Revisiting Over-smoothing and Over-squashing Using Ollivier-Ricci Curvature

Khang Nguyen [1 2] **Hieu Nong** [1] **Vinh Nguyen** [1] **Nhat Ho** [3] **Stanley Osher** [4] **Tan Nguyen** [5]

Abstract

Graph Neural Networks (GNNs) had been demonstrated to be inherently susceptible to the problems of over-smoothing and over-squashing. These issues prohibit the ability of GNNs to model complex graph interactions by limiting their effectiveness in taking into account distant information. Our study reveals the key connection between the local graph geometry and the occurrence of both of these issues, thereby providing a unified framework for studying them at a local scale using the Ollivier-Ricci curvature. Specifically, we demonstrate that over-smoothing is linked to positive graph curvature while over-squashing is linked to negative graph curvature. Based on our theory, we propose the Batch Ollivier-Ricci Flow, a novel rewiring algorithm capable of simultaneously addressing both over-smoothing and over-squashing.

1. Introduction

A collection of entities with a set of relations between them is among the simplest, and yet most general, types of structure. It came as no surprise that numerous real world data are naturally represented by graphs (Harary, 1967; Hsu & Lin, 2008; Estrada & Bonchev, 2013), motivating many recent advancements in the study of Graph Neural Networks (GNNs). This has lead to a wide range of successful applications, including physical modeling (Battaglia et al., 2016; Kipf et al., 2018; Sanchez-Gonzalez et al., 2018), chemical and biological inference (Duvenaud et al., 2015; Gilmer et al., 2017), recommender systems (van den Berg et al., 2017; Ying et al., 2018; Fan et al., 2019; Wu et al., 2019), generative models (Li et al., 2018b; Bojchevski et al., 2018; Shi et al., 2020), financial prediction (Chen et al., 2018b; Matsunaga et al., 2019; Yang et al., 2019), and knowledge graphs (Shang et al., 2019; Zhang et al., 2019).

Despite their success, popular GNN designs suffer from two notable setbacks that hamper their performance in practical applications that require long-range interactions. The first common problem encountered by GNNs is known as over-smoothing (Li et al., 2018a). Over-smoothing occurs when node features quickly converge to each other and become indistinguishable as the number of layers increases. This issue puts a limit on the depth of a GNN, prohibiting the model's capability of capturing complex relationships in the data. Another plight plaguing GNNs is known as over-squashing (Alon & Yahav, 2021). This phenomenon happens when the number of nodes within the receptive field of a particular node grows exponentially with the number of layers, leading to the squashing of exponentially-growing amount of information into fixed-size node features. Such over-squashing hinders the ability of GNNs to effectively process distant information and capture long-range dependencies between nodes in the graph, especially in the case of deep GNNs that require many layers.

Together, over-smoothing and over-squashing impair the performance of modern GNNs, impeding their application to many important settings that involve very large graphs. (Cai & Wang, 2020; Alon & Yahav, 2021). Understanding and alleviating either of these problems has been the main focus in many recent studies of GNNs (Oono & Suzuki, 2020; Cai & Wang, 2020; Zhao & Akoglu, 2020; Karhadkar et al., 2023). Notably, using a notion of graph curvature, Topping et al. (2022) suggested that over-squashing behaviors could arise from local graph structures. This pioneering idea implies that local graph properties can be used to study and improve GNN performance. It has been recently noted that over-smoothing and over-squashing are somewhat related problems (Karhadkar et al., 2023). Nevertheless, to the best of our knowledge, there has been no work in the literature that offers a common framework to understand these problems. Such a unified approach presents a potentially crucial theoretical contribution to our understanding of the over-smoothing and over-squashing issues. It enables the development of novel architectures and methods that can

[1] FPT Software AI Center, Vietnam [2] Faculty of Mathematics and Computer Science, University of Science, Vietnam National University Ho Chi Minh City, Vietnam [3] Department of Statistics and Data Sciences, University of Texas at Austin, USA [4] Department of Mathematics, University of California, Los Angeles, USA [5] Department of Mathematics, National University of Singapore, Singapore. Correspondence to: Khang Nguyen <khang.nguyenhoang.vn@gmail.com>, Tan Nguyen <tanmnguyen89@gmail.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

effectively learn complex and long-range graph interactions, thereby broadening the applications of GNNs on practical tasks.

Contribution. We present a unified theoretical framework to study both the over-smoothing and over-squashing phenomena in GNNs at the local level using the Ollivier-Ricci curvature (Ollivier, 2009), an inherent local geometric property of graphs. Our key contributions are three-fold:

1. We prove that very positively curved edges cause node representations to become similar, thereby establishing a link between the over-smoothing issue and high edge curvature.
2. We prove that low curvature value characterizes graph bottlenecks and demonstrate a connection between the over-squashing issue and negative edge curvature.
3. Based on our theoretical results, we propose the Batch Ollivier-Ricci Flow (BORF), a novel curvature-based rewiring method designed to mitigate the over-smoothing and over-squashing issues simultaneously.

Organization. We structure this paper as follows. First, we discuss related works in Section 2. In Section 3, we give a brief summary of the relevant backgrounds in the study of GNNs and provide a concise formulation for the Ollivier-Ricci curvature on graphs. In Section 4, we present our central analysis showing positive graph curvature is associated with over-smoothing, while negative graph curvature is associated with over-squashing. In Section 5, we introduce the novel graph rewiring method BORF, which modifies the local graph geometry to suppress over-smoothing and alleviate over-squashing inducing connections. We empirically demonstrate the superior performance of our method compared to other state-of-the-art rewiring methods in Section 6. The paper ends with concluding remarks in Section 7. Technical proofs and other additional materials are provided in the Appendix.

Notation. We denote scalars by lower- or upper-case letters and vectors and matrices by lower- and upper-case boldface letters, respectively. We use $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to denote a simple, connected graph $\mathcal{G}$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. Graph vertices are also referred to as nodes, and the characters u, v, w, p, q are reserved for representing them. We write $u \sim v$ if $(u, v) \in \mathcal{E}$. The shortest path distance between two vertices u, v is denoted by $d(u, v)$. We let $\mathcal{N}_u = \{p \in \mathcal{V} \mid p \sim u\}$ be the 1-hop neighborhood and $\tilde{\mathcal{N}_u} = \mathcal{N}_u \cup \{u\}$ be the extended neighborhood of u. The characters n, m are used to denote the degrees of the vertices u, v.

2. Related Work

Over-smoothing: First recognised by (Li et al., 2018a), who observed that GCN with non-linearity removed induces a smoothing effect on data features, over-smoothing has been one of the focal considerations in the study of GNNs. A dynamical system approach was used by (Oono & Suzuki, 2020) to show that under considerable assumptions, even GCN with ReLU can not escape this plight. Follow-up work by (Cai & Wang, 2020) generalized and improved this approach. Designing ways to alleviate or purposefully avoid the problem is a lively research area (Luan et al., 2019; Zhao & Akoglu, 2020; Rusch et al., 2022). Notably, randomly removing edges from the base graph consistently improves GNN performance (Rong et al., 2020).

Over-squashing: The inability of GNNs to effectively take into account distant information has long been observed (Xu et al., 2018). Alon & Yahav (2021) showed that this phenomenon cound be explained by the existence of local bottlenecks in the graph structure. It was shown by (Topping et al., 2022) that graph curvature provided an insightful way to study and address the over-squashing problem. Methods have been designed to tackle over-squashing, including those by (Banerjee et al., 2022) and (Karhadkar et al., 2023).

Graph curvature: Efforts have been made to extend the geometric notion of curvature to settings other than smooth manifolds, including on graphs (Bakry & Émery, 1985; Forman, 2003). Among these, the Ollivier's Ricci curvature (Ollivier, 2009) is arguably the superior attempt due to its proven compatibility with the classical notion of curvature in differential geometry (Ollivier, 2009; van der Hoorn et al., 2021). Graph curvature has been utilised in the study of complex networks (Ni et al., 2015; Sia et al., 2019), and a number of works have experimented with its use in GNNs (Topping et al., 2022; Bober et al., 2022).

3. Preliminaries

We begin by summarizing the relevant backgrounds on message passing neural networks and the over-smoothing and over-squashing issues of GNNs. We also provide a concise formulation for the Ollivier-Ricci curvature on graphs.

3.1. Message Passing Neural Networks

Message passing neural networks (MPNNs) (Gilmer et al., 2017) are a unified framework for a broad range of graph neural networks. It encompasses virtually every popular GNN design to date, including graph convolutional network (GCN) (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), graph attention network (GAT) (Veličković et al., 2018), graph isomorphism network (GIN) (Xu et al., 2019), etc. The key idea behind MPNNs is that by aggregating information from local neighborhoods, a neural network

Table 1. Popular GNNs are instances of Equation (1): GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), and GIN (Xu et al., 2019).

GNN	ψ_k	ϕ_k	$\bigoplus$
GCN [1]	linear	activation	mean
GraphSAGE [2]	linear	activation	mean
GAT	linear	activation	weighted mean
GIN [3]	identity	MLP	sum

can effectively use both node feature data and the graph topology to learn relevant information.

Let $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$ be the node feature matrix of a graph $\mathcal{G}$, where d is the number of feature channels. Let $\boldsymbol{X}^k$ be the node feature matrix at layer k, with the convention that $\boldsymbol{X}^0 = \boldsymbol{X}$. The features of node u at layer k is denoted by $\boldsymbol{X}_u^k$, and is exactly the transpose of the u-th row of $\boldsymbol{X}^k$. A general formulation for an MPNN can be given by

$$\boldsymbol{X}_u^{k+1} = \phi_k\left(\bigoplus_{p\in\tilde{\mathcal{N}}_u} \psi_k(\boldsymbol{X}_p^k)\right), \tag{1}$$

where ψ_k is a message function, $\bigoplus$ is an aggregating function, and ϕ_k is an update function. Table 1 summarizes the choice for ψ_k, ϕ_k, and $\bigoplus$ in four popular GNN architectures. We give further discussion on how Equation (1) accommodates different designs of GNNs in Appendix A. From now on, we will use MPNN and GNN interchangeably.

Traditionally, GNNs are designed to operate directly on the input graphs. In many cases, this leads to significant downsides due to possible undesirable characteristics of the dataset. Hence, it has been proposed that by conducting the learning process on a modified version of the input graphs, we can improve upon the scale and performance of graph models (Hamilton et al., 2017; Gasteiger et al., 2019). One such approach is known as graph rewiring, which involves modifying the set of edges $\mathcal{E}$ within a graph as a preprocessing step. We give a brief overview of two novel rewiring algorithms, SDRF (Topping et al., 2022) and FoSR (Karhadkar et al., 2023), along with a comparison between them and our proposed method in Section 5.

3.2. The Over-smoothing and Over-squashing Issues of GNNs

Over-smoothing has generally been described as the phenomenon where the feature representation of every node becomes similar to each other as the number of layers in a GNN increases (Li et al., 2018a). If over-smoothing occurs, for every two neighbor nodes u, v, it must happen that

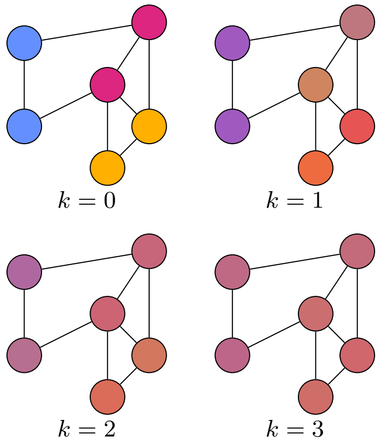


Figure 1. Over-smoothing induced by the averaging operation.

$$\left|\boldsymbol{X}_u^k - \boldsymbol{X}_v^k\right| \to 0 \text{ as } k \to \infty. \tag{2}$$

Equation (2) can be thought of as the local smoothing behavior, observed in the neighborhood of two nodes $u \sim v$.

A global formulation for feature representation similarity is obtained by summing up terms of the form $\left|\boldsymbol{X}_u^k - \boldsymbol{X}_v^k\right|$ for all neighbors u, v. Formally, we obtain a formulation for the global over-smoothing issue based on local observations

$$\sum_{(u,v)\in\mathcal{E}} \left|\boldsymbol{X}_u^k - \boldsymbol{X}_v^k\right| \to 0 \text{ as } k \to \infty. \tag{3}$$

That is, if the term $\sum_{(u,v)\in\mathcal{E}} \left|\boldsymbol{X}_u^k - \boldsymbol{X}_v^k\right|$ converges to zero, we say that the model experiences over-smoothing. This formulation is similar to the definition based on the node-wise Dirichlet energy utilised in (Rusch et al., 2022). The Dirichlet energy was first proposed as a viable way to measure the over-smoothing issue by (Cai & Wang, 2020). Figure 1 visualizes the over-smoothing behavior of a simple graph containing 6 nodes from 3 classes with different RGB color features. At the start, the nodes can easily be divided into 3 classes. When the mean operator is applied repeatedly for k times across k GNN layers, those nodes rapidly converge to having similar colors. At the final step $k = 3$, the nodes in the GNN have become virtually indistinguishable, suggesting that they have experienced over-smoothing.

