04±2.20	91.17±3.41	93.29±2.70	97.69±1.96	93.76±2.98
Vanilla	90.07±4.76	92.85±3.53	90.57±5.43	92.58±2.63	96.54±2.17	92.52±3.70
FitNet	63.50±8.25	70.28±11.26	67.86±10.11	73.93±9.17	92.76±2.49	73.67±8.25
LSP	70.49±11.52	77.31±8.33	74.40±12.75	77.92±7.74	91.50±4.82	78.32±9.03
MSKD	82.44±4.52	81.85±7.27	78.61±9.17	79.46±11.05	90.64±3.95	82.60±7.19
RepKD	**96.00±1.92**	**96.89±0.77**	**96.50±1.11**	**96.90±1.39**	**96.75±1.45**	**96.61±1.32**
GAT T.	91.72±3.31	93.33±3.36	91.60±5.21	92.42±4.06	73.36±10.82	88.49±5.35
GAT S.	91.94±3.33	90.74±4.07	91.99±3.43	91.75±3.74	99.47±1.01	93.18±3.12
Vanilla	90.81±4.78	90.28±4.77	90.06±5.31	91.21±3.37	97.11±5.72	91.89±4.79
FitNet	78.00±14.75	84.13±11.08	74.49±14.53	85.22±7.39	88.54±7.63	82.08±11.08
LSP	88.99±6.75	85.76±5.43	82.21±8.76	85.11±6.53	91.60±3.32	86.73±6.16
MSKD	88.10±7.37	84.47±5.16	81.97±9.20	85.44±8.70	91.04±4.00	86.21±6.88
RepKD	**95.33±1.80**	**95.86±0.39**	**97.96±1.65**	**97.56±0.55**	**98.96±0.63**	**97.13±1.00**

Table 6.2: Intra-model self-reproducibility score. **Bold** and Underline indicate the best and second best performance among the KD methods. Top GCN to GCN and bottom GAT to GAT. Visualisations in Figures B.1 and B.2.

and datasets, RepKD achieves greater self-reproducibility scores than all other KD methods, achieving an average self-reproducibility score of 96.61 and 97.13 for GCN and GAT architectures, respectively. This clearly shows the robustness and ability to conserve the reproducibility of distilled models of RepKD under different scenarios, one of the main objectives of the method.

Secondly, beyond superior self-reproducibility performance with respect to other KD methods **RepKD demonstrates its effectiveness in many cases by not only maintaining but enhancing the self-reproducibility score with respect to both student models (trained without KD) and teacher models.** In particular, we saw a percentage increase in self-reproducibility of 3.04% and 4.25%, for models trained with RepKD compared to models trained without KD, GCN S. and GAT S., respectively. Furthermore, substantial improvement was observed in models trained with RepKD in relation to the teacher models, in particular a percentage increase in self-reproducibility of 7.83% and 9.77% with respect to GCN T. and GAT T. This indicates that the method not only preserves reproducibility but also promotes consistent model weights across different data perturbation strategies. In turn, this improvement contributes to the increased interpretability of the distilled models.

Thirdly, we look at the standard deviation of the different methods. We can see that the average standard deviation of RepKD is 1.32 and 1.00 for GCN and GAT architectures, respectively. When comparing these values with respect to the other KD methods, we can see that **RepKD offers the greatest model stability**. This suggests that RepKD can find similar model weights (since reproducibility is based on the final layer model weights) independently of their initialization due to the use of ensembles. This inherent stability enhances the reliability of the models, as the self-reproducibility score remains

Model	C_1	C_2	C_3	C_4	BreastMNIST	Average
GCN T.	61.01±0.30	66.16±0.09	66.91±0.09	65.44±0.12	72.48±0.31	66.40±0.18
GCN S.	57.78±0.28	63.75±0.20	63.25±0.11	63.62±0.09	69.21±0.66	63.52±0.27
Vanilla	58.16±0.24	63.98±0.14	63.75±0.15	63.77±0.23	70.34±0.60	64.00±0.27
FitNet	60.73±0.57	64.21±0.39	65.20±0.31	64.45±0.20	73.32±0.08	65.58±0.31
LSP	61.21±0.27	65.58±0.31	66.43±0.08	65.78±0.13	73.41±0.17	66.48±0.19
MSKD	**62.18±0.14**	**65.72±0.38**	**66.51±0.26**	**65.79±0.09**	**73.99±0.04**	**66.84±0.18**
RepKD	60.13±0.18	61.95±0.05	64.02±0.06	63.19±0.20	72.68±0.13	64.39±0.13
GAT T.	62.03±0.31	67.77±0.09	65.29±0.23	66.40±0.55	72.92±0.09	66.88±0.26
GAT S.	57.52±0.49	61.17±0.07	60.29±0.19	59.96±0.08	55.59±0.08	58.91±0.18
Vanilla	58.05±0.38	63.12±0.19	61.13±0.03	60.65±0.15	56.91±0.08	59.97±0.16
FitNet	62.52±0.16	66.37±0.26	65.85±0.26	66.46±0.23	73.54±0.16	66.95±0.21
LSP	63.28±0.20	66.40±0.35	66.67±0.22	66.83±0.05	73.42±0.07	67.32±0.18
MSKD	**63.45±0.35**	**66.92±0.51**	**66.93±0.11**	**66.48±0.13**	73.60±0.18	**67.47±0.25**
RepKD	57.69±0.30	64.69±0.15	56.03±0.05	56.07±0.05	**74.10±0.04**	61.72±0.22

Table 6.3: Intra-model average test-set accuracy across 3, 5 and 10-fold cross validation. Top GCN to GCN and bottom GAT to GAT. **Bold** and Underline indicate the best and second best performance among the KD methods.

more consistent across different seeds in comparison to other KD methods.

We now shift our attention to Table 6.3, which shows the average 3, 5, and 10-fold cross-test accuracy for all models and datasets. It is important to analyse this since our KD model should also be able to distil performance (e.g., accuracy) from the teacher to the student. We can see, as expected, that MSKD, the state-of-the-art method for KD on graphs, achieves the best accuracy across all datasets and architectures. However, from Table 6.2 we can see that this is at the expense of self-reproducibility. In particular, RepKDs self-reproducibility score on average outperforms MSKD by 16.96% and 12.68%, for GCN and GAT architectures, respectively. This shows the difficulty of finding a balance between reproducibility and performance.

However, upon closer inspection, we can see that while preserving self-reproducibility, **RepKD manages to successfully distil performance into the student model**. On average across all datasets and for all architectures, models trained with RepKD compared to ones trained without KD saw an increase in accuracy from 63.52 to 64.39 (1.37% increase) and 58.91 to 61.72 (4.78% increase) for GCN and GAT architectures, respectively. Furthermore, we can see that **RepKD consistently outperforms Vanilla KD**. In particular, RepKD sees an average accuracy increase of 0.62% and 2.92% over Vanilla KD (for GCN and GAT architectures, respectively). Showing that while preserving self-reproducibility, it also manages to see considerable performance increases over other KD methods.

It is worth mentioning that our method outperformed other KD methods under one scenario when using the GAT architecture for the BreastMNIST dataset. We speculate that the outcome can be attributed to the nature of the data, where images are treated as graphs. Since GNNs and KDG methods are designed specifically for graph-structured

data, Furthermore, as mentioned before, we use the same model parameters found for GCN architectures, which in turn could indicate that certain models could potentially be experiencing suboptimal performance.

6.2.2 Cross-Model Performance

We now evaluate our method under a more difficult scenario, cross-model performance. The ability for a KD method to distil knowledge from a teacher and student with different GNN architectures; GCN to GAT and GAT to GCN.

Model	C_1	C_2	C_3	C_4	BreastMNIST	Average
GCN T.	89.14±3.74	88.88±3.39	90.60±4.46	90.17±5.46	89.21±5.45	89.60±4.50
GAT S.	91.94±3.33	90.74±4.07	91.99±3.43	91.75±3.74	99.47±1.01	93.18±3.12
Vanilla	89.49±4.88	84.63±9.63	86.76±9.48	88.40±6.58	**98.11±1.79**	89.48±6.47
FitNet	74.36±10.09	79.03±13.03	73.75±12.56	76.83±12.62	91.44±3.93	79.08±10.44
LSP	88.85±5.83	85.29±5.40	82.57±7.95	87.63±6.08	91.06±3.22	87.08±5.70
MSKD	84.47±9.77	83.06±6.95	85.31±4.78	83.67±4.83	91.79±3.51	85.66±5.97
RepKD	**95.11±2.73**	**92.97±1.82**	**93.07±2.70**	**94.03±1.14**	96.33±0.96	**94.30±1.87**
GAT T.	91.72±3.31	93.33±3.36	91.60±5.21	92.42±4.06	73.36±10.81	88.49±5.35
GCN S.	92.60±4.64	94.04±2.20	91.17±3.41	93.29±2.70	97.69±1.96	93.76±2.98
Vanilla	91.18±3.31	91.88±2.19	93.17±3.09	93.08±2.66	96.89±1.75	93.24±2.60
FitNet	76.93±9.81	79.32±8.56	78.24±9.22	79.74±4.68	90.06±4.16	80.86±7.29
LSP	73.31±7.64	80.11±3.66	81.97±7.07	83.82±5.18	92.38±3.09	82.32±5.33
MSKD	71.64±10.14	80.47±5.63	80.33±8.50	82.53±3.79	91.88±2.79	81.37±6.17
RepKD	**97.31±0.76**	**95.31±1.53**	**96.11±0.80**	**96.19±1.86**	**97.18±0.97**	**96.42±1.18**

Table 6.4: Cross-model self-reproducibility score. **Bold** and Underline indicate the best and second best performance among the KD methods. Top GCN to GAT and bottom GAT to GCN. Visualisations in Figures B.3 and B.4.

From Table 6.4 we can see results that are aligned with the previous section. Firstly, **across all architectures and datasets, RepKD achieves greater self-reproducibility scores than all other KD methods** (except for GCN T. to GAT S. for BreastMNIST), achieving an average self-reproducibility score of 94.30 and 96.42 for GCN T. to GAT S. and GAT T. to GCN S., respectively. Again, showing the robustness of RepKD. Furthermore, **RepKD again demonstrates its effectiveness by enhancing the self-reproducibility score with respect to both student models (trained without KD) and teacher models**. In particular, we saw a percentage increase in self-reproducibility of 1.21% and 2.84%, for models trained with RepKD with respect to models trained without KD, GCN T. to GAT S. and GAT T. to GCN S., respectively. Like before, improvement was observed in models trained with RepKD in relation to the teacher models, in particular a percentage increase in self-reproducibility of 5.25% and 8.97% for GCN T. and GAT T., respectively.
Again, we look at the standard deviation of the different models. We can see that the average standard deviation of RepKD is 1.87 and 1.18 for GCN T. to GAT S. and GAT T.

to GCN S., respectively. **As before, RepKD offers the greatest model stability**.

Model	C_1	C_2	C_3	C_4	BreastMNIST	Average
GCN T.	61.01±0.30	66.16±0.09	66.91±0.09	65.44±0.12	72.48±0.31	66.40±0.18
GAT S.	57.52±0.48	61.17±0.07	60.29±0.19	59.96±0.08	55.59±0.08	58.91±0.18
Vanilla	57.77±0.41	62.50±0.08	61.32±0.19	61.28±0.20	56.79±0.10	59.93±0.20
FitNet	63.47±0.20	66.87±0.04	66.64±0.24	66.68±0.02	73.50±0.06	67.43±0.11
LSP	63.43±0.28	66.27±0.19	66.21±0.23	66.68±0.08	73.44±0.11	67.21±0.18
MSKD	**63.80±0.22**	**67.46±0.46**	**67.41±0.13**	**66.88±0.10**	73.51±0.05	**67.81±0.19**
RepKD	56.01±0.08	64.32±0.37	56.06±0.13	56.06±0.08	**73.98±0.03**	61.28±0.14
GAT T.	62.03±0.31	67.77±0.09	65.29±0.23	66.40±0.55	72.92±0.09	66.88±0.26
GCN S.	57.78±0.28	63.75±0.20	63.25±0.11	63.62±0.09	69.21±0.66	63.52±0.27
Vanilla	58.60±0.31	64.19±0.15	63.70±0.24	63.68±0.30	70.36±0.31	64.12±0.26
FitNet	**62.97±0.12**	64.15±0.27	66.40±0.23	65.15±0.36	73.37±0.10	66.41±0.22
LSP	60.40±0.31	64.02±0.05	64.93±0.27	65.06±0.05	**73.65±0.10**	65.81±0.16
MSKD	60.79±0.24	**65.06±0.07**	**65.19±0.41**	**65.08±0.28**	73.43±0.16	**65.91±0.23**
RepKD	60.23±0.17	61.31±0.03	64.34±0.11	62.81±0.34	70.77±0.04	63.89±0.14

Table 6.5: Cross-model average test-set accuracy across 3, 5 and 10-fold cross validation. Top GCN to GAT and bottom GAT to GCN. **Bold** and Underline indicate the best and second best performance among the KD methods.

