UNIVERSITÄT DES SAARLANDES

ON THE IMPORTANCE OF GRAPH-TASK ALIGNMENT FOR GRAPH NEURAL NETWORKS

Master Thesis

by

Adarsh Jamadandi
Matriculation Number : 7009552

Advisor :
Prof. Dr. Rebekka Burkholz

1. Referee :
Prof. Dr. Rebekka Burkholz

2. Referee :
Prof. Dr. Verena Wolf

August 26, 2024

Contents

Sworn declaration

Acknowledgements

I would like to express my heartfelt gratitude to all those whose steadfast support has been instrumental in the completion of this thesis. First and foremost, I extend my profound appreciation to Dr. Rebekka Burkholz, my advisor and mentor, who not only provided me with the opportunity but also the freedom to explore compelling research questions within the nurturing environment of CISPA. Her expertise and unwavering encouragement have been pivotal in shaping both this thesis and my growth as a researcher. I would also like to thank Dr.Verena Wolf, who initially provided me with the opportunity to pursue research in Graph Neural Networks and also agreed to be a co-examiner for my defense. My time as part of the RelationalML lab has been one of the most enriching experiences of my life. I am particularly indebted to Celia Rubio-Madrigal for her continuous support, both as a collaborator on my projects and as a cherished friend. Her exceptional `TikZ` plots, created for our joint papers, have made their way into the thesis as well. I also fondly recall the early days of our group, shared with Advait Gadhikar and Nimrah Mustafa. I am grateful for the countless thought-provoking discussions we engaged in. I owe an immense debt of gratitude to my parents Lalita and Manjunath, whose constant encouragement and support have been my beacon during the most challenging times. To my brother Vikas, I extend my thanks for maintaining stability at home in my absence; you have truly grown into an inspiration. I must also convey my deepest apologies to my late grandmother, whose funeral I regrettably missed. I acknowledge that I would not be where I am today without the nurturing care provided by you and grandfather during my formative years when my parents were occupied with work. Finally, I would like to express my appreciation to my closest friends, Vinay, Dinesh, and Aashita, for their unwavering belief in me. It is safe to say that this thesis could not have been written without their support. Though, of course, any errors are mine alone.

Abstract

Given a graph and a chosen downstream task such as node classification, how do we know if the input graph is the optimal computational structure that can aid in the learning task? Graphs, like all other data modalities, are susceptible in addition to noise also topological bottlenecks that highly limit generalization performance. This problem is exacerbated in the context of Graph Neural Networks (GNNs), which popularly leverage the paradigm of *message passing*, because the input graph is not only the data for the model but also the computational structure on which information is diffused and aggregated to learn a final graph-level representation. In this thesis, we provably show that a key factor dictating the performance of a GNN is the alignment between the node labels and the latent community structure of the graph, which we call the *Graph-Task* alignment. We empirically demonstrate that rewiring approaches which rely on spectral gap do not take into account the pre-existing alignment and can only strengthen or dampen it. As a resolution, we propose a novel graph rewiring strategy that considers the node labels. This is non-trivial, since we do not have access to the test node labels. We propose a graph structure learning approach that uses the predictions made by a surrogate model as a proxy for effecting changes to the inter-class and intra-class edges directly. Through a comprehensive set of experiments on various benchmark datasets, we substantiate our claims and hypotheses.

Contributions

In this thesis - On the Importance of Graph-Task Alignment, we propose novel graph modification techniques that improve the generalization performance of Graph Neural Networks. The thesis is based on the following papers -

(i) Adarsh Jamadandi*, Celia Rubio-Madrigal* and Rebekka Burkholz (2024). Spectral Graph Pruning Against Over-squashing and Over-smoothing. Pre-print under-review.

(ii) Celia Rubio-Madrigal*, Adarsh Jamadandi* and Rebekka Burkholz (2024). SoLAR - Surrogate Label Aware Rewiring for Graph-Task Alignment. Pre-print under-review.

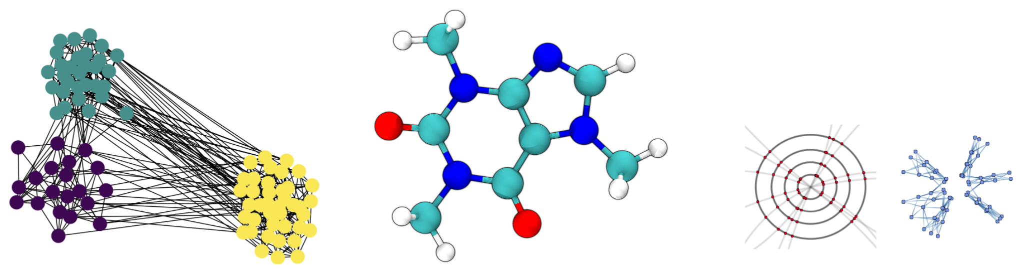
* represents joint first authors with equal contributions to the paper. For the first paper, with crucial inputs from Rebekka Burkholz, initial idea and all the experiments were carried out by Adarsh Jamadandi. Celia Rubio-Madrigal carried out theoretical proofs for Braess paradox, performing ring graph experiments and also establishing time complexity for the algorithms. For the second paper, with crucial inputs from Rebekka Burkholz, the label aware rewiring and all the experiments were carried out by Adarsh Jamadandi, Celia Rubio-Madrigal formulated proofs and analyses for the Stochastic Block Models.

Chapter I.

Introduction

Deep learning models for graphs commonly adopt the paradigm of message passing [26, 64, 23, 11], wherein, the input graph not only acts as the data for the model but is also the computational structure on which information is aggregated and diffused. Graph Neural Networks (GNNs) have found extensive applications in Chemistry [59], Biology [8], high-energy Physics [63, 67]. They have recently been used to improve real time ETA (Estimated Time of Arrival) for Google maps [19] and also to develop highly accurate weather forecasting model [39]. Despite their proliferation, GNNs are also known to have many inherent problems, for example, GNNs fail to distinguish simple sub-structures due to limited expressivity [40, 46, 55]. Some other problems include over-squashing [1, 72, 24], where topological bottlenecks lead to information congestion, affecting message passing and over-smoothing [42, 53, 54, 75, 37], where node features become indistinguishable due to repeated rounds of aggregation, disallowing for training deeper GNNs.

A common resolution for mitigating problems like over-squashing and over-smoothing is to operate on a different variant of the input graph that is more amenable to the message passing framework. This is termed as *Graph Rewiring*. The idea is to modify the edge structure of the graph based on certain criteria such as discrete Ricci curvature [72, 25, 50], spectral gap [36, 35] or transforming the input graph into expander graphs [62, 18, 3]. Each of these methods have their own drawbacks, for instance, Ricci curvature based methods fail to scale to large graphs, while spectral gap based methods that rely on adding edges to maximize the spectral gap might introduce detrimental over-smoothing [37]. In the chapters that follow, we will introduce these ideas and the methods proposed to mitigate them. This offers us an opportunity to see the contributions of this thesis in the context of the limitations of the aforementioned methods. We show through extensive experiments that the driving force behind the generalization



(a) Social Networks (b) Chemical Molecules (c) High-energy Physics [67].

Figure I.1.: Graphs are ubiquitous data structures that can model data from diverse fields including but not limited to social networks, chemical molecules, biological networks and even particle accelerator collisions can be modelled as graphs.

performance of GNNs can be attributed to the alignment between the inherent community structure and the node labels and how enhancing this concordance can better equip GNN models to solve the task at hand.

1. Graph Neural Networks

In this section, we briefly introduce the idea behind message passing and how GNNs are formulated. We will adopt the definition formalized in [11]. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph and let $\mathcal{N}_u = \{v \in \mathcal{V} | (v, u) \in \mathcal{E}\}$ denote the 1-hop neighborhood of the graph. Let $\mathbf{X} \in \mathbb{R}^{\mathcal{V} \times k}$ be the associated node features $\mathbf{x}_u$ of node u. The message passing scheme can be implemented on the graph as

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{V}} \psi(\mathbf{x}_u, \mathbf{x}_v)\right) \tag{I.1}$$

where $\bigoplus$ denotes permutation invariant aggregation function such as *SUM* or *MAX* and (ψ, ϕ) are learnable message passing and final readout functions respectively. Different variants of GNNs have been proposed, some of them include Graph Convolutional Networks (GCNs) [38], Graph Attention Networks (GAT) [73, 10], GraphSAGE [29] and many other specialised architectures operating on heterophilic graphs [57]. In this work, we will be using GCN [38] and GATv2 [10] as our backbone architectures for all our experiments.

1.1. Graph Convolutional Networks

GCN introduced in [38] performs convolutions on the graph analogous to Convolutional Neural Networks (CNNs). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected and unweighted graph with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges. The adjacency matrix $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ encodes the graph topology. The GCN [38, 14] operates on the degree normalized adjacency matrix $\hat{A} = \tilde{D}^{-1/2}(A+I)\tilde{D}^{-1/2}$. Our chosen downstream task is semi-supervised node classification, that is, given a graph, we have a set of nodes $\mathcal{V}_{train}$ whose labels $\mathcal{Y}_{train}$ are available, we are required to predict the labels of nodes $\mathcal{V}_{test} = \mathcal{V}\backslash\mathcal{V}_{train}$. Let $\mathbf{Z}$ be the predictions made by the GCN

$$\mathbf{Z} = \texttt{softmax}(\hat{A}\sigma(\hat{A}X\mathbf{\Theta^{(0)}})\mathbf{\Theta^{(1)}} \tag{I.2}$$

where $\sigma(\cdot)$ is a non-linear activation function such as ReLU and $\mathbf{\Theta}$ the weight matrix. We employ the cross-entropy loss [14] as our objective function,

$$\mathcal{L}(\mathcal{G}, \mathbf{\Theta}) = -\frac{1}{|\mathcal{V}_{label}|}\sum_{\sqsubseteq_{\rangle}\in\mathcal{V}_{label}} \mathbf{y}_i log(z_i) \tag{I.3}$$

1.2. Graph Attention Networks

While GCN and GraphSAGE weight the neighbours equally by using $\bigoplus = \{mean, max\}$ in Equation I.2,[73] propose an attention mechanism in the form of weighted average of the neighbours $\mathcal{N}_u$. We adopt the notations introduced in [10]. A score $e : \mathbb{R}^d \times \mathbb{R}^d$ is calculated for every edge (u, v), which captures how important the features of u are to the node v and is given by

$$e(\mathbf{h}_u, \mathbf{h}_v) = \texttt{LeakyReLU}(a^T \cdot [\mathbf{W}\mathbf{h}_u || \mathbf{W}\mathbf{h}_v]) \tag{I.4}$$

where $\mathbf{a} \in \mathbb{R}^{2d}$, $\mathbf{W} \in \mathbb{R}^{d\times d}$ are learnable parameters and $||$ denotes vector concatenation. The final attention scores are calculated as

$$\alpha_{u,v} = \texttt{softmax}_u(e(\mathbf{h}_u, \mathbf{h}_v)) = \frac{exp(e(\mathbf{h}_u, \mathbf{h}_v))}{\sum_{u\in\mathcal{N}_v} exp(e(\mathbf{h}_u, \mathbf{h}_v))} \tag{I.5}$$

where $\sum_{u\in\mathcal{N}_v}$ means the attention scores are normalized across all neighbours $v \in \mathcal{N}_u$. The final representation is learnt as

$$\hat{\mathbf{h}} = \sigma(\sum_{u\in\mathcal{N}_v} \alpha_{u,v} \cdot \mathbf{W}\mathbf{h}) \tag{I.6}$$

Authors in [10] modify Equation I.4 to

$$e(\mathbf{h}_u, \mathbf{h}_v) = \mathbf{a}^T\texttt{LeakyReLU}(\mathbf{W} \cdot [\mathbf{h}_u || \mathbf{h}_v]) \tag{I.7}$$

and show this simple modification allows the GAT to be more expressive than one proposed in [73]. This version is called GATv2 and we will be using this version for our future experiments.

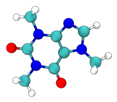
1.3. Thesis outline

In Chapter I, we discussed why graphs are a natural data modality to capture relational information and we introduced how Graph Neural Networks (GNNs) are formulated. We also discussed two variants of GNNs that will be used as backbone architectures for our future experiments. The rest of the thesis will be organized as follows, in Chapter II, we will introduce the notion of *over-squashing* and *over-smoothing*, the former results from failing to model information from long range interactions while the latter is due to too many rounds of information aggregation. Current literature espouses for a trade-off between these two phenomena, we show it is possible to mitigate both of them simultaneously. Chapter III will be dedicated to exploring the realm of graph rewiring and how it allows us to operate on a variant of input graph which is better equipped to handle message passing. Finally, in Chapter IV, we will discuss the role of latent community structure and its effect on the generalization performance of GNNs, followed by our introduction to our novel graph rewiring strategy `SoLAR` and present a suite of experiments to substantiate our claims.

Chapter II.

Over-squashing and Over-smoothing in Graph Neural Networks

1. Introduction

In the previous section, we introduced the idea of applying deep learning algorithms for graphs. We highlighted the different models that can be instantiated based on the aggregation scheme used. We also briefly introduced some inherent problems GNNs are plagued with, such as, over-squashing, over-smoothing, limited expressivity. In this section, our focus will be on over-squashing and over-smoothing. Consider the problem of predicting the properties of the caffeine molecule  using a GNN. To make this prediction, we need to propagate information from the red atom to the white atom which is located at a distance. Surprisingly, authors in [1] note that although GNNs are effective in propagating short-range information, they fail to model the long-range interactions, leading to a phenomenon they refer to as *over-squashing*. An information bottleneck prevents the GNN from capturing long-range interactions, leading to poor generalization performance. A simple fix would be to instantiate a deeper model, since conventional wisdom from training deep neural networks such as ResNets [30] has shown that increasing depth should generally help generalize better. However, in practice, most GNNs are trained with only 2-4 layers, and one of reasons for this is another limitation of GNNs dubbed over-smoothing[1]. As we saw earlier, GNNs operate by exchanging messages on the graph, this exchange/aggregate (smoothing) steps are necessary for the model to learn

[1]Note that, the training dynamics and gradient flow problems also severely hinder training deeper GNNs (although residual blocks and normalization techniques seem to help to a certain extent). That is not the focus of this thesis

important features necessary for the task to be solved, however, it is possible for the GNN to enter a regime of detrimental smoothing (over-smoothing) [42, 54, 53, 75, 61] that renders nodes of different classes indistinguishable. It seems that we are at an impasse, one one hand we have over-squashing that results from failing to capture all the information and on the other we have over-smoothing, that results from aggregating too much information. Does this mean, we have a trade-off that needs to be carefully balanced when training GNNs? To better understand if that is the case, we will devote this chapter to deconstruct the two phenomena, setting the stage for the next chapter which dives into rewiring based resolutions proposed to mitigate these issues.

2. What is Over-squashing?

We will adopt the notation introduced in [1]. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the problem radius r is defined as the required range of interaction between distant nodes to solve a task. This is characterized by the number of GNN layers (l) we instantiate. This implies a GNN should have $l \geq r$ to handle the long range interactions, however as pointed out in [1, 13], the number of nodes in each node's receptive field grows as $|\mathcal{N}_u^l| = \mathcal{O}(exp(l))$. That is, at every aggregation step, a node has to keep the learned information from its neighbourhood which grows exponentially, squashed into fixed-size vectors. We can use the [72] Jacobian - $\partial h_v^{(r)} / \partial x_u$ to characterize how the hidden representation $h_v^{(r)}$ of some node v is not influenced by node feature $\mathbf{x}_u$ of node u located at a distance r from v, due to over-squashing. That is, the node representations are desensitized to information from nodes that are located at the problem radius. One might wonder, surely increasing the number of GNN layers might help capture the information from the distant nodes? However, as authors in [1, 24] show that it does not help with over-squashing as the problem is strongly due to the *topology* of the input graph. A paradigm example is that of a barbell graph shown in Figure II.1, suppose a node from one of the communities need to send information across to a node in the other community, it can only do so, through a single edge connecting the two communities. Since all information has to be propagated through this edge only, this results in a bottleneck that leads to information congestion, affecting the efficacy of message passing. This bottleneck can be mathematically characterized by the spectral gap [16] and the Cheeger constant [12]. We will delve into these formulations in the subsequent chapter.

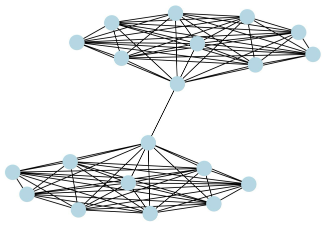


Figure II.1.: A barbell graph with two communities connected by a single edge is a paradigm example to demonstrate the presence of a topological bottleneck that can lead to information congestion when performing message passing.

