Mitigating Over-Smoothing and Over-Squashing using Augmentations of Forman-Ricci Curvature

Lukas Fesser
Harvard University
lukas_fesser@fas.harvard.edu

Melanie Weber
Harvard University
mweber@seas.harvard.edu

Abstract

While Graph Neural Networks (GNNs) have been successfully leveraged for learning on graph-structured data across domains, several potential pitfalls have been described recently. Those include the inability to accurately leverage information encoded in long-range connections (*over-squashing*), as well as difficulties distinguishing the learned representations of nearby nodes with growing network depth (*over-smoothing*). An effective way to characterize both effects is discrete curvature: Long-range connections that underlie over-squashing effects have low curvature, whereas edges that contribute to over-smoothing have high curvature. This observation has given rise to *rewiring* techniques, which add or remove edges to mitigate over-smoothing and over-squashing. Several rewiring approaches utilizing graph characteristics, such as curvature or the spectrum of the graph Laplacian, have been proposed. However, existing methods, especially those based on curvature, often require expensive subroutines and careful hyperparameter tuning, which limits their applicability to large-scale graphs. Here we propose a rewiring technique based on Augmented Forman-Ricci curvature (AFRC), a scalable curvature notation, which can be computed in linear time. We prove that AFRC effectively characterizes over-smoothing and over-squashing effects in message-passing GNNs. We complement our theoretical results with experiments, which demonstrate that the proposed approach achieves state-of-the-art performance while significantly reducing the computational cost in comparison with other methods. Utilizing fundamental properties of discrete curvature, we propose effective heuristics for hyperparameters in curvature-based rewiring, which avoids expensive hyperparameter searches, further improving the scalability of the proposed approach.

1 Introduction

Graph-structured data is ubiquitous in data science and machine learning applications across domains. Message-passing Graph Neural Networks (GNNs) have emerged as a powerful architecture for Deep Learning on graph-structured data, leading to many recent success stories in a wide range of disciplines, including biochemistry [11], drug discovery [40], recommender systems [37] and particle physics [29]. However, recent literature has uncovered limitations in the representation power of GNNs [38], many of which stem from or are amplified by the inability of message-passing graph neural networks to accurately leverage information encoded in long-range connections (*over-squashing* [1]), as well as difficulties distinguishing the learned representations of nearby nodes with growing network depth (*over-smoothing* [18]). As a result, there has been a surge of interest in characterizing over-squashing and over-smoothing mathematically and in developing tools for mitigating both effects.

Among the two, over-squashing has received the widest attention, driven by the importance of leveraging information encoded in long-range connections in both node- and graph-level tasks. Over-smoothing has been observed to impact, in particular, the performance of GNNs on node-level

Approach	O-Smo	O-Squ	Complexity	Hyperparams
SDRF [32]	✗	✓	$O(m\, d_{\max}^2)$	grid-search
RLEF [2]	✗	✓	$O\big(n^2 \sqrt{\log(n)}\, d_{\max}\big)$	grid-search
FoSR [16]	✗	✓	$O(n^2)$	grid-search
BORF [22]	✓	✓	$O(m\, d_{\max}^3)$	grid-search
GTR [3]	✗	✓	$O\big(m \text{ poly} \log n + n^2 \text{poly} \log n\big)$	grid-search
AFR-3 (this paper)	✓	✓	$O(m\, d_{\max})$	**heuristic**

Table 1: Comparison of state-of-the art rewiring approaches for mitigating over-squashing (O-Squ) and over-smoothing (O-Smo) regarding complexity and choice of hyperparameters (n denoting the number of vertices, m the number of edges, $d_{\max}$ the maximal node degree).

tasks. Characterizations of over-squashing and over-smoothing frequently utilize tools from Discrete Geometry, such as the spectrum of the Graph Laplacian [2, 3, 16] or discrete Ricci curvature [22, 32]. Discrete curvature has been linked previously to graph topology and "information flow" in graphs, which has given rise to several applications in network analysis and machine learning [9, 23, 30, 31, 35]. Classical Ricci curvature characterizes local volume growth rates on manifolds (*geodesic dispersion*); analogous notions in discrete spaces were introduced by Ollivier [24], Forman [10] and Maas and Erbar [8], among others.

Previous work on characterizing over-squashing and over-smoothing with discrete Ricci curvature has utilized Ollivier's notion [22] (short ORC), as well as a variation of Forman's notion [32] (FRC). Utilizing ORC is a natural approach, thanks to fundamental relations to the spectrum of the Graph Laplacian and other fundamental graph characteristics [15], which are known to characterize information flow on graphs. However, computing ORC requires solving an optimal transport problem for each edge in the graph, making a characterization of over-squashing and over-smoothing effects via ORC prohibitively expensive on large-scale graphs. In contrast, FRC can be computed efficiently even on massive graphs due to its simple combinatorial form. Recent work on *augmentations* of Forman's curvature, which incorporate higher-order structural information (e.g., cycles) into the curvature computation, has demonstrated their efficiency in graph-based learning, coming close to the accuracy reached by ORC-based methods at a fraction of the computational cost [9, 31]. In this work, we will demonstrate that augmented Forman curvature (AFRC) allows for an efficient characterization of over-squashing and over-smoothing. We relate both effects to AFRC theoretically, utilizing novel bounds on AFRC.

Besides characterizing over-squashing and over-smoothing, much recent interest has been dedicated to mitigating both effects in GNNs. *Graph rewiring*, which adds and removes edges to improve the information flow through the network, has emerged as a promising tool for improving the quality of the learned node embeddings and their utility in downstream tasks. Building on characterizations of over-squashing (and, to some extend, over-smoothing) discussed above, several rewiring techniques have been proposed [2, 3, 16, 22, 32]. Rewiring is integrated into GNN training as a preprocessing step, which alters the graph topology before learning node embeddings. It has been shown to improve accuracy on node- and graph-level tasks, specifically in graphs with long-range connections. Here, we argue that effective rewiring should only add a small computational overhead to merit its integration into GNN training protocols. Hence, the computational complexity of the corresponding preprocessing step, in tandem with the potential need for and cost of hyperparameter searches, is an important consideration. We introduce a graph rewiring technique based on augmentations of Forman's curvature (AFR-k), which can be computed in linear time and which comes with an efficiently computable heuristic for automating hyperparameter choices, avoiding the need for costly hyperparameter searches. Through computational experiments, we demonstrate that AFR-k has comparable or superior performance compared to state-of-the-art approaches, while being more computationally efficient.

Related Work. A growing body of literature considers the representational power of GNNs [20, 38] and related structural effects [1, 4, 6, 18, 25, 28]. This includes in particular challenges in leveraging information encoded in long-range connections (*over-squashing* [1]), as well as difficulties distinguishing representations of nearby nodes with growing network depth (*over-smoothing* [18]). Rewiring has emerged as a popular tool for mitigating over-squashing and over-smoothing. Methods based on the spectrum of the graph Laplacian [16], effective resistance [3], expander graphs [2, 5] and discrete curvature [22, 32] have been proposed (see Tab. 1). A connection between over-squashing and discrete curvature was first established in [32]. To the best of our knowledge, augmented Forman curvature has not been considered as a means for studying and mitigating over-squashing and over-smoothing in previous literature. Notably, the *balanced Forman curvature* studied in [32] is distinct from the notion considered in this paper. Forman's curvature, as well as discrete curvature more generally, has previously been used in graph-based machine learning, including for unsupervised node clustering (community detection) [9, 23, 30, 31], graph coarsening [36] and in representation learning [19, 34].

Contributions. Our main contributions are as follows:

- We prove that augmentations of Forman's Ricci curvature (AFRC) allow for an effective characterization of over-smoothing and over-squashing effects in message-passing GNNs.
- We propose an AFRC-based graph rewiring approach (AFR-k, Alg. 1), which is both scalable and performs competitively compared to state-of-the-art rewiring approaches on node- and graph-level tasks. Importantly, AFR-3 can be computed in linear time, allowing for application of the approach to large-scale graphs.
- We introduce a novel heuristic for choosing how many edges to add (or remove) in curvature-based rewiring techniques grounded in community detection. This heuristic allows for an effective implementation of our proposed approach, avoiding costly hyperparameter searches. Notably, the heuristic applies also to existing curvature-based rewiring approaches and hence may be of independent interest.