Table 6.5, which shows the average 3, 5, and 10-fold cross-test accuracy for all models and datasets. As before, we can see that MSKD achieves the best accuracy on average across all datasets and architectures. Again, at the expense of reproducibility, as seen in Table 6.4. RepKD, again, manages to successfully distil performance into the student model under the cross-model scenario. Models trained with RepKD compared to ones trained without KD saw an increase in accuracy from 58.91 to 61.28 (4.04% increase) and 63.52 to 63.89 (0.58% increase) for GCN T. to GAT S. and GAT T. to GCN S., respectively. Similar to the intra-model scenario, RepKD outperforms Vanilla KD when distilling from GCN to GAT but slightly underperforms when distilling from GAT to GCN. As a result, we can see that, in terms of the self-reproducibility score, **RepKD is robust to the GNN architecture used**. Moreover, looking at accuracy, the trend remains consistent with intra-model performance. RepKD consistently accomplishes the task of effectively transferring performance to the students across all scenarios, aligning with the expectations of any knowledge distillation approach.
What becomes evident from intra-model and cross-model performances is that **reproducibility and accuracy are inversely correlated**. We can see that in many cases, while accuracy increases, this usually comes at the expense of reproducibility. For example, when considering intra-model accuracy, MSKD and LSP are the first and second best methods, respectively (Table 6.3) but neither have the best reproducibility score, beaten by Vanilla KD and RepKD (Tables 6.2). However, RepKD manages to find a balance between reproducibility and accuracy while also increasing the latter. This will be a recurring theme throughout the next sections.

6.3 Ablation Study

In this section, we ablate different components of the loss function defined in equation 5.10, such that we can discuss and understand the importance of each loss and how they contribute to the performance of the proposed method. In particular, we look at the GCN teacher-to-GCN student scenario for all datasets. For the following metrics: self-reproducibility score, accuracy, and F1-score.

Metric	Dataset	RepKD	RepKD w/o $\mathcal{L}_{CE}$	RepKD w/o $\mathcal{L}_{ESP}$	RepKD w/o $\mathcal{L}_{ETS}$	RepKD w/o $\mathcal{L}_{IS}$
SR Score	C_1	96.00±1.92	95.90±2.89	95.97±1.95	**97.89±1.53**	95.49±1.31
	C_2	96.89±0.77	97.40±2.25	96.81±0.65	**97.72±0.17**	95.29±1.34
	C_3	**96.50±1.11**	95.61±1.48	**96.50±1.11**	92.97±0.58	95.04±2.96
	C_4	96.90±1.39	95.61±1.97	96.90±1.39	**97.14±0.25**	92.03±1.90
	BreastMNIST	**96.75±1.45**	92.11±3.15	**96.75±1.45**	95.25±0.50	92.04±3.34
	Average	**96.61±1.33**	95.33±2.35	96.59±1.31	96.19±0.61	93.98±2.17
Accuracy	C_1	60.13±0.96	55.20±1.82	60.13±0.98	55.25±0.11	**62.02±0.56**
	C_2	61.95±0.41	65.09±0.80	61.96±0.40	59.95±0.12	**62.78±0.37**
	C_3	64.02±0.24	62.48±1.05	64.05±0.24	60.83±0.27	**66.33±0.51**
	C_4	63.19±0.39	56.70±0.97	63.18±0.41	55.46±0.14	**64.91±0.16**
	BreastMNIST	72.68±0.34	69.11±1.39	72.68±0.34	71.98±0.07	**73.52±0.34**
	Average	64.40±0.47	61.72±1.21	64.40±0.47	60.70±0.14	**65.91±0.39**
F1-Score	C_1	**61.59±2.59**	35.40±10.98	61.56±2.59	44.31±1.25	61.47±1.09
	C_2	58.50±1.28	**68.26±3.80**	58.46±1.28	54.80±0.26	60.56±0.90
	C_3	60.39±1.01	65.57±5.06	60.43±1.16	59.38±1.07	**66.61±2.05**
	C_4	59.97±0.45	51.14±10.33	59.43±0.47	55.04±0.56	**60.52±0.63**
	BreastMNIST	78.01±0.83	73.17±0.28	78.01±0.83	73.64±0.08	**78.66±0.82**
	Average	63.69±1.23	58.71±6.09	63.58±1.26	57.43±0.64	**65.57±1.10**

Table 6.6: Ablation study results - self-reproducibility score (SR score) and average test accuracy and F1-Score across 3, 5 and 10-fold cross validation. **Bold** and Underline indicate the best and second best performance. "w/o" means without.

From Table 6.6 we can see RepKD achieves the highest average self-reproducibility and the second-best accuracy and F1-score across all datasets. As we shall see, the largest accuracy and F1-score come at the expense of reproducibility, and as such, we argue that the final RepKD loss $\mathcal{L}_{Final}$ strikes a balance between performance and reproducibility.

6.3.1 Teacher-Student Loss Parameter

The ensemble teacher-student loss $\mathcal{L}_{ETS}$, defined in equation 5.9, is used to distil knowledge from the teacher to each student in the ensemble individually in order to promote diversity and ensure performance. From Table 6.6 we can see that on average the self-reproducibility score decreases from 96.61 to 96.19 when ablating $\mathcal{L}_{ETS}$. This shows that $\mathcal{L}_{ETS}$ is effective in promoting diversity within the ensemble, which in turn allows

us to find reproducible models. Furthermore, and more importantly, when looking at Table 6.6 we can see that when $\mathcal{L}_{ETS}$ is ablated, we see the largest decrease in average accuracy and F1-score across all the ablated versions of RepKD; accuracy decreases from 64.40 to 60.70 and F1-score decreases from 63.69 to 57.43. This loss was incorporated to ensure that each student individually has similar performance as that of the teacher and to ensure good-performing students during the selection process. This highlights the importance of ensemble teacher-student loss for model performance and self-reproducibility. As a result, we showed that by abalating $\mathcal{L}_{ETS}$ we have validated the use of $\mathcal{L}_{ETS}$.

6.3.2 Intra-Student Loss Parameter

The intra-student loss $\mathcal{L}_{IS}$, defined in equation 5.8, promotes diversity within the ensemble by pushing the weights of the final layer away from each other. In particular, the cosine similarity is used for all pairs of students such that they are orthogonal to one another. This is illustrated in Figure 6.1 which shows the cosine similarity between all the weight vectors in the ensemble. We can see that when $\mathcal{L}_{IS}$ is used (left figure), all vectors are orthogonal to one another, as desired. In contrast, when $\mathcal{L}_{IS}$ (right figure) is not used, we can see that some weight vectors are correlated.
From Table 6.6 we can see that RepKD without $\mathcal{L}_{IS}$ sees on average the largest degradation of the self-reproducibility score, from 96.61 to 93.98. To explain this, we can look at $\mathcal{L}_{IS}$, which uses the cosine similarity between the weight vectors of the final layer between all pairs of students. However, this process confines the weight vectors to a limited region within the weight space. As a result, this increases the reproducibility of the models due to the restriction imposed by the orthogonality among the students due to the fact that the weight vectors are confined within a defined region.
However, this restriction in turn limits the student model performance; RepKD without $\mathcal{L}_{IS}$ achieves the largest average accuracy and F1-score, as seen from Table 6.6. Alleviating the orthogonality restriction imposed by $\mathcal{L}_{IS}$ allows for more flexibility of the vectors to explore the weight space during the distillation process. In turn, increasing the distilled model's performance comes at the expense of reproducibility. This highlights the difficulty of striking a balance between model self-reproducibility and performance. We justify using $\mathcal{L}_{IS}$ since we prioritise self-reproducibility over performance due to increased model interpretability and trustworthiness. However, these are choices that need to be carefully considered and decided for the intended use case of the distilled model.

6.3.3 Ensemble Losses

The ensemble losses, $\mathcal{L}_{CE}$ and $\mathcal{L}_{ESP}$, from Subsection 5.2.1, are designed for collaborative learning within the ensemble. From Table 6.6, we can see that self-reproducibility, accuracy, and F1-score all decrease when RepKD is trained without $\mathcal{L}_{CE}$. This comes as no surprise since $\mathcal{L}_{CE}$ is the only loss through which the ensemble learns from the ground truth labels. As such, during distillation, the student model cannot learn from the ground truth, and as such, we see sub-optimal performance. To explain the de-

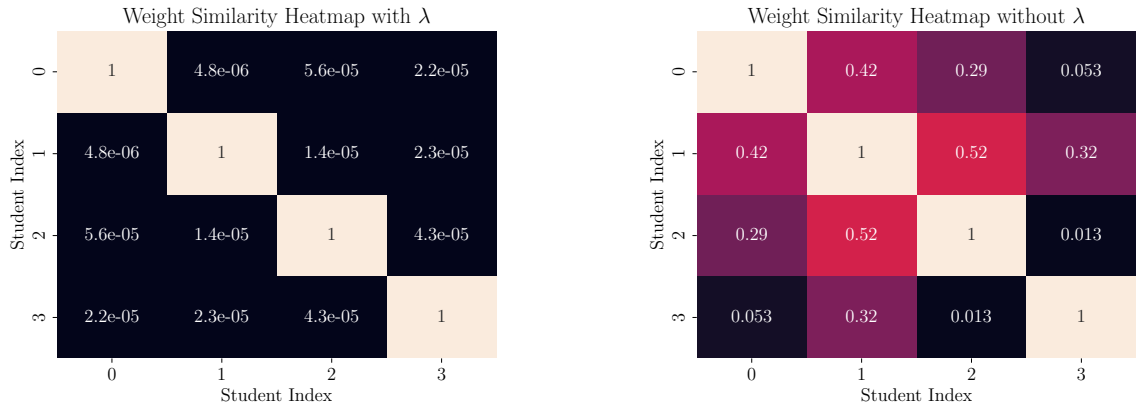

Figure 6.1: An example of the cosine similarity score between the weight vectors of the students in the ensemble for the C_1 dataset.

crease, we attribute this again to learning procedures. Since there are no ground truth labels, the other losses inject noise, which in turn leads to more variance in the learned weights, decreasing the self-reproducibility score.
Turning out attention to $\mathcal{L}_{ESP}$, defined in equation 5.6. Recall that this loss transfers knowledge through the learned graph structure of the ensemble and teacher. RepKD without $\mathcal{L}_{ESP}$ has the second-best average self-reproducibility score and accuracy but lags behind in terms of the F1-score. This highlights the importance of transferring graph structure knowledge from the teacher to the ensemble when dealing with graph-structured data, since without it we see a decrease in F1-score. thus justifying the use of $\mathcal{L}_{ESP}$ in the final RepKD loss $\mathcal{L}_{Final}$.
Furthermore, we provide supplementary experiments to further justify *why graph structure is used as knowledge in* $\mathcal{L}_{ESP}$. To do so, we considered two other forms of knowledge. Firstly, we considered knowledge distillation through the output **logits** of the ensemble and the teacher. Secondly, we considered matching the learned **node embeddings** of the ensemble (average node embedding of each student in the ensemble) and teacher, the same way knowledge is transferred in FitNet.
Table 6.7 shows that transferring knowledge through the ensemble local graph structure (our method) results in the highest self-reproducibility score and the second highest accuracy when compared to other forms of knowledge. Experiments were carried out with varying numbers of students in ensemble, and the same conclusions were found in Tables B.1 and B.2. As a result, these findings justify the type of knowledge used in $\mathcal{L}_{ESP}$, since we prioritised self-reproducibility over performance in RepKD.