2.1. How do we know Over-squashing really happens?

To demonstrate over-squashing really happens, authors [1] devise a synthetic task called the Tree-NeighboursMatch problem. The GNN is tasked with predicting the label for node marked $\textcircled{?}$, it can do so by learning that nodes $\textcircled{A}$, $\textcircled{B}$ have different number of ○ blue node neighbors. So the GNN prediction for the node marked $\textcircled{?}$ should be $\textcircled{C}$. However, for the prediction for the node in question, an exponential amount of information has to be propagated from all the leaves of this tree, which is compressed into a fixed size vector depending on the aggregation scheme we use. Thus causing an information bottleneck.

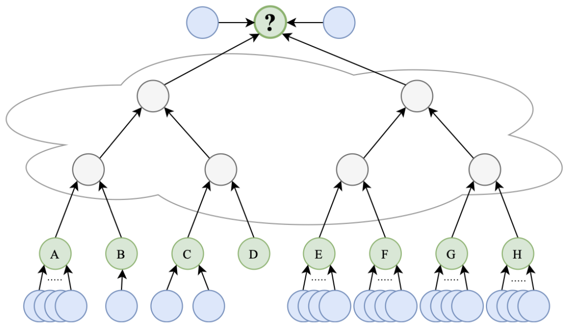


Figure II.2.: Tree-NeighboursMatch Problem. Figure credit : [1].

We instantiate a simple GCN [38] that is trained on this synthetic dataset, we plot the train accuracy vs the depth of the tree which represents the problem radius r in Figure II.3. For $r = 2, 3$, we obtain a train accuracy of 1.0, for depth, i.e $r = 4$, we can see the train accuracy is already decreasing rapidly and for increasing depth, the train accuracy is really low. This indicates, due to over-squashing, the GCN underfits the data and generalizes poorly during test time. We compare this with our proposed spectral based graph rewiring method called `Proxy-Add` that adds edges (more on this in Chapter III). Evidently, adding edges that maximize the spectral gap can help alleviate over-squashing. Thus, graph rewiring is a valid approach to mitigate over-squashing and many criteria such as Ricci curvature [72, 25, 50] and spectral gap [36, 35] based methods have been proposed. We will discuss each of these methods in detail in the next chapter.

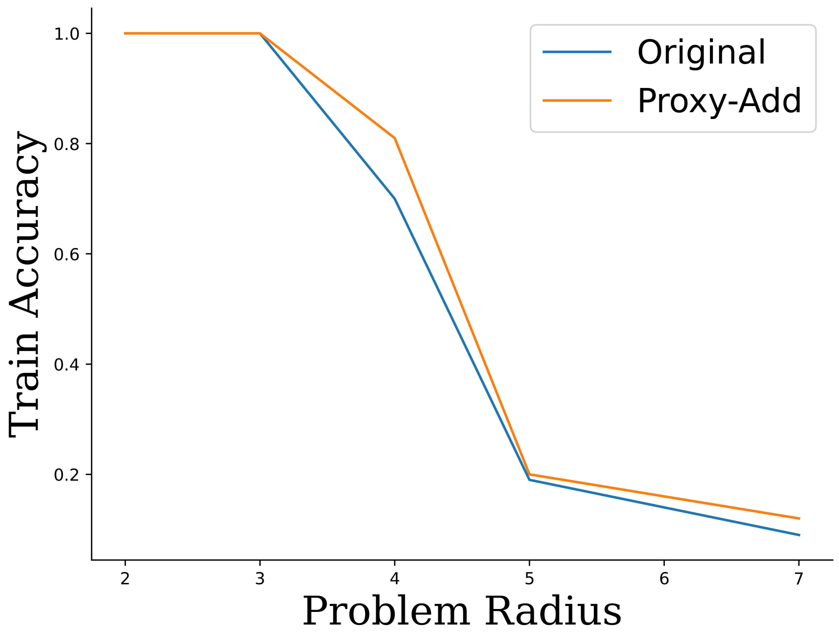


Figure II.3.: Problem radius vs. Train Accuracy for Tree-Neighbours match problem.

3. What is Over-smoothing?

Conventional wisdom from training neural networks for image and vision tasks suggest having deeper layers for the network will help in generalization [30]. Interestingly, this idea does not successfully translate for training deeper GNNs. Naively stacking multiple GNN layers harms generalization and results in node features tending to uninformative limit. This behaviour can be explained in two ways - the first obvious intuition is the problem of gradient flow, training deeper networks is usually hard due to problems like bad initialization [47] and vanishing gradients. The problem of vanishing gradients can be somewhat tackled by adding skip connections and normalization techniques like BatchNorm [34] and/or LayerNorm [2]. The second insight that sheds light on why training deeper GNNs might be difficult can be explained by the phenomenon of over-smoothing. [54] analyse the asymptotic behavior of GCNs, when the layers tend to infinity. Let $\tilde{A} = A + I$ and $\tilde{D} = D + I$ be the adjacent matrix and the degree matrix respectively. The normalized Laplacian with self-loops is defined as $\tilde{\Delta} = I - \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. Looking at the spectra of the Laplacian, one can show that when stacking multiple GCN layers, the node features undergo *smoothing* and will exhibit an information loss at an exponential rate. This also highlights the fact that, the phenomenon of over-smoothing is connected to the graph topology as well. Lot of works in the literature [42, 54, 53, 75, 61] have analysed the phenomenon of over-smoothing and have provided various fixes to improve training deeper GNNs. Most of these works use the notion of Dirichlet energy [61] to quantify the similarity between the embedding nodes,

$$E(X^{(l)}) = tr(X^{(l)^T}\tilde{\Delta}X^{(l)}) = \frac{1}{2}\sum a_{uv}||\frac{x_u^{(l)}}{\sqrt{1+d_u}} - \frac{x_v^{(l)}}{\sqrt{1+d_v}}||_2^2 \tag{II.1}$$

where, $X^{(l)}) = [x_1^{(l)}, x_2^{(l)}, ...x_n^{(l)}]^T \in \mathbb{R}^{n\times d}$ denotes the learned node embeddings at the lth layer. A smaller Dirichlet energy means the features are over-smoothed, while a very large Dirichlet energy [75] means over-separating, which means the node embeddings that should be mapped together are driven apart, which is again problematic to the learning task.

3.1. Is all Over-smoothing bad?

While Dirichlet energy provides a good notion of over-smoothing and helps quantify how similar the learned node embeddings are, it fails to convey what constitutes a good *smoothing* that is beneficial to the downstream task. Is all over-smoothing detrimental to the learning task? How do we characterize the usefulness of the smoothing? [37] provides an interesting alternative view of over-smoothing, which helps us answer these questions. We will adopt the notations introduced in [37]. The setup involves training a LinearGNN for ridge regression task on the features of the graph. The ridge regression can be formalized as

$$\hat{\beta}^{(l)} = argmin_{\beta}\frac{1}{2n}||Y - Z^{(l)}\beta||^2 + \lambda||\beta^2|| \tag{II.2}$$

where $Z^{(l)}$ represents the smoothed features of the graph after performing l^2 steps of mean aggregation. The empirical risk minimization is

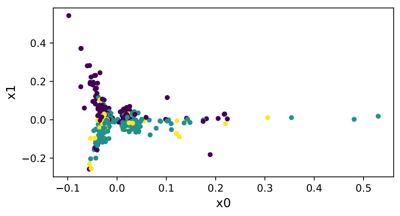
$$\mathcal{R}^{(l)} = n^{-1}||Y - \hat{Y}^{(l)}||^2 \tag{II.3}$$

where $\hat{Y}^{(k)} = Z^{(k)}\hat{\beta}^{(l)}$, represents the predictions made. It can be shown that there is an optimal $l^* > 0$ such that $\mathcal{R}^{(l^*)} < min(\mathcal{R}^{(l)}, \mathcal{R}^{(\infty)})$, that is, smoothing provably improves the risk. We can empirical illustrate this phenomenon. We will reproduce the results presented in [37] on the Cora [45] dataset for 500 steps of mean aggregation. This simple setup mimics the actual GNN training which involves aggregating and updating information from the graphs to aid in the learning task. In FigureII.4(a), the features represent when the the $\texttt{agg} = 0$, and subsequent Figures II.4(b) II.4(c) represent $\texttt{agg} = 10, 500$ respectively. The mean squared error (MSE) vs. the order of smoothing in a log-log plot is given in Figure II.4(d). This analysis allows us to understand that not all smoothing is detrimental. In fact, GNNs *smooth* the features to learn representations. The MSE plot reveals that for $\texttt{agg} = 0 - 50$, the MSE is low, indicating beneficial smoothing which aids the learning task. But for greater number of aggregation steps, the MSE slowly starts to rise, which is the regime of over-smoothing that is detrimental to the task, as we see a feature collapse. In the next section, we will see techniques introduced to mitigate over-squashing and over-smoothing.

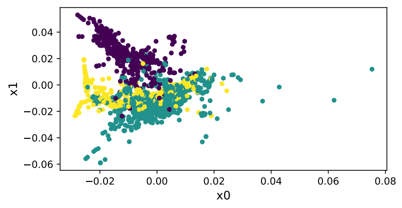
4. Conclusion

In this chapter, we have introduced the notion of over-squashing and over-smoothing. The former results from failing to propagate information due to bottlenecks while the latter is due to too much information aggregation. We demonstrate on the Tree-NeighboursMatch problem [1] that adding edges that maximize the spectral gap can help alleviate over-squashing. We also reproduce the results of [37] on over-smoothing and show empirically the existence of a regime very smoothing is beneficial to the learning task. Current literature that analyse these phenomena propose graph rewiring as a viable strategy to mitigate these problems with a caveat. Contemporaneous works such as [25, 50] propose a trade-off between over-squashing and over-smoothing, in the subsequent chapters, we show by leveraging the Braess paradox that we can tackle both of these problems simultaneously.

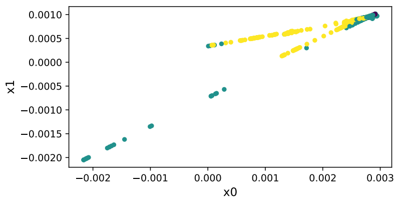
[2] Notational abuse : the number of layers and the number of mean aggregation steps are both represented as l. Technically they are similar, since increasing the number of layers is equivalent to just having 1 layer but performing l steps of aggregation.



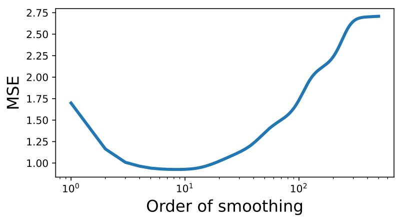
(a) Node features after $\mathtt{agg} = 0$.



(b) Node features after $\mathtt{agg} = 100$.



(c) Node features after $\mathtt{agg} = 500$.



(d) Order of smoothing vs MSE for Cora.

Figure II.4.: The phenomenon of over-smoothing refers to node features tending to uninformative limit due to repeated rounds of aggregation.

Chapter III.

Graph Rewiring for Optimal Computational Structure

1. Introduction

In the previous chapter we saw the problems of over-squashing and over-smoothing and how they affect the overall message passing scheme, consequently affecting the downstream task. This chapter will be devoted to exploring the different techniques that have been proposed to mitigate over-squashing and over-smoothing. We will focus on rewiring the input graph. The notion of graph rewiring involves effecting changes to the edge structure of the graph, this allows us to work on a variant of the input graph that is better equipped to handle message passing. It also allows to circumvent topological bottlenecks that might be present in the original input graph. To this end, various criteria such as the discrete Ricci curvature [72, 25, 50] and spectral gap [36, 35] have been proposed to rewire the graph. The curvature of the edge in a graph can provide useful information about bottlenecks as we will see in subsequent sections . Alternatively, probing the spectrum of the normalized Laplacian and looking at the eigenvalues, specifically the first non-zero eigenvalue, also called as the spectral gap of the graph provides useful information about convergence of random walks and mixing time for a diffusion process, and thus the presence of bottlenecks. These criteria can be leveraged to change the input graph in a meaningful way that might aid the task. In the sections that follow, we will discuss in detail the mathematical characterization of bottlenecks and how Ricci curvature based and spectral gap based criteria will help resolve these bottlenecks. We also present a novel proxy [1] spectral gap based rewiring

[1]This section will be based on A. Jamadandi, C. Rubio-Madrigal and R. Burkholz, "Spectral Pruning against Over-squashing and Over-smoothing [35] Under review.

technique that leverages the Braess paradox for the first time for Graph Neural Networks and show graph pruning can not only mitigate over-squashing but also prevent detrimental over-smoothing.

2. How to characterize the bottlenecks in graphs?

We will revisit the barbell graph in Figure III.1(a), we have two communities bridged by a single edge. Suppose a node from one of the communities needs to communicate with the node in the other community, this long-range dependency might not be captured by a GNN because of the bottleneck edge that connects the two communities. The Cheeger constant [12] defined as

$$h_S = \min_{S \subset V} \frac{|\partial S|}{\min\{Vol(S), Vol(S \backslash V)\}} \tag{III.1}$$

where $\partial S = \{(u, v) : u \in S, v \in \mathcal{V} \backslash S\}$ and $Vol(S) = \sum_{u \in S} d_u$ and d_u the degree of the node u, encodes the notion of the *bottleneckedness* of the graph. A graph with smaller Cheeger constant implies the presence of bottleneck edges.

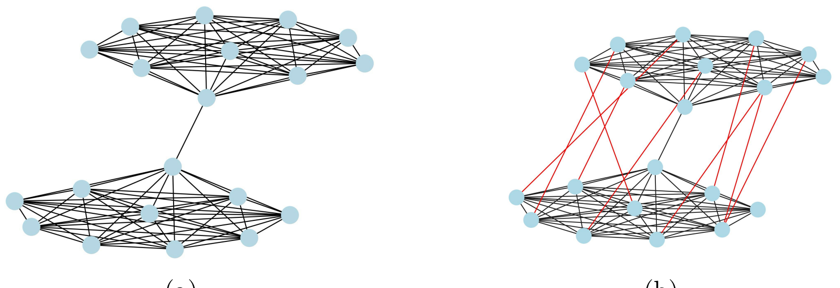

(a) .   (b) .

Figure III.1.: The barbell graph is a paradigm example to highlight the presence of topological bottlenecks in a graph. A single edge connecting two communities will lead to information congestion (Left). Graph rewiring by adding edges across two communities is a possible resolution to mitigate over-squashing.

Additionally, we can also look at the graph spectrum to understand the presence of bottlenecks. For a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The Laplacian of the graph is $\mathcal{L} = D - A$, where A is the adjacency matrix and D the diagonal degree matrix. The symmetric normalized graph Laplacian is defined as $\mathcal{L}_{\mathcal{G}} = D^{-1/2}\mathcal{L}D^{-1/2}$. Let $\{\lambda_0 < \lambda_1 < \lambda_2, ...\lambda_n\}$ be the eigenvalues of $\mathcal{L}_{\mathcal{G}}$ arranged in ascending order and let $\lambda_1(\mathcal{L}_{\mathcal{G}})$ be the first non-zero eigenvalue of the normalized graph Laplacian, which is also called the spectral gap of the graph. For the graph in Figure III.1(a), if we were to calculate the spectral gap, we can see that it is small, thus a smaller spectral gap can also be used as a measure for the bottleneck. In fact, the Cheeger inequality gives the relationship between Cheeger constant and the spectral gap of the graph.

$$h_G = min_{S \subset V} h_S = 2h_G \geq \lambda_1(\mathcal{L}_{\mathcal{G}}) \geq \frac{h_G^2}{2} \tag{III.2}$$

Thus, one can characterize the presence of bottlenecks either by looking at the Cheeger constant or the spectral gap of the graph. A smaller value indicates fewer edges connect communities leading to information congestion. This characterization lends us a strategy to possibly mitigate over-squashing. For instance, we can transform the barbell graph in Figure III.1(a) to the graph in Figure III.1(b) by adding edges (indicated in red), that optimize a certain quantity that describes the bottleneck. Now there is no single bottleneck edge responsible for information diffusion but several edges bearing the load and as we show, performing message passing on this modified graph leads to better generalization performance.