On the other hand, over-squashing is an inherent pitfall of GNNs that occurs when bottlenecks in the graph structure impede the graph's ability to propagate information among its vertices. We observe from Equation (1) that messages can only be transmitted by a distance of 1 at each layer. Hence, two nodes of distance K receive information from each other if and only if the GNN has at least K layers. For any given node u, the set of nodes whose messages can

[1] If the symmetrically normalized Laplacian is replaced by the normalized Laplacian. See Appendix A.

[2] Mean aggregator variant.

[3] GIN-0 variant.

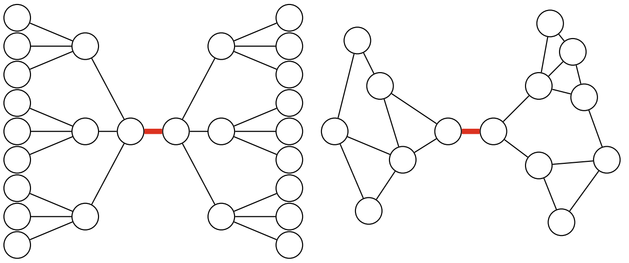


Figure 2. Bottlenecks inhibit the message passing capability of MPNNs. These bottlenecks are highlighted by bold red lines.

reach u is called the receptive field of u. As the number of layers increases, the size of each node's receptive field increases exponentially (Chen et al., 2018a). This causes messages between exponentially-growing number of distant vertices to be squashed into fixed size vectors, limiting the model's ability to capture long range dependencies (Alon & Yahav, 2021). As illustrated in Figure 2, graph bottlenecks contribute to this problem by enforcing the maximal rate of expansion to the receptive field, while providing minimal connection between either sides of the bottleneck. Thus, a graph containing many bottlenecks causes GNNs to suffer from over-squashing.

3.3. Ollivier-Ricci Curvature on Graph

The Ricci curvature is a geometric object ubiquitous in the field of differential geometry. At a local neighborhood of a Riemannian manifold, the Ricci curvature of the space characterizes the average geodesic dispersion, i.e., whether straight paths in a given direction of nearby points have the tendency to remain parallel (zero curvature), converge (positive curvature), or diverge (negative curvature). Crucially, the definition of the Ricci curvature depends on the ability to specify directions, or more precisely, tangent vectors, within the space considered.

To circumvent the lack of a tangent structure on graphs, the Ollivier-Ricci curvature (Ollivier, 2009) considers random walks from nearby points. We define a random walk μ on a graph $\mathcal{G}$ as a family of probability measure $\mu_u(\cdot)$ on the vertex set $\mathcal{V}$ for all $u \in \mathcal{V}$. For a vertex $p \in \mathcal{V}$, it is intuitive to think of $\mu_u(p)$ as the probability that a random walker starting from u will end up at p after some number of steps. Then, for any $u, v \in \mathcal{V}$, we can consider the L^1 Wasserstein transport distance $W_1(\mu_u, \mu_v)$ given by

$$W_1(\mu_u, \mu_v) = \inf_{\pi \in \Pi(\mu_u, \mu_v)} \left(\sum_{(p,q) \in \mathcal{V}^2} \pi(p,q) d(p,q) \right),$$

where $\Pi(\mu_u, \mu_v)$ is the family of joint probability distributions of μ_u and μ_v. This measures[4] the minimal distance that random walkers from u must travel to meet the random walkers from v. The Ollivier-Ricci curvature $\kappa(u, v)$ is then defined based on the ratio between the random walker distance $W_1(\mu_u, \mu_v)$ and the original distance $d(u, v)$

$$\kappa(u, v) = 1 - \frac{W_1(\mu_u, \mu_v)}{d(u, v)}. \tag{4}$$

Such a definition captures the behavior that $\kappa(u, v) = 0$ if the random walkers tend to stay at equal distance, $\kappa(u, v) < 0$ if they tend to diverge, and $\kappa(u, v) > 0$ if they tend to converge.

Since curvature is intrinsically a local concept, it makes sense to only examine small neighborhoods when defining any curvature notion. On Riemannian manifolds, various definitions of curvature are constructed using differentials and derivatives on arbitrarily small neighborhoods. On a graph, the smallest neighborhood has radius 1, and so it is natural to consider the uniform 1-step random walk μ given by

$$\mu_u(p) = \begin{cases} \frac{1}{\deg u} & \text{if } p \sim u, \\ 0 & \text{otherwise.} \end{cases}$$

Hence, the Ollivier-Ricci curvature on graphs κ is defined by Equation (4), where $W_1(\mu_u, \mu_v)$ is the optimal value of the objective function in the linear optimization problem

$$\begin{aligned} \text{minimize} & \sum_{p \in \mathcal{N}_u} \sum_{q \in \mathcal{N}_v} d(p,q)\pi(p,q), \\ \text{subject to} & \sum_{p \in \mathcal{N}_u} \pi(p,q) = \tfrac{1}{\deg v}, \\ & \sum_{q \in \mathcal{N}_v} \pi(p,q) = \tfrac{1}{\deg u}. \end{aligned} \tag{5}$$

Our analysis in Section 4 is based on this specific formulation of the Ollivier-Ricci curvature.

4. Analysis Based on Graph Curvature

Throughout this section, we assume $u \sim v \in \mathcal{V}$ are neighboring vertices with $\deg u = n, \deg v = m$, and $n \geq m$. We note that $u \sim v$ implies $0 \leq d(p,q) \leq 3$ for all neighbors p, q of u, v, and so $0 \leq W_1(\mu_u, \mu_v) \leq 3$. From Equation (4), the following bound holds

$$-2 \leq \kappa(u, v) \leq 1.$$

Hence, a curvature value close to 1 is considered very positive, while a value close to -2 is considered very negative.

To motivate our findings, we remark that the curvature $\kappa(u, v)$ characterizes how well-connected the neighborhoods $\tilde{\mathcal{N}}(u)$ and $\tilde{\mathcal{N}}(v)$ are. Figure 3 illustrates how different local graph structures give rise to different graph

[4] More precisely, $W_1(\mu_u, \mu_v)$ measures the minimal distance one must move the random walk from u in order to obtain the random walk from v.

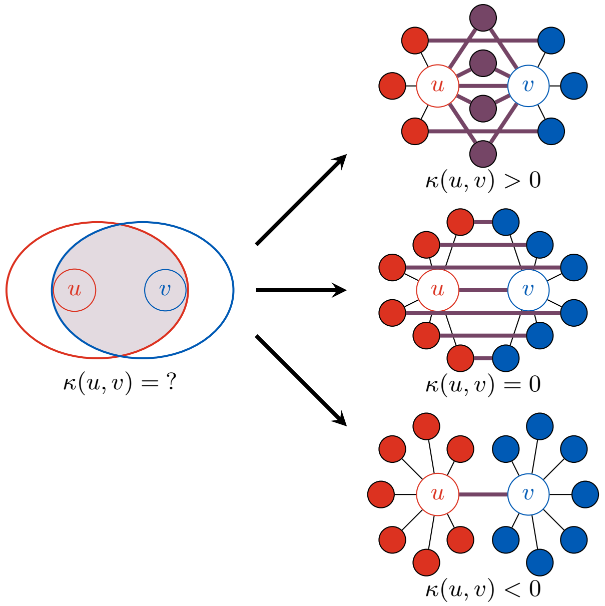


Figure 3. Different edge curvatures give rise to different local graph structures.

curvature. Red and blue are used to color the neighborhoods $\tilde{\mathcal{N}}_u \backslash \{v\}$ and $\tilde{\mathcal{N}}_v \backslash \{u\}$. The color violet is used to signal shared vertices or edges connecting from one neighborhood to the other. If the neighborhoods mostly coincide then the transport cost is very low, leading to a positive curvature value. In this case, messages can be transmitted freely and easily between both neighborhoods. In contrast, if the neighborhoods only have minimal connections then the transport cost is high, leading to a negative curvature value. Each such connection will then act as a bottleneck, limiting the effectiveness of the message-passing mechanism.

4.1. Positive Graph Curvature and Over-smoothing

We identify the key connection between positive graph curvature and the occurrence of the over-smoothing issue.

Lemma 4.1. *The following inequality holds*

$$\frac{|\mathcal{N}_u \cap \mathcal{N}_v|}{\max(m,n)} \geq \kappa(u,v).$$

Lemma 4.1 says that the curvature $\kappa(u,v)$ is a lower bound for the proportion of shared neighbors between u and v. A closer inspection of Equation (1) reveals a fundamental characteristic of GNNs: at the k-th layer, every node p broadcasts an identical message $\psi_k(\boldsymbol{X}_p^k)$ to each vertex u in its 1-hop neighborhood. These messages are then aggregated and used to update the features of u. If $\kappa(u,v)$ is very positive then the neighborhoods $\mathcal{N}_u$ and $\mathcal{N}_v$ mostly coincide. Hence, they incorporate roughly the same information, and their variance diminishes. This gives us significant insight into why over-smoothing happens, and is made precise by the following theorem.

Theorem 4.2. *Consider the update rule given by Equation (1). Suppose the edge curvature $\kappa(u,v) > 0$. For some k, assume the update function ϕ_k is L-Lipschitz, $\left|\boldsymbol{X}_p^k\right| \leq C$ for all $p \in \mathcal{N}(u) \cup \mathcal{N}(v)$, and the message function ψ_k is bounded, i.e. $|\psi_k(\boldsymbol{x})| \leq M|\boldsymbol{x}|, \forall \boldsymbol{x}$. There exists a positive function $h : (0,1) \to \mathbb{R}^+$ dependent on the constants L, M, C, n satisfying*

- *if $\bigoplus$ is the sum operation then h is constant;*
- *if $\bigoplus$ is the mean operation then h is decreasing;*

such that

$$\left|\boldsymbol{X}_u^{k+1} - \boldsymbol{X}_v^{k+1}\right| \leq (1 - \kappa(u,v))h(\kappa(u,v)). \tag{6}$$

In both cases, we clearly have

$$\lim_{x \to 1} (1-x)h(x) = 0. \tag{7}$$

This result applies to a wide range of GNNs, including those in Table 1 with the exception of GAT, due to the fact that GAT employs the attention mechanism to create a learnable weighted mean function as the aggregator. Nevertheless, if the variance between attention weights are low, we expect the general behavior to still hold true.

Theorem 4.2 conclusively shows that very positively curved edges force local node features to become similar. If the graph is positively curved everywhere or if it contains multiple subgraphs having this characteristic, we can expect that the node features will quickly converge to indistinguishable representations. In other words, positive edge curvature induces the mixing behavior observed by (Li et al., 2018a), causing over-smoothing to occur at a faster rate. This suggests the occurrence of over-smoothing in shallow GNNs can be explained by an abundance of positively curved edges.

Any global analysis of the issue based on local observations is hindered by the complexity in dealing with graph structures. Nevertheless, by restricting our attention to a more manageable class of graphs - the class of regular graphs, we obtain Proposition 4.3. This serves to illustrate how positive local graph curvature can affect the long term global behavior of a typical GNN.