6.3.4 Selection Process

Finally, we provide supplementary experiments to justify *why we pick the student in the ensemble with the highest weighted self-reproducibility score and accuracy?* Although these experiments are not explicitly intended as ablation studies, their inclusion in this section is due to the fact that they provide evidence for design choices in RepKD. We investigate five distinct selection criteria for choosing the student. These criteria en-

Model	C_1	C_2	C_3	C_4	BreastMNIST	Average
$\mathcal{L}_{ESP}$	96.00±1.92	**96.89±0.77**	**96.50±1.11**	**96.90±1.39**	**96.75±1.45**	**96.61±1.33**
$\mathcal{L}_{ESP}$ (**L**)	**97.22±1.72**	95.31±2.26	96.11±1.18	95.53±1.17	96.14±1.77	96.06±1.33
$\mathcal{L}_{ESP}$ (**E**)	93.43±5.40	93.56±4.88	92.17±6.04	92.94±2.82	92.74±4.07	92.97±4.64
w/o $\mathcal{L}_{ESP}$	95.97±1.95	96.81±0.65	**96.50±1.11**	**96.90±1.39**	**96.75±1.45**	96.59±1.31

Model	C_1	C_2	C_3	C_4	BreastMNIST	Average
$\mathcal{L}_{ESP}$	60.13±0.96	61.95±0.41	64.02±0.24	63.19±0.39	72.68±0.34	64.40±0.47
$\mathcal{L}_{ESP}$ (**L**)	**60.33±1.03**	**62.05±0.36**	**64.09±0.27**	**63.42±0.47**	**72.74±0.32**	**64.53±0.49**
$\mathcal{L}_{ESP}$ (**E**)	55.38±1.43	58.58±2.21	57.08±1.93	59.46±3.27	53.32±1.19	56.76±2.01
w/o $\mathcal{L}_{ESP}$	60.13±0.98	61.96±0.40	64.05±0.24	63.18±0.41	72.68±0.34	64.40±0.47

Table 6.7: Varying how knowledge is distilled through $\mathcal{L}_{ESP}$. Top table is the self-reproducibility score, bottom table is accuracy. **Bold** and Underline indicate the best and second best performance 'w/o" means without. "**L**" refers to logits and "**E**" refers to node embeddings, the type of knowledge used in $\mathcal{L}_{ESP}$.

compass selecting the student based on: self-reproducibility (max rep), max accuracy (max acc), max F1-score (max f1), average of the student's self-reproducibility score and F1-score (weighted f1), and average of the student's self-reproducibility score and accuracy (weighted acc).

Figure 6.2 shows the best student for the different selection criteria used. The best students can be found in the top-right sector of the plot; both scores are maximised. We can see from both plots that the "weighted acc" is more often than not located in this sector, suggesting that this heuristic can find students that balance both metrics. However, it is worth noting that the"weighted f1" heuristic often finds the same student and, as such, could also be used. Furthermore, it can be seen that our method is sensitive to the number of students used and, as such, should be tuned carefully to get the best performance from the final selected student.

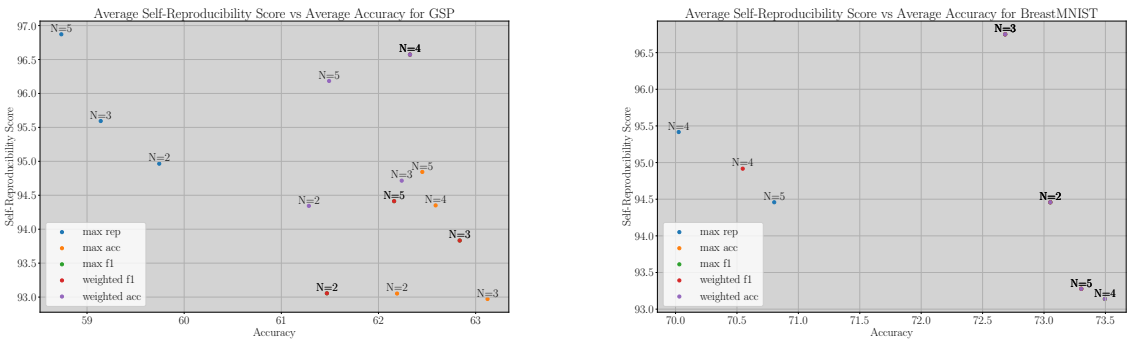

Figure 6.2: Average self-reproducibility score against accuracy for varying number of students in ensemble N for GSP (left) and BreastMNIST (right) datasets.

6.4 Parameter Sensitivity Study

In this section, we explore how varying different parameters affect **RepKD**. In particular, we look at the teacher-student loss parameter γ, intra-student loss parameter λ, soft target temperature parameter τ and the number of students in the ensemble N. The reason for this is to further highlight the difficulty of finding a balance between reproducibility and performance. For conciseness, we look at the GCN teacher-to-GCN student scenario for the GSP dataset. When varying hyperparameters, others remain fixed, as defined in Tables A.2 and A.3.

6.4.1 Teacher-Student Loss Parameter

Recall, γ controls $\mathcal{L}_{ETS}$, defined in equation 5.9, which is used to distil knowledge from the teacher to each student in the ensemble individually. A higher value means that the teacher influences the students more during the distillation process. We begin by varying the values of $\gamma = \{0, \frac{1}{9}, \frac{2}{9}, \frac{3}{9}, \frac{5}{9}, \frac{7}{9}, 1, 1.5\}$, they are in this range since we need to take into account τ^2 in equation 5.9 where $\tau = 3$ for all experiments.

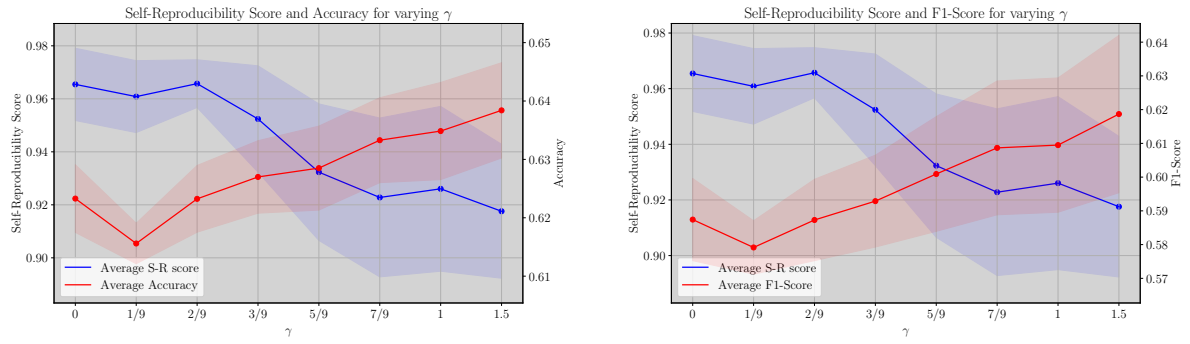

Figure 6.3: Parameter sensitivity study for γ.

From Figure 6.3 we can see that as γ **increases, self-reproducibility decreases, and performance (accuracy and F1-score) increases**, intersecting at $\gamma \approx \frac{5}{9}$. As γ increases, the teacher has more influence over the students during the distillation process, which in turn increases their performance. We suspect that the decrease in self-reproducibility is due to the fact that the distillation process is being guided by a less reproducible model. As a result, it incorporates noise during distillation, which, in turn, degrades the student's self-reproducibility score.

6.4.2 Intra-Student Loss Parameter

We now look at λ which controls $\mathcal{L}_{IS}$, defined in equation 5.8, and is used to push the weights in the final layer of each student away from each other. A high value of λ signifies increased differentiation among the students within the ensemble, whereas a lower value permits the students to be in closer proximity to each other. We vary λ in the following range: $\{0, 0.2, 0.4, 0.6, 0.8, 1, 1.5, 2\}$.

Figure 6.4 shows that as λ **increases, self-reproducibility increases and performance (accuracy and F1-score) decreases**, intersecting at $\lambda \approx 0.7$. This is similar to what we saw in Subsection 6.3.2 ($\lambda = 0$). As λ increases, $\mathcal{L}_{IS}$ has more influence, which results in the weight vectors being confined to a smaller region within the weight space. In turn, this increases the reproducibility of the models since the weights are constrained to a small space, thus not changing during training. However, this constraint reduces the ability of the weights of the model to update during the distillation process, and as a result, performance decreases as λ increases.

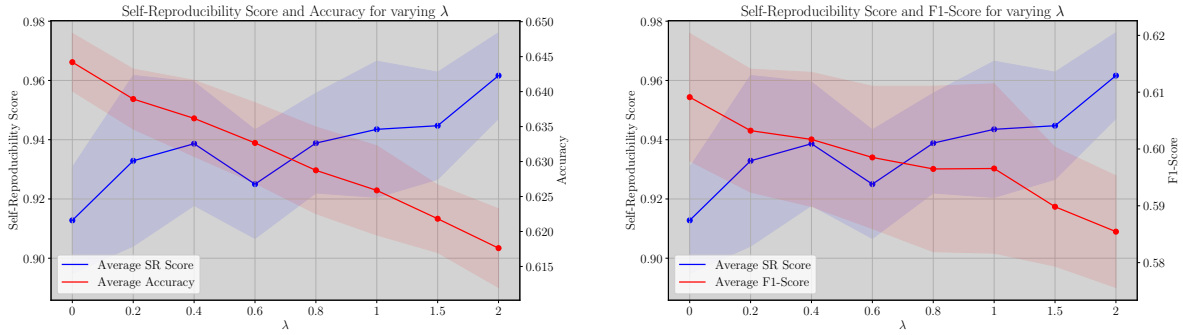

Figure 6.4: Parameter sensitivity study for λ.

6.4.3 Soft Target Temperature

We now turn our attention to the soft target temperature parameter τ, in $\mathcal{L}_{ETS}$, defined in equation 5.9, which is used to change the distribution of the output probability distributions we are matching. If $\tau > 1$, the resulting distribution becomes softer, causing the probabilities of various classes to be more evenly spread out. When $\tau < 1$, the predicted probability distribution becomes sharper, concentrating more probability mass on the most probable class. We vary τ in the following range: $\{0.5, 1, 2, 3, 4, 5\}$.
Figure 6.5, shows that as τ **increase, self-reproducibility increases and plateaus at** $\tau = 3$ **and performance (accuracy and F1-score) decreases**. When τ is small, the teacher's soft predictions become more diffuse. This means that the teacher assigns similar probabilities to all classes, and since the student is exposed to a more diverse set of class assignments, this in turn allows for better student generalisation and thus better performance. However, as τ increases, the teacher's predictions become overly confident, which reduces the ability of the student to learn; hence, we see a decrease in performance.
From the point of view of reproducibility, when τ is small, the distillation process becomes more uncertain due to the close resemblance of class distributions. In turn, this results in noisier learned weights during the distillation process, degrading the self-reproducibility score. Conversely, as τ increases, the teachers soft predictions become more stable, and as a result, there is less noise during distillation, stabilising the reproducibility. In fact, this is why we see a plateau at $\tau = 3$, since the teacher's soft predictions converge, resulting in a more stable learning procedure. thus stabilising the self-reproducibility score of the student model.