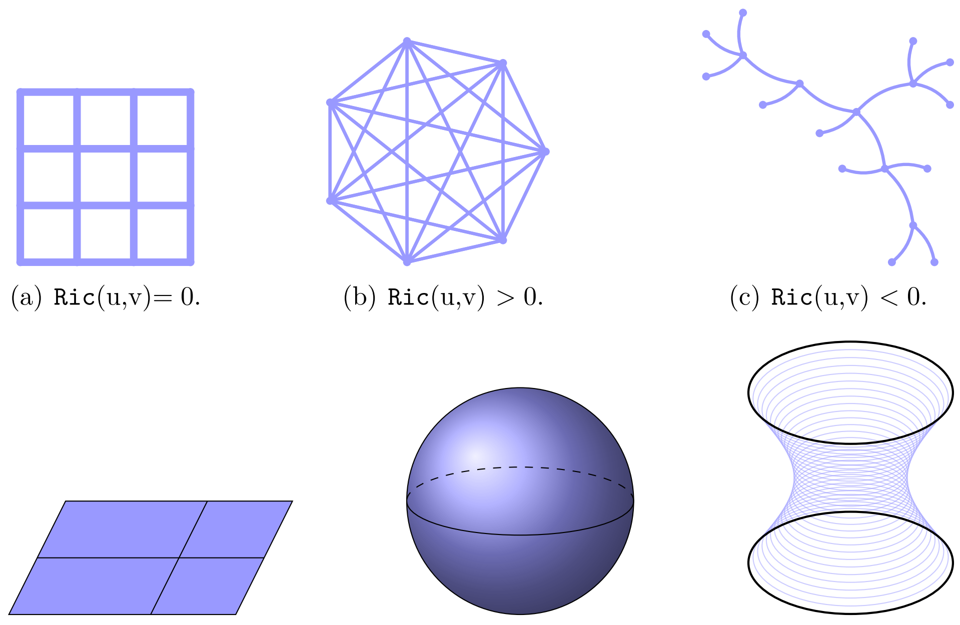
2.1. Discrete Ricci Curvature based Graph Rewiring

In this section, we will discuss the discrete Ricci curvature based graph rewiring technique proposed in [72]. The curvature (κ), is a geometrical quantity that measures how the ambient space *curves*, for example, if the curvature is zero, we have the Euclidean space and if the curvature is either positive or negative, we have the spherical space or the hyperbolic space respectively. The Ricci curvature [28] on the other hand measures the *geodesic dispersion* on the manifolds, that is, if the geodesics stay parallel ($\kappa = 0$), converge ($\kappa > 0$) or diverge ($\kappa < 0$). We can treat graphs as discrete manifolds and define analogously a discrete version of the Ricci curvature [52, 51, 68] on the edges of the graph. This is illustrated in Figure III.2, where the grid graph represents $\texttt{Ric}(u,v) = 0$, the $\texttt{Ric}(u,v) > 0$ represents a complete graph (triangles) and $\texttt{Ric}(u,v) < 0$ represents a tree-like structure. Authors in [72] propose a variant of the Ricci curvature called the balanced Forman curvature and its defined as follows

Definition III.2.1: Balanced Forman Curvature: [72] For any edge $u \sim v$ in a unweighted graph $\mathcal{G}$, we can define $\texttt{Ric}(u,v)$ is 0 if $\{d_u, d_v\} = 1$, else

$$\texttt{Ric}(u,v) = \frac{2}{d_u} + \frac{2}{d_v} + 2 - 2\frac{|\#_{\triangle}(i,j)|}{max\{d_u,d_v\}} + \frac{\#_{\triangle}(u,v)}{min\{d_u,d_v\}} + \frac{(\gamma_{max})^{-1}}{max\{d_u,d_v\}}(|\#_{\square}^{u}|+|\#_{\square}^{v}|) \tag{III.3}$$

where $\#_{\triangle}(u,v)$, represent the triangles in the graph,$\#_{\square}^{u}$ represent the 4-cycle formed by the edges $u \sim v$ and $\gamma_{max}(u,v)$ represent the maximum number of 4 cycles based at a common node p.


(a) `Ric`(u,v)$= 0$. (b) `Ric`(u,v) > 0. (c) `Ric`(u,v) < 0.

(d) $\kappa = 0$. (e) $\kappa > 0$. (f) $\kappa < 0$.

Figure III.2.

Proposition III.2.2: *Cheeger constant and the Balanced Forman Ricci Curvature : If **`Ric`**(u,v) $\geq p > 0$ for all $u \sim v$, then $\lambda_1/2 \geq h_G \geq p/2$.*

That is, the Cheeger constant can be influenced by changing the curvature of the graph. More specifically, [72] show negatively curved edges are responsible for the bottleneck and propose a discrete Ricci curvature based graph rewiring method, termed Stochastic Discrete Ricci Flow (SDRF), that can be used to surgically edit edges of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to obtain $\hat{\mathcal{G}} = (\mathcal{V}, \hat{\mathcal{E}})$, a modified version of the graph with different edge structure. The new graph is potentially bottleneck free and does not incur over-squashing. The proposed algorithm runs in $O(|\mathcal{E}|d_{max}^2)$.

2.2. Spectral Gap based Graph Rewiring

In this section, we will explore the idea of using spectral gap as a measure for bottlenecks. To reiterate, the spectral gap is the first non-zero eigenvalue of the normalized Laplacian denoted as λ_1. The spectral gap is inherently linked to the convergence of random walks and the mixing time of a diffusion process on the graph [31]. A larger spectral gap indicates that the random walk converges to a stationary distribution faster (and hence bottleneck-free). Thus, we can use this criterion to rewire the input graph. The idea is to sequentially add edges to the graph that results in highest spectral gap expansion. The problem with this strategy is that performing the eigenvalue decomposition every time the graph is changed is computationally expensive ($\mathcal{O}(n^2)$) and is not practically feasible. To circumvent this problem, authors in [36] propose a first order approximation of the spectral gap (FoSR) which is computationally cheaper.

2.3. First Order Approximation of Spectral Gap

We will use the formalism presented in [36], to discuss how to compute the spectral gap faster. The trick is instead of calculating the full eigenvalue decomposition, we can approximate the change in the eigenvalues and the eigenvectors for small perturbations made, in our case these perturbations are the edges that are added.

Proposition III.2.3: *First order change in spectral gap [36] : The first order change in spectral gap λ_1 when an edge $u \sim v$ is added is given by*

$$\lambda_1 \approx \frac{2x_u x_v}{(\sqrt{1+d_u})(\sqrt{1+d_v})} + 2\lambda_1 x_u^2 (\frac{\sqrt{d_u}}{\sqrt{1+d_u}} - 1) + 2\lambda_1 x_v^2 (\frac{\sqrt{d_v}}{\sqrt{1+d_v}} - 1) \quad \text{(III.4)}$$

where x is the eigenvector corresponding to the spectral gap and x_u denotes the u^{th} entry of the eigenvector. The authors propose to minimize

$$\lambda_{fosr} = \frac{2x_u x_v}{\sqrt{(1+d_u)(1+d_v)}} \quad \text{(III.5)}$$

At each iteration, FoSR aims to add an edge that minimizes Equation III.5 by calculating the second eigenvector x, the initial approximation for the eigenvector is got by few power iterations that is,

$$x \to D^{-1/2} A D^{-1/2} x - \frac{\langle x, \sqrt{d} \rangle}{2m} \sqrt{d} \quad \text{(III.6)}$$

Thus, the algorithm alternates between calculating the first order approximation of the spectral gap and computing the eigenvector by power method. The number of edges to add k is a hyperparameter that is dataset specific. The goal is to add a small number of edges to the graph, so that we don't change the original degree distribution of the graph too much while also inducing enough changes in the spectrum to ensure the graph is bottleneck free. The algorithm requires initial power iterations that run in $\mathcal{O}(m)$ and then involves minimizing the first order change in spectral gap by looking for all pairs of nodes in the graph and then choosing the best edge to add, which results in $\mathcal{O}(kn^2)$ complexity.

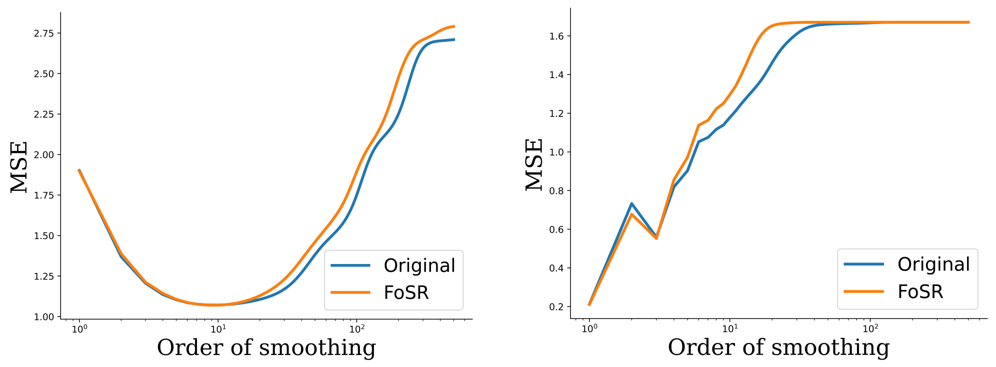
2.4. But what about Over-smoothing?

Graph rewiring by adding edges is an effective strategy to mitigate over-squashing. However, we will show that adding edges that maximize the spectral gap will inevitably lead to over-smoothing. This is expected because spectral gap based methods are purely topological, that is, they don't take into account the labels or the features that accompany the graph. This is both a feature and a bug, on one hand this lets us cheaply modify the graph to be bottleneck-free, the downside is that we will invariably end up connecting edges with different labels. This is problematic during message passing, because the GNN will aggregate information from nodes that should stay distinguishable, leading to over-smoothing. Especially for heterophilic[2] graphs, where the neighbourhood for a node might have nodes belonging to different classes.

The authors in [36] also note this problem and propose to use a variant of GCN called Relational-GCN (R-GCN) [4], that allows for providing special labels to the existing and newly added edges and empirically show that it is better for graph classification tasks. This reasoning is limited because in the case of graph classification, we eventually pool all the aggregated information from all the nodes and assign a single label to the entire graph. This means over-smoothing is really not a problem. However, for node classification tasks, we need the nodes to stay distinguishable and methods like FoSR accelerate the detrimental smoothing.

[2]Graphs can be broadly classified as homophilic and heterophilic graphs. The homophilic graph assumption says like nodes are connected to like nodes and GNNs usually operate under this assumption when aggregating neighbourhood information.

We show this empirically by adopting the Linear GNN test bed discussed in Section (§3.1). We plot the rate of smoothing vs the MSE for two real-world graphs Cora [45] which is a homophilic graph and Texas [56], a heterophilic graph in Figure III.3. We compare the performance of the original graph vs the graph modified by adding 100 edges using FoSR [36]. We can see in the plots below that adding edges which are supposed to help with over-squashing leads to accelerating the detrimental smoothing. This is worse for heterophilic graphs such as Texas. So does this mean spectral based methods might not be useful in practice? Or how do we strike a balance between over-squashing and over-smoothing when rewiring the graph? Is there a trade-off that needs to be taken care of when using spectral expansion methods?



(a) Order of smoothing vs MSE for Cora. (b) Order of smoothing vs MSE for Texas.

Figure III.3.: We perform ridge regression on the features of graphs Cora and Texas respectively with 500 steps of mean aggregation. The plot for order of smoothing vs MSE reveals that adding edges that maximise spectral gap although help mitigate over-squashing will invariably lead to over-smoothing.

3. The Road Not Taken : Braess Paradox and Graph Rewiring

In the previous sections we saw how we can formalize a proxy version of the spectral gap and use it as a criterion to add edges to a graph resulting in spectral gap expansion. We also saw that adding edges which maximize spectral gap although mitigates over-squashing, it will exacerbate the problem of over-smoothing. Is there a way we can tackle both of these problems simultaneously? Or is it a trade-off that needs to be balanced as advocated in [25]? These questions form the crux of this section. We introduce a novel graph modification strategy that allows us to tackle both over-squashing and over-smoothing simultaneously.

3.1. Braess Paradox

The Braess paradox [9] is an interesting phenomenon that states - if all the entities in a network selfishly choose their routes then increasing the capacity of the network will only worsen the overall performance of the network. For instance, in a traffic network if each commuter chooses their route selfishly, adding an extra road will increase the commute time for all travellers. Now this phenomenon is not a vague, isolated event but has profound implications for network science. The prevalence of this phenomenon has been highlighted in the context of graphs [16, 17]. In fact, authors in [21] take this analysis further and show that the spectral gap exhibits this paradox. To demonstrate the occurrence of this phenomenon we will devise a synthetic ring graph test bed. In Figure III.4, we have a ring graph $\mathcal{R}_8$ with 2 classes represented by orange and purple color. The following observations can be made :

(i) The original graph $\mathcal{G}$ has an edge connecting nodes $3-0$. Note that the edge is connecting nodes of different labels. The initial spectral gap is ~ 0.2829.

(ii) The graph $\mathcal{G}^{-}$ is obtained when we delete the edge $0-3$ resulting in spectral gap expansion ~ 0.2929. This is due to the Braess paradox.

(iii) The graph $\widetilde{\mathcal{G}}^{+}$ is obtained when we add an edge $4-2$ to the graph resulting in spectral gap decrease due to the Braess paradox.

Thus, we can see that the Braess paradox is a prevalent phenomenon and can be used to influence the spectral gap. This can be mathematically formalized in a lemma below which is called as the Eldan's criterion [21].

Lemma III.3.1: *Eldan's criterion [21]: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a finite graph, with f denoting the eigenvector and $\lambda_1(\mathcal{L}_{\mathcal{G}})$ the eigenvalue corresponding to the spectral gap. Let $\{u, v\} \notin \mathcal{V}$ be two vertices that are not connected by an edge. Denote $\hat{\mathcal{G}} = (\mathcal{V}, \hat{\mathcal{E}})$, the new graph obtained after adding an edge between $\{u, v\}$, i.e., $\hat{\mathcal{E}} := \mathcal{E} \cup \{u, v\}$. Denote with $\mathcal{P}_f := \langle f, \hat{f}_0 \rangle$ the projection of f onto the top*

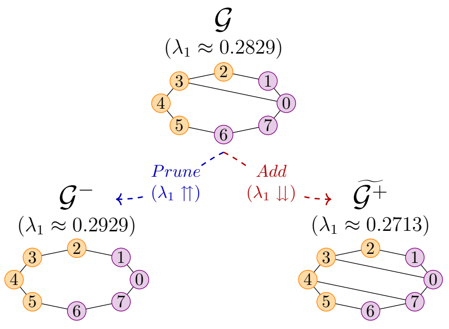

$\mathcal{G}$
($\lambda_1 \approx 0.2829$)
Prune
($\lambda_1 \upuparrows$)
Add
($\lambda_1 \downdownarrows$)
$\mathcal{G}^-$
($\lambda_1 \approx 0.2929$)
$\widetilde{\mathcal{G}^+}$
($\lambda_1 \approx 0.2713$)
0
1
2
3
4
5
6
7

Figure III.4.: The Braess paradox indicates that not all edge additions will result in spectral gap expansion. Conversely, we can delete an edge to increase the spectral gap.

eigenvector of $\hat{\mathcal{G}}$. *Define* $g\left(u, v, \mathcal{L}_{\mathcal{G}}\right) :=$

$$-\mathcal{P}_f^2 \lambda_1(\mathcal{L}_{\mathcal{G}}) - 2(1-\lambda_1(\mathcal{L}_{\mathcal{G}}))\left(\frac{\sqrt{d_u+1}-\sqrt{d_u}}{\sqrt{d_u+1}} f_u^2 + \frac{\sqrt{d_v+1}-\sqrt{d_v}}{\sqrt{d_v+1}} f_v^2\right) + \frac{2 f_u f_v}{\sqrt{d_u+1}\sqrt{d_v+1}}.$$

If $g\left(u, v, \mathcal{L}_{\mathcal{G}}\right) > 0$, then $\lambda_1(\mathcal{L}_{\mathcal{G}}) > \lambda_1(\mathcal{L}_{\hat{\mathcal{G}}})$. This implies that certain edge additions will *decrease* the spectral gap contrary to the popular belief. We can also think of the converse where certain edge deletions will increase the spectral gap.