Proposition 4.3. *Consider the update rule given by Equation (1). Assume the graph is regular. Suppose there exists a constant $\delta > 0$ such that for all edges $(u,v) \in \mathcal{E}$, the curvature is bounded by $\kappa(u,v) \geq \delta > 0$. For all $k \geq 1$, assume the functions ϕ_k are L-Lipschitz, $\bigoplus$ is realised as the mean operation, $\left|\boldsymbol{X}_p^0\right| \leq C$ for all $p \in \mathcal{V}$, and the message functions ψ_k are bounded linear operators, i.e.*

$|\psi_k(\boldsymbol{x})| \leq M|\boldsymbol{x}|, \forall \boldsymbol{x}$. The following inequality holds for $k \geq 1$ and any neighboring vertices $u \sim v$

$$\left|\boldsymbol{X}_u^k - \boldsymbol{X}_v^k\right| \leq \frac{2}{3} C \left(\frac{3LM\lfloor(1-\delta)n\rfloor}{n+1}\right)^k. \tag{8}$$

Furthermore, for any $u, v \in \mathcal{V}$ that are not necessarily neighbors, the following inequality holds

$$\left|\boldsymbol{X}_u^k - \boldsymbol{X}_v^k\right| \leq \frac{2}{3}\left\lfloor\frac{2}{\delta}\right\rfloor C \left(\frac{3LM\lfloor(1-\delta)n\rfloor}{n+1}\right)^k. \tag{9}$$

The conclusion of Proposition 4.3 says that if every edge curvature in a regular graph G is bounded from below by a sufficiently high constant δ then the difference between the features of any pair of neighboring nodes, or indeed, any pair of nodes at all, exponentially converges to 0 in a typical GNN. This leads to the over-smoothing issue formulated in Equation (3) since for appropriate constants $C_1, C_2 > 0$, we have

$$\sum_{(u,v)\in\mathcal{E}} \left|\boldsymbol{X}_u^k - \boldsymbol{X}_v^k\right| \leq C_1 e^{-C_2 k}.$$

In real-world graphs, it is often the case that not all edges in a graph is positively curved. Nevertheless, we expect an abundance of edges with overly high curvature will either cause or worsen the over-smoothing issue in GNNs.

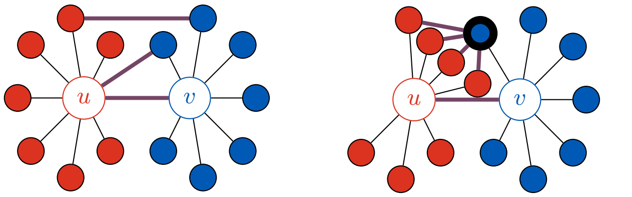
4.2. Negative Graph Curvature and Over-squashing

In this section, we demonstrate the intimate connection between negative graph curvature and the occurrence of local bottlenecks, which in turn causes over-squashing.

Message-passing across local neighborhoods is facilitated by connections of the form (p, q) with $p \in \tilde{\mathcal{N}}_u \backslash \{v\}$ and $q \in \tilde{\mathcal{N}}_v \backslash \{u\}$. As visualized by Figure 4a, such edges (colored in violet) provide information pathways between $\tilde{\mathcal{N}}_u$ and $\tilde{\mathcal{N}}_v$. However, a large number of these edges concentrated on a relatively few vertices will create new bottlenecks, instead of providing good message channels. Figure 4b illustrates this point, as there are way too many edges connecting to the emphasized node but too little edges connecting between other neighbors. Since $n \geq m$, there is a natural squashing of information as messages are transmitted from $\mathcal{N}_u$ to $\mathcal{N}_v$ of ratio $\frac{n}{m}$. We identify the edges that provide good pathways as those that do not exacerbate this ratio and restrict our attention to these edges.

We characterize the effect of edge curvature on graph bottlenecks in the following proposition.

Proposition 4.4. *Let $\tilde{\mathcal{E}}$ be union of the edge set $\mathcal{E}$ with the set of all possible self-loops. Let S be the subset of $\tilde{\mathcal{E}}$ containing edges of the form (p, q) with $p \in \tilde{\mathcal{N}}_u \backslash \{v\}$ and $q \in \tilde{\mathcal{N}}_v \backslash \{u\}$. Supposing each vertex w is a vertex of at most $\frac{n}{m}$ edges in S. The following inequality holds*



(a) Violet edges connecting $\tilde{\mathcal{N}}_u \backslash \{v\}$ and $\tilde{\mathcal{N}}_v \backslash \{u\}$ serve as information pathways.

(b) Edges may exacerbate the situation by creating new bottlenecks.

Figure 4. Identifying connections that enable effective message-passing at local neighborhoods.

$$|S| \leq \frac{n(\kappa(u,v)+2)}{2}. \tag{10}$$

Recall that the curvature is deemed very negative if it is close to -2. Proposition 4.4 shows that very negative edge curvature values prohibit the number of information pathways from $\tilde{\mathcal{N}}_u$ to $\tilde{\mathcal{N}}_v$, and very negatively curved edges induce local bottlenecks. This, in turn, contributes to the occurrence of the over-squashing issue as proposed by (Alon & Yahav, 2021). We note that an adequate measure for the over-squashing issue is currently lacking in the literature (see Appendix B). Inspired by the influence distribution introduced by Xu et al. (2018), the next theorem asserts that negative edge curvature directly causes the decaying importance of distant nodes in GNNs with non-linearity removed. This demonstrates the effect of edge curvature on the over-squashing issue.

Theorem 4.5. *Consider the update rule given by Equation (1). Suppose ψ_k, ϕ_k are linear operators for all k, and $\bigoplus$ is the sum operation. If u, v are neighboring vertices with neighborhoods as in Proposition 4.4 and S is defined similarly then for all $p \in \tilde{\mathcal{N}}_u \backslash \{v\}$, $q \in \tilde{\mathcal{N}}_v \backslash \{u\}$, we have*

$$\begin{aligned} \left[\frac{\partial \boldsymbol{X}_u^{k+2}}{\partial \boldsymbol{X}_q^k}\right] &= \alpha \sum_{w \in V} \left[\frac{\partial \boldsymbol{X}_u^{k+2}}{\partial \boldsymbol{X}_w^k}\right], \\ \left[\frac{\partial \boldsymbol{X}_v^{k+2}}{\partial \boldsymbol{X}_p^k}\right] &= \beta \sum_{w \in V} \left[\frac{\partial \boldsymbol{X}_v^{k+2}}{\partial \boldsymbol{X}_w^k}\right], \end{aligned} \tag{11}$$

where $\left[\frac{\boldsymbol{y}}{\boldsymbol{x}}\right]$ is used to denote the Jacobian of $\boldsymbol{y}$ with regard to $\boldsymbol{x}$, and α, β satisfy

$$\begin{aligned} \alpha &\leq \frac{|S|+2}{\sum_{w\in\tilde{\mathcal{N}}_v}(\deg(w)+1)}, \\ \beta &\leq \frac{|S|+2}{\sum_{w\in\tilde{\mathcal{N}}_u}(\deg(w)+1)}. \end{aligned} \tag{12}$$

To understand the meaning of Theorem 4.5, let us fix $k = 0$ and assume $\mathcal{G}$ is a regular graph with node degree n.

Equations (11) and (12), along with Proposition 4.4, say that the contribution by the vertex q to the vertex u relative to the contribution of all other vertices, measured by the scaling term α in regard to the Jacobians, is bounded by

$$\alpha \leq \frac{n(\kappa(u,v)+2)+4}{2(n+1)^2}.$$

Hence, if n is large and connections between the two neighborhoods are sparse, then $\kappa(u,v)$ is very negative. In such a case, we expect that the messages from each node of $\mathcal{N}_v$ make up only $\frac{2}{(n+1)^2}$ of the total sum of information. They thus hardly have any effect on u, even when the distance between them is only 2. An analogous result holds for the case when $\bigoplus$ is the mean operation without much modification.

We have thus shown that negative curvature characterizes graph bottlenecks. As such, GNNs that operate on graphs with a large volume of negatively curved edges are expected to suffer from over-squashing.

5. BORF: Batch Ollivier-Ricci Flow

Our theoretical results suggest the strikingly simple geometric connection between the over-smoothing and over-squashing issues: over-smoothing happens when there is a large proportion of edges with very positive curvature, while over-squashing occurs when there is a large proportion of edges with very negative curvature. As a natural extension, we propose the Batch Ollivier-Ricci Flow (BORF), a graph rewiring algorithm capable of simultaneously mitigating these issues by suppressing the over-smoothing and alleviateing the over-squashing inducing graph edges (see Algorithm 1).

For each of N batches, BORF first finds the h edges $(u_1,v_1),\ldots,(u_h,v_h)$ with minimal curvature and k edges $(u^1,v^1),\ldots,(u^k,v^k)$ with maximal curvature within the graph. Note that we index the minimally curved and maximally curved edges by subscripts and superscripts, respectively. Then, it tries to uniformly alleviate graph bottlenecks by adding connections to the set of h minimally curved edges. To save on computation time, BORF does not recalculate the graph curvature within each batch. Instead, for each edge with minimal curvature (u_j,v_j), it reuses the already calculated optimal transport plan π_j between μ_{u_j} and μ_{v_j} to decide which edge should be added. Recall that the formula for the optimal transport cost is

$$W_1(\mu_{u_j},\mu_{v_j}) = \sum_{(p,q)} \pi_j(p,q)d(p,q).$$

Hence, to minimize the transport cost, it makes sense to rewire the two nodes that contribute the most to this sum. Specifically, we choose to add to $\mathcal{G}$ the edge (p^*,q^*) such that

$$(p^*,q^*) = \operatorname{argmax} \pi_j(p,q)d(p,q).$$

Algorithm 1 Batch Ollivier-Ricci Flow (BORF)

```
Input: graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$, # rewiring batches $N$, # edges added per batch $h$, # edges removed per batch $k$
for $i=1$ to $N$ do
    Find $h$ edges $(u_1,v_1),\ldots,(u_h,v_h)$ with minimal Ollivier-Ricci curvature $\kappa$, along with each summand $\pi_j(p,q)d(p,q)$ in their optimal transportation cost sum for all $p,q\in\mathcal{V}$ and $j=\overline{1,h}$
    Find $k$ edges $(u^1,v^1),\ldots,(u^k,v^k)$ with maximal Ollivier-Ricci curvature $\kappa$
    for $j=1$ to $h$ do
        Add to $\mathcal{G}$ the edge $(p^*,q^*)$ given by
            $(p^*,q^*) = \operatorname{argmax} \pi_j(p,q)d(p,q)$
    end for
    Remove edges $(u^1,v^1),\ldots,(u^k,v^k)$ from $\mathcal{G}$
end for
```

If there are multiple candidates, we arbitrarily choose one. Finally, BORF removes the k maximally curved edges $(u^1,v^1),\ldots,(u^k,v^k)$ whose presence might prime the over-smoothing behavior to occur.

With such design, BORF can effectively limit both ends of the curvature spectrum, simultaneously suppressing over-smoothing and alleviating over-squashing inducing connections. Furthermore, depending on data characteristics, we may change the behaviours of BORF to either be a net edge add, net edge minus, or net zero rewiring algorithm. Thereby, BORF permits fine-tuned and fluid adjustments of the total number of edges and their curvature range.

Computational complexity. Let R be the number of edges to be rewired, $E=|\mathcal{E}|$ be the number of edges in the graph, D be the maximal degree of vertices on the graph, and $h+k$ be the number of edges rewired per batch by BORF. Solving the transportation problem to calculate the curvature for each edge using the network simplex algorithm is known to run in cubic time $O(D^3)$ (Ahuja et al., 1993). Once we have calculated the curvature of an edge, the optimal transport plan for that edge is already available for free. Hence, for each batch, the computational cost is $O(ED^3)$. The number of batches is $R/(h+k)$. Thus, the total cost for BORF is $O\left(RED^3/(h+k)\right)$ for each graph in consideration.

Many efficient methods for approximating the Wasserstein distance, which is the computational bottleneck in BORF, have been proposed. In particular, log-linear and linear time approximations for the Wasserstein distance have recently been developed in (Bonneel et al., 2014; Atasu & Mittelholzer, 2019), and some of them are already known to be topologically equivalent to the Wasserstein distance itself under appropriate assumptions (Bayraktar & Guo, 2021). By incorporating these approximations to calculate the Ollivier-Ricci curvature, we may significantly reduce the computational cost of BORF. We leave studying such modifications as directions for future work.

Other rewiring algorithms. SDRF (Topping et al., 2022) and FoSR (Karhadkar et al., 2023) are state-of-the-art rewiring algorithms for GNNs, designed with the purpose of alleviating the over-squashing issue. SDRF is based on the edge metric Balanced Forman curvature (BFC), which is actually a lower bound for the Ollivier-Ricci curvature. At its core, SDRF iteratively finds the edge with the lowest BFC, calculate the change in BFC for every possible edge that can be added, then add the one edge that affects the greatest change to the BFC of the aforementioned edge. On the other hand, FoSR is based on the heuristics that the spectral gap characterizes the connectivity of a graph. At each step, it approximates which missing edge would maximally improve the spectral gap and add that edge to the graph.