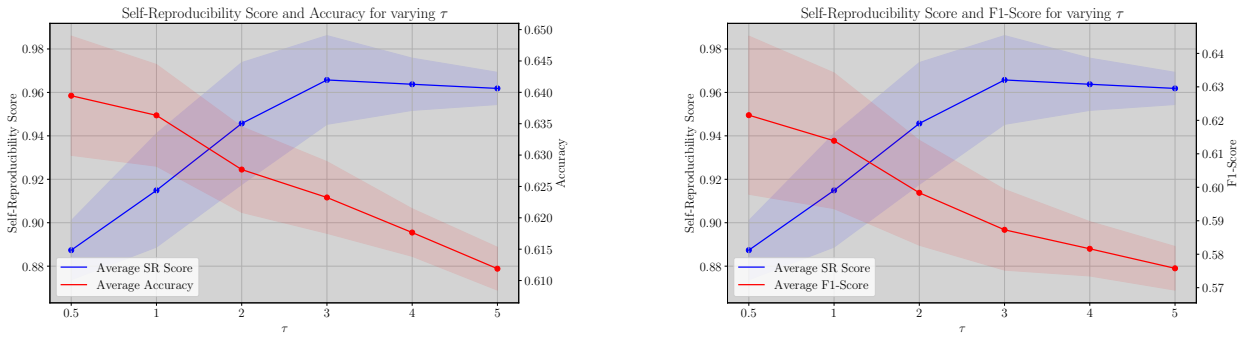

Figure 6.5: Parameter sensitivity study for τ.

6.4.4 Ensemble Size

Finally, we look at the number of students used in the ensemble N. We vary N from 2 to 5. RepKD does not use the ensemble during inference but instead selects one student from the ensemble based on the average self-reproducibility and accuracy score, as explained in Subsection 5.2.2. We can see from Figure 6.6 that **strictly increasing N does not increase self-reproducibility nor performance**. In fact, we can see that self-reproducibility and accuracy peak when $N = 4$. However, the F1-Score achieves its maximal value at $N = 3$. As a result, deciding on N depends on the intended use case of the final student, and as such, N should be carefully tuned.

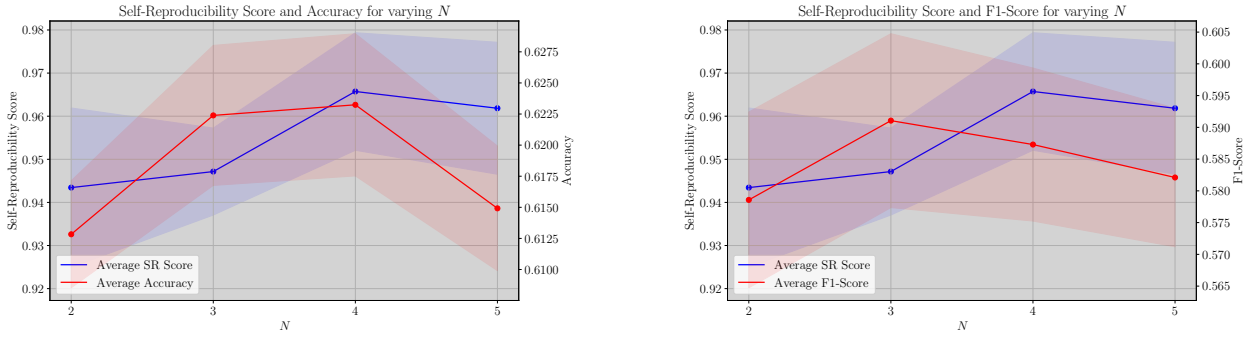

Figure 6.6: Parameter sensitivity study for number of students in ensemble N.

6.5 Model Efficiency

Ensuring our models competitiveness in terms of training time, memory usage, and parameter utilisation with respect to other KD methods is an important aspect to consider for their practical usage. A model that has a large training time or memory consumption in practice might not be the best solution or cannot be run given limited resources. Additionally, an efficient model facilitates cost reduction both during the training phase

and the inference stage. As a result, in this section, we look at our models efficiency during training and the performance of the final model, which is used during inference.

6.5.1 Training Efficiency

We look at the average training time and memory usage per epoch for 3-fold cross validation over 50 epochs (the standard deviation is omitted from reporting due to its negligible magnitude). For the GSP dataset, we report the statistics for the cortical morphological network (CMN) C_1, because the sizes of the different CMN datasets are the same. Furthermore, the model parameters from Section 6.2 were used for a fair comparison.

		Time Per Epoch (sec)						
Dataset	GNN	T.	S.	Vanilla	FitNet	LSP	MSKD	RepKD
GSP	GCN	0.31	0.23	0.41	0.43	1.61	2.30	2.56
	GAT	2.21	0.22	2.08	2.10	3.09	4.08	3.63
BreastMNIST	GCN	0.31	0.25	0.46	0.48	1.45	2.26	2.30
	GAT	2.21	0.40	2.05	2.10	3.09	3.93	3.77

		Memory Per Epoch (GB)						
Dataset	GNN	T.	S.	Vanilla	FitNet	LSP	MSKD	RepKD
GSP	GCN	0.42	0.39	0.28	0.29	0.29	0.28	0.29
	GAT	6.72	0.39	0.29	0.29	0.29	0.29	0.29
BreastMNIST	GCN	0.39	0.35	0.24	0.25	0.25	0.25	0.25
	GAT	5.18	0.50	0.25	0.25	0.25	0.25	0.25

Table 6.8: Training performance for all KD methods.

From Table 6.8 we can see that **RepKD and MSKD have similar training times across all datasets and GNN architectures**. It comes as no surprise since these two methods take the longest time to train since they make use of ensembles. Recall, MSKD makes use of multiple teachers while RepKD makes use of multiple students; in turn, this naturally adds additional training time due to multiple models being used during training. To explain why they have similar training times, we look at how knowledge is distilled. Both methods distil knowledge through graph structure, which in itself is an expensive procedure. In fact, LSP, which also uses graph structure as knowledge, has a larger training time than vanilla KD and FitNet. Non-graph-based KD methods take less time to train since matching logits or node embeddings does not require additional computation. As a result, we see that KDG requires more time to train since taking topological information into account involves additional computation. This means that, RepKD has comparable training times with state-of-the-art KD methods (MSKD). We provide additional results in Tables B.3 and B.4 for intra and cross model performance, and the same patterns emerge. Moreover, the tables show that as the number of students increases, this naturally increases the training time for RepKD.

From Table 6.8 (bottom), we can see that **RepKD uses the same amount of memory per epoch compared to all other KD methods across all datasets and GNN architectures.** Despite using ensembles, the student models are extremely light, which in turn adds a negligible memory overhead during the training produced. In fact, from Tables B.3 and B.4 we provide more evidence to show that our model in the worst case, when $N = 5$, did not see a significant memory increase, it remained constant. It is worth mentioning that the student models (S.) see more memory usage than the KD methods since we store additional metrics and results, which we do not store for the KD methods.

Lastly, we comment on the pre-training memory needed for each KD method. MKSD uses several pre-trained teachers, which all need to be stored, while our method and other KD methods make use of a single teacher. As a result, **RepKD has the same pre-training memory storage overhead as other KD methods and less than MSKD**.

6.5.2 Inference Efficiency

We now look at the results of the distilled model used during inference. Ideally, the distilled models should be lighter and quicker during inference. From Table 6.9 we can see that **RepKD reduces the number of trainable parameters** for GCN teacher to GCN student and GAT teacher to GAT **by 95.63% and 95.66%**, respectively. This is a notable enhancement in model compression efficiency, which is a fundamental objective of KD. Furthermore, we can see **substantial decreases in inference time**. Most notably, an **91.52% and 86.22% decrease** in inference time when using a GAT for both datasets. This is due to the expensive cost of evaluating multiple attention heads and concatenating them together. For the teacher model, the first layer has eight attention heads, while the student model only has one layer with one attention head.

RepKD successfully manages to reduce the number of parameters and decrease the inference time with respect to the teacher model while increasing reproducibility and retaining performance, as seen in Subsection 6.2. This is of great importance since any KD method should have the ability to reduce the size of the model while retaining performance, which RepKD accomplishes.

		# Parameters			**Inference Time (sec)**		
Dataset	GNN	Teacher	Student	Δ	Teacher	Student	Δ
GSP	GCN	2470	108	-95.63%	3.34x10-4	2.83x10-4	-15.27%
	GAT	2536	110	-95.66%	5.14x10-3	7.36x10-4	-91.52%
BreastMNIST	GCN	2470	108	-95.63%	3.46x10-3	2.76x10-4	-20.23%
	GAT	2536	110	-95.66%	5.08x10-3	7.00x10-4	-86.22%

Table 6.9: Inference performance for Teacher and Student Models. Δ indicates the percentage change from the Teacher to Student. For number of parameters and inference time, a smaller value is better. Average inference time over 100 samples.

6.6 Clinical Interpretability

In this section, our focus shifts to the interpretability of the distilled models. A fundamental aim behind enhancing the self-reproducibility score of the student models is to enable them to recognise reliable biomarkers or features consistently. This significantly enhances their credibility and trustworthiness. Specifically, our analysis focuses on the top biomarkers identified by each student model across all GSP datasets for the different combinations of teacher-to-student architectures, namely GCN-GCN, GCN-GAT, GAT-GAT, and GAT-GCN, employed during the RepKD distillation process.
For every student model selected by RepKD (derived through different teacher-to-student architecture pairings, resulting in a total of four), we extract the weights from the final layer of the model across the four CMNs: C_1, C_2, C_3 and C_4. To consolidate this information, we compute the average of these weight values, resulting in a single composite weight vector. As we have done previously to calculate the self-reproducibility score, we compute the absolute values of this vector to gauge its importance and subsequently arrange these weights in descending order. Finally, we look at the intersection among the top 10 biomarkers within these vectors, defined in Table C.2. This results in identifying the biomarkers (ROIs) that consistently appear across models utilising distinct teacher-to-student architectures. In Figures B.5, B.6, B.7 and B.8 we provide visual representations of the average weights for all teacher-to-student architecture pairings.

Index	Region of Interest (ROIs)
7	Inferior Parietal Cortex
12	Lingual gyrus
23	Precentral gyrus
27	Superior frontal gyrus

Table 6.10: Table of ROIs in the intersection within the top 10 ROIs of all student models derived from different teacher-to-student architectures.

From Table 6.10 we can see that there are four ROIs that appear consistently across the top 10 features for all student models trained with different teacher-to-student architectures. We attempt to explain the identified ROIs found by RepKD. In [85], they explore differences in sex by observing resting-state brain connectivity and found that the most discriminative regions for sex classification were located in the cingulate cortex, medial and lateral frontal cortex, temporoparietal regions, insula, and precuneus. When comparing these to the ones in Table 6.10, many are closely related. In particular, the superior frontal gyrus and medial frontal cortex are both part of the frontal cortex and have roles related to decision-making [86]. Moreover, the precentral gyrus and the lateral frontal cortex are also both parts of the frontal cortex concerned with motor control and voluntary movements [87]. Further supporting evidence can be found in [88] where they identify the morphological connection between the caudal anterior cingulate cortex and the superior frontal gyrus in the left hemisphere of the brain as one of the most discriminative ROI interactions. In light of these results, we can see that the ROIs identified by RepKD are closely related to ones found in other works.

6.7 Summary

In this chapter, we evaluated RepKD against the problem formulation defined in Section 5.1. In particular, we tested our method, **RepKD**, under different datasets and GNN architectures while comparing it to the state-of-the-art (SOTA). We saw that RepKD managed to **successfully increase the self-reproducibility score while preserving the performance of the student models**. We also saw that RepKD was **robust to the choice of GNN architecture**. We also carried out extensive tests to justify each of the different components of RepKD through ablation studies and parameter sensitivity studies.
Furthermore, we showed that during the training process, **RepKD exhibited comparable training times and incurred negligible memory overhead in comparison to other SOTA methods** (e.g., MSKD). More importantly, during inference, **RepKD managed to decrease the number of parameters by at least 95.63%**, a primary objective of any KD method. And decreased inference time by at least 15.27%. Moreover, we delved into the interpretability of the distilled models. The biomarkers we discovered, which closely correlate with existing literature, help distinguish gender by analysing cortical morphological networks.
Finally, across this chapter, our intention was to effectively communicate the challenge of achieving a balance between self-reproducibility and performance. This **challenge arises from the inverse relationship between these objectives**. Consequently, striking the right balance between these aims poses a significant challenge and requires careful attention during the design and tuning of KD methods.