3.2. Matrix Perturbation Theory for Proxy Spectral Gap

Now the question is can we leverage this criterion to rewire the graph? More specifically, can we prune the graph such that it increases the spectral gap? This would allow us to also afford computational savings since our GNNs could operate on a sparser graph. At first glance, this criterion seems to serve as an excellent proxy for spectral gap, however we can see that this criterion is limited, in the sense, it is expensive to evaluate this criterion for large graphs. For every edge that needs to be deleted, we would need to temporarily delete the edge, update the corresponding eigenvectors and eigenvalues, evaluate the criterion and decide whether to keep the edge or not. This is infeasible. To circumvent this complexity, we yet again leverage Matrix Perturbation Theory [70, 44] and use another approximation of the spectral gap [7] given by

$$\acute{\lambda} \approx \lambda + \Delta w_{u,v}((f_u - f_v)^2 - \lambda(f_u^2 + f_v^2)), \qquad \text{(III.7)}$$

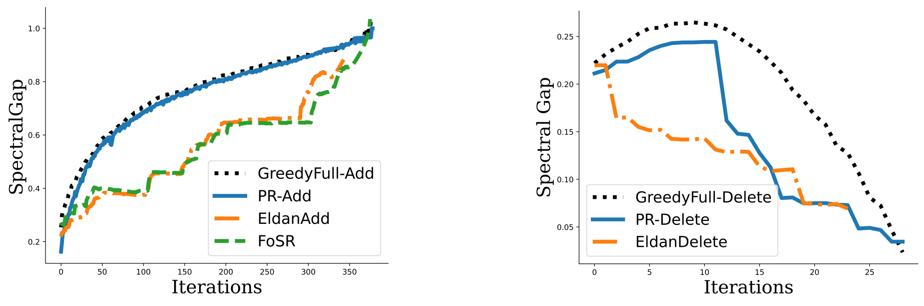
where λ is the initial eigenvalue; $\{f_u, f_v\}$ are entries of the leading eigenvector, $\Delta w_{u,v} = 1$ if we add an edge and $\Delta w_{u,v} = -1$ if we delete an edge. This is formalized in the Algorithm 1, where we present a greedy version of the Eldan criterion to prune edges in the graph.

Algorithm 1 Eldan based Greedy Graph Sparsification (ELDANDELETE)

```
Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, num. edges to prune $N$, spectral gap $\lambda_1(\mathcal{L}_\mathcal{G})$, top
  eigenvector $f$ of $\mathcal{G}$.
  repeat
    for $edges(u, v) \in \mathcal{E}$ do
      Consider $\hat{\mathcal{G}} = \mathcal{G} \setminus (u, v)$.
      {Note that the denser graph is the original $\mathcal{G}$, so we require approxima-
      tions of $\hat{f}$ and $\lambda_1(\mathcal{L}_{\hat{\mathcal{G}}})$ from the sparser $\hat{\mathcal{G}}$.}
      Estimate eigenvector $\hat{f}$ from $f$ based on the power iteration method.
      Estimate corresponding eigenvalue $\lambda_1(\mathcal{L}_{\hat{\mathcal{G}}})$ based on Eq. (III.7).
      Compute projection $\mathcal{P}_{\hat{f}}^2 = \langle \hat{f}, f_0 \rangle$.
      Compute Eldan's criterion $g(u, v, \mathcal{L}_{\hat{\mathcal{G}}})$.
    end for
    Find the edge that maximizes the criterion: $(u^-, v^-) = \underset{(u,v)\in\mathcal{E}}{\operatorname{argmax}}\, g(u, v, \mathcal{L}_{\hat{\mathcal{G}}})$
    $\hat{\mathcal{E}} = \hat{\mathcal{E}} \setminus (u^-, v^-)$.
    Update degrees $d_{u^-} = d_{u^-} - 1, d_{v^-} = d_{v^-} - 1$
    Update eigenvectors and eigenvalues of $\mathcal{G}$ accordingly.
  until $N$ edges deleted.
  Output : Sparse graph $\hat{\mathcal{G}} = (\mathcal{V}, \hat{\mathcal{E}})$.
```

3.3. Yet Another Spectral Gap Proxy

The Braess paradox via the Eldan criterion gives us a nice proxy for rewiring the graph. However, we see that although it provides theoretical guarantees for spectral gap increase when edges are deleted, it is computationally expensive to use in practice. To circumvent this problem, we propose to optimize the proxy spectral gap update given in Equation III.7 directly. This is formalized in Algorithm 2, where we use the proxy update equation as a ranking scheme to greedily delete edges. The proposed algorithm is computationally efficient as it runs in $\mathcal{O}\left(N \cdot (|\mathcal{E}| + s(\mathcal{G}))\right)$ where N is the number of edges to delete, and $s(\mathcal{G})$ denotes the complexity of the algorithm that updates the leading eigenvector and eigenvalue at the end of every iteration[3] So how good is this proxy? We plot the spectral gap expansion vs the number of edge modifications on a toy ER graph with $\mathcal{G} = (|\mathcal{V}|, |\mathcal{E}|) = (30, 58)$, comparing the spectral gap obtained by full eigenvalue decomposition, using the `ProxyAdd`, `ProxyDelete`, FoSR [36] and also the Eldan criterion in Figure III.5. We can see that our proposed proxy spectral gap methods are effective in inducing spectral gap expansions.



(a) Edge additions to improve spectral gap expansion.

(b) Edge deletions to improve spectral gap expansion.

Figure III.5.: We instantiate a toy ER graph with 30 nodes and 58 edges. We compare FoSR [36], our proxy spectral gap based methods, and our Eldan's criterion based edge methods.

4. Experiments

We will present a suite of experiments that highlights the efficacy of our proposed algorithms. In Figure III.6, we present the over-smoothing analyses for Cora, Citeseer which are homophilic datasets and Texas, Chameleon which are heterophilic datasets. We train a Linear GNN as in Section §3.1 for 500 steps of mean aggregation. We compare the results on the original graph which is unperturbed with graphs modified by adding edges using FoSR [36], `ProxyAdd` and deleted using `ProxyDelete`. Clearly, we can see that adding edges which maximize the spectral gap will accelerate the process of over-smoothing and this is worse for heterophilic graphs since we might connect nodes with different labels, leading to unnecessary information aggregation. On the other hand, our spectral pruning algorithm is successful in not only alleviating over-squashing

[3]We employ the `eigsh` function from the scipy library to obtain the initial estimate of the spectral gap and its corresponding eigenvector which internally uses the Implictly Restarted Lanczos Algorithm.

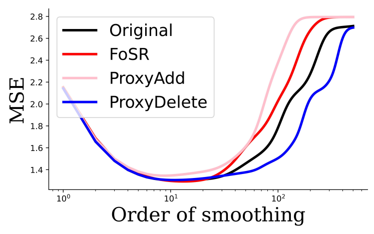
Algorithm 2 Proxy Spectral Gap based Greedy Graph Sparsification (ProxyDelete)

```
Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, num. edges to prune $N$, spectral gap $\lambda_1(\mathcal{L}_\mathcal{G})$,
  second eigenvector $f$.
  repeat
    for $(u, v) \in \mathcal{E}$ do
      Consider $\hat{\mathcal{G}} = \mathcal{G} \setminus (u, v)$.
      Calculate proxy value for the spectral gap of $\hat{\mathcal{G}}$ based on Eq. (III.7):
      $\lambda_1(\mathcal{L}_{\hat{\mathcal{G}}}) \approx \lambda_1(\mathcal{L}_\mathcal{G}) - ((f_u - f_v)^2 - \lambda_1(\mathcal{L}_\mathcal{G}) \cdot (f_u^2 + f_v^2))$
    end for
    Find the edge that maximizes the proxy: $(u^-, v^-) = \underset{(u,v)\in\mathcal{E}}{\operatorname{argmax}} \lambda_1(\mathcal{L}_{\hat{\mathcal{G}}})$.
    Update graph edges: $\mathcal{E} = \mathcal{E} \setminus (u^-, v^-)$.
    Update degrees: $d_{u^-} = d_{u^-} - 1, d_{v^-} = d_{v^-} - 1$
    Update eigenvectors and eigenvalues of $\mathcal{G}$ accordingly.
  until $N$ edges deleted.
  Output : Sparse graph $\hat{\mathcal{G}} = (\mathcal{V}, \hat{\mathcal{E}})$.
```

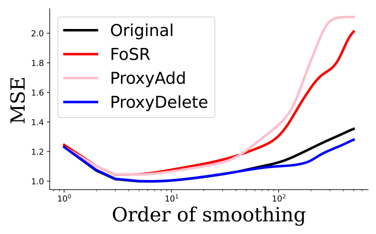
by increasing the spectral gap but also slows down detrimental smoothing and it is especially useful in heterophilic settings. We surmise this is because, spectral pruning will allow us to delete edges between nodes of different labels and thus effectively avoiding aggregating messages from nodes that need to stay distinguishable.

4.1. Long Range Graph Benchmark

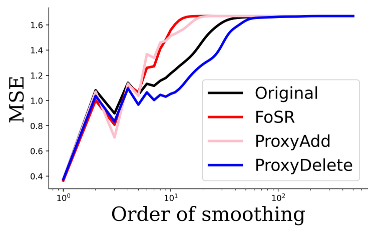
The Long Range Graph Benchmark (LRGB) datasets have been introduced [20] specifically to test methods that claim to mitigate over-squashing. The data statistics are provided in Table III.1. These datasets panning domains like computer vision and chemistry require propagating long-range information to accurately learn representations and the only way to ensure our GNNs perform well on these datasets is if they are able to overcome over-squashing. We adopt the code base of [71] for our experiments. We compare our methods `ProxyAdd` and `ProxyDelete` with a recently proposed strong baseline called DRew [27] in Table III.2 with a GCN backbone [38]. We can see that our proposed rewiring methods help alleviate over-squashing and improve performance significantly over the baselines.



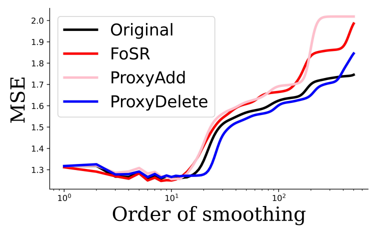
(a) Cora dataset with 200 edges added (FoSR, ProxyAdd) and 20 deleted (ProxyDelete).



(b) Citeseer dataset with 200 edges added (FoSR, ProxyAdd) and 100 deleted (ProxyDelete).



(c) Texas dataset with 200 edges added (FoSR, ProxyAdd) and 5 deleted (ProxyDelete) .



(d) Chameleon dataset with 200 edges added (FoSR, ProxyAdd) and 250 deleted (ProxyDelete).

Figure III.6.: We show on real-world graphs that spectral pruning can not only mitigate over-squashing by improving the spectral gap but also slows down the rate of smoothing, thus effectively preventing over-smoothing as well.

4.2. Large Heterophilic dataset

Our performance improvements on the LRGB datasets reinforce the effectiveness of our proposed rewiring schemes for mitigating over-squashing. But we also want our method to alleviate the problem of detrimental over-smoothing. Unfortunately, there are no standard benchmark datasets to measure over-smoothing, this is understandable since any GNN model operating on the simplest of graph with either increased rounds of aggregation steps or by increasing model depth is prone to over-smoothing. Thus, one of the ways to quantify if the proposed method is successful in tackling over-smoothing is to assess if you are able to train deeper GNN models without causing performance drop. We use the newly introduced large heterophilic graphs [57] and train $5, 10, 20$ layers GCN [38] and GAT [73] on Roman-empire, Amazon-ratings and

Table III.1.: Statistics of the LRGB Dataset

Dataset	Domain	Task	TotalGraphs	TotalNodes	TotalEdges	Metric
PascalVOC-SP	Computer Vision	Node Classification	11,355	5,443,545	30,777,444	macro F1
Peptides-func	Chemistry	Graph Classification	15,535	2,344,859	4,773,974	AP
Peptides-Struct	Chemistry	Graph Regression	15,535	2,344,859	4,773,974	MAE

Table III.2.: Results on Long Range Graph Benchmark Datasets

Dataset	PascalVOC-SP (Test F1 ↑)	Peptides-Func (Test AP↑)	Peptides-struct (Test MAE ↓)
GCN Baseline	0.1268±0.0060	0.5930±0.0023	0.3496±0.0013
DRew+GCN	0.1848±0.0107	**0.6996±0.0076**	0.2781±0.0028
FoSR+GCN	0.2157±0.0057	0.6526±0.0014	0.2499±0.0006
ProxyAdd+GCN	**0.2213±0.0011**	0.6789±0.0002	**0.2465±0.0004**
ProxyDelete+GCN	**0.2170±0.0015**	0.6908±0.0007	**0.2470±0.0080**

Minesweeper datasets. We compare the GNN (GCN,GAT) operating on the original graph, GNN+FoSR which is graph modified by adding edges using FoSR [36], GNN+EldanAdd and EldanDelete which uses the Eldan criterion (§III.3.1) for rewiring the graph and finally our `ProxyAdd` and `ProxyDelete` methods. We use the code base of [57] to run the experiments. Note that all the architectures for these experiments are augmented with normalization techniques [2, 34] and skip connections [30] to stabilise the training. The results are presented in Tables III.3,III.4 and III.5. Evidently, for increasing model depth we can see that the performance is not degraded and infact we boost performance across model depth. This effectively confirms our hypothesis that spectral pruning can help with over-smoothing especially for heterophilic graphs because it will delete edges across nodes with different labels prventing unnecessary aggregation.

5. Conclusion

In this chapter we have introduced the idea of rewiring the input graph to obtain a computational structure that is amenable to message passing. GNNs are known to suffer from problems like over-squashing, which results from topological bottlenecks that prevent information flow, and over-smoothing, a phenomenon where too much information aggregation results in feature collapse rendering nodes indistinguishable. While contemporaneous works [25, 50, 36] suggest a trade-off between alleviating over-squashing and mitigating over-

Table III.3.: Node classification on Roman-Empire dataset.

Method	#EdgesAdded	Accuracy	#EdgesDeleted	Accuracy	Layers
GCN	-	70.30±0.73	-	70.30±0.73	5
GCN+FoSR	50	73.60±1.11	-	-	5
GCN+Eldan	50	72.11±0.80	50	**79.14±0.73**	5
GCN+ProxyGap	50	**77.54±0.74**	20	77.45±0.68	5
GAT	-	80.89±0.70	-	80.89±0.70	5
GAT+FoSR	50	81.88±1.07	-	-	5
GAT+Eldan	50	81.13±0.50	100	82.12±0.69	5
GAT+ProxyGap	50	**86.07±0.46**	20	**86.00±0.36**	5
GCN	-	68.89±0.77	-	68.89±0.77	10
GCN+FoSR	100	73.85±1.26	-	-	10
GCN+Eldan	100	75.39±0.96	100	**80.40±0.54**	10
GCN+ProxyGap	20	78.31±0.47	20	78.19±0.71	10
GAT	-	80.23±0.59	-	80.23±0.59	10
GAT+FoSR	100	81.37±1.14	-	-	10
GAT+Eldan	100	**87.19±0.38**	20	86.90±0.37	10
GAT+ProxyGap	20	83.45±0.42	20	86.44±0.40	10
GCN	-	67.77±0.90	-	67.77±0.90	20
GCN+FoSR	100	75.14±1.43	-	-	20
GCN+Eldan	100	75.52±1.16	20	**80.37±0.70**	20
GCN+ProxyGap	50	77.96±0.65	20	78.03±0.71	20
GAT	-	79.22±0.70	-	79.22±0.70	20
GAT+FoSR	100	80.64±1.12	-	80.64±1.12	20
GAT+Eldan	100	**86.79±0.58**	50	86.70±0.50	20
GAT+ProxyGap	10	86.25±0.63	20	86.15±0.61	20

Table III.4.: Node classification on Amazon-Ratings.

Method	#EdgesAdded	Accuracy	#EdgesDeleted	Accuracy	Layers
GCN	-	47.20±0.33	-	47.20±0.33	10
GCN+FoSR	25	49.68±0.73	-	-	10
GCN+Eldan	25	48.71±0.99	100	**50.15±0.50**	10
GCN+ProxyGap	10	49.72±0.41	50	**49.75±0.46**	10
GAT	-	47.43±0.44	-	47.43±0.44	10
GAT+FoSR	25	51.36±0.62	-	-	10
GAT+Eldan	25	51.68±0.60	50	**51.80±0.27**	10
GAT+ProxyGap	20	49.06±0.92	100	**51.72±0.30**	10
GCN	-	47.32±0.59	-	47.32±0.59	20
GCN+FoSR	100	49.57±0.39	-	-	20
GCN+Eldan	50	**49.66±0.31**	20	48.32±0.76	20
GCN+ProxyGap	50	49.48±0.59	500	**49.58±0.59**	20
GAT	-	47.31±0.46	-	47.31±0.46	20
GAT+FoSR	100	51.31±0.44	-	-	20
GAT+Eldan	20	51.40±0.36	20	**51.64±0.44**	20
GAT+ProxyGap	50	47.53±0.90	20	**51.69±0.46**	20

Table III.5.: Node classification on Minesweeper.