2 Background and Notation

2.1 Message-Passing Graph Neural Networks

Many popular architectures for Graph Machine Learning utilize the *Message-Passing paradigm* [12, 14]. Message-passing Graph Neural Networks (short *GNNs*) iteratively compute node representations as a function of the representations of their neighbors with node attributes in the input graph determining the node representation at initialization. We can view each iteration as a *GNN layer*; the number of iterations performed to compute the final representation can be thought of as the *depth* of the GNN. Formally, if we let $\mathbf{X}_u^k$ denote the features of node u at layer k, then a general formulation for a message-passing Graph Neural Network is

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

Here, ϕ_k denotes an update function, $\bigoplus$ an aggregation function, and ψ_k a message function. $\tilde{\mathcal{N}}_u = \mathcal{N}_u \cup \{u\}$ is the extended neighborhood of u. Widely used examples of this general formulation include GCN [17], GIN [38] and GAT [33].

2.2 Discrete Ricci Curvature on Graphs

Ricci curvature is a classical tool in Differential Geometry, which establishes a connection between the geometry of the manifold and local volume growth. Discrete notions of curvature have been proposed via *curvature analogies*, i.e., notions that maintain classical relationships with other geometric characteristics. Forman [10] introduced a notion of curvature on CW complexes, which allows for a discretization of a crucial relationship between Ricci curvature and Laplacians, the Bochner-Weizenböck equation. Here, we utilize an edge-level version of Forman's curvature, which also allows for evaluating curvature contributions of higher-order structures. Specifically, we consider notions that evaluate higher-order information encoded in

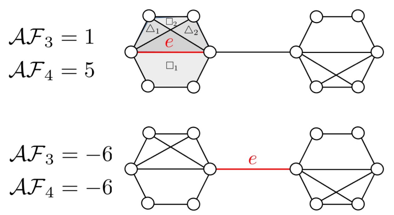

Figure 1: Augmented Forman-Ricci Curvature.

cycles of order $\leq k$ (denoted as $\mathcal{AF}_k$), focusing on the cases $k = 3$ and $k = 4$:

$$\mathcal{AF}_3(u, v) = 4 - \deg(u) - \deg(v) + 3\triangle(u, v) \tag{1}$$

$$\mathcal{AF}_4(u, v) = 4 - \deg(u) - \deg(v) + 3\triangle(u, v) + 2\square(u, v) \ , \tag{2}$$

where $\triangle(u, v)$ and $\square(u, v)$ denote the number of triangles and quadrangles containing the edge (u, v) (see Fig. 1 for an example). The derivation of those notions follows directly from [10] and can be found, e.g., in [31].

2.3 Over-squashing and Over-smoothing

Oversquashing. It has been observed that bottlenecks in the information flow between distant nodes form as the number of layers in a GNN increases [1]. The resulting information loss can significantly decrease the effectiveness of message-passing and reduce the utility of the learned node representations in node- and graph-level tasks. This effect is particularly pronounced in long-range connections between distant nodes, such as edges that connect distinct clusters in the graph. Such edges are characterized by low (negative) Ricci curvature in the sense of Ollivier (ORC), giving rise to the curvature-based analysis of over-squashing [22, 32]. We will show below, that this holds also for AFRC.

Oversmoothing. At the same time, increasing the number of layers can introduce "shortcuts" between communities, induced by the representations of dissimilar nodes (e.g., nodes belonging to different clusters) becoming indistinguishable under the information flow induced by message-passing. First described by [18], this effect is known to impact node-level tasks, such as node clustering. It has been previously shown that oversmoothing arises in positively curved regions of the graph in the sense of Ollivier [22]. We will show below, that this holds also for AFRC.

Rewiring. Graph rewiring describes tools that alter the graph topology by adding or removing edges to improve the information flow in the network. A taxonomy of state-of-the art rewiring techniques and the AFRC-based approach proposed in this paper with respect to the complexity of the resulting preprocessing step and the choice of hyperparameters can be found in Table 1.

3 Characterizing Over-smoothing and Over-squashing with $\mathcal{AF}_4$

Before we can relate the augmented Forman-Ricci curvature of an edge to over-smoothing and over-squashing in GNNs, we require upper and lower bounds on $\mathcal{AF}_3(u, v)$ and $\mathcal{AF}_4(u, v)$. Unlike the Ollivier-Ricci curvature, the augmentations of the Forman-Ricci curvature considered in this paper are not generally bounded independently of the underlying graph. We can prove the following for any (undirected and unweighted) graph $G = (V, E)$:

Theorem 3.1. For $(u, v) \in E$, let $m = \deg(u) \geq \deg(v) = n$. Then for $\mathcal{AF}_3$ and $\mathcal{AF}_4$, we have

$$\begin{aligned} 4 - m - n &\leq \mathcal{AF}_3(u, v) \leq n + 1 \\ 4 - m - n &\leq \mathcal{AF}_4(u, v) \leq 2mn - 3n + 3 \end{aligned} \tag{3}$$

Note that these edge-specific bounds can easily be extended to graph-level bounds by noting that $n \leq m \leq |V| - 1$. All other results in this section can be extended to graph-level bounds in a similar fashion. The proof for this result can be found in appendix A.1.1. Our theoretical results and their derivations are largely inspired by [22], although we require several adjustments such as the bounds above, since we are using the AFRC, not the ORC.

From the definition in the last section, it is clear that $\mathcal{AF}_4(u, v)$ characterizes how well connected the neighborhoods of u and v are. If many of the nodes in $\mathcal{N}_u$ are connected to v or nodes in $\mathcal{N}_v$, or vice-versa, then $\mathcal{AF}_4$ will be close to its upper bound. Conversely, if $\tilde{\mathcal{N}}_u$ and $\tilde{\mathcal{N}}_v$ have only minimal connections, then $\mathcal{AF}_4(u, v)$ will be close to its lower bound. In the second case, the existing connections will act as bottlenecks and will hinder the message-passing mechanism.

3.1 $\mathcal{AF}_4$ and Over-smoothing

Going back to the definition of the message-passing paradigm in the previous section, we note that at the k-th layer of a GNN, every node p sends the same message $\psi_k\left(\mathbf{X}_p^k\right)$ to each node u in its neighborhood. The GNN then aggregates these messages to update the features of node u. If $\mathcal{AF}_3$ or $\mathcal{AF}_4$ is very positive, then the neighborhoods of u and v are largely identical, so they receive nearly the same messages, and the difference between their features decreases. The following result makes this intuition for why over-smoothing happens more rigorous. For the proof, see appendix A.1.2.

Theorem 3.2. Consider an updating rule as defined in section 2.1 and suppose that $\mathcal{AF}_4(u,v) > 0$. For some k, assume the update function ϕ_k is L-Lipschitz, $|\mathbf{X}_p^k| \leq C$ is bounded for all $p \in \mathcal{N}(u) \cup \mathcal{N}(v)$, and the message passing function is bounded, i.e. $|\psi_k(\mathbf{x})| \leq M|\mathbf{x}|, \forall \mathbf{x}$. Let $\bigoplus$ be the sum operation. Then there exists a constant $H > 0$, depending on L, M, and C, such that

$$\left|\mathbf{X}_u^{k+1} - \mathbf{X}_v^{k+1}\right| \leq H(2mn - 3n + 3 - \mathcal{AF}_4(u,v)) \tag{4}$$

where $m = \deg(u) \geq \deg(v) = n > 1$.

We can prove analogous results for $\mathcal{AF}_3(u,v)$ when $\bigoplus$ is the mean or the sum operation (A.2.2). Note that Theorem 3.2 applies to most GNN architectures, with the exception of GAT, since GAT uses a learnable weighted mean operation for aggregation. Similar to previous results on the ORC, Theorem 3.2 shows that edges whose augmented Forman-Ricci curvature is close to their upper bound force local node features to become similar. If $\mathcal{AF}_4(e) \approx 2mn - 3n + 3$ for most edges e, then we can expect the node features to quickly converge to each other even if the GNN is relatively shallow. For regular graphs, we can extend our analysis and show that the difference between the features of neighboring nodes u and v decays exponentially fast. If in addition, we assume that the diameter of the graph is bounded, then this is true for any pair of nodes u, v.