Chapter 7

Conclusion

In this project, we explored the intersection between GNN, knowledge distillation, and reproducibility and termed this new domain *reproducible offline knowledge distillation for GNN*. In Chapter 4 we created a novel score that can be used to quantify the reproducibility of any GNN for graph classification tasks. More importantly, we motivated research into this novel domain where we saw that for a range of different KD and KDG methods, the students reproducibility score degraded during the distillation process.
These findings then motivated us to propose a novel *Reproducibility aware Knowledge Distillation method* (RepKD) in Chapter 5. In particular, we proposed the first one-to-many teacher-student method for knowledge distillation on graphs. Furthermore, we introduced novel loss functions inspired by distance-aware deep ensembles [77] and anti-distillation techniques [75] which ensured diversity and stability within the ensemble, which in turn helped with increasing reproducibility. Unlike other KD or KDG methods, RepKD makes use of a two-step process, where the second involves selecting the best student model within the ensemble. We showed that this was an imperative step of our method in order to produce performant, reproducible, lightweight student models.
In Chapter 6 we extensively tested and compared our method, RepKD, with state-of-the-art KD and KDG methods. We saw that RepKD achieved the highest reproducibility score across all datasets for both intra- and cross-model distillation when compared to other KD methods. This was achieved while not only retaining but increasing the performance of the student model; in fact, RepKD managed to consistently outperform vanilla KD. RepKD is the first KDG and KD method that produces both reproducible and performant student models. Moreover, in this chapter, we justified the design choices by carrying out extensive ablation studies. Furthermore, we saw that RepKD manages to reduce models parameters and inference time by considerable amounts a main goal of any KD method. Finally, throughout this, we conveyed the difficulty of finding a balance between both self-reproducibility and performance due to the contrasting nature of these two goals.

7.1 Future Work

As an innovative work in the intersection of three different areas; GNNs, KD and reproducibility, there are several directions of future work one could explore:

- **Exploring more complex datasets** - In our project, we focused on the GSP and BreastMNIST datasets, both with less than 1,000 samples consisting of small graphs. Further evaluation can be done on larger datasets such as PPI [7] or Pubmed [89].
- **Beyond classification** - In this project, we concentrate on issues with binary classification. Future research may therefore focus on multi-class classification or regression tasks, such as age prediction using CMNs.
- **GNN architectures** - We only examined two GNN architectures due to the time-consuming process of training models across 3, 5, and 10 fold cross validation and across 10 seeds. More research could be done to determine how RepKD performs when utilising various GNN architectures, like GraphSAGE [7].
- **Limitations of the self-reproducibility score** - In Section 4.2 we highlight the limitation of the self-reproducibility score. Future research can be done to address these constraints, in order to enhance the score.

We saw that the main drawback of our method is that, despite outperforming all the state-of-the-art KD methods in terms of reproducibility, our method trailed behind in terms of performance (accuracy and F1-score) due to the difficulties of balancing both aims. Furthermore, we saw that our method had training times that were longer than some KD methods due to the use of ensembles, and as such, we propose two avenues of future work to further improve RepKD:

- **Further enhance performance** - In Section 6.2 we saw that RepKD consistently outperformed vanilla KD but lagged behind other KD approaches in terms of accuracy. Perhaps the use of a less constrictive loss function could improve model accuracy while maintaining reproducibility, as was discussed in Section 6.3. Alternatively, different forms of knowledge can be incorporated into RepKD to further improve performance.
- **Decrease training time** - In Subsection 6.5.1, we saw that RepKD has one of the longest training times, similar to MSKD. Incorporating quantization [21, 22] or pruning [20] in conjunction with RepKD could, in turn, reduce the training time of the model while retaining performance.

7.2 Final Remarks

Finally, to conclude, our project combined three domains, consolidating them into one, and demonstrated that the goal of obtaining distilled models that are both reproducible and performant proved to be a formidable challenge. Therefore, we hope that our proposed method and research can act as a pivotal starting point for future exploration into reproducible offline knowledge distillation for GNNs.

Bibliography

[1] Ahmed Nebli, Mohammed Amine Gharsallaoui, Zeynep Gürler, Islem Rekik, Alzheimer's Disease Neuroimaging Initiative, et al. Quantifying the reproducibility of graph neural networks using multigraph data representation. *Neural Networks*, 148:254–265, 2022. pages 3, 4, 13, 17, 18, 21, 23, 28

[2] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. *Advances in neural information processing systems*, 30, 2017. pages 3

[3] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30:595–608, 2016. pages 3

[4] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. pages 3

[5] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. pages 3, 7, 8, 23, 35

[6] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019. pages 3, 7, 8

[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. pages 3, 7, 8, 9, 53

[8] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017. pages 3, 7, 8, 23, 35

[9] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018. pages 3, 7, 8, 9

[10] Paul Y Wang, Sandalika Sapra, Vivek Kurien George, and Gabriel A Silva. Generalizable machine learning in neuroscience using graph neural networks. *Frontiers in artificial intelligence*, 4:618372, 2021. pages 3

[11] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019. pages 3, 7, 8

[12] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 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, 2018. pages 3

[13] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 922–929, 2019. pages 3

[14] Victoria Zayats and Mari Ostendorf. Conversation modeling on reddit using a graph-structured lstm. *Transactions of the Association for Computational Linguistics*, 6:121–132, 2018. pages 3

[15] Dongya Wu, Xin Li, and Jun Feng. Multi-hops functional connectivity improves individual prediction of fusiform face activation via a graph neural network. *Frontiers in neuroscience*, 14:596109, 2021. pages 3

[16] Sofia Ira Ktena, Sarah Parisot, Enzo Ferrante, Martin Rajchl, Matthew Lee, Ben Glocker, and Daniel Rueckert. Metric learning with spectral graph convolutions on brain connectivity networks. *NeuroImage*, 169:431–442, 2018. pages 3

[17] Xinyi Gao, Wentao Zhang, Yingxia Shao, Quoc Viet Hung Nguyen, Bin Cui, and Hongzhi Yin. Efficient graph neural network inference at large scale. *arXiv preprint arXiv:2211.00495*, 2022. pages 3, 8, 9

[18] Yijun Tian, Shichao Pei, Xiangliang Zhang, Chuxu Zhang, and Nitesh V Chawla. Knowledge distillation on graphs: A survey. *arXiv preprint arXiv:2302.00219*, 2023. pages 3, 10, 11, 12, 15

[19] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 257–266, 2019. pages 3, 8, 9

[20] Hongkuan Zhou, Ajitesh Srivastava, Hanqing Zeng, Rajgopal Kannan, and Viktor Prasanna. Accelerating large scale real-time gnn inference using channel pruning. *arXiv preprint arXiv:2105.04528*, 2021. pages 3, 9, 53

[21] Shyam A Tailor, Javier Fernandez-Marques, and Nicholas D Lane. Degree-quant: Quantization-aware training for graph neural networks. *arXiv preprint arXiv:2008.05000*, 2020. pages 3, 9, 53

[22] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021. pages 3, 9, 53

[23] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021. pages 3, 9, 12, 14

[24] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. pages 3, 10, 14, 23, 30, 34, 36

[25] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. pages 3, 14, 23, 34, 36

[26] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Distilling knowledge from graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7074–7083, 2020. pages 3, 11, 14, 23, 28, 29, 34, 36

[27] Chunhai Zhang, Jie Liu, Kai Dang, and Wenzheng Zhang. Multi-scale distillation from multiple graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4337–4344, 2022. pages 3, 11, 15, 23, 28, 29, 34

[28] Alaa Bessadok, Mohamed Ali Mahjoub, and Islem Rekik. Graph neural networks in network neuroscience. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5833–5848, 2022. pages 4, 18

[29] Emanuel Schwarz and Sabine Bahn. Biomarker discovery in psychiatric disorders. *Electrophoresis*, 29(13):2884–2890, 2008. pages 4

[30] Mehmet Yiğit Balık, Arwa Rekik, and Islem Rekik. Investigating the predictive reproducibility of federated graph neural networks using medical datasets. In *Predictive Intelligence in Medicine: 5th International Workshop, PRIME 2022, Held in Conjunction with MICCAI 2022, Singapore, September 22, 2022, Proceedings*, pages 160–171. Springer, 2022. pages 4, 13, 18, 21, 23, 34

[31] Avram J Holmes, Marisa O Hollinshead, Timothy M O'keefe, Victor I Petrov, Gabriele R Fariello, Lawrence L Wald, Bruce Fischl, Bruce R Rosen, Ross W Mair, Joshua L Roffman, et al. Brain genomics superstruct project initial data release with structural, functional, and behavioral measures. *Scientific data*, 2(1):1–16, 2015. pages 5, 23, 33

[32] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023. pages 5, 23, 34

[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020. pages 7, 8

[34] Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B Wiltschko. A gentle introduction to graph neural networks. *Distill*, 6(9):e33, 2021. pages 7

[35] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017. pages 7

[36] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020. pages 8

[37] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020. pages 8

[38] James M Joyce. Kullback-leibler divergence. In *International encyclopedia of statistical science*, pages 720–722. Springer, 2011. pages 10

[39] Wentao Zhang, Xupeng Miao, Yingxia Shao, Jiawei Jiang, Lei Chen, Olivier Ruas, and Bin Cui. Reliable data distillation on graph convolutional network. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1399–1414, 2020. pages 11

[40] Chaitanya K Joshi, Fayao Liu, Xu Xun, Jie Lin, and Chuan Sheng Foo. On representation knowledge distillation for graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. pages 11

[41] Yuzhao Chen, Yatao Bian, Xi Xiao, Yu Rong, Tingyang Xu, and Junzhou Huang. On self-distilling graph neural network. *arXiv preprint arXiv:2011.02255*, 2020. pages 12

[42] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. Graph-less neural networks: Teaching old mlps new tricks via distillation. *arXiv preprint arXiv:2110.08727*, 2021. pages 12, 15

[43] Yijun Tian, Chuxu Zhang, Zhichun Guo, Xiangliang Zhang, and Nitesh V Chawla. Nosmog: Learning noise-robust and structure-aware mlps on graphs. *arXiv preprint arXiv:2208.10010*, 2022. pages 12, 15

[44] Cheng Yang, Jiawei Liu, and Chuan Shi. Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework. In *Proceedings of the web conference 2021*, pages 1227–1237, 2021. pages 12, 15

[45] Umar Asif, Jianbin Tang, and Stefan Harrer. Ensemble knowledge distillation for learning improved and efficient networks. *arXiv preprint arXiv:1909.08097*, 2019. pages 12, 15, 27, 28, 30

[46] Xiaoqin Chang, Sophia Yat Mei Lee, Suyang Zhu, Shoushan Li, and Guodong Zhou. One-teacher and multiple-student knowledge distillation on sentiment classification. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 7042–7052, 2022. pages 12, 15, 27

[47] Danielle S Bassett and Olaf Sporns. Network neuroscience. *Nature neuroscience*, 20(3):353–364, 2017. pages 13

[48] Yushun Dong, Binchi Zhang, Yiling Yuan, Na Zou, Qi Wang, and Jundong Li. Reliant: Fair knowledge distillation for graph neural networks. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pages 154–162. SIAM, 2023. pages 15

[49] Yichen Zhou, Zhengze Zhou, and Giles Hooker. Approximation trees: Statistical stability in model distillation. *arXiv preprint arXiv:1808.07573*, 2018. pages 16, 17

[50] Yunzhe Zhou, Peiru Xu, and Giles Hooker. A generic approach for statistical stability in model distillation. *arXiv preprint arXiv:2211.12631*, 2022. pages 16, 17

[51] Yanqiao Zhu, Yuanqi Du, Yinkai Wang, Yichen Xu, Jieyu Zhang, Qiang Liu, and Shu Wu. A survey on deep graph generation: Methods and applications. *arXiv preprint arXiv:2203.06714*, 2022. pages 16