Method	#EdgesAdded	Accuracy	#EdgesDeleted	Test ROC	Layers
GCN	-	88.57± 0.64	-	88.57± 0.64	10
GCN+FoSR	50	90.15±0.55	-	-	10
GCN+Eldan	100	**90.11±0.50**	50	89.49±0.60	10
GCN+ProxyGap	20	**89.59±0.50**	20	89.57±0.49	10
GAT	-	93.60±0.64	-	93.60±0.64	10
GAT+FoSR	100	93.14±0.43	-	-	10
GAT+Eldan	50	93.26±0.48	100	**93.82±0.56**	10
GAT+ProxyGap	20	93.60±0.69	20	**93.65±0.84**	10
GCN	-	87.41±0.65	-	87.41±0.65	20
GCN+FoSR	100	89.64±0.55	-	-	20
GCN+Eldan	72	**89.70±0.57**	10	88.90±0.44	20
GCN+ProxyGap	20	**89.46±0.50**	50	89.35±0.30	20
GAT	-	93.92±0.52	-	93.92±0.52	20
GAT+FoSR	50	93.56±0.64	-	-	20
GAT+Eldan	10	93.92±0.44	20	**95.48±0.64**	20
GAT+ProxyGap	20	**94.89±0.67**	20	94.64±0.81	20

smoothing, we challenge this assumption by proposing a novel spectral gap based graph pruning strategy that leverages the Braess paradox to show that deleting edges can increase spectral gap and thus alleviating over-squashing and also slows down detrimental smoothing by deleting edges between nodes of different labels by preventing unnecessary aggregation. We have performed extensive experiments on a large range of benchmark datasets and show we outperform most recent competitive baselines. This chapter provides an ingress to the idea of having a *computational structure* that is decoupled from the actual input graph which contains enough information to solve the task at hand. In the next chapter we will delve more deeply into this idea.

Chapter IV.

Graph-Task Alignment

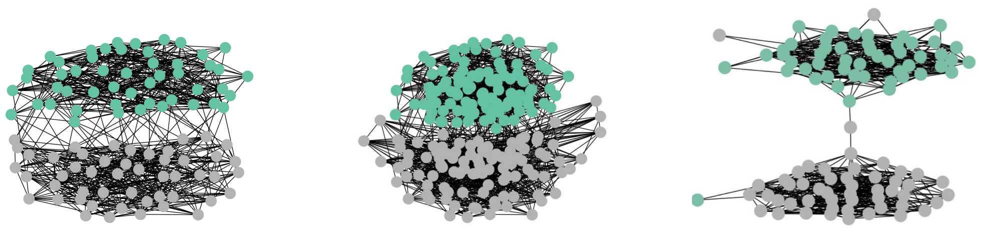
1. Introduction

In the previous chapter, we introduced the idea of a *computational structure* that is better equipped to handle the message passing. This computational structure can be decoupled from the input graph by inducing changes to the graph based on criteria such as discrete Ricci curvature [72, 25, 50], spectral gap [36, 35], effective resistance [6] and even converting the graph into an expander graph [18, 3]. In this chapter, we ask - Why does spectral maximization via graph rewiring work? What are the conditions that need to be met for the spectral rewiring to work? Should we always maximize the spectral gap? Investigating these questions in detail forms the basis of this chapter, and as a result of these analyses we propose what is called as the *graph-task alignment*, that dictates the success of GNNs when solving a downstream task such as node classification. The graph-task alignment refers to the alignment that exists between the node labels and the latent community structure of the graph, a high degree of alignment allows the GNN to solve the task much easier. We show that, rewiring methods such as [36, 35] which maximize the spectral gap overestimate their ability to enhance GNN performance. This is because, the spectral rewiring methods rely purely on topological information to modify the graph, and as a result cannot change the pre-existing alignment, but can either amplify or dampen the existing alignment. Note that the idea of finding an optimal computational structure has close ties to the paradigm of *graph structure learning* (GSL) methods [76], where the idea is to jointly optimize the graph structure and the GNN model in an end-to-end manner, such that, the graph structure is suitable for the prediction task. Some examples include Iterative Deep Graph Learning (IDGL) [15], where the objective is to use a weighted cosine similarity measure to learn the graph structure from the node embeddings and vice-versa till an optimal graph structure suitable for the prediction task is obtained. And a graph rewiring method (Deep Heterophily Graph Rewiring (DHGR))[5] that uses a similarity measure of the label/features of the node neighbours to add homophilic edges and prune heterophilic edges in a graph. Both of these methods rely on using non-robust measures such as cosine similarity [69] as a criterion to induce changes to the graph structure. In contrast to these approaches, we propose a novel graph rewiring strategy which we call - `S☼LAR - Surogate Label Aware Rewiring`[1], that adds/deletes inter-class/intra-class edges based on node labels. This is challenging since our chosen downstream task is node classification and we do not have access to labels of test nodes. To circumvent this requirement, we propose to train a surrogate GNN model, whose predictions are used as proxy labels to guide the rewiring strategy. We show through a suite of experiments that, such label-aware rewiring can indeed better align the input graph to the task at hand, enhancing the generalization performance.

2. Graph-Task Alignment in Stochastic Block Model

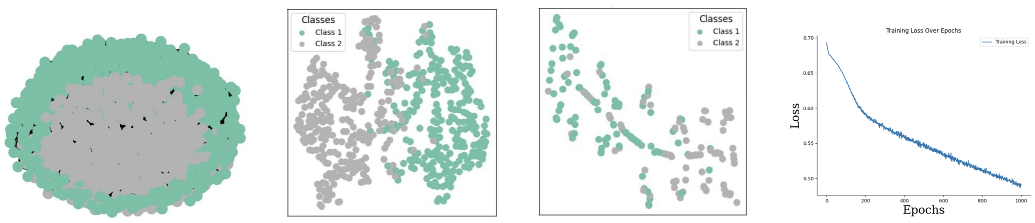
Spectral gap based rewiring techniques [36, 35] aim to maximize the second eigenvalue of the normalized Laplacian, as this is closely linked to the idea of convergence of random walks [41]. That is, given a diffusion process on the vertices of the graph, how fast can a process *mix* or more specifically settle for a stationary distribution is controlled by the spectral gap. A smaller gap involves a bottleneck (cf. 2.2), which has also been shown to cause over-squashing [1, 72] and rewiring the graph by adding or deleting edges that maximize the spectral gap transforms the input graph into a computational structure which is bottleneck-free. Consequently, a GNN operating on this modified graph has been shown to achieve better generalization performance [72, 36, 25, 50, 35]. In this section, we are interested in the question - *Does spectral gap maximization always lead to improved GNN performance?*. To answer this question, we instantiate synthetic datasets based on the Stochastic Block Model [32], which is a generative random graph model, where communities are blocks and the preferential attachments are dictated by the intra-class (p) and inter-class probabilities q respectively.

[1]This chapter will be based on the paper - C Rubio-Madrigal, A Jamadandi and R Burkholz, 'SoLAR - Surrogate Label Aware Rewiring for Graph-Task Alignment [60].



(a) Original SBM. (b) Maximizing spectral gap. (c) Minimizing spectral gap.

Figure IV.1.: Visualizing the effect of spectral graph maximization and minimization by deleting edges on a perfectly aligned SBM with 100 nodes.



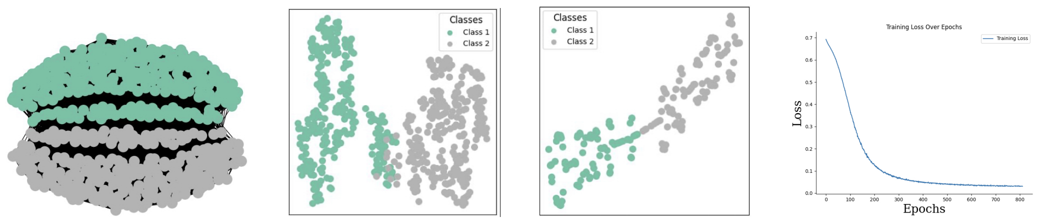
(a) SBM after spectral gap maximization. (b) Node embeddings before training a GCN. (c) Node embeddings after training a GCN. (d) Loss vs Epochs.

Figure IV.2.: GCN on the SBM after pruning edges that maximize the spectral gap.

2.1. To Maximize or Not to Maximize

The Stochastic Block Model (SBM $(p, q, \mathcal{C})$) is a generative model for random graphs with planted communities. It is characterized by the intra-class probability p and the inter-class probability q and the number of communities/blocks $\mathcal{C}$. All our experiments will have a binary community structure. Consider a 2 community SBM depicted in Figure IV.1(a). In the considered example, we have $p = 0.3$ and $q = 0.03$, meaning there are distinctly 2 communities with only few inter-community edges. Lets apply a community detection algorithm such as modularity maximization [49] to partition the graph into communities. We can measure the Normalized Mutual Information (NMI)

$$NMI(L_{class}, L_{cluster}) = \frac{2I(L_{class}; L_{cluster})}{H(L_{class}) + H(L_{cluster})} \tag{IV.1}$$



(a) SBM after spectral gap minimization.

(b) Node embeddings before training a GCN.

(c) Node embeddings after training a GCN.

(d) Loss vs Epochs.

Figure IV.3.: GCN on SBM after pruning edges that minimize the spectral gap.

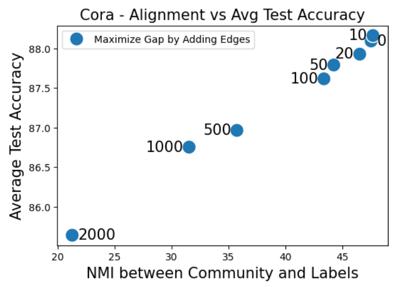
where L_{class} refers to the class labels, $L_{cluster}$ is the community labels, $H(\cdot)$ is the entropy and $I(;)$ measures the mutual information between the class labels and the assigned community labels. We get a NMI of ~ 1.0 for the graph in Figure IV.1(a) indicating that the node class labels align well with assigned community labels, or more specifically, by setting $p > q$, we instantiated a SBM which has a high degree of alignment between the node labels and the community labels, allowing a community detection algorithm to easily find the optimal partition. Applying a spectral gap maximization algorithm such as Algorithm2) to the original SBM graph will result in a graph with intra-community (due to the alignment, the intra-community edges are same as the edges connecting nodes with the same labels) edges deleted (Figure IV.1(b)), weakening the existing community structure. However,if we apply the same algorithm but minimize the spectral gap we will end up with graph in Figure IV.1(c), this yields a better NMI than the graph in Figure IV.1(b). This is because minimizing the spectral gap deletes inter-class edges which helps the community detection algorithm find the optimal partition. This leads to our theoretical result - *The higher the spectral gap of the SBM, the weaker the community strength of the graph.*

Theorem IV.2.1: *Let G be a (p-q)-SBM with N nodes in 2 equally-sized communities and intra/inter-edge probabilities $p > q$. Let G^{del} be a (p'-q)-SBM where $p' < p$, and G^{add} be a (p-q')-SBM where $q' > q$. The (expected) spectral gap of G is smaller than those of G^{del} and G^{add}: $\lambda_1(G) < \lambda_1(G^{del})$, and $\lambda_1(G) < \lambda_1(G^{add})$. In fact, the spectral gap is approximately $\propto \frac{q-p}{q+p}$.*

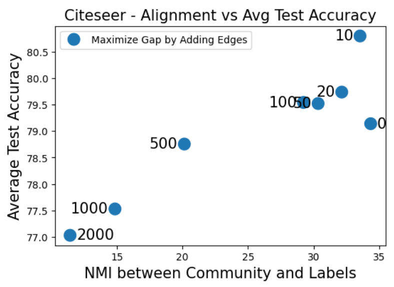
Thus, not taking into account the pre-existing alignment between the node labels and the community structure of the graph when maximizing the spectral gap leads to reduced performance. To further substantiate this hypothesis, we generate a SBM $(p, q, \mathcal{C}) = (0.3, 0.1, 2)$, with node features generated from a normal distribution $x \in X \sim \mathcal{N}(-1, 1)$. We train a 2-layered GCN [38] to perform community detection, the NMI is calculated between the ground truth community labels and the GCN assigned predictions. Figure IV.2 shows the SBM after pruning the edges that maximize spectral gap (Figure IV.2(a)), the node embeddings before and after training a GCN (Figures IV.2(b)IV.2(c)) and the training loss (Figure IV.2(d)). Similarly in Figure IV.3, we can see the SBM after pruning edges that minimize the gap. Evidently, maximizing the spectral gap diminishes the existing node label and community structure alignment, resulting in a poorer training loss. We also log different metrics in Table IV.1, before and after pruning the edges in the graph. We measure the Node Label Informativeness (NLI) and the adjusted Homophily score proposed in [58], which measure how much information a neighboring node gives to make a decision about the node under consideration and the level of homophily respectively. We also report the NMI between the predicted labels and the ground truth labels. Clearly, when there is a high degree of alignment between the community structure and the node labels, minimizing the spectral gap is beneficial to solving the task.

Table IV.1.: Different metrics for the SBM graph after training a GCN.

SpectralGap Before	SpectralGap After	NLI Before	NLI After	Homophily Before	Homophily After	NMI
0.4933	0.8010	0.18	0.02	0.49	0.15	0.006
0.4933	0.2047	0.18	0.46	0.49	0.73	0.430



(a) GCN test accuracy vs NMI for Cora.



(b) GCN test accuracy vs NMI for Citeseer.

Figure IV.4.: Maximizing the spectral gap (using [35]) on Cora and Citeseer reduces both the graph-task alignment and the test accuracy.

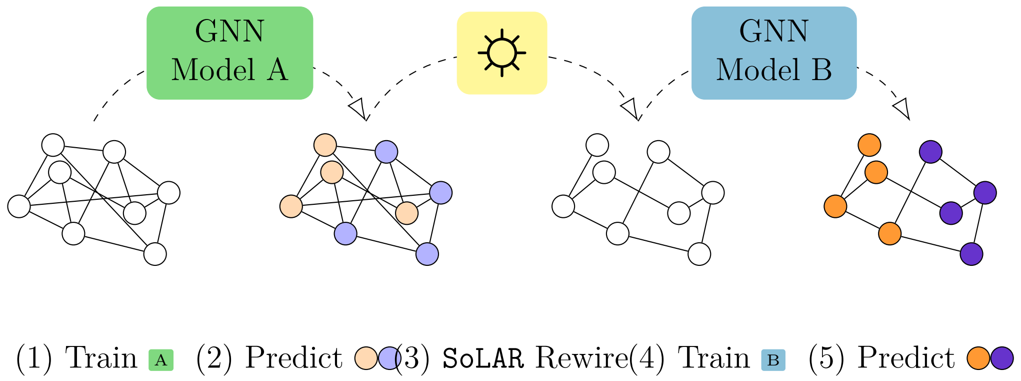


Figure IV.5.: `SoLAR`: Surrogate Label Aware Rewiring. A first model A is trained on the original graph (1), and used to predict its test labels (2). The graph is then rewired (3) based on these predictions: adding same-class edges, and/or deleting different-class edges. A second model B is trained on the new graph (4). This trained model can be used to test performance (5), but also circle back to step (2) for an iterative version of `SoLAR`. We write **A☼B** to indicate different model combinations, used in the given order.

The SBM is a simple paradigmatic example to demonstrate the nuances of aligning the graph with the task at hand and the effect of spectral rewiring, however in real-world graphs this is not trivial as the alignment can take more complex forms. For instance, homophilic graphs, where nodes with same labels are highly likely to be connected resemble the SBM setup we described earlier, but in heterophilic graphs, the nodes might be connected to different label nodes. To demonstrate this, we train a GCN [38] on Cora and Citeseer, two homophilic graphs, for different edge additions that maximize the spectral gap via our algorithm 2. We measure the NMI between the ground truth labels and the GCN assigned labels in Figure IV.4. Clearly, adding edges that maximize the spectral gap adds edges between nodes of different labels, weakening the latent community structure (as seen by the decrease in NMI), which further affects the GCN performance.

3. Surrogate Label Aware Rewiring

We saw in the previous section that spectral based graph rewiring techniques fail to induce any kind of graph-task alignment but can only enhance/dampen

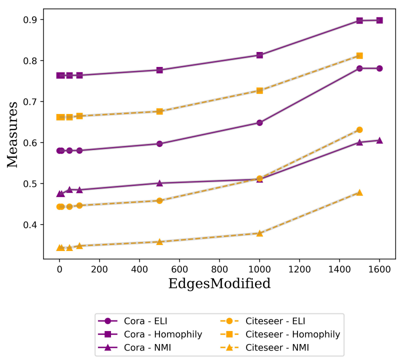
Table IV.2.: Node classification using one-shot SoLAR on large heterophilic graphs.