Propositon 3.3. Consider an updating rule as before. Assume the graph is regular. Suppose there exists a constant δ, such that for all edges $(u,v) \in E$, $\mathcal{AF}_4(u,v) \geq \delta > 0$. For all k, assume the update functions ϕ_k are L-Lipschitz, $\bigoplus$ is the mean operation, $|\mathbf{X}_p^0| \leq C$ is bounded for all $p \in V$, and the message passing functions are bounded linear operators, i.e. $|\psi_k(\mathbf{x})| \leq M|\mathbf{x}|, \forall \mathbf{x}$. Then the following inequality holds for $k \geq 1$ and any neighboring vertices $u \sim v$

$$\left|\mathbf{X}_u^k - \mathbf{X}_v^k\right| \leq \frac{1}{3}C\left(\frac{3LM(2mn - 3n + 3 - \delta)}{n+1}\right)^k \tag{5}$$

If in addition, the diameter of G is bounded, i.e. $d = \max_{u \in V} \max_{v \in V} d(u,v) \leq D$, then for any $u, v \in V$, not necessarily connected, we have

$$\left|\mathbf{X}_u^k - \mathbf{X}_v^k\right| \leq \frac{1}{3}DC\left(\frac{3LM(2mn - 3n + 3 - \delta)}{n+1}\right)^k \tag{6}$$

For the proof, see appendix A.1.3. The above result gives us the aforementioned exponential decay in differences between node features, since for appropriately chosen $C_1, C_2 > 0$, we have

$$\sum_{(u,v)\in E} \left|\mathbf{X}_u^k - \mathbf{X}_v^k\right| \leq C_1 e^{-C_2 k} \tag{7}$$

Even if most real-word graphs possess negatively curved edges, we still expect an abundance of edges whose AFRC is close to its upper bound to lead to over-smoothing in GNNs.

3.2 $\mathcal{AF}_4$ and Over-squashing

In this subsection, we relate $\mathcal{AF}_4$ to the occurrence of bottlenecks in a graph, which in turn cause the over-squashing phenomenon. Message-passing between neighborhoods requires connections of the form (p,q) for $p \in \tilde{\mathcal{N}}_u \setminus \{v\}$ and $q \in \tilde{\mathcal{N}}_v \setminus \{u\}$. As the next result shows, the number of these connections can be upper-bounded using $\mathcal{AF}_4$. The proof for this can be found in appendix A.1.4.

Proposition 3.4. Let $(u,v) \in E$ and let $S \subset E$ be the set of all (p,q) with $p \in \tilde{N}_u \setminus \{v\}$ and $q \in \tilde{N}_v \setminus \{u\}$. Then

$$|S| \leq \frac{\mathcal{AF}_4(u,v) + \deg(u) + \deg(v) - 4}{2} \tag{8}$$

An immediate consequence of this result is that if $\mathcal{AF}_4$ is close to its lower bound $4 - \deg(u) - \deg(v)$, then the number of connections between $\tilde{\mathcal{N}}_u$ and $\tilde{\mathcal{N}}_v$ has to be small. Hence these edges will induce bottlenecks in G, which in turns causes over-squashing.

4 $\mathcal{AF}_3$- and $\mathcal{AF}_4$-based Rewiring

4.1 AFR-3 and AFR-4

Following our theoretical results in the last section, an $\mathcal{AF}_3$- or $\mathcal{AF}_4$-based rewiring algorithm should remove edges whose curvature is close to the upper bound to avoid over-smoothing, and add additional edges to the neighborhoods of those edges whose curvature is particularly close to the lower bound to mitigate over-squashing. In line with this intuition, we propose *AFR-k*, a novel algorithm for graph rewiring that, like the ORC-based BORF, can address over-smoothing and over-squashing simultaneously, while being significantly cheaper to run.

Algorithm 1 AFR-3

```
Require: A graph $G(V, E)$, heuristic (Bool), # edges to add $h$, # edges to remove $l$
  for $e \in E$ do
      compute $\mathcal{AF}_3(e)$
  end for
  Sort $e \in E$ by $\mathcal{AF}_3(e)$
  if heuristic == True then
      - Compute lower threshold $\Delta_L$. For each $(u, v) \in E$ with $\mathcal{AF}_3(u, v) < \Delta_L$,
      choose $w \in \mathcal{N}_u \setminus \mathcal{N}_v$ uniformly at random, add $(w, v)$ to $E$.
      - Compute upper threshold $\Delta_U$ and remove all edges $(u, v)$ with $\mathcal{AF}_3(u, v) > \Delta_U$
  else
      - Find $h$ edges with lowest $\mathcal{AF}_3$ values, for each of these edges,
      choose $w \in \mathcal{N}_u \setminus \mathcal{N}_v$ uniformly at random, add $(w, v)$ to $E$.
      - Find $k$ edges with highest $\mathcal{AF}_3$ values and remove them from $E$.
  end if
  return a rewired graph $G' = (V, E')$ with new edge set $E'$.
```

Assume that we want to add h edges to E to reduce over-squashing, and remove l edges from E to mitigate over-smoothing. Then for AFR-3 without heuristics for edge addition or removal, we first compute $\mathcal{AF}_3(e)$ for all $e \in E$. Then, we sort the the edges by curvature from lowest to highest. For each of the h edges with lowest curvature values, we choose $w \in \mathcal{N}_u \setminus \mathcal{N}_v$ uniformly at random if deg$(u) \geq$ deg(v) and add (w, v) to E. If no such w exists, we continue on to the next edge. Finally, we remove the l edges with the highest curvature values from E. We return the rewired graph $G' = (V, E')$, where E' is the rewired edge set. AFR-4 follows by analogy, only that we use $\mathcal{AF}_4$ instead of $\mathcal{AF}_3$. While $\mathcal{AF}_3$ is cheaper to run than $\mathcal{AF}_4$, both are more economical than their ORC-based cousin BORF, which is due to the next result.

Theorem 4.1. Computing the ORC scales as $O\left(|E|\, d_{\text{max}}^3\right)$, while computing $\mathcal{AF}_3$ scales as $O\left(|E|\, d_{\text{max}}\right)$, and computing $\mathcal{AF}_4$ scales as $O\left(|E|\, d_{\text{max}}^2\right)$.

Here, d_{max} is the highest degree of a node in G. For the proof, see appendix A.1.5. Note that we could run AFR-3 (AFR-4) as a batch-based algorithm like BORF. For N batches, we would recompute $\mathcal{AF}_3$ ($\mathcal{AF}_4$) N times, and for each batch we would identify the k edges with highest $\mathcal{AF}_3$ ($\mathcal{AF}_4$) values and the h edges with lowest $\mathcal{AF}_3$ ($\mathcal{AF}_4$) values and proceed with them as before. In practice, we find that this rarely improves performance. Results on this are presented in appendix A.7.4.

4.2 Heuristics for adding and removing Edges

A natural question to ask upon seeing the AFR algorithm is how many edges one should add or remove for a given graph G. For previous methods such as BORF, answering this question involved costly hyperparameter tuning. In this subsection, we use results from community detection in network science to motivate heuristics to automatically determine h and k.

Curvature Thresholds. Suppose we are given a graph $G = (V, E)$ and a partition of the network into communities. As previously mentioned, edges within communities tend to have positive Ollivier-Ricci curvature, while edges between communities tend to have negative curvature values. Theoretical evidence for this observation for graphs with community structure has been given in [13, 23, 31].

[30] use this observation to propose a community detection algorithm: remove the most negatively curved edge e from G, recompute the ORC for all edges sharing a node with e and repeat these two

steps until all negatively curved edges have been removed. The connected components in the resulting graph G' are the communities. For $\mathcal{AF}_3$ or $\mathcal{AF}_4$, a natural threshold to distinguish between inter- and intra-community edges is missing, so [9] propose to use a Gaussian mixture model with two modes. They fit two normal distributions $\mathcal{N}_1(\mu_1, \sigma_1)$ and $\mathcal{N}_2(\mu_2, \sigma_2)$ with $\mu_1 > \mu_2$ to the curvature distribution and then determine the curvature threshold

$$\Delta_L = \frac{\sigma_2}{\sigma_1 + \sigma_2}\mu_1 + \frac{\sigma_1}{\sigma_1 + \sigma_2}\mu_2 \tag{9}$$

Using this as the threshold between inter- and intra-community edges, their $\mathcal{AF}_3$-based sequential community detection algorithm achieves competitive performance while being significantly cheaper to run than the ORC-based original. Based on this, we propose the following heuristic to determine the number h of edges to add for AFR.

Heuristic for edge addition. Instead of choosing the number h of edges to add by hand, we calculate the curvature threshold Δ_L as above. We then add a new edge for each edge e with $\mathcal{AF}_3(e) < \Delta_L$ ($\mathcal{AF}_4(e) < \Delta_L$) using the same procedure as before.