Comparison to SDRF. BORF shares some similarities to SDRF, but with notable differences. Since SDRF is based on BFC, it can only accurately enforce a lower bound on the Ollivier-Ricci curvature. Furthermore, its design lacks the capability to be used as a net edge minus rewiring algorithm. As such, it is ill-equipped to deal with the over-smoothing issue or to be used on denser graphs. Another difference is BORF calculates the graph curvature very infrequently, while SDRF has to constantly recalculate for each possible new edge. Finally, by rewiring edges in batch, BORF affects a uniform change across the graph. This helps to preserve the graph topology and prevent the possibility that a small outlier subgraph gets continually rewired, while other parts of the graph do not see any geometrical improvement.

Comparison to FoSR. Unlike BORF and SDRF, FoSR does not have the ability to remove edges. Hence, despite over-smoothing and over-squashing being problems on the two ends of the same spectrum, the algorithm is incapable of addressing the first issue. It is also very challenging to predict where new edges will be added and what changes FoSR would make to the graph topology. This might complicate attempts by users to analyse the performance changes made by the algorithm.

6. Experiments

In this section, we empirically verify the effectiveness of BORF on a variety of tasks compared to other rewiring alternatives. We seek to demonstrate the potential of curvature-based rewiring methods, and more generally, geometric-aware techniques in improving the performance of GNNs. Our codes for the experiments are available at `https://github.com/hieubkvn123/revisiting-gnn-curvature`.

Datasets. We conduct our experiments on a range of widely used node classification and graph classification tasks. For node classification, we report our results on the datasets CORA, CITESEER (Yang et al., 2016), TEXAS, CORNELL, WISCONSIN (Pei et al., 2020) and CHAMELEON (Rozemberczki et al., 2021). For graph classification, we validate our method on the following benchmarks: ENZYMES, IMDB, MUTAG and PROTEINS from the TUDataset (Morris et al., 2020). A summary of dataset statistics is available in Appendix F.

Experiment details. We compare BORF to no graph rewiring and two other state-of-the-art rewiring methods: SDRF (Topping et al., 2022) and FoSR (Karhadkar et al., 2023). In designing our experiments, we prioritize fairness and comprehensiveness, rather than aiming to obtain the best possible performance for each dataset presented. We applied each method as a preprocessing step to all graphs in the datasets considered, before feeding the rewired graph data into a GNN to evaluate performance. For baseline GNNs, we employed the popular graph architectures GCN (Kipf & Welling, 2017) and GIN (Xu et al., 2019). For each task and baseline model, we used the same settings of GNN and optimization hyper-parameters across all rewiring methods to rule out hyper-parameter tuning as a source of performance gain. The setting for each rewiring option was obtained by tuning every hyper-parameter available for each method with the exception of the temperature τ of SDRF, which we set to ∞. Each configuration is evaluated using the validation set. The test set accuracy of the configuration with the best validation performance is then recorded. For each experiment, we accumulate the result across 100 random trials and report the mean test accuracy, along with the 95% confidence interval. Further experiment details are available in Appendix D.

Results. Table 2 and Table 3 summarize our experiment results for node classification and graph classification datasets, respectively. BORF outperforms all other methods in every node classification tasks on both GCN and GIN. It is worth mentioning that the 95% confidence interval of BORF is almost always smaller than other methods, indicating a consistent level of performance. This result agrees with what is expected since SDRF and FoSR are not suited for dealing with the over-smoothing issue, which heavily degrades the model's performance on node classification tasks. On graph classification datasets, BORF achieves higher test accuracy compared to other rewiring options in most settings.

Ablation study on edge addition/removal. To investigate the role of edge addition and removal, we compare BORF's performance on GCN at high depths when using the best rewiring settings found in previous experiments against versions of BORF where edges are only removed (only alleviates over-smoothing), edges are only added (only alleviates over-squashing), or edges are removed and added equally.

Table 2. Classification accuracies of GCN and GIN with None, SDRF, FoSR, and BORF rewiring on various node classification datasets. Best results are highlighted in bold.

	GCN				GIN			
Data set	None	SDRF	FoSR	BORF	None	SDRF	FoSR	BORF
Cora	86.7 ± 0.3	86.3 ± 0.3	85.9 ± 0.3	$\mathbf{87.5 \pm 0.2}$	76.0 ± 0.6	74.9 ± 0.1	75.1 ± 0.8	$\mathbf{78.4 \pm 0.4}$
Citeseer	72.3 ± 0.3	72.6 ± 0.3	72.3 ± 0.3	$\mathbf{73.8 \pm 0.2}$	59.3 ± 0.9	60.3 ± 0.8	61.7 ± 0.7	$\mathbf{63.1 \pm 0.8}$
Texas	44.2 ± 1.5	43.9 ± 1.6	46.0 ± 1.6	$\mathbf{49.4 \pm 1.2}$	53.5 ± 3.1	50.3 ± 3.7	47.0 ± 3.7	$\mathbf{63.1 \pm 1.7}$
Cornell	41.5 ± 1.8	42.2 ± 1.5	40.2 ± 1.6	$\mathbf{50.8 \pm 1.1}$	36.5 ± 2.2	40.0 ± 2.1	35.6 ± 2.4	$\mathbf{48.6 \pm 1.2}$
Wisconsin	44.6 ± 1.4	46.2 ± 1.2	48.3 ± 1.3	$\mathbf{50.3 \pm 0.9}$	48.5 ± 2.2	48.8 ± 1.9	48.5 ± 2.1	$\mathbf{54.9 \pm 1.2}$
Chameleon	59.2 ± 0.6	59.4 ± 0.5	59.3 ± 0.6	$\mathbf{61.5 \pm 0.4}$	58.1 ± 2.1	58.4 ± 2.1	56.3 ± 2.2	$\mathbf{65.3 \pm 0.8}$

Table 3. Classification accuracies of GCN and GIN with None, SDRF, FoSR, and BORF rewiring on various graph classification datasets. Best results are highlighted in bold.

	GCN				GIN			
Data set	None	SDRF	FoSR	BORF	None	SDRF	FoSR	BORF
Enzymes	25.5 ± 1.3	26.1 ± 1.1	$\mathbf{27.4 \pm 1.1}$	24.7 ± 1.0	31.3 ± 1.2	33.5 ± 1.3	25.3 ± 1.2	$\mathbf{35.5 \pm 1.2}$
Imdb	49.3 ± 1.0	49.1 ± 0.9	49.6 ± 0.8	$\mathbf{50.1 \pm 0.9}$	69.0 ± 1.3	68.6 ± 1.2	69.5 ± 1.1	$\mathbf{71.3 \pm 1.5}$
Mutag	68.8 ± 2.1	70.5 ± 2.1	75.6 ± 1.7	$\mathbf{75.8 \pm 1.9}$	75.5 ± 2.9	77.3 ± 2.3	75.2 ± 3.0	$\mathbf{80.8 \pm 2.5}$
Proteins	70.6 ± 1.0	71.4 ± 0.8	$\mathbf{72.3 \pm 0.9}$	71.0 ± 0.8	69.7 ± 1.0	72.2 ± 0.9	$\mathbf{74.2 \pm 0.8}$	71.3 ± 1.0

Table 4. Classification accuracies of GCN at depths 5, 7, and 9 with different BORF rewiring options on Cornell and Mutag datasets.

Data set	# layers	None	Best settings	Only remove	Only add	remove & add equally
Cornell	5	41.3 ± 1.4	45.5 ± 1.1	46.4 ± 1.2	44.7 ± 1.3	45.9 ± 1.2
	7	39.5 ± 1.7	41.5 ± 1.5	43.2 ± 1.3	42.8 ± 1.4	41.8 ± 1.3
	9	35.5 ± 1.4	40.9 ± 1.3	41.9 ± 1.6	40.3 ± 2.0	39.9 ± 1.6
Mutag	5	67.7 ± 1.6	75.4 ± 2.1	68.5 ± 2.8	76.1 ± 2.2	71.8 ± 1.2
	7	64.1 ± 2.1	72.1 ± 1.3	65.1 ± 1.5	75.2 ± 2.4	66.2 ± 1.9
	9	63.1 ± 1.2	69.7 ± 1.5	60.7 ± 2.5	70.4 ± 1.7	61.3 ± 1.5

Further experimental details are described in Section E.1. We report our experimental results in Table 4.

At all depths, the GNN can learn relevant information from the datasets preprocessed with BORF more effectively, even when the rewiring algorithm is constrained to only alleviate either over-smoothing or over-squashing. Indeed, the unrewired datasets record the worst performance in 5 out of 6 scenarios considered. This indicates that both issues negatively impact the classification accuracy as the depth count increases. Our results suggest both BORF's abilities to reduce over-smoothing and over-squashing play an essential role in improving the performance of GNN models.

7. Conclusion

In this paper, we established the novel correspondence between the Ollivier-Ricci curvature on graphs with both the over-smoothing and the over-squashing issues. In specific, we showed that positive graph curvature is associated with over-smoothing, while negative graph curvature is associated with over-squashing. Based on our theoretical results, we proposed Batch Ollivier-Ricci Flow, a novel curvature based rewiring method that can effectively improve GNN performance by tackling both the over-smoothing and over-squashing problems at the same time. It is interesting to note that by different definitions of the random walk μ_u, we may be able to capture different behaviors of the local graph structures using graph curvature.

Limitations and societal impacts. Our current formulation only demonstrates that the over-squashing issue is related to negative Ollivier-Ricci curvature at a local scale under some constraints on the graph structure and GNN design. We leave providing a more general treatment of the problem for future work. Our theory and proposed method have no discernible negative societal impact.

Acknowledgements

This material is based on research sponsored by the NSF Grant# 2030859 to the Computing Research Association for the CIFellows Project (CIF2020-UCLA-38). SJO acknowledges support from the ONR N00014-20-1-2093/N00014-20-1-2787 and the NSF DMS 2208272 and 1952339.

References

Ahuja, R., Magnanti, T., and Orlin, J. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993. ISBN 9780136175490.

Alon, U. and Yahav, E. 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.

Atasu, K. and Mittelholzer, T. Linear-complexity data-parallel earth mover's distance approximations. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 364–373. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/atasu19a.html.

Bakry, D. and Émery, M. Diffusions hypercontractives. In Azéma, J. and Yor, M. (eds.), *Séminaire de Probabilités XIX 1983/84*, pp. 177–206, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg. ISBN 978-3-540-39397-9.

Banerjee, P. K., Karhadkar, K., Wang, Y. G., Alon, U., and Montúfar, G. Oversquashing in GNNs through the lens of information contraction and graph expansion. In *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1–8, 2022.

Battaglia, P., Pascanu, R., Lai, M., Jimenez Rezende, D., and kavukcuoglu, k. Interaction networks for learning about objects, relations and physics. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/3147da8ab4a0437c15ef51a5cc7f2dc4-Paper.pdf.

Bayraktar, E. and Guo, G. Strong equivalence between metrics of Wasserstein type. *Electronic Communications in Probability*, 26:1 – 13, 2021. doi: 10.1214/21-ECP383. URL https://doi.org/10.1214/21-ECP383.

Bober, J., Monod, A., Saucan, E., and Webster, K. N. Rewiring networks for graph neural network training using discrete geometry. *arXiv preprint arXiv:2207.08026*, 2022.

Bojchevski, A., Shchur, O., Zügner, D., and Günnemann, S. NetGAN: Generating graphs via random walks. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 610–619. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/bojchevski18a.html.

Bonneel, N., Rabin, J., Peyre, G., and Pfister, H. Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 2014.

Bourne, D. P., Cushing, D., Liu, S., Münch, F., and Peyerimhoff, N. Ollivier–Ricci idleness functions of graphs. *SIAM Journal on Discrete Mathematics*, 32(2): 1408–1424, 2018. doi: 10.1137/17M1134469. URL https://doi.org/10.1137/17M1134469.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. arXiv, 2021. doi: 10.48550/ARXIV.2104.13478. URL https://arxiv.org/abs/2104.13478.

Cai, C. and Wang, Y. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

Chen, J., Zhu, J., and Song, L. Stochastic training of graph convolutional networks with variance reduction. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 942–950. PMLR, 10–15 Jul 2018a. URL https://proceedings.mlr.press/v80/chen18p.html.

Chen, Y., Wei, Z., and Huang, X. Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pp. 1655–1658, New York, NY, USA, 2018b. Association for Computing Machinery. ISBN 9781450360142. doi: 10.1145/3269206.3269269. URL https://doi.org/10.1145/3269206.3269269.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/f9be311e65d81a9ad8150a60844bb94c-Paper.pdf.

Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., and Beaini, D. Long range graph benchmark. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 22326–22340. Curran Associates, Inc., 2022.