Method	Roman-Empire	Amazon-Ratings	Penn94
GCN	77.74±0.60	47.66±0.54	82.29±0.77
GATv2	82.52±0.50	47.66±0.95	81.85±3.02
GCN+FoSR	73.60±1.11	49.68±0.73	69.73±7.83
GATv2+FoSR	81.88±1.07	51.36±0.62	72.56±5.55
GCN☼GCN+Delete	80.90±0.14	50.30±0.09	83.59±1.40
GCN☼GCN+Add	81.13±0.21	49.86±0.11	**83.65±1.69**
GATv2☼GATv2+Delete	**84.32±0.80**	**52.06±0.00**	83.58±1.60
GATv2☼GATv2+Add	**84.27±0.40**	**52.08±0.09**	**83.60±1.32**

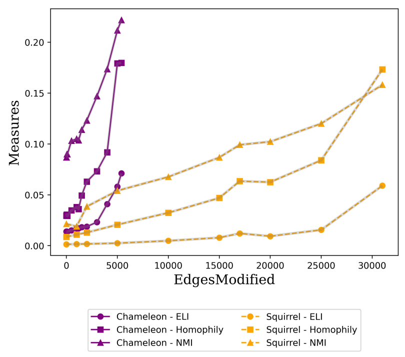
the existing alignment. In fact, since the spectral gap based rewiring techniques rely purely on the topological information, it is possible they can enhance a badly aligned graph or might even dampen an already existing good alignment. Methods like [36, 35] rely on modifying a small number of edges so that the GNN performance on the downstream task is improved, but changing the original degree distribution of the graph (as seen in Figure IV.4) too much, will have a negative effect on the generalization performance. To circumvent these limitations we need a graph rewiring strategy that can induce the graph-task alignment directly, and to do this, we need a way to ensure the node labels align with the latent community structure of the graph. We propose a novel graph rewiring strategy called S☼LAR - Surrogate Label Aware Rewiring, that modifies the graph based on node labels. Given a unweighted and undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $|\mathcal{V}|$ represents the nodes and $|\mathcal{E}|$ the edges, the adjacency matrix $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ encodes the graph topology. The degree normalized adjacency matrix is given by $\hat{A} = \tilde{D}^{-1/2}(A+I)\tilde{D}^{-1/2}$, here X is the associated node features. The task is to perform node classification in a transductive setting. That is, given a set of nodes $\mathcal{V}_{train}$ whose labels $\mathcal{Y}_{train}$ are available, we are required to predict the labels of nodes $\mathcal{V}_{test} = \mathcal{V} \backslash \mathcal{V}_{train}$. Let $\mathbf{Z}$ be the predictions made by the GCN [38]

$$\mathbf{Z} = softmax(\hat{A}\sigma(\hat{A}X\mathbf{\Theta^{(0)}})\mathbf{\Theta^{(1)}} \tag{IV.2}$$

where $\sigma(\cdot)$ is a non-linear activation function such as ReLU and $\mathbf{\Theta}$ the weight matrix. We are interested in rewiring the graph based on node labels, note that we have access to the labels of the training nodes but not the test nodes. A simple yet highly effective strategy is to train a GNN model (a surrogate model) to obtain predictions on the test nodes and use them as proxy labels for graph rewiring. Our rewiring process works in two stages. In the first stage, we instantiate a surrogate GNN $Z_1 = f_{surrogate}(\mathcal{G}, \mathbf{\Theta})$ (such as in Equation (IV.2)), and train it to convergence to obtain a set of predicted labels. We then use the predicted labels, $Z_1 = \mathcal{Y}_{proxy}$, to rewire the graph by either deleting inter-class edges and/or adding intra-class edges to obtain a rewired graph $\hat{\mathcal{G}} = (\mathcal{V}, \hat{\mathcal{E}})$ —we use the predictions on the test set, as we already have access to the ground truth labels for the nodes in the train set. In the second stage, we instantiate a second 'training' GNN $f_{train}(\hat{\mathcal{G}}, \mathbf{\Theta})$, which operates on the new computational structure. Note that $f(\mathcal{G}, \mathbf{\Theta})$ can be any model which is expressive enough. We present two variants of our rewiring procedure called one-shot `SoLAR` and Iterative `SoLAR`, where we can either modify the graph in a one-shot way or iteratively modify the graph. We present a comprehensive set of experiments in the next section that highlight the effectiveness of our proposed approach.



(a) ELI,NMI and Homophily scores after deleting edges using GCN☼GCN for homophilic graphs.



(b) ELI,NMI and Homophily scores after deleting edges using GCN☼GCN for heterophilic graphs.

Figure IV.6.: The effect of one-shot rewiring on ELI, homophily and NMI on Cora, Citeseer, Chameleon and Squirrel datasets

4. Experiments

4.1. One-shot `SoLAR`

We perform a comprehensive set of experiments to validate our hypothesis. We conduct experiments on Cora [45], Citeseer [65], Pubmed [48], CS, Physics and Photo [66] which are homophilic datasets. And Cornell, Texas, Squirrel, Chameleon, Squirrel, Actor, Roman-empire, Amazon-ratings [57] and Penn94 [43] which are heterophilic datasets. Our main backbone models are SGC [74], GCN[38] and GATv2 [10]. We denote the model combinations as GNN1☼GNN2, where GNN1 = {SGC,GCN, GATv2} and GNN2 = {SGC,GCN,GATv2} with extensions Add/Delete to represent whether we are adding intra-class edges or deleting inter-class edges. The first model is the surrogate model whose predictions are used as proxy labels and the second model represents the final training model. We also present results for both additions and deletions simultaneously. Our baseline comparisons are against FoSR [36] which adds edges based on spectral gap maximization, Proxydelmin [35] which deletes edges that minimize the spectral gap. Note that any expressive GNN model can be used including more complex architectures like Graph Transformers, but since the rewiring framework involves training 2 GNN models, we would want the training to be computationally less expensive. We also resort to deleting/adding edges randomly during the rewiring to further ensure the method is computationally friendly. In Tables IV.2, IV.5,IV.6, we present the results for the large heterophilic datasets, homophilic and other small and medium sized heterophilic datasets for one-shot `SoLAR` respectively with GCN and GATv2 model combinations. In Tables IV.3 and IV.4, we present results with Simplified Graph Convolution (SGC) [74], a version of GCN with all the weight matrices collapsed into one matrix and with non-linearity such as ReLU removed.

4.2. Iterative-`SoLAR`

In Figures IV.7(a),IV.7(b), we present the results for the iterative version of `SoLAR` which uses a combination of GCN☼GATv2☼GCN models. The GCN model is trained first to provide proxy labels, based on the labels a subset of edges are added/deleted, for instance for Cora we can approximately delete 1500 inter-class edges, instead of deleting all the edges in a one-shot manner, we can choose to delete 750 edges in 2 iterations. After deleting 750 edges we now train a GATv2 and then update the predictions of the proxy labels from GATv2 delete the remaining 750 edges and finally train a GCN model again. In Figures IV.8(a) and IV.8(b) we present results for GATv2☼GCN☼GATv2 models.Clearly, from the figures we can see that iterative version yields significantly higher performance than one-shot methods.

5. Discussion

In this chapter, we have proposed the idea of graph-task alignment as one of the key factors influencing the GNN performance. Furthering our analysis, we

Table IV.3.: Node classification results on homophilic graphs with SGC.

Method	Cora	Citeseer	Pubmed	CS	Physics	Photo
SGC	88.78±0.48	80.51±0.59	82.47±0.41	93.39±0.18	95.21 ± 0.06	86.48±1.00
GCN	87.94±3.35	79.38±3.48	81.99±1.42	92.44±0.67	94.49 ± 0.04	**92.89±1.23**
GCN☼SGCDelete	88.10±0.48	80.14±0.64	82.12±0.32	93.68±0.13	**94.97±0.03**	89.93±0.83
GCN☼SGCAdd	89.02±0.48	79.14±0.72	82.06±0.37	93.43±0.18	OOM	87.15±0.98
GATv2☼SGCDelete	**89.55±0.56**	**82.28±0.89**	**82.55±0.36**	**93.77±0.22**	94.48±0.07	89.96±0.89
GATv2☼SGCAdd	89.16±0.50	80.85±0.83	81.96±0.38	93.44±0.18	OOM	87.26±0.98

Table IV.4.: Node classification results on heterophilic graphs with SGC.

Method	Cornell	Texas	Wisconsin	Chameleon	Squirrel	Actor
SGC	65.14±1.70	73.70±1.70	66.04±1.40	55.26±1.12	45.16±1.12	29.23 ±0.55
GCN	68.31±8.13	73.47±10.13	66.14±9.23	54.64±6.94	43.25±6.32	28.26±3.22
GCN☼SGCDelete	67.89±1.75	74.89±2.04	69.37±1.19	57.79±1.29	45.85±1.35	28.32±0.57
GCN☼SGCAdd	68.39±1.89	74.63±1.95	67.53±1.38	53.87±1.26	43.08±1.25	26.85±0.52
GATv2☼SGCDelete	75.86±1.86	83.13±2.13	74.04±1.40	**66.82±2.11**	**47.71±1.35**	**30.32±0.83**
GATv2☼SGCAdd	**83.73±2.16**	**86.40±2.28**	**81.09±1.83**	64.20±2.07	45.45±1.22	27.01±0.60

can validate why this node label aware graph rewiring works. The Edge Label Informativeness (ELI) proposed in [58] has been shown to correlate well with the GNN performance. Given an edge $(u, v) \in |\mathcal{E}|$, the class labels of the nodes are y_u and y_v respectively. We are interested in measuring how much does knowing the label y_u give information about predicting the label y_v. We can measure the normalized mutual information as,

$$ELI = \frac{H(y_u) - H(y_u|y_v)}{H(y_u)} \tag{IV.3}$$

where we can write, $I(y_u, y_v) = H(y_u) - H(y_u|y_v)$ and $H(\cdot)$ represents the entropy. In Figures IV.6(a) and IV.6(b), we measure the ELI, the homophily score proposed in [58] and NMI (Equation IV.1) which is measured by performing community detection on the rewired graph for Cora, Citeseer, Squirrel and Chameleon datasets after deleting inter-class edges using GCN☼GCN. For increasing edge deletions, we can see that the measures consistently increase for both homophilic and heterophilic graphs, this supports our hypothesis that our proposed rewiring scheme helps in the downstream task by improving the available information in the neighbourhood by eliminating confusing edges that might lead to detrimental information aggregation. In Figure IV.9, we plot the T-SNE plots for the node embeddings for Cora and Squirrel after training it on GCN and GCN☼GCN Deletions. We can see that the node embeddings belonging to different classes are well separated, further reinforcing our hypothesis that node label based rewiring is beneficial since it is successful in inducing the

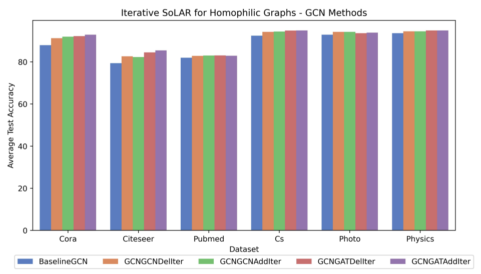
Table IV.5.: Node classification on homophilic graphs using one-shot `SoLAR`.

Method	Cora	Citeseer	Pubmed	CS	Physics	Photo
GCN	87.94±3.35	79.38±3.48	81.99±1.42	92.44±0.67	93.64±0.16	92.89±1.23
GATv2	89.13±3.13	81.92±4.81	81.83±1.04	91.90±1.59	94.07±0.44	91.22±2.18
GCN+FoSR	88.74±2.70	79.48±3.77	82.22±1.24	93.54±0.80	94.72±0.21	90.57±3.82
GATv2+FoSR	89.72±2.91	81.75±4.86	81.29±2.31	92.35±1.21	93.96±0.40	90.48±2.57
GCN+Proxydelmin	89.35±2.92	79.56±2.96	82.89±1.53	93.66±0.83	94.61±0.27	92.46±2.16
GATv2+Proxydelmin	89.61±3.00	81.57±3.56	81.59±1.43	93.31±1.21	94.43±0.33	93.86±1.61
GCN✡GCN+Delete	**90.17±2.82**	82.22±4.01	82.61±1.16	93.00±0.21	93.96±0.10	93.75±0.99
GCN✡GCN+Add	90.06±2.56	**83.26±4.44**	**83.05±2.50**	92.46±0.56	**95.47±0.31**	92.13±0.32
GATv2✡GATv2+Delete	90.06±3.31	**83.01±4.32**	82.41±2.46	**94.16±1.79**	**95.01±0.54**	**93.78±1.30**
GATv2✡GATv2+Add	89.63±3.16	81.78±4.44	81.32±1.66	92.79±1.58	94.25±0.46	**93.36±1.93**
GCN✡GATv2+Delete	**90.23±0.59**	81.48±0.77	83.15±0.26	93.96±0.15	87.01±2.09	94.80±0.03
GCN✡GATv2+Add	90.01±0.58	81.42±0.85	82.29±0.31	93.41±0.22	84.61±2.51	94.30±0.05
GATv2✡GCN+Delete	**90.42±0.65**	**83.93±0.90**	**83.20±0.28**	93.38±0.26	92.08±0.62	94.56±0.04
GATv2✡GCN+Add	**90.47±0.60**	**83.44±0.86**	82.71±0.27	93.65±0.17	92.47±0.44	94.67±0.03

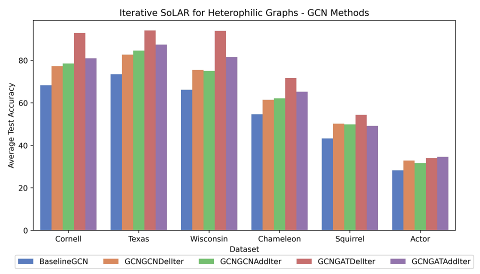
Table IV.6.: Node classification on heterophilic graphs using one-shot `SoLAR`.

Method	Cornell	Texas	Wisconsin	Chameleon	Squirrel	Actor
GCN	68.31±8.13	73.47±10.13	66.14±9.23	54.64±6.94	43.25±6.32	28.26±3.22
GATv2	86.84±9.78	89.01±10.43	87.56±9.20	61.79±10.20	45.71±5.12	29.41±2.98
GCN+FoSR	71.64±9.80	73.93±10.23	65.85±7.73	54.40±6.58	42.80±6.40	28.66±3.21
GATv2+FoSR	76.12±6.51	78.15±7.81	74.08±9.01	46.48 ± 4.97	47.40±7.17	27.45±3.61
GCN+Proxydelmin	81.94±7.96	83.46±10.90	70.63±7.68	53.64±6.00	41.26±5.35	28.58±2.93
GATv2+Proxydelmin	85.40±7.64	83.44±9.52	79.78±11.26	66.15±12.01	45.02±5.13	**32.37±4.36**
GCN✡GCN+Delete	68.35±8.54	74.12±9.89	67.85±7.14	57.19 ± 6.45	44.50±6.29	29.25±3.50
GCN✡GCN+Add	69.42±8.93	74.20±10.26	68.51±7.20	56.43 ± 6.16	44.04±6.34	28.16±3.22
GATv2✡GATv2+Delete	**87.40±9.89**	**90.14±10.64**	**88.32±9.08**	**68.89±11.50**	**49.10±5.59**	30.31±4.29
GATv2✡GATv2+Add	87.12±9.59	87.97±10.95	87.76±9.57	66.35±11.18	46.44±6.00	29.46±4.67
GCN ✡GATv2+Delete	84.03±2.12	86.91±2.23	84.53±1.95	60.11±1.59	47.98±1.17	30.02±0.73
GCN ✡GATv2+Add	83.11±1.88	85.01±2.10	85.96±1.70	54.99±1.42	43.17±1.10	30.09±0.93
GATv2 ✡GCN+Delete	78.63±2.01	84.65±2.12	77.65±1.86	68.60±2.20	47.89±1.36	30.91±0.94
GATv2 ✡GCN+Add	85.37±2.22	87.43±2.28	83.00±1.96	68.27±2.34	47.70±1.23	29.15±0.85

alignment. Our approach works for both homophilic and heterophilic settings contrary to methods which use non-robust feature similarity measures [33, 5] or require expensive k-hop rewiring during training [27] to be effective. Our approach is also more powerful in that it is capable of directly influencing measures (Figure IV.6) that are critical for GNN performance, as opposed to methods that rely purely on the topological characteristics of the input graph [72, 36, 50, 25, 35] like the spectral gap. However, it is important to note that the quality of rewiring largely depends on the surrogate model's ability to provide accurate labels. If the predicted labels are too noisy, they will also not be very informative for the rewiring, and may even amplify issues that were already present in the initial model.