Heuristic for edge removal. To identify particularly positively curved edges which might lead to over-smoothing, we compute the upper curvature threshold Δ_U as

$$\Delta_U = \mu_1 + \sigma_1 \tag{10}$$

We remove all edges e with $\mathcal{AF}_3(e) > \Delta_U$ ($\mathcal{AF}_4(e) > \Delta_U$), i.e. all edges whose curvature is more than one standard deviation above the mean of the normal distribution with higher curvature values.

5 Experiments

In this section, we experimentally demonstrate the effectiveness and computational efficiency of our proposed AFR rewiring algorithm and of our heuristics for edge addition and removal. We first compare AFR against other rewiring alternatives on a variety of tasks, including node and graph classification, as well as long-range tasks. Details on the rewiring strategies we compare against can be found in appendix A.4. We also show that our heuristics allow AFR to achieve competitive performance on all tasks without costly hyperparameter tuning. Our code will be made publicly available upon publication.

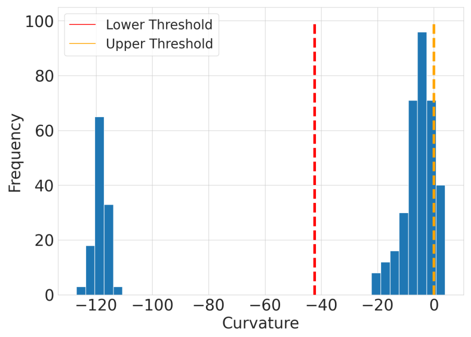

Figure 2: Upper and lower thresholds for the Wisconsin dataset ($\mathcal{AF}_3$).

Experimental details. Our experiments are designed as follows: for a given rewiring strategy, we first apply it as a preprocessing step to all graphs in the datasets considered. We then train a GNN on a part of the rewired graphs and evaluate its performance on a withheld set of test graphs. We use a train/val/test split of 50/25/25. As GNN architectures, we consider GCN and GIN. Settings and optimization hyper-parameters are held constant across tasks and baseline models for all rewiring methods, so we can rule out hyper-parameter tuning as a source of performance gain. When not using our heuristics, we obtain the settings for the individual rewiring option via hyperparameter tuning. The only hyperparameter choice which we do not optimize using grid search is τ in SDRF, which we set to ∞, in line with [22]. For all rewiring methods and hyperparameter choices, we record the test set accuracy of the settings with the highest validation accuracy. As there is a certain stochasticity involved, especially when training GNNs, we accumulate experimental results across 100 random trials. We report the mean test accuracy, along with the 95% confidence interval. Details on all data sets can be found in appendix A.9.

5.1 Results using hyperparameter search

Tables 2 and 3 present the results of our experiments for node and graph classification with hyperparameter tuning. The exact hyperparameter settings for each dataset can be found in appendix A.3, where we also present additional results using GIN as an architecture (A.7.1). For the experiments with GCNs presented here, AFR-3 and AFR-4 outperform all other rewiring strategies and the no-rewiring baseline on all node classification datasets and on four of our five graph classification datasets. We expect AFR-3, AFR-4, and BORF to attain generally higher accuracies than FoSR and SDRF, because unlike FoSR and SDRF, they can address over-smoothing by removing edges.

	GCN					
DATA SET	AFR-3	AFR-4	BORF	SDRF	FOSR	NONE
CORA	87.5 ± 0.5	$\mathbf{88.1 \pm 0.5}$	87.9 ± 0.7	86.4 ± 2.1	86.9 ± 2.0	86.6 ± 0.8
CITESEER	$\mathbf{74.4 \pm 1.0}$	73.3 ± 0.6	73.4 ± 0.6	72.6 ± 2.2	73.5 ± 2.0	71.7 ± 0.7
TEXAS	49.7 ± 0.5	$\mathbf{51.4 \pm 0.5}$	48.9 ± 0.5	43.6 ± 1.2	46.9 ± 1.2	44.1 ± 0.5
CORNELL	$\mathbf{49.7 \pm 3.4}$	48.9 ± 3.3	48.1 ± 2.9	43.1 ± 1.2	43.9 ± 1.1	46.8 ± 3.0
WISCON.	47.3 ± 2.4	$\mathbf{52.2 \pm 2.4}$	46.5 ± 2.6	47.1 ± 1.0	48.5 ± 1.0	44.6 ± 2.9
CHAMEL.	62.3 ± 0.9	$\mathbf{62.5 \pm 0.9}$	61.4 ± 0.9	59.5 ± 0.4	59.3 ± 1.9	59.1 ± 1.4
COCO	$\mathbf{10.1 \pm 1.1}$	9.8 ± 1.1	10.1 ± 1.2	8.5 ± 1.0	9.3 ± 1.4	7.8 ± 0.4
PASCAL	14.3 ± 1.5	$\mathbf{14.4 \pm 1.4}$	14.1 ± 1.1	11.7 ± 0.9	13.8 ± 1.3	10.4 ± 0.6

Table 2: Classification accuracies of GCN with AFR-3, AFR-4, BORF, SDRF, FoSR, or no rewiring strategy using best hyperparameters. Highest accuracies on any given dataset are highlighted in bold. We report F1 scores for the LGRB COCO and PASCAL datasets.

Comparing AFR-3 and AFR-4. In section 3, we used $\mathcal{AF}_4$ for our theoretical results due to it allowing us to better analyze over-smoothing. However, for many real-world datasets, $\mathcal{AF}_3$ and $\mathcal{AF}_4$ are highly correlated [9]. We find that this also holds true for the datasets considered here (see appendix A.9), which might explain why AFR-3, AFR-4, and BORF all achieve comparable accuracies, with AFR-3 and AFR-4 often performing marginally better. This also suggests that characterizing over-smoothing and over-squashing using $\mathcal{AF}_3$ or $\mathcal{AF}_4$ is sufficient. The computationally more expensive ORC is not needed. In practice, one would always use AFR-3 due to competitive performance and excellent scalability.

	GCN					
DATA SET	AFR-3	AFR-4	BORF	SDRF	FOSR	NONE
MUTAG	69.7 ± 2.1	68.7 ± 1.9	68.2 ± 2.4	65.1 ± 2.2	$\mathbf{70.0 \pm 2.2}$	62.7 ± 2.1
ENZYMES	25.9 ± 1.2	$\mathbf{26.3 \pm 1.2}$	26.0 ± 1.2	24.3 ± 1.2	24.9 ± 1.1	25.4 ± 1.3
IMDB	$\mathbf{50.4 \pm 0.9}$	49.8 ± 1.1	48.6 ± 0.9	48.6 ± 0.9	48.3 ± 0.9	48.1 ± 1.0
PROTEINS	$\mathbf{62.7 \pm 0.8}$	61.2 ± 0.9	61.5 ± 0.7	59.5 ± 0.8	59.3 ± 0.9	59.6 ± 0.9
PEPTIDES	$\mathbf{44.7 \pm 2.8}$	44.4 ± 2.8	43.9 ± 2.6	41.8 ± 1.5	44.3 ± 2.2	40.5 ± 2.1

Table 3: Classification accuracies of GCN with AFR-3, AFR-4, BORF, SDRF, FoSR, or no rewiring strategy using best hyperparameters. Highest accuracies on any given dataset are highlighted in bold.

5.2 Results using heuristics for edge addition and removal

Using the same datasets as before, we now test our heuristics for replacing hyperparameter tuning. We study the effects of using the thresholds proposed in section 4 on AFR-3, AFR-4, and BORF. Tables 4 and 5 present the accuracies attained by GCN and GIN architectures using our heuristics for finding upper and lower thresholds. Comparing these to the results in Tables 2 and 3, we see that our heuristics outperform hyperparameter tuning on four out of six node classification datasets, and are competitive on the other two. Similarly on the graph classification tasks, where our heuristics achieve superior performance on four out of five tasks.

For BORF, we use zero as the lower threshold to identify bottleneck edges. We conduct additional experiments in appendix A.7.7 which show that the lower thresholds which we get from fitting two normal distributions to the ORC distributions are in fact very close to zero on all datasets considered here. Further ablations that study the effects of our two heuristics individually can also be found in appendices A.7.2 and A.7.3.