Estrada, E. and Bonchev, D. *Chemical Graph Theory*, pp. 1538–1558. Discrete Mathematics and Its Applications Series. Taylor & Francis, 12 2013. ISBN 9781439880180. doi: 10.1201/b16132-92.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. Graph neural networks for social recommendation. In *The World Wide Web Conference*, WWW '19, pp. 417–426, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313488. URL https://doi.org/10.1145/3308558.3313488.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. POT: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL http://jmlr.org/papers/v22/20-451.html.

Forman. Bochner's method for cell complexes and combinatorial Ricci curvature. *Discrete & Computational Geometry*, 29(3):323–374, Feb 2003. ISSN 1432-0444. doi: 10.1007/s00454-002-0743-x. URL https://doi.org/10.1007/s00454-002-0743-x.

Gasteiger, J., Weiß enberger, S., and Günnemann, S. Diffusion improves graph learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/23c894276a2c5a16470e6a31f4618d73-Paper.pdf.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/gilmer17a.html.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

Harary, F. *Graph Theory and Theoretical Physics*. Academic Press, 1967.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585:357–362, 2020. doi: 10.1038/s41586-020-2649-2.

Hsu, L. and Lin, C. *Graph Theory and Interconnection Networks*. CRC Press, 2008. ISBN 9781420044829.

Karhadkar, K., Banerjee, P. K., and Montúfar, G. FoSR: First-order spectral rewiring for addressing oversquashing in GNNs. In *International Conference on Learning Representations*, 2023.

Kipf, T. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2688–2697. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/kipf18a.html.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018a. ISBN 978-1-57735-800-8.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. Learning deep generative models of graphs. *arXiv preprint arXiv:2207.08026*, 2018b.

Luan, S., Zhao, M., Chang, X.-W., and Precup, D. Break the ceiling: Stronger multi-scale deep graph convolutional networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Matsunaga, D., Suzumura, T., and Takahashi, T. Exploring graph neural networks for stock market predictions with rolling window analysis. *arXiv preprint arXiv:1909.10660*, 2019.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL www.graphlearning.io.

Ni, C.-C., Lin, Y.-Y., Gao, J., David Gu, X., and Saucan, E. Ricci curvature of the Internet topology. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 2758–2766, 2015. doi: 10.1109/INFOCOM.2015.7218668.

Ollivier, Y. Ricci curvature of Markov chains on metric spaces. *Journal of Functional Analysis*, 256(3):810–864, 2009. ISSN 0022-1236. doi: 10.1016/j.jfa.2008.11.001.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1ldO2EFPr.

Paeng, S.-H. Volume and diameter of a graph and Ollivier's Ricci curvature. *Eur. J. Comb.*, 33:1808–1819, 2012.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-GCN: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1e2agrFvS.

Rong, Y., Huang, W., Xu, T., and Huang, J. DropEdge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hkx1qkrKPr.

Rozemberczki, B., Allen, C., and Sarkar, R. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2), 05 2021. ISSN 2051-1329. doi: 10.1093/comnet/cnab014. URL https://doi.org/10.1093/comnet/cnab014.

Rusch, T. K., Chamberlain, B., Rowbottom, J., Mishra, S., and Bronstein, M. Graph-coupled oscillator networks. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18888–18909. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/rusch22a.html.

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. Graph networks as learnable physics engines for inference and control. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4470–4479. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/sanchez-gonzalez18a.html.

Scaman, K. and Virmaux, A. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/d54e99a6c03704e95e6965532dec148b-Paper.pdf.

Shang, C., Tang, Y., Huang, J., Bi, J., He, X., and Zhou, B. End-to-end structure-aware convolutional networks for knowledge base completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33 (01):3060–3067, Jul. 2019. doi: 10.1609/aaai.v33i01.33013060. URL https://ojs.aaai.org/index.php/AAAI/article/view/4164.

Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., and Tang, J. GraphAF: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1esMkHYPr.

Sia, J., Jonckheere, E., and Bogdan, P. Ollivier-Ricci curvature-based method to community detection in complex networks. *Scientific Reports*, 9(1): 9800, Jul 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-46079-x. URL https://doi.org/10.1038/s41598-019-46079-x.

Tantau, T. *The TikZ and PGF Packages*, 2023. URL https://tikz.dev/.

Topping, J., Giovanni, F. D., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=7UmjRGzp-A.

van den Berg, R., Kipf, T. N., and Welling, M. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

van der Hoorn, P., Cunningham, W. J., Lippner, G., Trugenberger, C., and Krioukov, D. Ollivier-Ricci curvature convergence in random geometric graphs. *Phys. Rev. Res.*, 3:013211, Mar 2021. doi: 10.1103/PhysRevResearch.3.013211. URL https://link.aps.org/doi/10.1103/PhysRevResearch.3.013211.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Wu, Q., Zhang, H., Gao, X., He, P., Weng, P., Gao, H., and Chen, G. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *The World Wide Web Conference*, WWW '19, pp. 2091–2102, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313442. URL https://doi.org/10.1145/3308558.3313442.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5453–5462. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/xu18c.html.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yang, Y., Wei, Z., Chen, Q., and Wu, L. Using external knowledge for financial event prediction based on graph neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pp. 2161–2164, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369763. doi: 10.1145/3357384.3358156. URL https://doi.org/10.1145/3357384.3358156.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 40–48, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/yanga16.html.

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

Zhang, F., Liu, X., Tang, J., Dong, Y., Yao, P., Zhang, J., Gu, X., Wang, Y., Shao, B., Li, R., and Wang, K. OAG: Toward linking large-scale heterogeneous entity graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pp. 2585–2595, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330785. URL https://doi.org/10.1145/3292500.3330785.

Zhao, L. and Akoglu, L. PairNorm: Tackling oversmoothing in GNNs. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkecl1rtwB.

Supplement for "Revisiting Over-smoothing and Over-squashing using Ollivier-Ricci Curvature"

In this supplementary material, we first present how Equation (1) accommodates different designs of GNNs in Appendix A. Then, we discuss the lack of an appropriate measure for the over-squashing issue in Appendix B. Skipped proofs within the main text are provided in Appendix C. We present our experiment settings in Appendix D and additional experiment results in Appendix E. Dataset statistics are available in Appendix F. Finally, we provide our hardware specifications and list of libraries in Appendix G.

A. Message Passing Neural Networks

In its most general form, a typical layer of a message passing neural network is given by the following update rule (Bronstein et al., 2021):

$$\boldsymbol{H}_u = \phi\left(\boldsymbol{X}_u, \bigoplus_{v \in \mathcal{N}_u} \Lambda(\boldsymbol{X}_u, \boldsymbol{X}_v)\right).$$

Here, Λ , $\bigoplus$ and ϕ are the message, aggregate and update functions. Different designs of MPNNs amount to different choices for these functions. The additional input of $\boldsymbol{X}_u$ to ϕ represents an optional skip-connection.

In practice, we found that the skip connection of each vertex u is often implemented by considering a message passing scheme where each node sends a message to itself. This can be thought of as adding self-loops to the graph, and its impact has been studied by Xu et al. (2018) and Topping et al., (2022). Then, Λ could be realized as a learnable affine transformation ψ of $\boldsymbol{X}_v$, the aggregating function $\bigoplus$ could either be chosen as a sum, mean, weighted mean, or max operation, and ϕ is a suitable activation function. Hence, we arrive at Equation (1), which we restate below

$$\boldsymbol{X}_u^{k+1} = \phi_k\left(\bigoplus_{p \in \tilde{\mathcal{N}}_u} \psi_k(\boldsymbol{X}_p^k)\right).$$

For example, graph convolutional network (GCN) (Kipf & Welling, 2017) defines its layer as

$$\boldsymbol{H}_u = \sigma\left(\sum_{v \in \tilde{\mathcal{N}}_u} \frac{1}{c_{uv}} \boldsymbol{W}\boldsymbol{X}_v\right),$$

with $c_{uv} = \sqrt{|\tilde{\mathcal{N}}_u||\tilde{\mathcal{N}}_v|}$. The mean aggregator variant (but not other variants) of GraphSAGE (Hamilton et al., 2017) uses the same formulation but with $c_{uv} = |\tilde{\mathcal{N}}_u|$. Both choices of c_{uv} have the exact same spectral characteristics and act identically in theory (Li et al., 2018a), which leads to an averaging behavior based on node degrees. Similarly, graph attention networks (GAT) (Veličković et al., 2018) defines its single head layer as

$$\boldsymbol{H}_u = \sigma\left(\sum_{v \in \tilde{\mathcal{N}}_u} a_{uv} \boldsymbol{W}\boldsymbol{X}_v\right).$$

The difference here being a_{uv} is now a learnable function given by the attention mechanism (Vaswani et al., 2017). Finally, graph isomorphism network (GIN) (Xu et al., 2019) is formulated by

$$\boldsymbol{H}_u = \mathrm{MLP}\left((1+\epsilon)\boldsymbol{X}_u + \sum_{v \in \mathcal{N}_u} \boldsymbol{X}_v\right),$$

where MLP is a multilayer perceptron. GIN achieves its best performance with the model GIN-0, in which ϵ is set to be 0.

We remark that most nonlinear activation functions such as ReLU, Leaky ReLU, Tanh, Sigmoid, softmax, etc., has a simple and explicit Lipschitz-constant (which equals to 1 more often than not) (Scaman & Virmaux, 2018).

B. Measuring Over-squashing

It is intuitive to think that if N vertices in $\tilde{\mathcal{N}}_u$ contribute to the feature representation of some vertex u by the permutation invariant update rule 1, we should expect each such vertex to provide $\frac{1}{N}$ of the total contribution. If this is repeated over and over, such as in a tree, the exponentially decaying dependence of distant vertices is to be expected. However, quantifying this phenomenon is actually quite difficult. The first work to call attention to the over-squashing problem (Alon & Yahav, 2021) measures whether breaking the bottleneck improves the results of long-range problems, instead of measuring the decaying importance itself.

A definition for the over-squashing issue that took inspiration from the vanishing gradient problem was given by (Topping et al., 2022), where it was suggested that $\frac{\partial \boldsymbol{X}_u^k}{\partial \boldsymbol{X}_v^0}$ can be used to evaluate the decreasing importance of distant vertex v to u. However, this quantity does not actually measure the squashing behavior experienced by all GNNs, but only by those using an aggregation function with a natural decaying effect such as GCNs. Furthermore, we argue that in Theorem 4 of (Topping et al., 2022), there is no general squashing behavior being described. Given some negatively curved edge (i, j), the conclusion of the theorem provides a bound for the average effect of messages from the node i to a set of neighbors Q_j of the node j. In other words, this describes how one single node on one specific side of the edge (the side with the lower degree) can transmit messages to other nodes and not the other way around. To visualize this, let us look at Figure 4a. The theorem measures how effectively the red node u can send messages to a number of blue nodes in the neighborhood of v. Clearly, there is no squashing of information from the surrounding receptive field being considered: the message was sent from one single node, and there is no other message to be squashed. In reality, this theorem is describing the average dilution of information sent out by the node i to other nodes k of distance 2 from i. This dilution is caused by the choice of using the normalized adjacency matrix as the aggregating function. If we use the sum aggregating function as in GIN, then such diluting process does not take place.

Clearly, it is the *relative* importance of a vertex compared to the contribution of all other vertices that is at the heart of the matter. To this end, we have found that the closest notion to our description actually predates the observation of the over-squashing issue. Xu et al. (2018) introduced the notion of influence distribution to quantify the relative importance of vertices to each other. It is defined as

$$I_u(v) = \frac{\mathrm{sum}\left(\left[\frac{\partial \boldsymbol{X}_u^k}{\partial \boldsymbol{X}_v^0}\right]\right)}{\sum_{p \in V} \mathrm{sum}\left(\left[\frac{\partial \boldsymbol{X}_u^k}{\partial \boldsymbol{X}_p^0}\right]\right)}$$

where the sum is taken over all entries of each Jacobian matrix. Unfortunately, this definition is quite unwieldy to use in any sort of analysis. We would like to remark that the theoretical proofs in (Xu et al., 2018) are only partially correct. They have made a mistake by claiming

$$\mathbb{E}\left(\frac{X_1}{\sum_{i=1}^n X_i}\right) = \frac{\mathbb{E}(X_1)}{\sum_{i=1}^n \mathbb{E}(X_i)}$$

for random variables X_i.

C. Proofs

In this Appendix, we provide proofs for key results in the paper. We state without proof the following lemma, which is Lemma 4.1 in (Bourne et al., 2018).