[52] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2020. pages 16

[53] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. pages 17

[54] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017. pages 17

[55] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. pages 17

[56] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. arxiv 2013. *arXiv preprint arXiv:1312.6034*, 2019. pages 17

[57] Yujia Zhang, Kuangyan Song, Yiming Sun, Sarah Tan, and Madeleine Udell. " why should you trust my explanation?" understanding uncertainty in lime explanations. *arXiv preprint arXiv:1904.12991*, 2019. pages 17

[58] Ulrich Aïvodji, Hiromi Arai, Olivier Fortineau, Sébastien Gambs, Satoshi Hara, and Alain Tapp. Fairwashing: the risk of rationalization. In *International Conference on Machine Learning*, pages 161–170. PMLR, 2019. pages 17

[59] Zhengze Zhou, Giles Hooker, and Fei Wang. S-lime: Stabilized-lime for model explanation. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 2429–2438, 2021. pages 17

[60] Dan Jin, Bo Zhou, Ying Han, Jiaji Ren, Tong Han, Bing Liu, Jie Lu, Chengyuan Song, Pan Wang, Dawei Wang, et al. Generalizable, reproducible, and neuroscientifically interpretable imaging biomarkers for alzheimer's disease. *Advanced Science*, 7(14):2000675, 2020. pages 17

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. pages 17

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. pages 17

[63] Yuhui Du, Zening Fu, Jing Sui, Shuang Gao, Ying Xing, Dongdong Lin, Mustafa Salman, Anees Abrol, Md Abdur Rahaman, Jiayu Chen, et al. Neuromark: An automated and adaptive ica based pipeline to identify reproducible fmri markers of brain disorders. *NeuroImage: Clinical*, 28:102375, 2020. pages 17

[64] Armin Iraji, Zening Fu, Ashkan Faghiri, Marlena Duda, Jaiyu Chen, Srinivas Rachakonda, Thomas Patrick DeRamus, Peter Kochunov, Bhim M Adhikari, Aysenil Belger, et al. Canonical and replicable multi-scale intrinsic connectivity networks in 100k+ resting-state fmri datasets. *bioRxiv*, pages 2022–09, 2022. pages 17

[65] Cooper J Mellema, Kevin P Nguyen, Alex Treacher, and Albert Montillo. Reproducible neuroimaging features for diagnosis of autism spectrum disorder with machine learning. *Scientific reports*, 12(1):3057, 2022. pages 17

[66] James V Stone. Independent component analysis: an introduction. *Trends in cognitive sciences*, 6(2):59–64, 2002. pages 17

[67] Nicolas Georges, Islem Mhiri, Islem Rekik, Alzheimer's Disease Neuroimaging Initiative, et al. Identifying the best data-driven feature selection method for boosting reproducibility in classification tasks. *Pattern Recognition*, 101:107183, 2020. pages 18

[68] Zengyou He and Weichuan Yu. Stable feature selection for biomarker discovery. *Computational biology and chemistry*, 34(4):215–225, 2010. pages 18

[69] Alexandros Kalousis, Julien Prados, and Melanie Hilario. Stability of feature selection algorithms: a study on high-dimensional spaces. *Knowledge and information systems*, 12:95–116, 2007. pages 18

[70] Anne-Laure Boulesteix and Martin Slawski. Stability and aggregation of ranked gene lists. *Briefings in bioinformatics*, 10(5):556–568, 2009. pages 18, 28

[71] Chad A Davis, Fabian Gerick, Volker Hintermair, Caroline C Friedel, Katrin Fundel, Robert Küffner, and Ralf Zimmer. Reliable gene signatures for microarray classification: assessment of stability and performance. *Bioinformatics*, 22(19): 2356–2363, 2006. pages 18, 28

[72] Francis R Bach. Bolasso: model consistent lasso estimation through the bootstrap. In *Proceedings of the 25th international conference on Machine learning*, pages 33–40, 2008. pages 18, 28

[73] Utkarsh Mahadeo Khaire and R Dhanalakshmi. Stability of feature selection algorithm: A review. *Journal of King Saud University-Computer and Information Sciences*, 34(4):1060–1073, 2022. pages 22

[74] Max Klabunde and Florian Lemmerich. On the prediction instability of graph neural networks. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part III*, pages 187–202. Springer, 2023. pages 24

[75] Gil I Shamir and Lorenzo Coviello. Anti-distillation: Improving reproducibility of deep networks. *arXiv preprint arXiv:2010.09923*, 2020. pages 27, 29, 52

[76] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017. pages 28

[77] Francesco D'Angelo and Vincent Fortuin. Repulsive deep ensembles are bayesian. *Advances in Neural Information Processing Systems*, 34:3451–3465, 2021. pages 29, 52

[78] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. pages 31

[79] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019. pages 31

[80] Bruce Fischl. Freesurfer. *Neuroimage*, 62(2):774–781, 2012. pages 33

[81] Ines Mahjoub, Mohamed Ali Mahjoub, and Islem Rekik. Brain multiplexes reveal morphological connectional biomarkers fingerprinting late brain dementia states. *Scientific reports*, 8(1):1–14, 2018. pages 33

[82] Colin R Buchanan, Mark E Bastin, Stuart J Ritchie, David C Liewald, James W Madole, Elliot M Tucker-Drob, Ian J Deary, and Simon R Cox. The effect of network thresholding and weighting on structural brain networks in the uk biobank. *NeuroImage*, 211:116443, 2020. pages 34

[83] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. pages 35

[84] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. *arXiv preprint arXiv:1912.09893*, 2019. pages 36

[85] Susanne Weis, Kaustubh R Patil, Felix Hoffstaedter, Alessandra Nostro, BT Thomas Yeo, and Simon B Eickhoff. Sex classification by resting state brain connectivity. *Cerebral cortex*, 30(2):824–835, 2020. pages 50

[86] Rami M El-Baba and Mark P Schury. Neuroanatomy, frontal cortex. 2020. pages 50

[87] Linnea Banker and Prasanna Tadi. Neuroanatomy, precentral gyrus. 2019. pages 50

[88] Ahmed Nebli and Islem Rekik. Gender differences in cortical morphological networks. *Brain imaging and behavior*, 14(5):1831–1839, 2020. pages 50

[89] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3): 93–93, 2008. pages 53

Appendix A

Parameter Configuration

Model	GSP	BreastMNIST
GCN Teacher	$\{lr = $ 1x10-4$, \text{dropout} = 0\}$	$\{lr = $ 1x10-4$, \text{dropout} = 0\}$
GCN Student	$\{lr = $ 1x10-4$, \text{dropout} = 0\}$	$\{lr = $ 1x10-5$, \text{dropout} = 0\}$
GAT Teacher	$\{lr = $ 1x10-4$, \text{dropout} = 0.1\}$	$\{lr = $ 1x10-3$, \text{dropout} = 0.1\}$
GAT Student	$\{lr = $ 1x10-4$, \text{dropout} = 0\}$	$\{lr = $ 1x10-6$, \text{dropout} = 0\}$

Table A.1: Table of GNN teacher and student specific hyperparameters.

Model	GSP	BreastMNIST
Vanilla	$\{lr = $ 1x10-4$, \alpha = 1, \beta = \frac{2}{9}\}$	$\{lr = $ 1x10-4$, \alpha = 0.8, \beta = \frac{9}{9}\}$
FitNet	$\{lr = $ 1x10-3$, \alpha = 1, \beta = \frac{2}{9}, \gamma = 0.5\}^*$	$\{lr = $ 1x10-3$, \alpha = 2, \beta = \frac{7}{9}, \gamma = 0.1\}$
LSP	$\{lr = $ 1x10-3$, \alpha = 1, \beta = \frac{2}{9}\}$	$\{lr = $ 1x10-3$, \alpha = 1, \beta = \frac{2}{9}\}$
MSKD	$\{lr = $ 1x10-3$, \alpha = 1, \beta = \frac{4}{9}, \gamma = 4\}$	$\{lr = $ 1x10-3$, \alpha = 1, \beta = \frac{4}{9}, \gamma = 4\}$
RepKD	$\{lr = $ 1x10-4$, \alpha = 1, \beta = 2, \gamma = \frac{2}{9}, \lambda = 1\}$	$\{lr = $ 1x10-4$, \alpha = 1, \beta = 7, \gamma = \frac{7}{9}, \lambda = 1\}$

Table A.2: Table of model specific hyperparameters. * for C_2 and C_4 datasets $\gamma = 0.2$.

Teacher GNN	Student GNN	C_1	C_2	C_3	C_4	BreastMNIST
GCN	GCN	4	4	4	4	3
GCN	GAT	2	2	2	2	2
GAT	GAT	2	2	2	2	2
GAT	GCN	3	4	3	3	3

Table A.3: Table of number of students N used in ensemble for RepKD.

Hyperparameter	Search Space
Optimiser Learning Rate lr	{1x10-3, 1x10-4, 1x10-5, 1x10-6}
KD α	{0.1, 0.2, 0.4, 0.6, 0.8, 1, 2}
KD β	$\{\frac{1}{9}, \frac{2}{9}, \frac{3}{9}, \frac{4}{9}, \frac{5}{9}, \frac{6}{9}, \frac{7}{9}, \frac{8}{9}, \frac{9}{9}, \frac{10}{9}, \frac{11}{9}, \frac{12}{9}\}$
KD γ	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 2}
RepKD β	{1, 2, 3, 5, 7, 9, 13.5}
RepKD γ	$\{\frac{1}{9}, \frac{2}{9}, \frac{3}{9}, \frac{4}{9}, \frac{5}{9}, \frac{6}{9}, \frac{7}{9}, \frac{8}{9}, \frac{9}{9}, \frac{10}{9}, \frac{11}{9}, \frac{12}{9}\}$
RepKD λ	{0.2, 0.4, 0.6, 0.8, 1, 1.5, 2}
Number of Students in Ensemble N (if used)	{2, 3, 4, 5}

Table A.4: Table of hyperparameter search space for model specific parameters. Notice that KD β and RepKD γ, which are both used for response-based knowledge, are in the range from $\frac{1}{9}$ to $\frac{12}{9}$. The reason for this is because we need to take into account τ^2, as defined in equation 2.6, since $\tau = 3$ for all experiments.

Appendix B

Supplementary Results

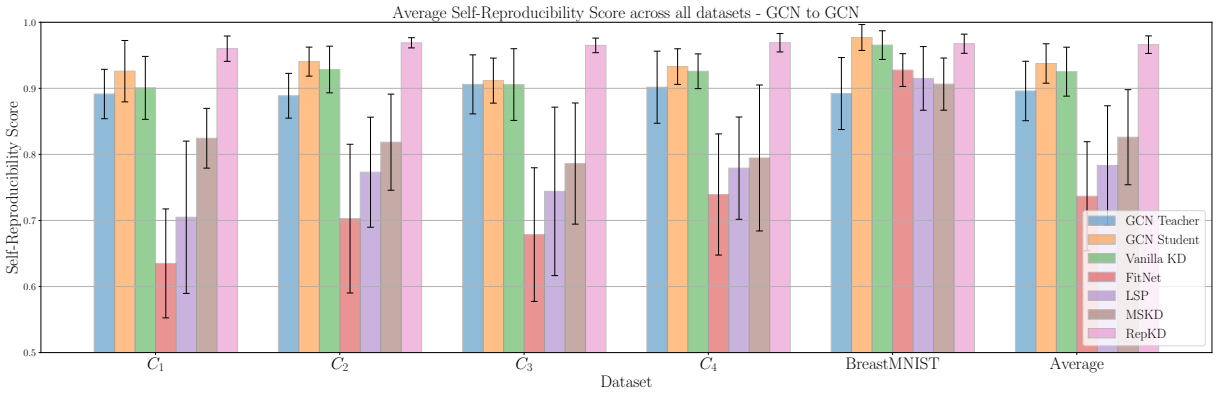

Figure B.1: Visualisations of average self-reproducibility score across all datasets for GCN-GCN teacher-student distillation.