Long-range tasks. We also evaluated AFR-3 and AFR-4 on the Peptides-func graph classification dataset and on the PascalVOC-SP and COCO-SP node classification datasets, which are part of the long-range tasks introduced by [7]. As Table 3 shows, AFR-3 and AFR-4 outperform all other rewiring methods on the graph classification task and significantly improve on the no-rewiring baseline. Our heuristics are also clearly still applicable, as they result in comparable performance

	GCN			GIN		
DATA SET	AFR-3	AFR-4	BORF	AFR-3	AFR-4	BORF
CORA	87.8 ± 0.7	87.9 ± 0.9	87.6 ± 0.7	77.9 ± 1.2	78.0 ± 1.1	78.4 ± 1.1
CITESEER	74.6 ± 0.7	74.7 ± 0.7	74.2 ± 0.8	65.1 ± 0.7	64.7 ± 0.6	64.6 ± 0.6
TEXAS	52.4 ± 3.3	48.4 ± 3.2	50.5 ± 2.8	68.7 ± 3.1	63.8 ± 3.8	62.3 ± 2.0
CORNELL	50.5 ± 4.3	46.2 ± 2.6	44.6 ± 1.9	51.9 ± 4.2	47.3 ± 2.3	48.4 ± 2.5
WISCON.	52.4 ± 2.6	49.0 ± 2.0	48.2 ± 3.0	58.4 ± 2.9	58.7 ± 2.3	53.5 ± 2.6
CHAMEL.	62.2 ± 1.2	62.1 ± 1.2	61.1 ± 1.4	67.1 ± 2.1	65.9 ± 2.2	66.3 ± 2.2
COCO	10.3 ± 1.3	9.6 ± 1.2	10.5 ± 1.2	13.1 ± 2.1	13.5 ± 2.2	13.6 ± 2.2
PASCAL	14.2 ± 1.5	14.3 ± 1.5	14.8 ± 1.1	16.0 ± 1.7	15.8 ± 1.7	16.4 ± 1.8

Table 4: Node classification accuracies of GCN and GIN with AFR-3, AFR-4, and BORF using our heuristics to avoid hyperparameter tuning.

	GCN			GIN		
DATA SET	AFR-3	AFR-4	BORF	AFR-3	AFR-4	BORF
MUTAG	71.4 ± 2.2	69.9 ± 2.6	68.5 ± 1.9	70.9 ± 2.7	73.4 ± 2.4	75.4 ± 2.8
ENZYMES	26.1 ± 1.0	25.5 ± 1.0	23.3 ± 1.2	37.6 ± 1.2	32.1 ± 1.4	31.9 ± 1.2
IMDB	50.1 ± 0.9	49.0 ± 0.9	49.4 ± 1.0	68.9 ± 1.1	67.8 ± 1.2	67.7 ± 1.5
PROTEINS	62.2 ± 0.8	61.2 ± 0.9	61.0 ± 0.9	73.0 ± 1.5	72.7 ± 1.3	72.3 ± 1.2
PEPTIDES	44.8 ± 2.8	43.6 ± 2.5	44.3 ± 2.8	49.2 ± 1.5	50.7 ± 1.6	50.1 ± 1.6

Table 5: Graph classification accuracies of GCN and GIN with AFR-3, AFR-4, and BORF using our heuristics to avoid hyperparameter tuning.

(Table 5). Our experiments with long-range node classification yield similar results, as can be seen in Tables 2 and 4. Additional experiments with the LRGB datasets using GIN can be found in A.7.1.

6 Discussion

In this paper we have introduced formal characterizations of over-squashing and over-smoothing effects using augementations of Forman's Ricci curvature, a simple and scalable notion of discrete curvature. Based on this characterization, we proposed a scalable graph rewiring approach, which exhibits performance comparable or superior to state-of-the-art rewiring approaches on node- and graph-level tasks. We further introduce an effective heuristic for hyperparameter choices in curvature-based graph rewiring, which removes the need to perform expensive hyperparameter searches.

There are several avenues for further investigation. We believe that the complexity of rewiring approaches merits careful consideration and should be evaluated in the context of expected performance gains in applications. This includes in particular the choice of hyperparameters, which for many state-of-the-art rewiring approaches requires expensive grid searches. While we propose an effective heuristic for curvature-based approaches (both existing and proposed herein), we believe that a broader study on transparent hyperparameter choices across rewiring approaches is merited. While performance gains in node- and graph-level tasks resulting from rewiring have been established empirically, a mathematical analysis is still largely lacking and an important direction for future work. Our proposed hyperparameter heuristic is linked to the topology and global geometric properties of the input graph. We believe that similar connections could be established for rewiring approaches that do not rely on curvature. Building on this, a systematic investigation of the suitability of different rewiring and corresponding hyperparameter choices dependent on graph topology would be a valuable direction for further study. Similarly, differences between the effectiveness of rewiring in homophilous vs. heterophilous graphs strike us as an important direction for future work.

7 Limitations

The present paper does not explicitly consider the impact of graph topology on the efficiency of AFR-k. Specifically, our proposed heuristic implicitly assumes that the graph has community structure, which is true in many, but not all, applications. Our experiments are restricted to one node- and graph-level task each. While this is in line with experiments presented in related works, a wider range of tasks would give a more complete picture.

References

[1] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=i80OPhOCVH2`. 1, 2, 4

[2] Pradeep Kr Banerjee, Kedar Karhadkar, Yu Guang Wang, Uri Alon, and Guido Montúfar. Oversquashing in gnns through the lens of information contraction and graph expansion. In *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–8. IEEE, 2022. 2

[3] Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding oversquashing in gnns through the lens of effective resistance. In *International Conference on Machine Learning*, pages 2528–2547. PMLR, 2023. 2

[4] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020. 2

[5] Andreea Deac, Marc Lackenby, and Petar Veličković. Expander graph propagation. In *Proceedings of the First Learning on Graphs Conference*, 2022. 2

[6] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M. Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7865–7885. PMLR, 23–29 Jul 2023. 2

[7] Vijay Prakash Dwivedi, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL `https://openreview.net/forum?id=in7XC5RcjEn`. 8, 27

[8] M. Erbar and J. Maas. Ricci curvature of finite markov chains via convexity of the entropy. *Archive for Rational Mechanics and Analysis*, 206:997–1038, 2012. 2

[9] Lukas Fesser, Sergio Serrano de Haro Iváñez, Karel Devriendt, Melanie Weber, and Renaud Lambiotte. Augmentations of forman's ricci curvature and their applications in community detection. *arXiv preprint arXiv:2306.06474*, 2023. 2, 3, 7, 8

[10] Robin Forman. Bochner's Method for Cell Complexes and Combinatorial Ricci Curvature. volume 29, pages 323–374, 2003. 2, 3, 4, 16

[11] Vladimir Gligorijević, P Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C Taylor, Ian M Fisk, Hera Vlamakis, et al. Structure-based protein function prediction using graph convolutional networks. *Nature communications*, 12(1):3168, 2021. 1

[12] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks*, volume 2, pages 729–734, 2005. 3

[13] Adam Gosztolai and Alexis Arnaudon. Unfolding the multiscale structure of networks with dynamical Ollivier-Ricci curvature. *Nature Communications*, 12(1), December 2021. 6

[14] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *NIPS*, pages 1024–1034, 2017. 3

[15] J. Jost and S. Liu. Ollivier's Ricci curvature, local clustering and curvature-dimension inequalities on graphs. *Discrete & Computational Geometry*, 51(2):300–322, 2014. 2

[16] Kedar Karhadkar, Pradeep Kr Banerjee, and Guido Montúfar. Fosr: First-order spectral rewiring for addressing oversquashing in gnns. *arXiv preprint arXiv:2210.11790*, 2022. 2

[17] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017. 3

[18] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 1, 2, 4

[19] Shane Lubold, Arun G. Chandrasekhar, and Tyler H. McCormick. Identifying the latent space geometry of network models through analysis of curvature, May 2022. arXiv:2012.10559 [cs, math, stat]. 3

[20] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019. 2

[21] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *CoRR*, abs/2007.08663, 2020. URL `https://arxiv.org/abs/2007.08663`. 27

[22] Khang Nguyen, Nong Minh Hieu, Vinh Duc Nguyen, Nhat Ho, Stanley Osher, and Tan Minh Nguyen. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *International Conference on Machine Learning*, pages 25956–25979. PMLR, 2023. 2, 3, 4, 7, 19, 20

[23] Chien-Chun Ni, Yu-Yao Lin, Feng Luo, and Jie Gao. Community detection on networks with ricci flow. *Scientific reports*, 9(1):1–12, 2019. 2, 3, 6