Lemma C.1. *Let μ_1, μ_2 be probability measures on a space V. Then there exists an optimal transport plan π transporting μ_1 to μ_2 with the following property: For all $x \in V$ with $\mu_1(x) \leq \mu_2(x)$, we have $\pi(x, x) = \mu_1(x)$.*

C.1. Proof of Lemma 4.1

Proof. Without loss of generality, assume $n \geq m$. Let π be an optimal transport plan between μ_u and μ_v satisfying the condition in Lemma C.1. That is, $\pi(p, p) = \frac{1}{n}$ for all $p \in \mathcal{N}(u) \cap \mathcal{N}(v)$. We have

$$\begin{aligned} W_1(\mu_u, \mu_v) &= \sum_{p \in \mathcal{N}_u} \sum_{q \in \mathcal{N}_v} \pi(p, q) d(p, q) \\ &= \sum_{\substack{(p,q) \in \mathcal{N}_u \times \mathcal{N}_v \\ p \neq q}} \pi(p, q) d(p, q) + \sum_{p \in \mathcal{N}(u) \cap \mathcal{N}(v)} \pi(p, p) d(p, p). \end{aligned}$$

It is obvious that $d(p,p) = 0$ for any vertex p and $d(p,q) \geq 1$ for any vertices $p \neq q$. We have

$$\begin{aligned} W_1(\mu_u, \mu_v) &\geq \sum_{\substack{(p,q)\in\mathcal{N}_u\times\mathcal{N}_v \\ p\neq q}} \pi(p,q) + 0 \\ &= 1 - \sum_{p\in\mathcal{N}_u\cap\mathcal{N}_v} \pi(p,p) \\ &= 1 - \frac{|\mathcal{N}_u \cap \mathcal{N}_v|}{n}. \end{aligned}$$

Hence, we have

$$\frac{|\mathcal{N}_u \cap \mathcal{N}_v|}{\max(m,n)} \geq 1 - W_1(\mu_u, \mu_v) = \kappa(u,v).$$

□

C.2. Proof of Theorem 4.2

As ϕ_k is L-Lipschitz, we have

$$\begin{aligned} \left|\boldsymbol{X}_u^{k+1} - \boldsymbol{X}_v^{k+1}\right| &= \left|\phi_k\left(\bigoplus_{p\in\tilde{\mathcal{N}}_u} \psi_k(\boldsymbol{X}_p^k)\right) - \phi_k\left(\bigoplus_{q\in\tilde{\mathcal{N}}_v} \psi_k(\boldsymbol{X}_q^k)\right)\right| \\ &\leq L\left|\bigoplus_{p\in\tilde{\mathcal{N}}_u} \psi_k(\boldsymbol{X}_p^k) - \bigoplus_{q\in\tilde{\mathcal{N}}_v} \psi_k(\boldsymbol{X}_q^k)\right|. \end{aligned} \tag{13}$$

Theorem 4.1 tells us that

$$|\tilde{\mathcal{N}}_v \backslash \tilde{\mathcal{N}}_u| \leq |\tilde{\mathcal{N}}_u \backslash \tilde{\mathcal{N}}_v| = n + 1 - |\mathcal{N}_u \cap \mathcal{N}_v| - 2 \leq n - n\kappa(u,v).$$

Hence, there are at most $\lfloor(1-\kappa(u,v))n\rfloor$ vertices in the extended neighborhood of u that is not present in the extended neighborhood of v and vice versa. The symmetric difference $\tilde{\mathcal{N}}_u \vartriangle \tilde{\mathcal{N}}_v$ satisfies

$$|\tilde{\mathcal{N}}_u \vartriangle \tilde{\mathcal{N}}_v| = |(\tilde{\mathcal{N}}_u \backslash \tilde{\mathcal{N}}_v) \cup (\tilde{\mathcal{N}}_v \backslash \tilde{\mathcal{N}}_u)| \leq 2(1-\kappa(u,v))n.$$

- If $\bigoplus$ is realized as the sum operation, we obtain from Equation (13)

$$\begin{aligned} \left|\boldsymbol{X}_u^{k+1} - \boldsymbol{X}_v^{k+1}\right| &\leq L\left|\sum_{p\in\mathcal{N}_u\cup\{u\}} \psi_k(\boldsymbol{X}_p^k) - \sum_{q\in\mathcal{N}_v\cup\{v\}} \psi_k(\boldsymbol{X}_q^k)\right| \\ &= L\left|\sum_{p\in\tilde{\mathcal{N}}_u\backslash\tilde{\mathcal{N}}_v} \psi_k(\boldsymbol{X}_p^k) - \sum_{q\in\tilde{\mathcal{N}}_v\backslash\tilde{\mathcal{N}}_u} \psi_k(\boldsymbol{X}_q^k)\right| \\ &\leq L \sum_{p\in\tilde{\mathcal{N}}_u\vartriangle\tilde{\mathcal{N}}_v} \left|\psi_k(\boldsymbol{X}_p^k)\right| \\ &\leq (1-\kappa(u,v))2LCMn. \end{aligned}$$

  We can now set $h \equiv 2LCMn$.

- If $\bigoplus$ is realized as the mean operation, we obtain from Equation (13)

$$\begin{aligned}\left|\boldsymbol{X}_u^{k+1}-\boldsymbol{X}_v^{k+1}\right| &\leq L\left|\sum_{p\in\tilde{\mathcal{N}}_u}\frac{1}{n+1}\psi_k(\boldsymbol{X}_p^k)-\sum_{q\in\tilde{\mathcal{N}}_v}\frac{1}{m+1}\psi_k(\boldsymbol{X}_q^k)\right|\\ &\leq L\sum_{p\in(\tilde{\mathcal{N}}_u\cap\tilde{\mathcal{N}}_v)}\left(\frac{1}{m+1}-\frac{1}{n+1}\right)\left|\psi_k(\boldsymbol{X}_p^k)\right|\\ &\quad+L\left|\sum_{p\in\tilde{\mathcal{N}}_u\setminus\tilde{\mathcal{N}}_v}\frac{1}{n+1}\psi_k(\boldsymbol{X}_p^k)-\sum_{q\in\tilde{\mathcal{N}}_v\setminus\tilde{\mathcal{N}}_u}\frac{1}{m+1}\psi_k(\boldsymbol{X}_q^k)\right|. \qquad (14)\end{aligned}$$

We have $n\geq m=|\mathcal{N}_v|\geq|\mathcal{N}_u\cap\mathcal{N}_v|\geq\kappa(u,v)n$, and

$$\frac{1}{m+1}-\frac{1}{n+1}\leq\frac{1}{m}-\frac{1}{n}\leq\frac{1}{\kappa(u,v)n}-\frac{1}{n}=\frac{1-\kappa(u,v)}{\kappa(u,v)n}.$$

Therefore, Equation (14) gives

$$\begin{aligned}\left|\boldsymbol{X}_u^{k+1}-\boldsymbol{X}_v^{k+1}\right| &\leq L\sum_{p\in\tilde{\mathcal{N}}_u\cap\tilde{\mathcal{N}}_v}\frac{1-\kappa(u,v)}{\kappa(u,v)n}\left|\psi_k(\boldsymbol{X}_p^k)\right|+L\sum_{p\in\tilde{\mathcal{N}}_u\triangle\tilde{\mathcal{N}}_v}\frac{1}{\kappa(u,v)n+1}\left|\psi_k(\boldsymbol{X}_p^k)\right|\\ &\leq L(n+1)\frac{1-\kappa(u,v)}{\kappa(u,v)n}CM+2(1-\kappa(u,v))nL\frac{1}{\kappa(u,v)n+1}CM\\ &\leq(1-\kappa(u,v))LCM\left(\frac{n+1}{\kappa(u,v)n}+2\frac{n}{\kappa(u,v)n+1}\right).\end{aligned}$$

We can now set $h(x)=LCM(\frac{n+1}{xn}+2\frac{n}{nx+1})$.

Clearly, the functions h as defined satisfy the conditions given in Theorem 4.2.

C.3. Proof of Proposition 4.3

We will use proof by induction. For all edges $u\sim v$, repeating the argument in Theorem 4.2, we get $|\tilde{\mathcal{N}}_u\triangle\tilde{\mathcal{N}}_v|\leq 2(1-\delta)n$. Then, the base case $k=1$ follows since

$$\begin{aligned}|\boldsymbol{X}_u^1-\boldsymbol{X}_v^1| &= \left|\phi_1\left(\frac{1}{n+1}\sum_{p\in\tilde{\mathcal{N}}_u}\psi(\boldsymbol{X}_p)\right)-\phi_1\left(\frac{1}{n+1}\sum_{q\in\tilde{\mathcal{N}}_v}\psi(\boldsymbol{X}_q)\right)\right|\\ &\leq L\left|\frac{1}{n+1}\sum_{p\in\tilde{\mathcal{N}}_u}\psi(\boldsymbol{X}_p)-\frac{1}{n+1}\sum_{q\in\tilde{\mathcal{N}}_v}\psi(\boldsymbol{X}_q)\right|\\ &=\frac{L}{n+1}\left|\sum_{p\in\tilde{\mathcal{N}}_u\setminus\tilde{\mathcal{N}}_v}\psi(\boldsymbol{X}_p)-\sum_{q\in\tilde{\mathcal{N}}_v\setminus\tilde{\mathcal{N}}_u}\psi(\boldsymbol{X}_q)\right|\\ &\leq\frac{L}{n+1}\sum_{p\in\tilde{\mathcal{N}}_u\triangle\tilde{\mathcal{N}}_v}|\psi(\boldsymbol{X}_p)|\\ &\leq\frac{2\lfloor(1-\delta)n\rfloor}{n+1}LCM.\end{aligned}$$

Suppose the statement is true for k and consider the case $k+1$. We have for all $u \sim v$:

$$\begin{aligned} \left|\boldsymbol{X}_u^{k+1} - \boldsymbol{X}_v^{k+1}\right| &\leq L\frac{1}{n+1}\left|\sum_{p\in\tilde{\mathcal{N}}_u}\psi_k(\boldsymbol{X}_p^k) - \sum_{q\in\tilde{\mathcal{N}}_v}\psi_k(\boldsymbol{X}_q^k)\right| \\ &= L\frac{1}{n+1}\left|\sum_{p\in\tilde{\mathcal{N}}_u\backslash\tilde{\mathcal{N}}_v}\psi_k(\boldsymbol{X}_p^k) - \sum_{q\in\tilde{\mathcal{N}}_v\backslash\tilde{\mathcal{N}}_u}\psi_k(\boldsymbol{X}_q^k)\right| \\ &= L\frac{1}{n+1}\left|\psi_k\left(\sum_{p\in\tilde{\mathcal{N}}_u\backslash\tilde{\mathcal{N}}_v}\boldsymbol{X}_p^k - \sum_{q\in\tilde{\mathcal{N}}_v\backslash\tilde{\mathcal{N}}_u}\boldsymbol{X}_q^k\right)\right| \\ &\leq LM\frac{1}{n+1}\left|\sum_{p\in\tilde{\mathcal{N}}_u\backslash\tilde{\mathcal{N}}_v}\boldsymbol{X}_p^k - \sum_{q\in\tilde{\mathcal{N}}_v\backslash\tilde{\mathcal{N}}_u}\boldsymbol{X}_q^k\right|. \end{aligned} \tag{15}$$

For each $p \in \tilde{\mathcal{N}}_u\backslash\tilde{\mathcal{N}}_v$, match it with one and only one $q \in \tilde{\mathcal{N}}_v\backslash\tilde{\mathcal{N}}_u$. For any node pair (p,q), they are connected by the path $p \sim u \sim v \sim q$, where the difference in norm of features at layer k of each 1-hop connection is at most $\frac{2}{3}C\left(\frac{3LM\lfloor(1-\delta)n\rfloor}{n+1}\right)^k$. Hence, we have

$$|\boldsymbol{X}_p^k - \boldsymbol{X}_q^k| \leq 2C\left(\frac{3LM\lfloor(1-\delta)n\rfloor}{n+1}\right)^k.$$

Substituting this into equation (15), and by noting that there are at most $\lfloor(1-\delta)n\rfloor$ pairs, we get

$$\begin{aligned} |\boldsymbol{X}_u^{k+1} - \boldsymbol{X}_v^{k+1}| &\leq LM\frac{1}{n+1}\sum_{(p,q)}\left|\boldsymbol{X}_p^k - \boldsymbol{X}_q^k\right| \\ &\leq LM\frac{1}{n+1}\lfloor(1-\delta)n\rfloor 2C\left(\frac{3LM\lfloor(1-\delta)n\rfloor}{n+1}\right)^k \\ &= \frac{2}{3}C\left(\frac{3LM\lfloor(1-\delta)n\rfloor}{n+1}\right)^{k+1}. \end{aligned}$$

By induction, we have shown inequality (8) holds for all $k \geq 1$ and $u \sim v$.