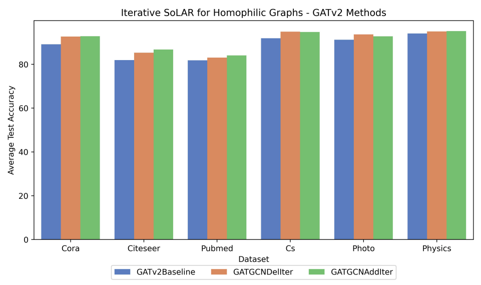


(a) Node classification on homophilic graphs using Iterative-`SoLAR` with GCN backbone.

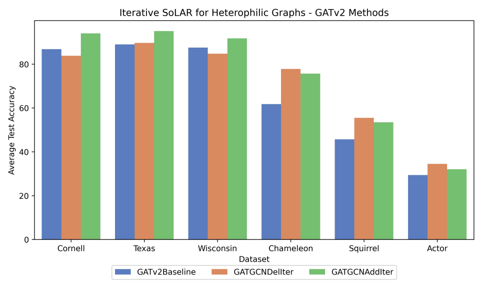


(b) Node classification on heterophilic graphs using Iterative-`SoLAR` with GCN backbone.

Figure IV.7.: Iterative-`SoLAR`+GCN.

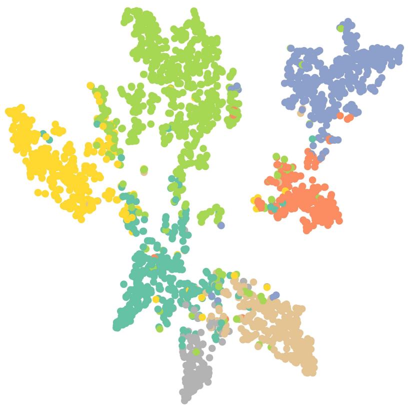


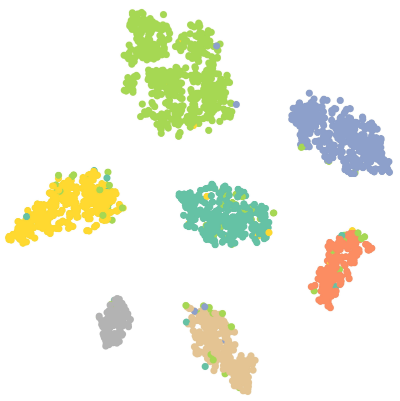
(a) Node classification on homophilic graphs using Iterative-`SoLAR` with GATv2 backbone.

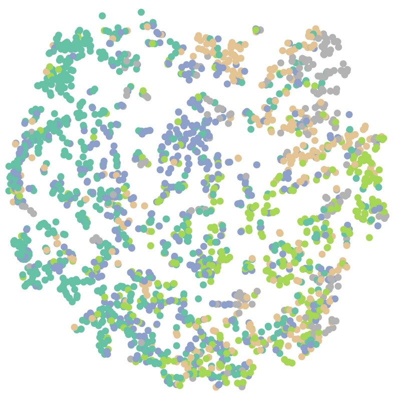


(b) Node classification on heterophilic graphs using Iterative-`SoLAR` with GATv2 backbone.

Figure IV.8.: Iterative-`SoLAR`+GATv2.


(a) GCN trained on the original Cora graph.


(b) GCN☼GCN trained on Cora with 1500 inter-class edge deletions.


(c) GCN trained on the original Squirrel graph.

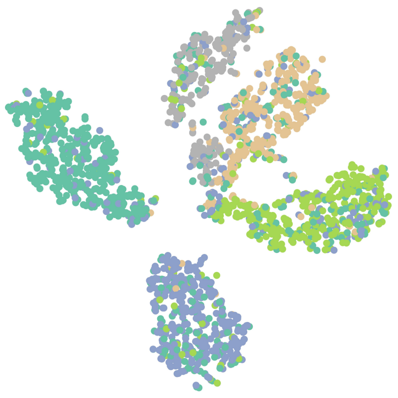

(d) GCN☼GCN trained on Squirrel with 20K inter-class edge deletions.

Figure IV.9.: We plot T-SNE for Cora and Squirrel datasets after training a GCN on the original graph and the rewired graph.

Bibliography

[1] Uri Alon and Eran Yahav. "On the Bottleneck of Graph Neural Networks and its Practical Implications". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=i80OPhOCVH2.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML].

[3] Pradeep Kr. Banerjee et al. "Oversquashing in GNNs through the Lens of Information Contraction and Graph Expansion". In: *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. Monticello, IL, USA: IEEE Press, 2022, pp. 1–8. DOI: 10.1109/Allerton49937.2022.9929363. URL: https://doi.org/10.1109/Allerton49937.2022.9929363.

[4] Peter W Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).

[5] Wendong Bi et al. *Make Heterophily Graphs Better Fit GNN: A Graph Rewiring Approach*. 2022. arXiv: 2209.08264 [cs.LG].

[6] Mitchell Black et al. "Understanding oversquashing in GNNs through the lens of effective resistance". In: *Proceedings of the 40th International Conference on Machine Learning*. ICML'23. Honolulu, Hawaii, USA: JMLR.org, 2023.

[7] Aleksandar Bojchevski and Stephan Günnemann. "Adversarial Attacks on Node Embeddings via Graph Poisoning". In: *Proceedings of the 36th International Conference on Machine Learning, ICML*. Proceedings of Machine Learning Research. PMLR, 2019.

[8] Pietro Bongini et al. "BioGNN: How Graph Neural Networks Can Solve Biological Problems". In: *Artificial Intelligence and Machine Learning for Healthcare*. Springer, 2023, pp. 211–231.

[9] Dietrich Braess, Anna Nagurney, and Tina Wakolbinger. “On a Paradox of Traffic Planning”. In: *Transportation Science* 39.4 (2005), pp. 446–450. DOI: 10.1287/trsc.1050.0127. eprint: https://doi.org/10.1287/trsc.1050.0127. URL: https://doi.org/10.1287/trsc.1050.0127.

[10] Shaked Brody, Uri Alon, and Eran Yahav. “How Attentive are Graph Attention Networks?” In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=F72ximsx7C1.

[11] Michael M. Bronstein et al. “Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges”. In: *CoRR* abs/2104.13478 (2021). arXiv: 2104.13478. URL: https://arxiv.org/abs/2104.13478.

[12] Jeff Cheeger. “A lower bound for the smallest eigenvalue of the Laplacian”. In: 1969.

[13] Jianfei Chen, Jun Zhu, and Le Song. “Stochastic Training of Graph Convolutional Networks with Variance Reduction”. In: *International Conference on Machine Learning*. 2018, pp. 941–949.

[14] Tianlong Chen et al. “A Unified Lottery Ticket Hypothesis for Graph Neural Networks”. In: *International Conference on Machine Learning*. 2021.

[15] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. “Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings”. In: *CoRR* abs/2006.13009 (2020). arXiv: 2006.13009. URL: https://arxiv.org/abs/2006.13009.

[16] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[17] Fan Chung and Stephen J. Young. “Braess’s Paradox in Large Sparse Graphs”. In: *Internet and Network Economics*. Springer Berlin Heidelberg, 2010.

[18] Andreea Deac, Marc Lackenby, and Petar Veličković. “Expander Graph Propagation”. In: *The First Learning on Graphs Conference*. 2022. URL: https://openreview.net/forum?id=IKevTLt3rT.

[19] Austin Derrow-Pinion et al. “ETA Prediction with Graph Neural Networks in Google Maps”. In: *CoRR* abs/2108.11482 (2021).

[20] Vijay Prakash Dwivedi et al. *Long Range Graph Benchmark*. 2023. arXiv: 2206.08164 [cs.LG].

[21] Ronen Eldan, Miklós Z. Rácz, and Tselil Schramm. “Braess’s paradox for the spectral gap in random graphs and delocalization of eigenvectors”. In: *Random Structures & Algorithms* 50 (2017).

[22] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. “Diffusion Improves Graph Learning”. In: *Conference on Neural Information Processing Systems (NeurIPS)*. 2019.

[23] Justin Gilmer et al. “Neural Message Passing for Quantum Chemistry”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML’17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1263–1272.

[24] Francesco Di Giovanni et al. *On Over-Squashing in Message Passing Neural Networks: The Impact of Width, Depth, and Topology*. 2023. arXiv: `2302.02941 [cs.LG]`.

[25] Jhony H. Giraldo et al. “On the Trade-off between Over-Smoothing and Over-Squashing in Deep Graph Neural Networks”. In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. CIKM ’23. Birmingham, United Kingdom: Association for Computing Machinery, 2023, pp. 566–576. DOI: `10.1145/3583780.3614997`. URL: `https://doi.org/10.1145/3583780.3614997`.

[26] M. Gori, G. Monfardini, and F. Scarselli. “A new model for learning in graph domains”. In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2. 2005, 729–734 vol. 2. DOI: `10.1109/IJCNN.2005.1555942`.

[27] Benjamin Gutteridge et al. “DRew: Dynamically Rewired Message Passing with Delay”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 12252–12267.

[28] Richard Hamilton. “The ricci flow on surfaces”. In: *Mathematics and general relativity, Proceedings of the AMS-IMS-SIAM Joint Summer Research Conference in the Mathematical Sciences on Mathematics in General Relativity*. 1988.

[29] William L. Hamilton, Zhitao Ying, and Jure Leskovec. “Inductive Representation Learning on Large Graphs”. In: *NIPS*. 2017, pp. 1024–1034.

[30] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`.

[31] Christopher Hoffman, Matthew Kahle, and Elliot Paquette. "Spectral gaps of random graphs and applications". In: *International Mathematics Research Notices* 2021.11 (2021), pp. 8353–8404.

[32] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. "Stochastic blockmodels: First steps". In: *Social Networks* 5.2 (1983), pp. 109–137. ISSN: 0378-8733. DOI: `https://doi.org/10.1016/0378-8733(83)90021-7`. URL: `https://www.sciencedirect.com/science/article/pii/0378873383900217`.

[33] Qian Huang et al. "Combining Label Propagation and Simple Models Out-performs Graph Neural Networks". In: *CoRR* abs/2010.13993 (2020).

[34] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* 2015. arXiv: `1502.03167 [cs.LG]`.

[35] Adarsh Jamadandi, Celia Rubio-Madrigal, and Rebekka Burkholz. *Spectral Graph Pruning Against Over-Squashing and Over-Smoothing.* 2024. arXiv: `2404.04612 [cs.LG]`.

[36] Kedar Karhadkar, Pradeep Kr. Banerjee, and Guido Montufar. "FoSR: First-order spectral rewiring for addressing oversquashing in GNNs". In: *The Eleventh International Conference on Learning Representations.* 2023. URL: `https://openreview.net/forum?id=3YjQfCLdrzz`.

[37] Nicolas Keriven. "Not too little, not too much: a theoretical analysis of graph (over)smoothing". In: *The First Learning on Graphs Conference.* 2022. URL: `https://openreview.net/forum?id=KQNsbAmJEug`.

[38] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *ICLR.* 2017.

[39] Remi Lam et al. "Learning skillful medium-range global weather forecasting". In: *Science* 382.6677 (2023), pp. 1416–1421.

[40] Adrien Leman. "THE REDUCTION OF A GRAPH TO CANONICAL FORM AND THE ALGEBRA WHICH APPEARS THEREIN". In: 1968.

[41] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times.* American Mathematical Society, 2006.

[42] Guohao Li et al. "DeepGCNs: Can GCNs Go as Deep as CNNs?" In: *The IEEE International Conference on Computer Vision (ICCV).* 2019.

[43] Derek Lim et al. “Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021. URL: https://openreview.net/forum?id=DfGu8WwT0d.

[44] Ulrike von Luxburg. “A tutorial on spectral clustering”. In: *Statistics and Computing* 17.4 (2007), pp. 395–416. DOI: 10.1007/s11222-007-9033-z. URL: https://doi.org/10.1007/s11222-007-9033-z.

[45] Andrew Kachites McCallum et al. “Automating the Construction of Internet Portals with Machine Learning”. In: *Information Retrieval* 3.2 (2000), pp. 127–163. DOI: 10.1023/A:1009953814988. URL: https://doi.org/10.1023/A:1009953814988.

[46] Christopher Morris et al. “Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (2019), pp. 4602–4609. DOI: 10.1609/aaai.v33i01.33014602. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4384.

[47] Nimrah Mustafa, Aleksandar Bojchevski, and Rebekka Burkholz. “Are GATs Out of Balance?” In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://openreview.net/forum?id=qY7UqLoora.

[48] Galileo Namata et al. “Query-driven Active Surveying for Collective Classification”. In: 2012.

[49] M. E. Newman. “Modularity and community structure in networks”. In: *Proceedings of the National Academy of Sciences* 103.23 (2006), pp. 8577–8582. DOI: 10.1073/pnas.0601602103.

[50] Khang Nguyen et al. *Revisiting Over-smoothing and Over-squashing Using Ollivier-Ricci Curvature*. 2023. arXiv: 2211.15779 [cs.LG].

[51] Chien-Chun Ni et al. “Network Alignment by Discrete Ollivier-Ricci Flow”. In: *Graph Drawing and Network Visualization*. Ed. by Therese Biedl and Andreas Kerren. Springer International Publishing, 2018, pp. 447–462.

[52] Chien-Chun Ni et al. “Ricci curvature of the Internet topology”. In: *2015 IEEE Conference on Computer Communications (INFOCOM)*. 2015, pp. 2758–2766. DOI: 10.1109/INFOCOM.2015.7218668.

[53] Hoang NT and Takanori Maehara. “Revisiting Graph Neural Networks: All We Have is Low-Pass Filters”. In: *ArXiv* abs/1905.09550 (2019).

[54] Kenta Oono and Taiji Suzuki. "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification". In: *International Conference on Learning Representations*. 2020.

[55] Pál András Papp et al. "DropGNN: Random Dropouts Increase the Expressiveness of Graph Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021. URL: https://openreview.net/forum?id=fpQojkIV5q8.

[56] Hongbin Pei et al. "Geom-GCN: Geometric Graph Convolutional Networks". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=S1e2agrFvS.

[57] Oleg Platonov et al. "A critical look at evaluation of GNNs under heterophily: Are we really making progress?" In: *The Eleventh International Conference on Learning Representations*. 2023.

[58] Oleg Platonov et al. "Characterizing Graph Datasets for Node Classification: Homophily-Heterophily Dichotomy and Beyond". In: *The Second Learning on Graphs Conference*. 2023. URL: https://openreview.net/forum?id=D4GLZkTphJ.

[59] Patrick Reiser et al. "Graph neural networks for materials science and chemistry". In: *Communications Materials* 3.1 (2022), p. 93. URL: https://doi.org/10.1038/s43246-022-00315-6.

[60] Celia Rubio-Madrigal, Adarsh Jamadandi, and Rebekka Burkholz. *SoLAR : Surrogate Label Aware Graph Rewiring for Graph-Task Alignment*. 2024. URL: https://adarshmj.github.io/assets/publications/SOLAR__Surrogate_Label_Aware_Rewiring_for_Graph_Task_Alignment_in_GNNs.pdf.

[61] T. Konstantin Rusch, Michael M. Bronstein, and Siddhartha Mishra. *A Survey on Oversmoothing in Graph Neural Networks*. 2023. arXiv: 2303.10993 [cs.LG].

[62] Justin Salez. *Sparse expanders have negative curvature*. 2021. arXiv: 2101.08242 [math.PR].

[63] Alvaro Sanchez-Gonzalez et al. "Learning to Simulate Complex Physics with Graph Networks". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 8459–8468. URL: https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html.

[64] Franco Scarselli et al. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605.

[65] Prithviraj Sen et al. "Collective Classification in Network Data". In: *AI Magazine* 29.3 (2008), p. 93. DOI: 10.1609/aimag.v29i3.2157. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/2157.

[66] Oleksandr Shchur et al. *Pitfalls of Graph Neural Network Evaluation*. 2019. arXiv: 1811.05868 [cs.LG].

[67] Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. "Graph neural networks in particle physics". In: *Machine Learning: Science and Technology* 2.2 (2021), p. 021001. DOI: 10.1088/2632-2153/abbf9a. URL: https://doi.org/10.1088/2632-2153/abbf9a.

[68] R P Sreejith et al. "Forman curvature for complex networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2016.6 (2016), p. 063206. DOI: 10.1088/1742-5468/2016/06/063206. URL: https://dx.doi.org/10.1088/1742-5468/2016/06/063206.