[24] Y. Ollivier. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 256(3):810–864, 2009. 2, 16

[25] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020. 2

[26] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *CoRR*, abs/2002.05287, 2020. URL `https://arxiv.org/abs/2002.05287`. 27

[27] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *CoRR*, abs/1909.13021, 2019. URL `http://arxiv.org/abs/1909.13021`. 27

[28] T Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-coupled oscillator networks. In *International Conference on Machine Learning*, pages 18888–18909. PMLR, 2022. 2

[29] Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, 2020. 1

[30] Jayson Sia, Edmond Jonckheere, and Paul Bogdan. Ollivier-ricci curvature-based method to community detection in complex networks. *Scientific reports*, 9(1):1–12, 2019. 2, 3, 6

[31] Yu Tian, Zachary Lubberts, and Melanie Weber. Curvature-based clustering on graphs. *arXiv preprint arXiv:2307.10155*, 2023. 2, 3, 4, 6

[32] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022. 2, 3, 4

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *ICLR*, 2018. 3

[34] Melanie Weber. Neighborhood growth determines geometric priors for relational representation learning. In *International Conference on Artificial Intelligence and Statistics*, volume 108, pages 266–276, 2020. 3

[35] Melanie Weber, Emil Saucan, and Jürgen Jost. Characterizing complex networks with forman-ricci curvature and associated geometric flows. *Journal of Complex Networks*, 5(4):527–550, 2017. 2

[36] Melanie Weber, Johannes Stelzer, Emil Saucan, Alexander Naitsat, Gabriele Lohmann, and Jürgen Jost. Curvature-based methods for brain network analysis. *arXiv:1707.00180*, 2017. 3

[37] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022. 1

[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 1, 2, 3

[39] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *CoRR*, abs/1603.08861, 2016. URL `http://arxiv.org/abs/1603.08861`. 27

[40] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018. 1

A Appendix

Contents

A.1 Proofs of theoretical results in the main text

A.1.1 Theorem 3.1

Proof: Let $G = (V, E)$ be an unweighted, undirected graph, let $(u, v) \in E$ and let t and q denote the number of triangles and quadrangles, respectively, containing the edge (u, v). The lower bound in both cases ($\mathcal{AF}_3$ and $\mathcal{AF}_4$) follows straight from the definition and the fact that t and q are non-negative by definition. The lower bounds are attained when there are no triangles (resp. no triangles and quadrangles) containing (u, v) in E.

For the upper bound on $\mathcal{AF}_3$, note that $t \leq n - 1$ (and $n \geq 1$ by assumption). Also note that the only way to increase $\mathcal{AF}_3(u, v)$ is to add triangles, which increases $\mathcal{AF}_3(u, v)$ by one (as we also increase the degrees of u and v by one each). Hence

$$\begin{aligned} \mathcal{AF}_3(u,v) &= 4 - m - n + 3t \leq 4 - m - n + 3(n-1) \\ &= 2n - m + 1 \leq n + 1 \end{aligned} \tag{11}$$

The upper bound is attained when $t = n - 1$ and $n = m$. The upper bound on $\mathcal{AF}_3(u, v)$ follows from the observation that $q \leq (m - 1)(n - 1)$, so

$$\begin{aligned} \mathcal{AF}_4(u,v) &= 4 - m - n + 3t + 2q \leq 4 - m - n + 3(n-1) + 2(m-1)(n-1) \\ &= 2n - m + 1 + 2mn - 2n - 2m + 2 = 2mn - 3m + 3 \\ &\leq 2mn - 3n + 3 \end{aligned} \tag{12}$$

Note that the bound is attained when $t = n + 1, q = (m - 1)(n - 1)$, and $m = n$.

□

A.1.2 Theorem 3.2

Proof: ϕ_k is L-Lipschitz, so

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

By **Lemma 1** and using that $t \leq m - 1$ and $q \leq (m - 1)(n - 1)$, we have after some algebra

$$\begin{aligned} \left|\tilde{N}_u\Delta\tilde{N}_v\right| &\leq n + m - 2t - 2 \leq 3m - 3t - 3 \\ &\leq (2mn - 2n - 2m + 2) + 3m - 3 - 3t - 2q \\ &= 2mn - 3n + 3 - (4 - m - n + 3t + 2q) \\ &= 2mn - 3n + 3 - \mathcal{AF}_4(u,v) \end{aligned} \tag{14}$$

Finally, using the above and the assumptions that ψ_k is bounded and that $|\mathbf{X}_p^k| \leq C$ for all $p \in \mathcal{N}(u) \cup \mathcal{N}(v)$,

$$L\sum_{p\in\tilde{N}_u\Delta\tilde{N}_v} \left|\psi_k\left(\mathbf{X}_p^k\right)\right| \leq LCM\left(2mn - 3n + 3 - \mathcal{AF}_4(u,v)\right) \tag{15}$$

This proves the statement with $H := LCM$.

□

A.1.3 Proposition 3.3

Proof: Since $\mathcal{AF}_4(u,v) \in \mathbb{N}$ for all $(u,v) \in E$, we may assume that $\delta \in \mathbb{N}$. We proceed by induction. For all edges $u \sim v$, **Lemma 1** tells us that

$$\begin{aligned} \left|\tilde{N}_u \Delta \tilde{N}_v\right| &\leq \deg(u) + \deg(v) - 2t - 2 \leq 3m - 3t - 3 \\ &\leq 2mn - 3n + 3 - \mathcal{AF}_4(u,v) \leq 2mn - 3n + 3 - \delta \end{aligned} \tag{16}$$

The base case $k = 1$ follows, since

$$\begin{aligned} \left|\mathbf{X}_u^1 - \mathbf{X}_v^1\right| &= \left|\phi_1\left(\frac{1}{n+1}\sum_{p\in\tilde{N}_u}\psi\left(\mathbf{X}_p\right)\right) - \phi_1\left(\frac{1}{n+1}\sum_{q\in\tilde{N}_v}\psi\left(\mathbf{X}_q\right)\right)\right| \\ &\leq L\left|\frac{1}{n+1}\sum_{p\in\tilde{N}_u}\psi\left(\mathbf{X}_p\right) - \frac{1}{n+1}\sum_{q\in\tilde{N}_v}\psi\left(\mathbf{X}_q\right)\right| \\ &= \frac{L}{n+1}\left|\sum_{p\in\tilde{N}_u\setminus\tilde{N}_v}\psi\left(\mathbf{X}_p\right) - \sum_{q\in\tilde{N}_v\setminus\tilde{N}_u}\psi\left(\mathbf{X}_q\right)\right| \\ &\leq \frac{L}{n+1}\sum_{p\in\tilde{N}_u\Delta\tilde{N}_v}\left|\psi\left(\mathbf{X}_p\right)\right| \\ &\leq \left(\frac{2mn-3n+3-\delta}{n+1}\right)LCM \end{aligned} \tag{17}$$

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

$$\begin{aligned} \left|\mathbf{X}_u^k - \mathbf{X}_v^k\right| &\leq \frac{L}{n+1}\left|\sum_{p\in\tilde{N}_u}\psi_k\left(\mathbf{X}_p^k\right) - \sum_{q\in\tilde{N}_v}\psi_k\left(\mathbf{X}_q^k\right)\right| \\ &= \frac{L}{n+1}\left|\sum_{p\in\tilde{N}_u\setminus\tilde{N}_v}\psi_K\left(\mathbf{X}_p^k\right) - \sum_{q\in\tilde{N}_v\setminus\tilde{N}_u}\psi_k\left(\mathbf{X}_q^k\right)\right| \\ &= \frac{L}{n+1}\left|\psi_k\left(\sum_{p\in\tilde{N}_u\setminus\tilde{N}_v}\mathbf{X}_p^k - \sum_{q\in\tilde{N}_v\setminus\tilde{N}_u}\mathbf{X}_q^k\right)\right| \\ &\leq \frac{LM}{n+1}\left|\sum_{p\in\tilde{N}_u\setminus\tilde{N}_v}\mathbf{X}_p^k - \sum_{q\in\tilde{N}_v\setminus\tilde{N}_u}\mathbf{X}_q^k\right| \end{aligned} \tag{18}$$

For each $p \in \tilde{N}_u \setminus \tilde{N}_v$, match it with one and only one $q \in \tilde{N}_v \setminus \tilde{N}_u$. For any node pair, they are connected by a node 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 $\frac{1}{3}C\left(\frac{3LM}{n+1}(2mn-3n+3-\delta)\right)^k$. Hence, we have

$$\left|\mathbf{X}_p^k - \mathbf{X}_q^k\right| \leq C\left(\frac{3LM(2mn-3n+3-\delta)}{n+1}\right)^k \tag{19}$$

Substitute this back into the above and note that there are at most $2mn - 3n + 3 - \delta$ pairs to get

$$\begin{aligned} \left|\mathbf{X}_u^{k+1} - \mathbf{X}_v^{k+1}\right| &\leq \frac{LM}{n+1} \sum_{(p,q)} \left|\mathbf{X}_p^k - \mathbf{X}_q^k\right| \\ &\leq \frac{LM}{n+1} (2mn - 3n + 3 - \delta)\, C \left(\frac{3LM(2mn - 3n + 3 - \delta)}{n+1}\right)^k \\ &= \frac{1}{3} C \left(\frac{3LM(2mn - 3n + 3 - \delta)}{n+1}\right)^{k+1} \end{aligned} \tag{20}$$

This proves that the desired inequality holds for all $k \geq 1$ and $u \sim v$.