It is known that if the curvature of every edge in a graph is positive and bounded away from zero by $\delta > 0$ then the diameter of the graph does not exceed $\lfloor 2/\delta\rfloor$ (Paeng, 2012). Hence, for any two nodes $u, v \in \mathcal{V}$, the shortest path between them is of length at most $\lfloor 2/\delta\rfloor$. Apply the inequality (8) for each pair of neighboring nodes on this shortest path, we obtain the inequality (9).

C.4. Proof of Proposition 4.4

Note that S consists of elements of the form (p,q) where either $p = q$ or $p \neq q$. The first type corresponds to mutual neighbors of u, v, while the second type corresponds to neighbors of u, v that share an edge. Denote the number of edges of the first type as n_0 and the number of edges of the second type as n_1. A transport plan π between μ_u and μ_v can be obtained as followed.

- For every vertex p such that $(p,p) \in S$, the mass of $\mu_u(p) = \frac{1}{n}$ remains in place at p with cost $\pi(p,p) \times 0 = \frac{1}{n} \times 0 = 0$
- For each edge $(p,q) \in S$ with $p \neq q$, transport the mass of $\frac{1}{n}$ from p to q with cost $\pi(p,q) \times 1 = \frac{1}{n} \times 1 = \frac{1}{n}$. The assumption that each vertex w is a vertex of at most $\frac{n}{m}$ edges ensures that the total mass transported to each vertex is no greater than $\frac{1}{m}$.
- The remaining mass is $1 - n_0\frac{1}{n} - n_1\frac{1}{n}$. Transport this amount arbitrarily to obtain a valid optimal transport plan.

This transport plan has cost

$$\sum_{p\in\tilde{\mathcal{N}}_u}\sum_{q\in\tilde{\mathcal{N}}_v}\pi(p,q)d(p,q) \leq n_0 0 + n_1\frac{1}{n} + \left(1 - n_0\frac{1}{n} - n_1\frac{1}{n}\right)3 = 3 - 3n_0\frac{1}{n} - 2n_1\frac{1}{n}.$$

Table 5. SDRF's best hyper-parameter settings.

	GCN			GIN		
DATASET	# ITERATION	C^+	# REWIRED	# ITERATION	C^+	# REWIRED
CORA	12	0	24	50	∞	50
CITESEER	175	∞	175	25	∞	25
TEXAS	87	0	174	37	0	74
CORNELL	100	0	200	25	0	50
WISCONSIN	25	0	50	150	∞	150
CHAMELEON	50	0	100	87	0	174
ENZYMES	15	0	30	5	0	10
IMDB	10	0	20	10	∞	10
MUTAG	20	∞	20	10	0	20
PROTEINS	5	0	10	15	0	30

We have

$$\kappa(u,v) = 1 - W_1(\mu_u,\mu_v) \geq 1 - \left(3 - 3n_0\frac{1}{n} - 2n_1\frac{1}{n}\right) = -2 + \frac{3n_0 + 2n_1}{n}.$$

Therefore, we obtain

$$|S| = n_0 + n_1 \leq \frac{n}{2}\frac{3n_0 + 2n_1}{n} \leq n\frac{\kappa(u,v)+2}{2}.$$

We can observe from the proof that a stronger result holds: $3n_0 + 2n_1 \leq n(\kappa(u,v)+2)$.

C.5. Proof of Theorem 4.5

Since ϕ_k and ψ_k are linear operators for all k, their Jacobians J_{ϕ_k}, J_{ψ_k} are constant matrices. By inspection of Equation (1), we see that a vertex $w \in V$ can only transmit a message to u if there exists a vertex w' such that $w' \in \tilde{\mathcal{N}}_u \cap \tilde{\mathcal{N}}_w$. Moreover, the chain rule gives

$$\left[\frac{\partial \boldsymbol{X}_u^{k+2}}{\partial \boldsymbol{X}_w^k}\right] = \sum_{w' \in \tilde{\mathcal{N}}_u \cap \tilde{\mathcal{N}}_w} \left[\frac{\partial \boldsymbol{X}_u^{k+2}}{\partial \boldsymbol{X}_{w'}^{k+1}}\right]\left[\frac{\partial \boldsymbol{X}_{w'}^{k+1}}{\partial \boldsymbol{X}_w^k}\right] = \sum_{w' \in \tilde{\mathcal{N}}_u \cap \tilde{\mathcal{N}}_w} J_{\phi_{k+1}} J_{\psi_{k+1}} J_{\phi_k} J_{\psi_k}.$$

Therefore, $\left[\frac{\partial \boldsymbol{X}_u^{k+2}}{\partial \boldsymbol{X}_w^k}\right]$ is the number of distinct paths (that might contain self-loops) from w to u times $J_{\phi_{k+1}} J_{\psi_{k+1}} J_{\phi_k} J_{\psi_k}$.

The number of distinct paths without self-loops from $q \in \tilde{\mathcal{N}}_v \backslash \{u\}$ to u is not greater than $|S|$ as defined in Proposition 4.4. With self-loops, this rises to at most $|S| + 2$, which corresponds to the case where $q \sim u$.

On the other hand, $\sum_{w \in V}\left[\frac{\partial \boldsymbol{X}_u^{k+2}}{\partial \boldsymbol{X}_w^k}\right]$ equals the number of distinct paths with self-loops with one end at u times $J_{\phi_{k+1}} J_{\psi_{k+1}} J_{\phi_k} J_{\psi_k}$. Clearly, we have

$$\sum_{w \in V}\left[\frac{\partial \boldsymbol{X}_u^{k+2}}{\partial \boldsymbol{X}_w^k}\right] = \left(\sum_{w \in \tilde{\mathcal{N}}_u} (\deg(w) + 1)\right) J_{\phi_{k+1}} J_{\psi_{k+1}} J_{\phi_k} J_{\psi_k}.$$

Let

$$\alpha = \frac{|S| + 2}{\sum_{w \in \tilde{\mathcal{N}}_u} (\deg(w) + 1)},$$

then Proposition 4.4 gives us the required inequality. We can choose β by the same process.

D. Experiment Settings

D.1. Rewiring hyper-parameters

We report the best rewiring settings for every task and baseline GNN architecture. For SDRF, we set the temperature $\tau = \infty$ and only tuned the Ric upper bound C^+ and iteration count. For FoSR, we tuned the iteration count. For BORF, we

Table 6. FoSR's best iteration count settings.

DATASET	GCN	GIN
CORA	150	50
CITESEER	100	200
TEXAS	50	150
CORNELL	125	75
WISCONSIN	175	25
CHAMELEON	50	25
ENZYMES	40	5
IMDB	5	20
MUTAG	10	20
PROTEINS	30	10

Table 7. BORF's best hyper-parameter settings.

	GCN				GIN			
DATASET	n	h	k	# REWIRED	n	h	k	# REWIRED
CORA	3	20	10	90	3	20	30	150
CITESEER	3	20	10	90	3	10	20	90
TEXAS	3	30	10	120	1	20	10	30
CORNELL	2	20	30	100	3	10	20	90
WISCONSIN	2	30	20	100	2	50	30	160
CHAMELEON	3	20	20	120	3	30	30	180
ENZYMES	1	3	2	5	3	3	1	12
IMDB	1	3	0	3	1	4	2	6
MUTAG	1	20	3	23	1	3	1	4
PROTEINS	3	4	1	15	2	4	3	14

tuned the number of batches n, number of edges added per batch h, and number of edges removed per batch k. The exact hyper-parameters for SDRF, FoSR and BORF are available in Table 5, Table 6, and Table 7, respectively. We also report the total amount of edges each method rewired, which equals the total number of added and removed connections for SDRF and BORF. The number of edges rewired by FoSR is the same as the iteration count.

For node classification tasks, each dataset is one big graph. The column titled "# REWIRED" reports the total number of edges rewired in this graph. For graph classification tasks, each dataset is a collection of smaller graphs. The column titled "# REWIRED" reports the number of edges rewired for each of these smaller graphs. We did not vary the number of edges rewired for each smaller graph in graph classification datasets.

D.2. Architecture and experiment settings

For graph and node classification, we utilized fixed model architectures with fixed numbers of GNN layers across all datasets. We used 3 GNN layers for node classification and 4 GNN layers for graph classification tasks. All the intermediate GNN layers (except for the input and output layers) have the same number of input and output dimensions specified by the hidden dimensions. After every GNN layer, we also added a drop-out layer with a fixed drop-out probability and an activation layer. The specific hidden dimensions, drop-out probabilities, and final activation layers for both node and graph classification tasks are specified in the architecture settings in Table 8.

For each graph and node classification experiment, we randomly split the dataset into train, validation and test sets 100 times corresponding to 100 trials. For each trial, the GNN model is trained on the train set using the Adam optimizer and validated using the validation set. The test accuracy corresponding to the best accuracy on the validation set is recorded as the test accuracy of the current trial. After all 100 trials are finished, the mean test accuracy and the 95% confidence interval across all trials are computed and recorded in Tables 2 and 3. We also implemented a callback that stops the training process upon no improvement on the validation accuracy for 100 epochs. The train and validation fractions used to split the dataset

Table 8. Experiment settings for node and graph classification tasks (Note: The train fraction is with respect to the entire dataset while the validation fraction is with respect to the train set).

TASK	LEARNING RATE	#TRIALS/RUN	STOP PATIENCE	TRAIN FRACTION	VALIDATION FRACTION
NODE	0.001	100	100	0.6	0.2
GRAPH	0.001	100	100	0.8	0.1

Table 9. Architecture settings for node and graph classification tasks.

TASK	DROP-OUT PROBABILITY	#GNN LAYERS	HIDDEN DIMENSIONS	FINAL ACTIVATION
NODE	0.5	3	128	RELU
GRAPH	0.5	4	64	RELU

is specified in Table 9.

E. Additional experiments and ablation studies

E.1. Ablation study - Effect of edge addition/removal

In this section, we investigate the role of edge addition and removal in helping BORF improve GNN performance. We compare BORF's performance at high depths when using the best rewiring settings found in previous experiments against versions of BORF where it only removes edges, only adds edges (only alleviates over-squashing), or adds & removes edges equally. Specifically, we consider the following cases

- BORF with the best settings for 3 GNN layers on node classification datasets and 4 GNN layers for graph classification datasets, as reported in Table 7.
- BORF removes the same number of edges k as reported in Table 7 but adds $h = 0$ new edge.
- BORF adds the same number of edges h as reported in Table 7 but removes $k = 0$ edge.
- BORF adds and removes an equal amount of edges $h = k$, taken to be the average of h and k as reported in Table 7.

We use GCN as the baseline GNN and conduct our ablation study on all datasets used in previous experiments at depths 5, 7, and 9. Other experiment details are kept identical to previous experiments, as documented in Section D.2. We report the experiment results in tables 10 and 11. We note that the rewiring hyper-parameters were not tuned for higher depths. As such, the results in the columns "BEST SETTINGS" are not the best performance achievable with BORF. With more depth-specific tuning, we expect to be able to obtain even better classification accuracy for both node classification and graph classification tasks.

We observe that the un-rewired datasets record the worst performance in most scenarios considered. This indicates that GCN frequently suffers from both over-smoothing and over-squashing at high depths and that BORF helps the GNN achieve better performance even when it is restricted to only relieving over-smoothing or only relieving over-squashing.

E.2. Experiment results on long-range graph benchmark

We provide additional empirical results in this section to demonstrate the effectiveness of BORF on the long-range graph classification dataset Peptides-func introduced by (Dwivedi et al., 2022). In table 12, We compare three rewiring algorithms: SDRF, FoSR, and BORF by tuning the hyper-parameters of each method and report the mean average precision (mAP) of the best setting for each algorithm. For each hyper-parameter setting, we run our experiment with 4 random splits and 4 random seeds per split, similar to the setting in (Dwivedi et al., 2022). However, for our experiments, we utilized a lower hidden dimensions of 64 rather than 300. For both SDRF and FoSR, we performed hyper-parameters tuning by trying different numbers of iterations from 25 to 200 with an increment step of 25 to find the optimal number of rewiring iterations. For SDRF, we only tuned the hyper-parameter C^+ using values 0 and ∞. For BORF, we set the range of rewiring batches

Table 10. Classification accuracies of GCN at depths 5, 7, and 9 with different BORF rewiring options on node classification datasets.