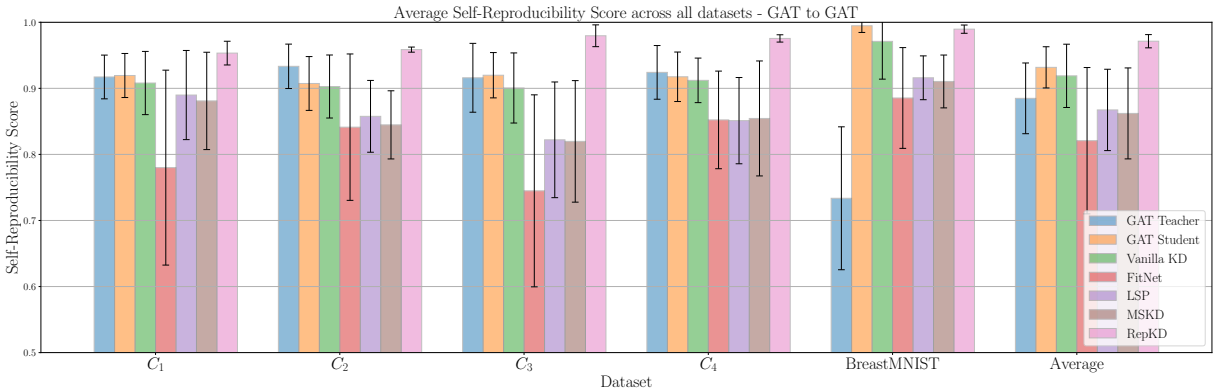

Figure B.2: Visualisations of average self-reproducibility score across all datasets for GAT-GAT teacher-student distillation.

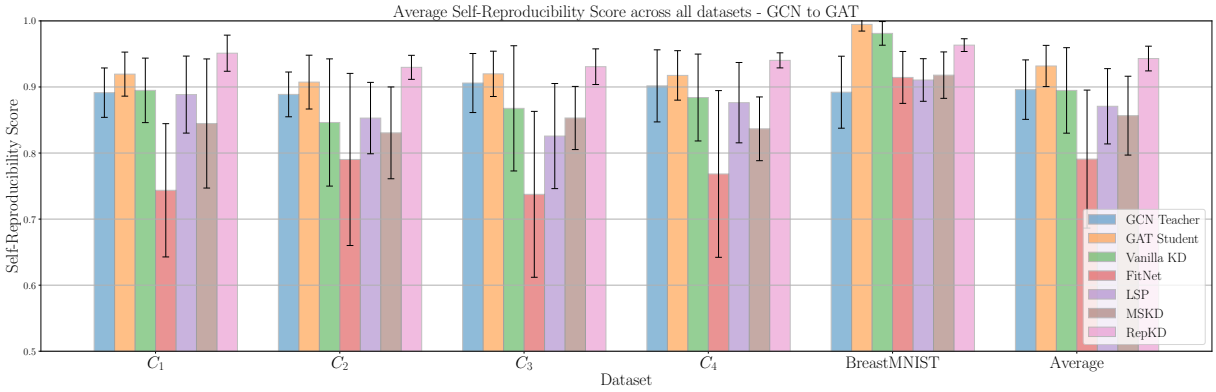

Figure B.3: Visualisations of average self-reproducibility score across all datasets for GCN-GAT teacher-student distillation.

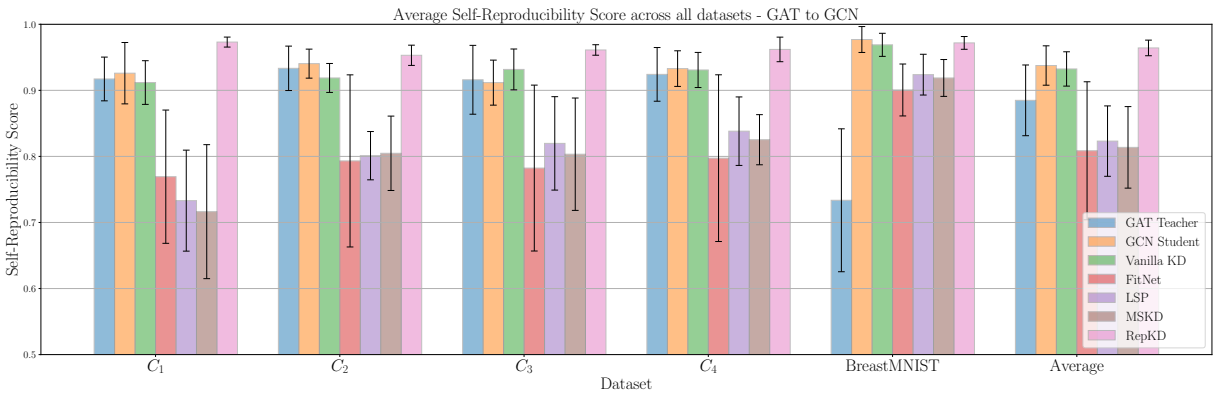

Figure B.4: Visualisations of average self-reproducibility score across all datasets for GAT-GCN teacher-student distillation.

Dataset	N=2 RepKD ($\mathcal{L}_{ESP}$)	N=2 RepKD ($\mathcal{L}_{ESP}$ w/ **L**)	N=2 RepKD ($\mathcal{L}_{ESP}$ w/ **E**)	N=3 RepKD ($\mathcal{L}_{ESP}$)	N=3 RepKD ($\mathcal{L}_{ESP}$ w/ **L**)	N=3 RepKD ($\mathcal{L}_{ESP}$ w/ **E**)
C_1	93.83±1.86	94.06±2.22	88.58±6.08	96.83±0.73	96.88±0.43	91.47±5.02
C_2	94.44±1.94	92.97±1.89	89.88±8.38	93.19±1.41	93.21±2.85	92.83±4.31
C_3	95.79±1.76	94.39±2.61	86.96±6.64	96.67±0.86	95.78±1.03	91.38±4.12
C_4	93.31±1.83	94.61±1.90	91.68±5.71	92.17±0.91	91.26±1.80	94.13±5.13
Average	94.34±1.85	94.01±2.16	89.27±6.70	94.72±0.98	94.28±1.53	92.45±4.65

Dataset	N=4 RepKD ($\mathcal{L}_{ESP}$)	N=4 RepKD ($\mathcal{L}_{ESP}$ w/ **L**)	N=4 RepKD ($\mathcal{L}_{ESP}$ w/ **E**)	N=5 RepKD ($\mathcal{L}_{ESP}$)	N=5 RepKD ($\mathcal{L}_{ESP}$ w/ **L**)	N=5 RepKD ($\mathcal{L}_{ESP}$ w/ **E**)
C_1	96.00±1.92	97.22±1.72	93.43±5.40	95.81±2.02	96.53±2.43	92.88±4.34
C_2	96.89±0.77	95.31±2.26	93.56±4.88	97.83±0.85	94.17±1.57	93.88±3.89
C_3	96.50±1.11	96.11±1.18	92.17±6.04	97.72±0.90	97.56±1.45	90.75±5.51
C_4	96.90±1.39	95.53±1.17	92.94±2.82	93.38±1.94	93.78±1.93	95.97±3.20
Average	96.57±1.30	96.04±1.58	93.02±4.78	96.18±1.43	95.51±1.85	93.37±4.23

Table B.1: Self-reproducibility score - Varying how knowledge is distilled through $\mathcal{L}_{ESP}$ for different number of students in ensemble. **Bold** and Underline indicate the best and second best performance 'w/o" means without. "**L**" refers to logits and "**E**" refers to node embeddings, the type of knowledge used in $\mathcal{L}_{ESP}$.

Dataset	N=2 RepKD ($\mathcal{L}_{ESP}$)	N=2 RepKD ($\mathcal{L}_{ESP}$ w/ **L**)	N=2 RepKD ($\mathcal{L}_{ESP}$ w/ **E**)	N=3 RepKD ($\mathcal{L}_{ESP}$)	N=3 RepKD ($\mathcal{L}_{ESP}$ w/ **L**)	N=3 RepKD ($\mathcal{L}_{ESP}$ w/ **E**)
C_1	56.39±0.42	56.78±0.56	55.60±1.16	61.13±0.60	61.35±0.74	55.50±1.55
C_2	60.04±0.31	60.21±0.26	58.57±2.75	60.16±0.73	60.47±0.34	56.70±1.64
C_3	65.35±0.57	65.39±0.66	58.41±3.56	64.19±0.29	64.23±0.42	56.82±2.13
C_4	63.36±0.38	63.75±0.42	58.86±3.59	63.47±0.54	63.57±0.49	57.86±2.44
Average	61.28±0.43	61.53±0.50	57.86±2.94	62.24±0.56	62.41±0.52	56.72±1.97

Dataset	N=4 RepKD ($\mathcal{L}_{ESP}$)	N=4 RepKD ($\mathcal{L}_{ESP}$ w/ **L**)	N=4 RepKD ($\mathcal{L}_{ESP}$ w/ **E**)	N=5 RepKD ($\mathcal{L}_{ESP}$)	N=5 RepKD ($\mathcal{L}_{ESP}$ w/ **L**)	N=5 RepKD ($\mathcal{L}_{ESP}$ w/ **E**)
C_1	60.13±0.96	60.33±1.03	55.38±1.43	58.92±0.73	57.11±0.61	56.26±1.90
C_2	61.95±0.41	62.05±0.36	58.58±2.21	61.43±0.24	63.25±0.34	58.75±2.12
C_3	64.02±0.24	64.09±0.27	57.08±1.93	63.88±0.29	63.36±0.53	59.31±1.79
C_4	63.19±0.39	63.42±0.47	59.46±3.27	61.74±0.58	64.91±0.37	56.77±1.27
Average	62.32±0.57	62.47±0.61	57.62±2.31	61.49±0.50	62.16±0.48	57.77±1.80

Table B.2: Accuracy - Varying how knowledge is distilled through $\mathcal{L}_{ESP}$ for different number of students in ensemble. **Bold** and Underline indicate the best and second best performance 'w/o" means without. "**L**" refers to logits and "**E**" refers to node embeddings, the type of knowledge used in $\mathcal{L}_{ESP}$.

Dataset	T. GNN to S. GNN	Time Per Epoch (sec)									
		T.	S.	Vanilla	FitNet	LSP	MSKD	RepKD (2)	RepKD (3)	RepKD (4)	RepKD (5)
GSP	GCN-GCN	0.31	0.23	0.41	0.43	1.61	2.30	1.87	2.17	2.56	2.99
	GAT-GAT	2.21	0.22	2.08	2.10	3.09	4.08	3.63	4.11	4.67	5.21
BreastMNIST	GCN-GCN	0.31	0.25	0.46	0.48	1.45	2.26	1.99	2.30	2.75	3.33
	GAT-GAT	2.21	0.40	2.05	2.10	3.09	3.93	3.77	4.36	4.94	5.54

Dataset	T. GNN to S. GNN	Memory Usage Per Epoch (GB)									
		T.	S.	Vanilla	FitNet	LSP	MSKD	RepKD (2)	RepKD (3)	RepKD (4)	RepKD (5)
GSP	GCN-GCN	0.42	0.39	0.28	0.29	0.29	0.28	0.29	0.29	0.29	0.29
	GAT-GAT	6.72	0.39	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29
BreastMNIST	GCN-GCN	0.39	0.35	0.24	0.25	0.25	0.25	0.24	0.25	0.25	0.25
	GAT-GAT	5.18	0.50	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25

Table B.3: Training intra-model performance for all KD methods. RepKD (N) indicates the number of students N in ensemble.

Dataset	T. GNN to S. GNN	Time Per Epoch (sec)									
		T.	S.	Vanilla	FitNet	LSP	MSKD	RepKD (2)	RepKD (3)	RepKD (4)	RepKD (5)
GSP	GCN-GAT	0.31	0.22	0.60	0.61	1.63	2.49	2.31	2.65	3.17	3.75
	GAT-GCN	2.21	0.23	1.91	1.96	2.95	3.88	3.36	3.69	4.10	4.53
BreastMNIST	GCN-GAT	0.31	0.40	0.61	0.64	1.66	2.41	2.27	2.77	3.29	4.00
	GAT-GCN	2.21	0.25	1.91	1.95	2.95	3.81	3.55	3.85	4.30	4.61

Dataset	Architecture	Memory Per Epoch (GB)									
		T.	S.	Vanilla	FitNet	LSP	MSKD	RepKD (2)	RepKD (3)	RepKD (4)	RepKD (5)
GSP	GCN-GAT	0.42	0.39	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29
	GAT-GCN	6.72	0.39	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29
BreastMNIST	GCN-GAT	0.39	0.50	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
	GAT-GCN	5.18	0.35	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25

Table B.4: Training cross-model performance for all KD methods. RepKD (N) indicates the number of students N in ensemble.