□

A.1.4 Proposition 3.4

Proof: Denote the number of triangles containing (u, v) by $\triangle(u, v) =: t$ and the number of quadrangles by $\square(u, v) =: q$. Then $|S| = t + q$. By definition,

$$\mathcal{AF}_4(u, v) = 4 - \text{deg}(u) - \text{deg}(v) + 3t + 2q = 4 - \text{deg}(u) - \text{deg}(v) + t + 2|S|. \tag{21}$$

Rewriting this, we get

$$|S| = \frac{\mathcal{AF}_4(u, v) + \text{deg}(u) + \text{deg}(v) - 4 - t}{2}, \tag{22}$$

but $t \geq 0$, so the inequality follows.

□

A.1.5 Theorem 4.1

Proof: The complexity of computing discrete curbature in the sense of Ollivier and Forman follows directly from the definitions [10, 24]; we will briefly recall it below for completeness. Computing ORC involves the computation of the W_1-distance between measures defined on the neighborhoods of the nodes adjacent to the edge, which can be done in $O(d_{max}^3)$ (via the Hungarian algorithm), where $d_{\max}$ is the maximal node degree. For computing $\mathcal{AF}_3$, we note that the costliest operation involved is counting the number of triangles containing a given edge $(u, v) \in E$. We can do this by determining the neighborhoods $\mathcal{N}_u$ and $\mathcal{N}_v$ of u and v, respectively, which scales linearly in $\max\{d_u, d_v\} \leq d_{\max}$, before taking the intersection of the two neighborhoods, again in linear time. Doing this for every edges shows that computing $\mathcal{AF}_3$ indeed scales as $O(|E|\, d_{\max})$. For $\mathcal{AF}_4$, note that counting the 4-cycles containing (u, v) is the costliest operation involved. For a given edge, this scales quadratically in $\max\{d_u, d_v\} \leq d_{\max}$: for each $w_1 \in \mathcal{N}_u$ and for each $w_2 \in \mathcal{N}_v \setminus \mathcal{N}_u$, we check whether $(w_1, w_2) \in E$.

□

A.2 Additional theoretical results

A.2.1 Lemma 1

Lemma 1: Let $\tilde{N}_u = N_u \cup \{u\}$, i.e. $\tilde{N}_u$ contains the neighborhood of the vertex u and u itself. Let $G = (V, E)$ be a graph and suppose that $(u, v) \in E$. Then

$$\left|\tilde{N}_u \Delta \tilde{N}_v\right| \leq \text{deg}(u) + \text{deg}(v) - 2t - 2 \tag{23}$$

Proof: Note that $\left|\tilde{N}_u \Delta \tilde{N}_v\right| = \left|\left(\tilde{N}_u \setminus \tilde{N}_v\right) \cup \left(\tilde{N}_v \setminus \tilde{N}_u\right)\right|$, and that for $\left|\tilde{N}_u \setminus \tilde{N}_v\right|$, we have

$$\left|\tilde{N}_u \setminus \tilde{N}_v\right| = |N_u \setminus N_v| - 1 = |N_u \setminus N_u \cap N_v| - 1 = |N_u| - |N_u \cap N_v| - 1 = \text{deg}(u) - t - 1 \tag{24}$$

The result follows by symmetry in u and v and because

$$\left|\left(\tilde{N}_u \setminus \tilde{N}_v\right) \cup \left(\tilde{N}_v \setminus \tilde{N}_u\right)\right| \leq \left|\left(\tilde{N}_u \setminus \tilde{N}_v\right)\right| + \left|\left(\tilde{N}_v \setminus \tilde{N}_u\right)\right| \tag{25}$$

□

A.2.2 Extension of Theorem 3.2

Theorem (Extension of Theorem 3.2): Consider an updating rule as defined above and suppose that $\mathcal{AF}_3(u,v) > 0$. For some k, assume the update function ϕ_k is L-Lipschitz, $|\mathbf{X}_p^k| \leq C$ is bounded for all $p \in \mathcal{N}(u) \cup \mathcal{N}(v)$, and the message passing function is bounded, i.e. $|\psi_k(\mathbf{x})| \leq M|\mathbf{x}|, \forall \mathbf{x}$. Let $\bigoplus$ be the mean operation. Then there exists a constant $H > 0$, depending on L, M, and C, such that

$$\left|\mathbf{X}_u^{k+1} - \mathbf{X}_v^{k+1}\right| \leq (n+1-\mathcal{AF}_3(u,v))h(\mathcal{AF}_3(u,v)) \tag{26}$$

where $m = \deg(u) \geq \deg(v) = n > 1$. Furthermore, we have

$$\lim_{\mathcal{AF}_3(u,v)\to n+1} (n+1-\mathcal{AF}_3(u,v))h(\mathcal{AF}_3(u,v)) = 0 \tag{27}$$

Proof: We have

$$\begin{aligned}
\left|\mathbf{X}_u^{k+1} - \mathbf{X}_v^{k+1}\right| &\leq L\left|\left(\bigoplus_{p\in\tilde{N}_u} \psi_k\left(\mathbf{X}_p^k\right)\right) - \left(\bigoplus_{q\in\tilde{N}_v} \psi_k\left(\mathbf{X}_q^k\right)\right)\right| \\
&= L\left|\sum_{p\in\tilde{N}_u} \frac{1}{n+1}\psi_k\left(\mathbf{X}_p^k\right) - \sum_{q\in\tilde{N}_v} \frac{1}{m+1}\psi_k\left(\mathbf{X}_q^k\right)\right| \\
&\leq L \sum_{p\in\left(\tilde{N}_u\cap\tilde{N}_v\right)} \left(\frac{1}{m+1} - \frac{1}{n+1}\right)\left|\psi_k\left(\mathbf{X}_p^k\right)\right| \\
&\quad + L\left|\sum_{p\in\tilde{N}_u\setminus\tilde{N}_v} \frac{1}{n+1}\psi_k\left(\mathbf{X}_p^k\right) - \sum_{q\in\tilde{N}_v\setminus\tilde{N}_u} \frac{1}{m+1}\psi_k\left(\mathbf{X}_q^k\right)\right|
\end{aligned} \tag{28}$$

Note that $\mathcal{AF}_3(u,v) \leq \min\{m+1, n+1\}$, so

$$\frac{1}{m+1} - \frac{1}{n+1} \leq \frac{1}{\mathcal{AF}_3(u,v)} - \frac{1}{n+1} = \frac{n+1-\mathcal{AF}_3(u,v)}{(n+1)\mathcal{AF}_3(u,v)} \tag{29}$$

Hence the above equation now implies

$$\begin{aligned}
\left|\mathbf{X}_u^{k+1} - \mathbf{X}_v^{k+1}\right| &\leq L \sum_{p\in\left(\tilde{N}_u\cap\tilde{N}_v\right)} \frac{n+1-\mathcal{AF}_3(u,v)}{(n+1)\mathcal{AF}_3(u,v)}\left|\psi_k\left(\mathbf{X}_p^k\right)\right| + L\sum_{p\in\tilde{N}_u\Delta\tilde{N}_v} \frac{1}{\mathcal{AF}_3(u,v)}\left|\psi_k\left(\mathbf{X}_p^k\right)\right| \\
&\leq LCM(n+1)\frac{n+1-\mathcal{AF}_3(u,v)}{(n+1)\mathcal{AF}_3(u,v)} + LCM\left(n+1-\mathcal{AF}_3(u,v)\right)\frac{1}{\mathcal{AF}_3(u,v)} \\
&= 2LCM\frac{n+1-\mathcal{AF}_3(u,v)}{\mathcal{AF}_3(u,v)}
\end{aligned} \tag{30}$$

This proves the statement with $h(\mathcal{AF}_3(u,v)) := \frac{2LCM}{\mathcal{AF}_3(u,v)}$.