Dataset	#layers	None	Best settings	Only add	Only remove	Remove + add equally
Cora	5	84.2 ± 0.8	85.1 ± 1.3	84.6 ± 1.1	84.9 ± 0.9	85.0 ± 0.7
	7	82.1 ± 1.1	82.6 ± 0.9	81.8 ± 1.2	82.2 ± 0.7	83.1 ± 1.1
	9	78.9 ± 0.7	78.4 ± 1.2	77.5 ± 0.9	78.2 ± 1.4	77.9 ± 1.2
Citeseer	5	69.2 ± 1.1	72.1 ± 0.8	70.5 ± 1.4	71.2 ± 1.2	71.8 ± 1.5
	7	65.4 ± 1.5	68.3 ± 0.9	66.8 ± 1.1	67.5 ± 0.9	68.1 ± 1.2
	9	61.7 ± 0.9	62.6 ± 1.6	61.5 ± 1.2	62.2 ± 1.3	63.2 ± 1.5
Texas	5	40.3 ± 1.3	43.3 ± 0.7	43.5 ± 0.8	41.1 ± 1.0	42.8 ± 0.9
	7	37.2 ± 1.1	40.1 ± 1.6	41.4 ± 1.2	39.6 ± 0.5	40.8 ± 0.6
	9	32.1 ± 0.9	34.4 ± 1.2	36.5 ± 1.1	35.1 ± 0.7	35.8 ± 0.6
Cornell	5	41.3 ± 1.4	45.5 ± 1.1	44.7 ± 1.3	46.4 ± 1.2	45.9 ± 1.2
	7	39.5 ± 1.7	41.5 ± 1.5	42.8 ± 1.4	43.2 ± 1.3	41.8 ± 1.3
	9	35.5 ± 1.4	40.9 ± 1.3	40.3 ± 2.0	41.9 ± 1.6	39.9 ± 1.6
Wisconsin	5	40.1 ± 1.2	47.2 ± 0.6	44.4 ± 0.9	45.2 ± 1.1	45.1 ± 0.7
	7	35.5 ± 1.1	42.3 ± 1.0	39.7 ± 0.9	40.9 ± 0.7	39.2 ± 1.2
	9	31.2 ± 0.8	36.4 ± 0.6	34.2 ± 0.9	35.8 ± 1.2	34.7 ± 1.1
Chameleon	5	57.2 ± 0.4	58.8 ± 0.5	58.1 ± 0.7	58.6 ± 0.9	58.9 ± 1.1
	7	55.3 ± 0.5	56.1 ± 0.8	55.9 ± 0.6	56.3 ± 0.4	55.8 ± 0.9
	9	51.1 ± 0.7	52.0 ± 0.4	51.8 ± 0.6	51.3 ± 0.8	52.1 ± 1.2

Table 11. Classification accuracies of GCN at depths 5, 7, and 9 with different BORF rewiring options on graph classification datasets.

Dataset	#layers	None	Best settings	Only add	Only remove	Remove + add equally
Enzymes	5	24.2 ± 1.3	24.5 ± 1.0	24.8 ± 1.6	23.8 ± 1.1	25.0 ± 1.4
	7	21.7 ± 1.5	22.3 ± 1.2	21.9 ± 1.7	21.1 ± 1.2	22.1 ± 1.3
	9	20.8 ± 0.9	20.4 ± 1.1	20.8 ± 1.2	19.7 ± 1.3	20.1 ± 1.2
Imdb	5	47.6 ± 0.8	49.1 ± 1.0	48.8 ± 1.1	46.9 ± 1.4	48.6 ± 1.2
	7	44.3 ± 1.5	45.4 ± 1.2	44.7 ± 1.1	43.7 ± 1.3	44.2 ± 1.4
	9	39.2 ± 1.0	41.8 ± 1.1	40.7 ± 1.6	39.3 ± 1.1	41.3 ± 1.2
Mutag	5	67.7 ± 1.6	75.4 ± 2.1	76.1 ± 2.2	68.5 ± 2.8	71.8 ± 1.2
	7	64.1 ± 2.1	72.1 ± 1.3	75.2 ± 2.4	65.1 ± 1.5	66.2 ± 1.9
	9	63.1 ± 1.2	69.7 ± 1.5	70.4 ± 1.7	60.7 ± 2.5	61.3 ± 1.5
Proteins	5	69.2 ± 0.8	70.1 ± 1.0	69.5 ± 1.1	69.3 ± 1.2	69.9 ± 1.4
	7	67.3 ± 1.0	68.1 ± 0.9	68.3 ± 0.8	67.5 ± 1.1	67.9 ± 1.2
	9	64.5 ± 1.1	65.1 ± 1.1	64.9 ± 1.2	64.7 ± 1.0	65.0 ± 1.3

Table 12. Classification accuracies of GCN and GIN with None, SDRF, FoSR, and BORF rewiring on the Peptides-func dataset. Best results are highlighted in bold.

Layer type	None	SDRF	FoSR	BORF
GCN	40.1 ± 2.1	41.5 ± 1.5	$\mathbf{44.2 \pm 2.1}$	43.9 ± 2.7
GIN	46.1 ± 2.4	46.3 ± 1.7	48.3 ± 1.8	$\mathbf{50.2 \pm 1.7}$

from 1 to 4. For each number of rewiring batches, we tested for the following pairs of edge addition - removal settings: 30 - 20, 40 - 10. In table 13, we report the best settings of SDRF, FoSR, and BORF for both GCN and GIN layer types.

We observe that on GCN, BORF's performance is comparable with that of FoSR. Both of these rewiring algorithms significantly improve the model performance compared to the baseline and SDRF. On GIN, BORF is the best performer

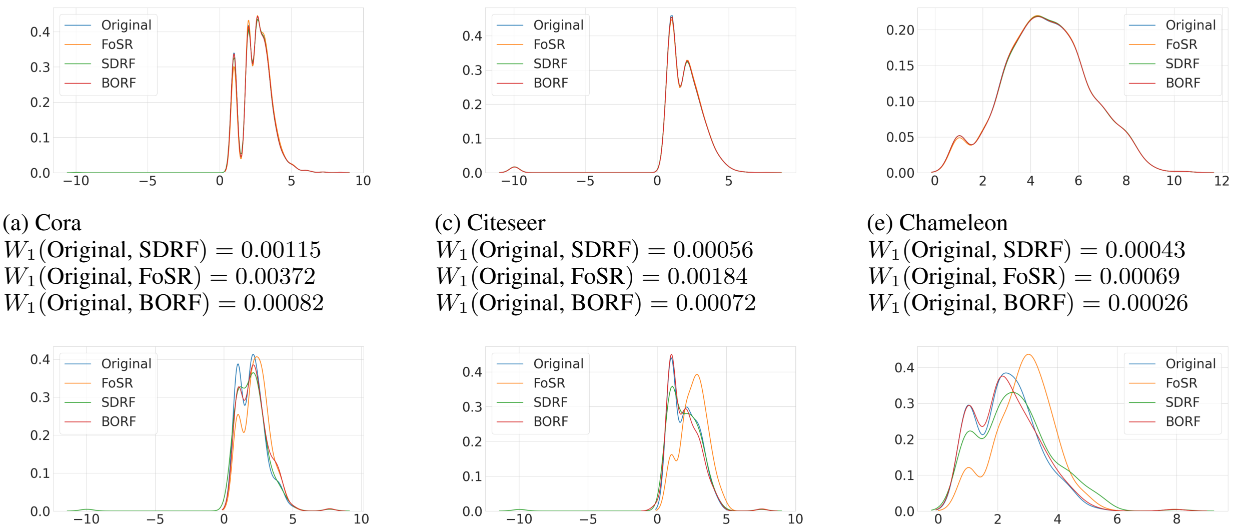
Table 13. Hyper-parameter settings of SDRF, FoSR and BORF for GCN and GIN tested on the Peptides-func graph classification dataset.

LAYER TYPE	SDRF	FOSR	BORF
GCN	100 ITERATIONS - $C^+ = 0$	25 ITERATIONS	4 BATCHES - ADD 30 - REMOVE 20
GIN	50 ITERATIONS - $C^+ = 0$	150 ITERATIONS	2 BATCHES - ADD 40 - REMOVE 10

overall.

E.3. Graph topology changes by different rewiring algorithms

In this section, we provide empirical data comparing the change in graph topology enacted by the rewiring algorithms SDRF, FoSR, and BORF. Similar to (Topping et al., 2022), for each dataset, we record the node degrees' base 2 logarithm distribution before and after applying BORF, SDRF and FoSR rewiring settings tuned for GCN as documented in Section D.1. We visualize the difference by utilizing the kernel density functions. The L_1 Wasserstein distance between the kernel density functions of these rewiring methods and the original node degree density is also calculated to measure the extent of topological change after rewiring.



(a) Cora
W_1(Original, SDRF) $= 0.00115$
W_1(Original, FoSR) $= 0.00372$
W_1(Original, BORF) $= 0.00082$

(c) Citeseer
W_1(Original, SDRF) $= 0.00056$
W_1(Original, FoSR) $= 0.00184$
W_1(Original, BORF) $= 0.00072$

(e) Chameleon
W_1(Original, SDRF) $= 0.00043$
W_1(Original, FoSR) $= 0.00069$
W_1(Original, BORF) $= 0.00026$

(b) Texas
W_1(Original, SDRF) $= 0.00831$
W_1(Original, FoSR) $= 0.01024$
W_1(Original, BORF) $= 0.00971$

(d) Cornell
W_1(Original, SDRF) $= 0.00857$
W_1(Original, FoSR) $= 0.01534$
W_1(Original, BORF) $= 0.00641$

(f) Wisconsin
W_1(Original, SDRF) $= 0.00836$
W_1(Original, FoSR) $= 0.01723$
W_1(Original, BORF) $= 0.00451$

Figure 5. Kernel density functions of graph degrees' base 2 logarithm before and after rewiring using BORF, SDRF and FoSR.

Our data shows that both BORF and SDRF change the graph topology only minimally according to this metric, while rewiring with FoSR can have a much more drastic effect.

F. Dataset Statistics

We provide a summary of statistics of all datasets used in Table 14 and Table 15. We also report the mean and standard deviation of the Ollivier Ricci curvature for each dataset. On node classification tasks, this is exactly the statistics of the set of edge curvature values. On graph classification tasks, this is the statistics of the mean curvature value of all graphs within the dataset.

Table 14. Statistics of node classification datasets.

	CORNELL	TEXAS	WISCONSIN	CORA	CITESEER	CHAMELEON
#NODES	140	135	184	2485	2120	832
#EDGES	219	251	362	5069	3679	12355
#FEATURES	1703	1703	1703	1433	3703	2323
#CLASSES	5	5	5	7	6	5
DIRECTED	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE
ORC MEAN	-0.39	-0.24	-0.59	-0.19	-0.31	0.64
ORC STD	0.52	0.45	0.71	0.68	0.78	0.58

Table 15. Statistics of graph classification datasets.

	ENZYMES	IMDB	MUTAG	PROTEINS
#GRAPHS	600	1000	188	1113
#NODES	2-126	12-136	10 - 28	4-620
#EDGES	2 - 298	52 - 2498	20 - 66	10 - 2098
AVG #NODES	32.63	19.77	17.93	39.06
AVG #EDGES	124.27	193.062	39.58	145.63
#CLASSES	6	2	2	2
DIRECTED	FALSE	FALSE	FALSE	FALSE
ORC MEAN	0.13	0.58	-0.27	0.17
ORC STD	0.15	0.19	0.05	0.20

Table 16. Server specifications for conducting all experiments.

SERVER ID	COMPONENTS	SPECIFICATIONS
1	ARCHITECTURE	X86_64
	OS	UBUNTU 20.04.5 LTS X86_64
	CPU	INTEL I7-10700KF (16) @ 5.100GHZ
	GPU	NVIDIA GEFORCE RTX 2080 TI REV. A
	RAM	12GB
2	ARCHITECTURE	X86_64
	OS	UBUNTU 20.04.5 LTS X86_64
	CPU	AMD EPYC 7742 64-CORE
	GPU	NVIDIA A100 TENSOR CORE
	RAM	40GB

G. Hardware Specifications and Libraries

All experiments were implemented in Python using PyTorch (Paszke et al., 2019), Numpy (Harris et al., 2020), PyG (PyTorch Geometric) (Fey & Lenssen, 2019), POT (Python Optimal Transport) (Flamary et al., 2021) with figures created using TikZ (Tantau, 2023). PyTorch, PyG and NumPy are made available under the BSD license, POT under MIT license, and TikZ under the GNU General Public license.

We conducted our experiments on two local servers with the specifications laid out in Table 16.