[69] Harald Steck, Chaitanya Ekanadham, and Nathan Kallus. "Is Cosine-Similarity of Embeddings Really About Similarity?" In: *Companion Proceedings of the ACM on Web Conference 2024*. Vol. 201. WWW '24. ACM, May 2024, pp. 887–890. DOI: 10.1145/3589335.3651526. URL: http://dx.doi.org/10.1145/3589335.3651526.

[70] G.W. Stewart and J. Sun. *Matrix Perturbation Theory*. Computer Science and Scientific Computing. Elsevier Science, 1990. ISBN: 9780126702309. URL: https://books.google.de/books?id=l78PAQAAMAAJ.

[71] Jan Tönshoff et al. *Where Did the Gap Go? Reassessing the Long-Range Graph Benchmark*. 2023. arXiv: 2309.00367 [cs.LG].

[72] Jake Topping et al. "Understanding over-squashing and bottlenecks on graphs via curvature". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=7UmjRGzp-A.

[73] Petar Veličković et al. "Graph Attention Networks". In: *ICLR*. 2018.

[74] Felix Wu et al. "Simplifying Graph Convolutional Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 6861–6871. URL: https://proceedings.mlr.press/v97/wu19e.html.

[75] Kaixiong Zhou et al. “Dirichlet energy constrained learning for deep graph neural networks”. In: *Advances in neural information processing systems* (2021).

[76] Zhiyao Zhou et al. *OpenGSL: A Comprehensive Benchmark for Graph Structure Learning*. 2023. arXiv: `2306.10280 [cs.LG]`. URL: `https://arxiv.org/abs/2306.10280`.

Appendix A.

Implementation details

1. Reproducibility

We use Pytorch Geometric (`https://pytorch-geometric.readthedocs.io/en/latest/`) and Deep graph library (DGL) (`https://www.dgl.ai`) for all our experiments. For experiments concerning spectral gap based rewiring we use a 60/20/20 split for training,validation and testing respectively for datasets Cora, Citeseer, Pubmed, Cornell, Texas, Wisconsin, Squirrel, Chameleon and Actor. The final test accuracy is reported averaged over 10 splits, with each split further averaged over 5 random seeds. For datasets Roman-empire, Amazon-ratings and Minesweeper we use the experimental setup proposed by authors in [57]. For experiments on `SoLAR`, we use the splits proposed by authors in [66] and the final test accuracy is reported averaged over 100 splits. The code base to reproduce all the results is available here -

- `SoLAR` - `https://github.com/AdarshMJ/SoLAR`
- Spectral gap based pruning -`https://github.com/AdarshMJ/SpectralPruningBraess`

2. Hyperparameters for `SoLAR`

Below we present the hyperparameters used for `SoLAR` both one-shot and iterative versions, along with the empirical runtimes and the dataset statistics.

Table A.1.: Statistics of the graphs used. We use the largest connected component for all our experiments.

Dataset	#Nodes	#Edges
Cora	2,708	10,138
Citeseer	3,327	7,358
Pubmed	19,717	88,648
Cornell	183	277
Texas	183	279
Wisconsin	251	450
Chameleon	890	8,854
Squirrel	2,223	57,850
Actor	7,600	26,659
CS	18,333	1,63,788
Physics	34,493	4,95,924
Photo	7,650	2,38,162
Roman-empire	22,662	32,927
Amazon-ratings	24,492	93,050
Penn94	41,554	13,62,229

Table A.2.: Hyperparameters for one shot GCN☼GCN+Del

Dataset	EdgesDeleted	LR	HiddenDimension	Runtime
Cora	1500	0.01	32	71.43
Citeseer	1500	0.01	32	84.08
Pubmed	10000	0.01	32	90.79
Cornell	100	0.001	128	86.76
Texas	100	0.001	128	73.94
Wisconsin	100	0.001	128	77.23
Chameleon	5400	0.001	128	76.82
Squirrel	310000	0.001	128	78.70
Actor	16000	0.001	128	80.12
CS	22000	0.01	128	200.90
Physics	30000	0.01	128	412.47
Photo	35000	0.01	512	263.11

Table A.3.: Hyperparameters for one shot GCN☼GCN+Add

Dataset	EdgesAdded	LR	HiddenDimension	Runtime
Cora	6929	0.01	32	89.43
Citeseer	7168	0.01	32	70.88
Pubmed	352	0.01	32	94.07
Cornell	55	0.001	128	88.76
Texas	54	0.001	128	74.33
Wisconsin	41	0.001	128	86.68
Chameleon	4088	0.001	128	70.35
Squirrel	12349	0.001	128	74.85
Actor	12215	0.001	128	78.38
CS	8680	0.01	32	129.36
Physics	45991	0.01	32	351.85
Photo	26846	0.01	32	108.79

Table A.4.: Hyperparameters for one shot GAT☼GAT+Add

Dataset	EdgesAdded	LR	HiddenDimension	Runtime
Cora	9711	0.001	32	149.73
Citeseer	11996	0.001	32	192.35
Pubmed	17647	0.001	32	594.85
Cornell	37	0.001	32	134.70
Texas	55	0.001	32	123.80
Wisconsin	49	0.001	32	135.06
Chameleon	4167	0.001	32	105.65
Squirrel	20754	0.001	32	313.19
Actor	30251	0.001	32	388.99
CS	27592	0.001	32	2292.85
Physics	46700	0.001	32	1761.90
Photo	27713	0.01	32	456.02

Table A.5.: Hyperparameters for one shot GAT☼GAT+Del

Dataset	EdgesDeleted	LR	HiddenDimension	Runtime
Cora	1700	0.001	32	105.27
Citeseer	1500	0.001	32	116.28
Pubmed	14126	0.001	32	395.73
Cornell	120	0.001	32	100.89
Texas	120	0.001	32	131.33
Wisconsin	120	0.001	32	131.69
Chameleon	6000	0.001	32	169.09
Squirrel	35000	0.001	32	212.64
Actor	30000	0.001	32	139.54
CS	30000	0.001	32	1579.53
Physics	30000	0.001	32	3766.81
Photo	40264	0.001	32	450.34

Table A.6.: Hyperparameters for Iterative-SoLAR for GCN✡GCN edge deletions.

Model	Dataset	LR	HiddenDimension	LR	HiddenDimension	Runtime
GCN✡GCNDelIter	Cora	0.01	32	0.01	32	144.65
GCN✡GCNDelIter	Citeseer	0.01	32	0.01	32	147.60
GCN✡GCNDelIter	Pubmed	0.01	32	0.01	32	411.81
GCN✡GCNDelIter	CS	0.01	32	0.01	32	1347.70
GCN✡GCNDelIter	Photo	0.01	32	0.01	32	2063.57
GCN✡GCNDelIter	Physics	0.01	32	0.01	32	4969.71
GCN✡GCNDelIter	Cornell	0.001	128	0.001	128	141.09
GCN✡GCNDelIter	Texas	0.001	128	0.001	128	142.33
GCN✡GCNDelIter	Wisconsin	0.001	128	0.001	128	142.33
GCN✡GCNDelIter	Chameleon	0.001	128	0.001	128	157.65
GCN✡GCNDelIter	Squirrel	0.001	128	0.001	128	395.13
GCN✡GCNDelIter	Actor	0.001	128	0.001	128	229.87

Table A.7.: Hyperparameters for Iterative-SoLAR for GCN✡GCN edge additions.

Model	Dataset	LR	HiddenDimension	LR	HiddenDimension	Runtime
GCN✡GCNAddIter	Cora	0.01	32	0.01	32	409.66
GCN✡GCNAddIter	Citeseer	0.01	32	0.01	32	696.17
GCN✡GCNAddIter	Pubmed	0.01	32	0.01	32	4393.44
GCN✡GCNAddIter	CS	0.01	32	0.01	32	974.78
GCN✡GCNAddIter	Photo	0.01	32	0.01	32	2272.33
GCN✡GCNAddIter	Physics	0.01	32	0.01	32	10310.70
GCN✡GCNAddIter	Cornell	0.001	128	0.001	128	212.67
GCN✡GCNAddIter	Texas	0.001	128	0.001	128	204.39
GCN✡GCNAddIter	Wisconsin	0.001	128	0.001	128	204.20
GCN✡GCNAddIter	Chameleon	0.001	128	0.001	128	509.40
GCN✡GCNAddIter	Squirrel	0.001	128	0.001	128	938.76
GCN✡GCNAddIter	Actor	0.001	128	0.001	128	918.85

3. Hyperparameters for Spectral Rewiring

Below we present the results for node classification before and after spectral gap based rewiring on datasets: Cora [45], Citeseer [65] and Pubmed [48],Cornell,Wisconsin,Texas, Chameleon, Squirrel, Actor [58]. We compare GCN [38] without any modifications to the original graph, DIGL by [22], SDRF by [72], and FoSR by [36]. We adopt the public implementations available and tune the hyperparameters to improve the performance if possible. Our results are presented in Table A.10. We compare GCN with no edge modifications, GCN+DIGL, GCN+SDRF, GCN+FoSR, GCN+EldanDelete where we delete the edges, GCN+EldanAdd where we add the edges according to the criterion from Lemma III.3.1 and ProxyAdd and ProxyDelete which use

Table A.8.: Hyperparameters for Iterative-SoLAR for GCN☼GATv2 edge deletions.

Model	Dataset	LR	HiddenDimension	LR	HiddenDimension	Runtime
GCN☼GATv2DelIter	Cora	0.01	32	0.001	32	190.72
GCN☼GATv2DelIter	Citeseer	0.01	32	0.001	64	187.22
GCN☼GATv2DelIter	Pubmed	0.01	32	0.001	32	468.10
GCN☼GATv2DelIter	CS	0.01	128	0.001	32	173.93
GCN☼GATv2DelIter	Photo	0.01	512	0.001	32	175.24
GCN☼GATv2DelIter	Physics	0.01	32	0.001	32	175.44
GCN☼GATv2DelIter	Cornell	0.001	128	0.001	32	194.10
GCN☼GATv2DelIter	Texas	0.001	128	0.001	32	426.81
GCN☼GATv2DelIter	Wisconsin	0.001	128	0.001	32	263.64
GCN☼GATv2DelIter	Chameleon	0.001	128	0.001	32	2156.36
GCN☼GATv2DelIter	Squirrel	0.001	128	0.001	32	2977.70
GCN☼GATv2DelIter	Actor	0.001	128	0.001	32	6292.84

Table A.9.: Hyperparameters for Iterative-SoLAR for GCN☼GATv2 edge additions.

Model	Dataset	LR	HiddenDimension	LR	HiddenDimension	Runtime
GCN☼GATv2AddIter	Cora	0.01	32	0.001	32	311.17
GCN☼GATv2AddIter	Citeseer	0.01	32	0.001	32	305.38
GCN☼GATv2AddIter	Pubmed	0.01	32	0.001	32	2815.99
GCN☼GATv2AddIter	CS	0.01	128	0.001	32	198.06
GCN☼GATv2AddIter	Photo	0.01	512	0.001	32	241.03
GCN☼GATv2AddIter	Physics	0.01	32	0.001	32	238.01
GCN☼GATv2AddIter	Cornell	0.001	128	0.001	32	305.70
GCN☼GATv2AddIter	Texas	0.001	128	0.001	32	587.08
GCN☼GATv2AddIter	Wisconsin	0.001	128	0.001	32	986.07
GCN☼GATv2AddIter	Chameleon	0.001	128	0.001	32	4318.28
GCN☼GATv2AddIter	Squirrel	0.001	128	0.001	32	1479.35
GCN☼GATv2AddIter	Actor	0.001	128	0.001	32	6694.26

Equation (III.7) to optimize the spectral gap directly. The top performance is highlighted in **bold**.

Table A.10.: We compare the performance of GCN augmented with different graph rewiring methods on node classification.

Method	Cora $\mathcal{H} = 0.8041$	Citeseer $\mathcal{H} = 0.7347$	Pubmed $\mathcal{H} = 0.8023$	Cornell $\mathcal{H} = 0.1227$	Wisconsin $\mathcal{H} = 0.1777$	Texas $\mathcal{H} = 0.060$	Actor $\mathcal{H} = 0.2167$	Chameleon $\mathcal{H} = 0.2474$	Squirrel $\mathcal{H} = 0.2174$
GCN	87.22±0.40	77.35±0.70	86.96±0.17	50.74±7.24	53.52±7.80	50.40±10.01	29.12±0.24	31.15±0.84	26.00±0.69
GCN+DIGL	83.21±0.79	73.29±0.17	78.84±0.008	42.04±4.43	44.22±5.02	57.35±6.46	26.33±1.22	38.95±0.99	32.45±0.88
GCN+SDRF	87.84±0.68	78.43±0.62	87.36±0.14	53.54±2.65	58.78±3.22	60.25±4.97	31.67±0.36	41.30±1.36	38.98±0.46
GCN+FoSR	**91.44±3.16**	**82.13±4.29**	**91.49±1.89**	53.91±8.67	58.63±9.55	63.50±11.07	38.01±7.48	46.64±7.70	50.73±6.93
GCN+EldanDelete	87.60±0.18	78.68±0.54	87.33±0.07	65.13±13.02	**67.84±7.65**	70.53±6.70	**43.65±9.88**	52.51±8.12	48.89±7.89
GCN+EldanAdd	88.38±0.12	79.45 ±0.37	87.17±0.14	**69.05±6.17**	64.08±6.58	67.10±8.91	43.64±10.00	48.09±7.30	**51.66±6.50**
GCN+ProxyAdd	89.10±0.70	78.94±0.54	87.54±0.24	66.54±9.56	67.75±7.96	**74.21±10.64**	43.45±9.93	54.30±6.27	48.85±6.14
GCN+ProxyDelete	87.51±0.81	78.68 ±0.55	87.39±0.11	66.60 ± 6.50	66.36±7.17	72.36±7.88	43.52±9.64	**55.88±5.48**	48.90±7.68

Table A.11.: Hyperparameters for GCN+our proposed rewiring algorithms.

Dataset	LR	HiddenDimension	Dropout	EldanAdd	EldanDelete	ProxyAdd	ProxyDelete
Cora	0.01	32	0.3130296	50	20	100	100
Citeseer	0.01	32	0.4130296	50	20	50	50
Pubmed	0.01	128	0.3130296	50	100	20	50
Cornell	0.001	128	0.4130296	100	5	50	20
Wisconsin	0.001	128	0.5130296	100	5	50	10
Texas	0.001	128	0.4130296	100	5	50	76
Actor	0.001	128	0.2130296	100	10	25	500
Chameleon	0.001	128	0.2130296	100	50	50	200
Squirrel	0.001	128	0.5130296	50	100	10	1000

Table A.12.: Hyperparameters for SDRF.

Dataset	LR	Dropout	Hidden Dimension	SDRF Iterations	τ	C^+
Cora	0.01	0.3130296	32	100	163	0.95
Citeseer	0.01	0.2130296	32	84	180	0.22
Pubmed	0.01	0.4130296	128	166	115	1443
Cornell	0.001	0.2130296	128	126	145	0.88
Wisconsin	0.001	0.2130296	128	89	22	1.64
Texas	0.001	0.2130296	128	136	12	7.95
Actor	0.01	0.4130296	128	3249	106	7.91
Chameleon	0.01	0.2130296	128	2441	252	2.84
Squirrel	0.01	0.2130296	128	1396	436	5.88

Table A.13.: Hyperparameters for FoSR.

Dataset	LR	Dropout	Hidden Dimension	FoSR Iterations
Cora	0.01	0.5130296	128	50
Citeseer	0.01	0.3130296	128	10
Pubmed	0.01	0.4130296	128	50
Cornell	0.001	0.2130296	128	100
Wisconsin	0.001	0.2130296	128	100
Texas	0.001	0.4130296	128	100
Actor	0.01	0.4130296	128	100
Chameleon	0.01	0.4130296	128	100
Squirrel	0.01	0.2130296	128	100

Table A.14.: Hyperparameters for DIGL.

Dataset	LR	Dropout	Hidden Dimension	α	κ
Cora	0.01	0.41	32	0.0773	128
Citeseer	0.01	0.31	32	0.1076	-
Pubmed	0.01	0.41	128	0.1155	128
Cornell	0.001	0.41	128	0.1795	64
Wisconsin	0.001	0.31	128	0.1246	-
Texas	0.001	0.41	128	0.0206	32
Actor	0.01	0.21	128	0.0656	-
Chameleon	0.01	0.41	128	0.0244	64
Squirrel	0.01	0.41	128	0.0395	32