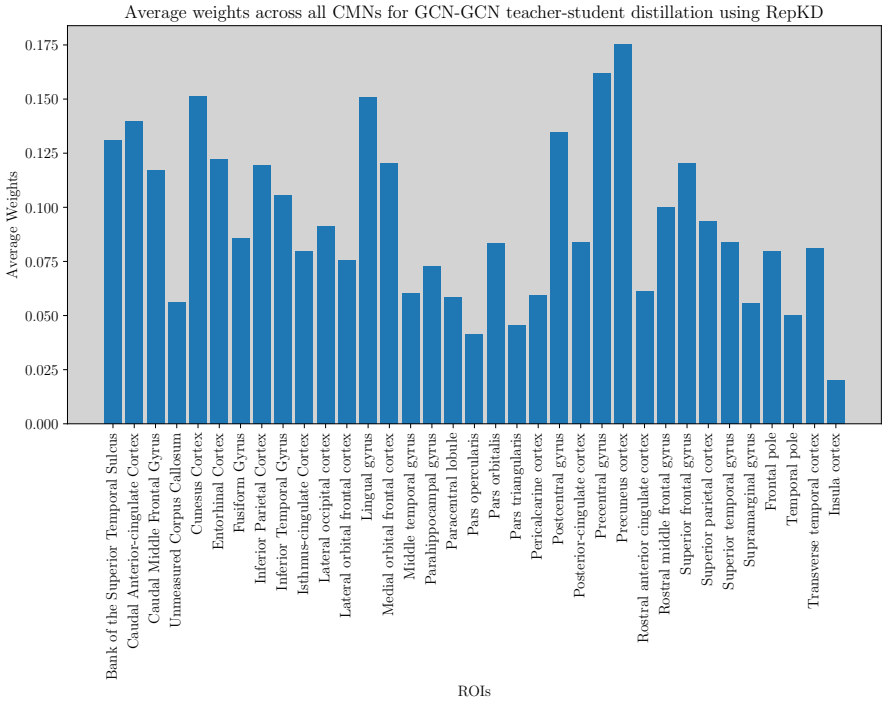

Figure B.5: Average weights across all CMNs for GCN-GCN teacher-student distillation using RepKD.

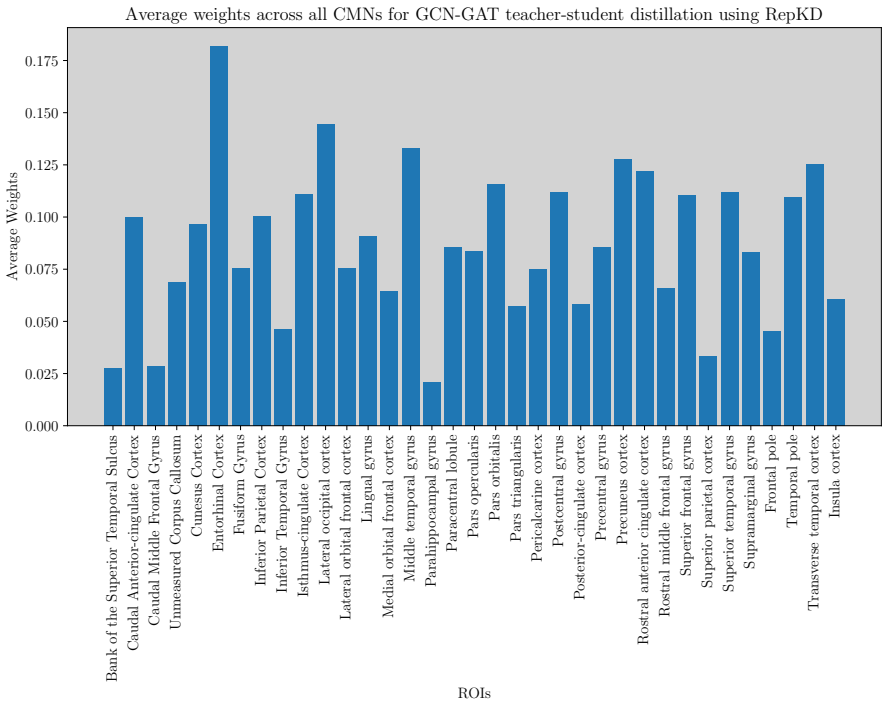

Figure B.6: Average weights across all CMNs for GCN-GAT teacher-student distillation using RepKD.

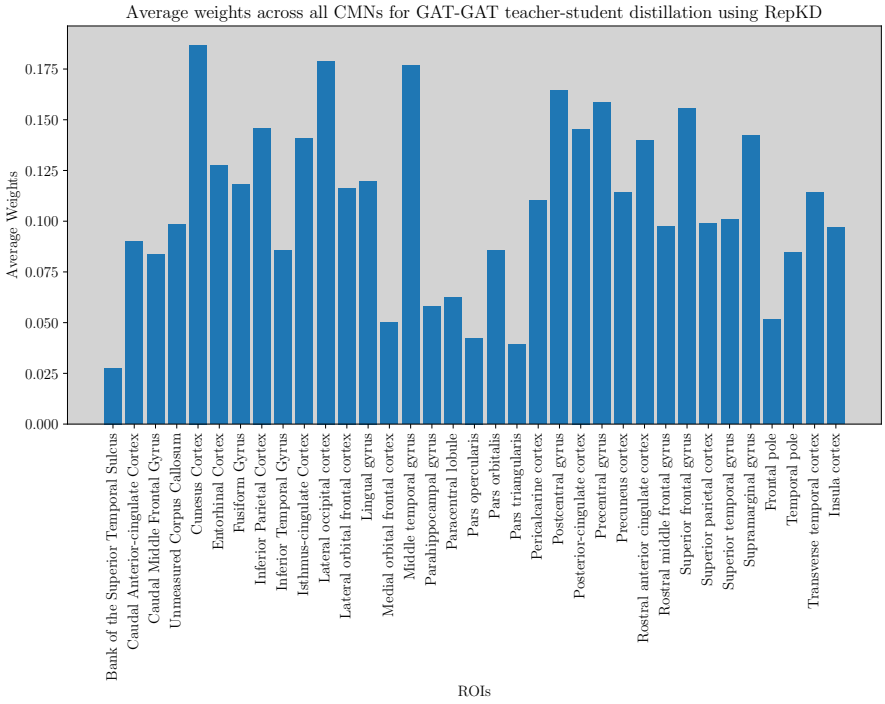

Figure B.7: Average weights across all CMNs for GAT-GAT teacher-student distillation using RepKD.

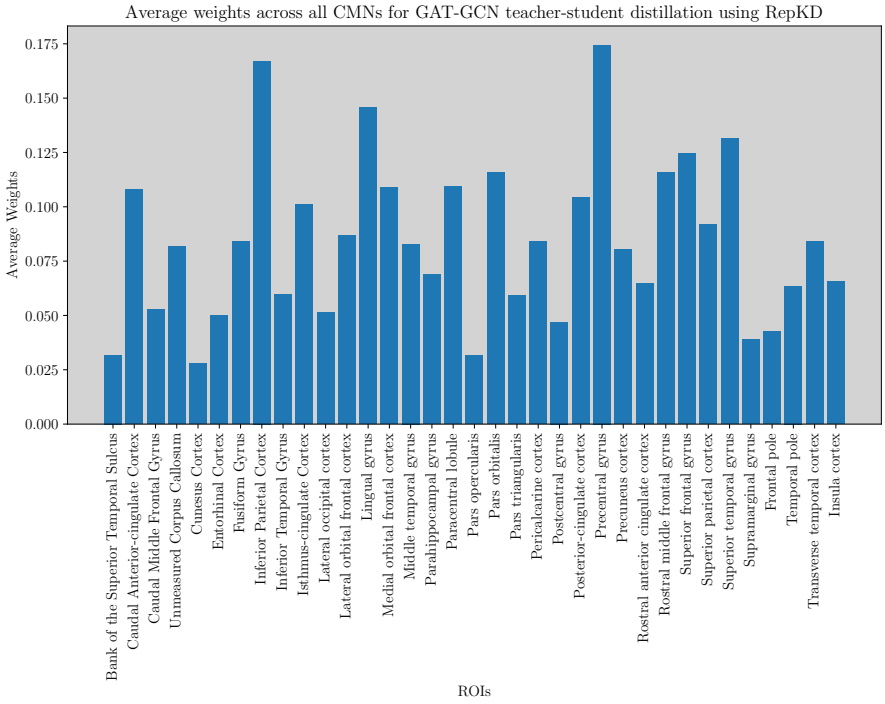

Figure B.8: Average weights across all CMNs for GAT-GAT teacher-student distillation using RepKD.

Appendix C

Additional Information

In this section we provide with additional information for the code repository used for this project. In particular we give an overview of the different directories and components, for each giving a brief explanation such that they are understood. More importantly, in the directory `src/demo/` we provide a toy example of how to use our code. The code repository can be found here: https://github.com/LorenzoStigliano/thesis-imperial.

Directory / File	Description
`src/`	Contains code for the project.
`src/models/`	Contains GNN model implementations and definitions for GCN and GAT.
`src/models_config/`	Stores configuration files for different model architectures for knowledge distilation methods used in main_*.py files.
`src/notebooks/`	Contains Jupyter notebooks used for experimentation and visualization throughout the project.
`src/scripts/`	Holds scripts that can be run to replicate results for thesis.
`src/trainers/`	Contains training code for all knowledge distillation methods baselines (vanilla KD, FitNet, LSP and MSKD) as well as RepKD training method (`trainers/rep_kd_ensemble/`) for ensemble sizes of 2, 3, 4 or 5.
`src/utils/`	Stores utility functions and helper modules, used to create datasets and for analysis. More importantly `utils/config.py` is where the configurations of the project structure need to be defined.
`src/demo/`	Holds a demonstration example of how to use the proposed method.
`src/main_*.py`	These `main_*.py` files are used as the main functions to train various models in the project. They correspond to different knowledge distillation methods. Additionally, the files in `model_config/` are used in conjunction with these scripts to configure model architectures and training parameters. If custom model configurations need to be defined in the one of the files in `model_config/` directory, then imported into the appropriate model you want to train.

Table C.1: Overview of code repository directory structure.

Index	Brain Regions of Interest (ROIs)
1	Bank of the Superior Temporal Sulcus
2	Caudal Anterior-cingulate Cortex
3	Caudal Middle Frontal Gyrus
4	Unmeasured Corpus Callosum
5	Cunesus Cortex
6	Entorhinal Cortex
7	Fusiform Gyrus
8	Inferior Parietal Cortex
9	Inferior Temporal Gyrus
10	Isthmus-cingulate Cortex
11	Lateral occipital cortex
12	Lateral orbital frontal cortex
13	Lingual gyrus
14	Medial orbital frontal cortex
15	Middle temporal gyrus
16	Parahippocampal gyrus
17	Paracentral lobule
18	Pars opercularis
19	Pars orbitalis
20	Pars triangularis
21	Pericalcarine cortex
22	Postcentral gyrus
23	Posterior-cingulate cortex
24	Precentral gyrus
25	Precuneus cortex
26	Rostral anterior cingulate cortex
27	Rostral middle frontal gyrus
28	Superior frontal gyrus
29	Superior parietal cortex
30	Superior temporal gyrus
31	Supramarginal gyrus
32	Frontal pole
33	Temporal pole
34	Transverse temporal cortex
35	Insula cortex

Table C.2: List of ROIs for GSP dataset.