□

A.2.3 Extension of Proposition 3.3

Theorem (Extension of Proposition 3.3): Consider an updating rule as defined above. Assume the graph is regular. Suppose there exists a constant δ, such that for all edges $(u,v) \in E$, $\mathcal{AF}_3(u,v) \geq \delta > 0$. For all k, assume the update functions ϕ_k are L-Lipschitz, $\bigoplus$ is the mean operation, $|\mathbf{X}_p^0| \leq C$ is bounded for all $p \in V$, and the message passing functions are bounded linear operators, i.e. $|\psi_k(\mathbf{x})| \leq M|\mathbf{x}|, \forall \mathbf{x}$. Then the following inequality holds for $k \geq 1$ and any neighboring vertices $u \sim v$

$$\left|\mathbf{X}_u^k - \mathbf{X}_v^k\right| \leq \frac{1}{3} C \left(\frac{3LM(n+1-\delta)}{n+1}\right)^k \tag{31}$$

Proof: Since $\mathcal{AF}_3(u,v) \in \mathbb{N}$ for all $(u,v) \in E$, we may assume that $\delta \in \mathbb{N}$. We proceed by induction. For all edges $u \sim v$, **Lemma 1** tells us that

$$\begin{aligned} \left|\tilde{N}_u \Delta \tilde{N}_v\right| &\leq \deg(u) + \deg(v) - 2t - 2 \\ &\leq n + 1 - \mathcal{AF}_3(u,v) \leq n + 1 - \delta \end{aligned} \tag{32}$$

The base case $k = 1$ follows, since

$$\begin{aligned} \left|\mathbf{X}_u^1 - \mathbf{X}_v^1\right| &\leq L \left| \frac{1}{n+1} \sum_{p \in \tilde{N}_u} \psi\left(\mathbf{X}_p\right) - \frac{1}{n+1} \sum_{q \in \tilde{N}_v} \psi\left(\mathbf{X}_q\right) \right| \\ &= \frac{L}{n+1} \left| \sum_{p \in \tilde{N}_u \setminus \tilde{N}_v} \psi\left(\mathbf{X}_p\right) - \sum_{q \in \tilde{N}_v \setminus \tilde{N}_u} \psi\left(\mathbf{X}_q\right) \right| \\ &\leq \frac{L}{n+1} \sum_{p \in \tilde{N}_u \Delta \tilde{N}_v} \left|\psi\left(\mathbf{X}_p\right)\right| \\ &\leq \frac{(n+1-\delta)}{n+1} LCM \end{aligned} \tag{33}$$

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

$$\begin{aligned} \left|\mathbf{X}_u^k - \mathbf{X}_v^k\right| &\leq \frac{L}{n+1} \left| \sum_{p \in \tilde{N}_u} \psi_k\left(\mathbf{X}_p^k\right) - \sum_{q \in \tilde{N}_v} \psi_k\left(\mathbf{X}_q^k\right) \right| \\ &= \frac{L}{n+1} \left| \sum_{p \in \tilde{N}_u \setminus \tilde{N}_v} \psi_K\left(\mathbf{X}_p^k\right) - \sum_{q \in \tilde{N}_v \setminus \tilde{N}_u} \psi_k\left(\mathbf{X}_q^k\right) \right| \\ &= \frac{L}{n+1} \left| \psi_k \left(\sum_{p \in \tilde{N}_u \setminus \tilde{N}_v} \mathbf{X}_p^k - \sum_{q \in \tilde{N}_v \setminus \tilde{N}_u} \mathbf{X}_q^k \right) \right| \\ &\leq \frac{LM}{n+1} \left| \sum_{p \in \tilde{N}_u \setminus \tilde{N}_v} \mathbf{X}_p^k - \sum_{q \in \tilde{N}_v \setminus \tilde{N}_u} \mathbf{X}_q^k \right| \end{aligned} \tag{34}$$

For each $p \in \tilde{N}_u \setminus \tilde{N}_v$, match it with one and only one $q \in \tilde{N}_v \setminus \tilde{N}_u$. For any node pair, they are connected by a node 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 $\frac{1}{3} C \left(\frac{3LM(n+1-\delta)}{n+1}\right)^k$. Hence, we have

$$\left|\mathbf{X}_p^k - \mathbf{X}_q^k\right| \leq C \left(\frac{3LM(n+1-\delta)}{n+1}\right)^k \tag{35}$$

Substitute this back into the above and note that there are at most $n + 1 - \delta$ pairs to get

$$\begin{aligned}
\left|\mathbf{X}_u^{k+1} - \mathbf{X}_v^{k+1}\right| &\leq \frac{LM}{n+1} \sum_{(p,q)} \left|\mathbf{X}_p^k - \mathbf{X}_q^k\right| \\
&\leq \frac{LM}{n+1} \left(n + 1 - \delta(u,v)\right) C \left(3LM(n+1-\delta)\right)^k \\
&= \frac{1}{3} C \left(\frac{3LM(n+1-\delta)}{n+1}\right)^{k+1}
\end{aligned} \tag{36}$$

This proves that the desired inequality holds for all $k \geq 1$ and $u \sim v$.

□

A.2.4 Theorem 4.5 in Nguyen et al. (2023)

The following result from (Nguyen et al. 2023) also applies when using augmentations of the Forman-Ricci curvature. It shows that edges with $\mathcal{AF}_4$ close to the lower bound result in a decaying importance of distant nodes in GNNs without non-linearities.

Theorem 4.5. Consider the same updating rule as before, let ψ_k and ϕ_k be linear operators for all k, and let $\bigoplus$ be the sum operation. Let u and v be neighboring vertices with neighborhoods as in the previous theorem and let S be defined similarly. Then for all $p \in \tilde{N}_u \setminus \{v\}$ and $q \in \tilde{N}_v \setminus \{u\}$, we have

$$\left[\frac{\delta \mathbf{X}_u^{k+2}}{\delta \mathbf{X}_q^k}\right] = \alpha \sum_{w \in V} \left[\frac{\delta \mathbf{X}_u^{k+2}}{\delta \mathbf{X}_w^k}\right]; \qquad \left[\frac{\delta \mathbf{X}_v^{k+2}}{\delta \mathbf{X}_p^k}\right] = \beta \sum_{w \in V} \left[\frac{\delta \mathbf{X}_v^{k+2}}{\delta \mathbf{X}_w^k}\right] \tag{37}$$

where $\left[\frac{\mathbf{y}}{\mathbf{x}}\right]$ denotes the Jacobian of $\mathbf{y}$ w.r.t. $\mathbf{x}$, and α and β satisfy

$$\alpha \leq \frac{|S| + 2}{\sum_{w \in \tilde{N}_v} (\deg(w) + 1)}; \qquad \beta \leq \frac{|S| + 2}{\sum_{w \in \tilde{N}_u} (\deg(w) + 1)} \tag{38}$$

For the proof, see [22].

A.3 Best hyperparameter settings

Our hyperparameter choices are largely based on the hyperparameters reported for BORF in [22]. We used their values as starting values for a grid search, but found the changes in accuracy to be minimal across datasets. We similarly experimented with different hyperparameter choices for AFR-3, AFR-4, and BORF, but again found the resulting differences in accuracy to be marginal. We therefore decided to use the hyperparameters reported here for AFR-3, AFR-4, and BORF.

	GCN			GIN		
DATA SET	n	h	l	n	h	l
CORA	3	20	10	3	20	30
CITESEER	3	20	10	3	10	20
TEXAS	3	30	10	3	20	10
CORNELL	2	20	30	3	10	20
WISCONSIN	2	30	20	2	50	30
CHAMELEON	3	20	20	3	30	30
MUTAG	1	20	3	1	3	1
ENZYMES	1	3	2	3	3	1
IMDB	1	3	0	1	4	2
PROTEINS	3	4	1	2	4	3

Table 6: Best hyperparameter settings for the datasets and architectures considered in this paper. Here, n is the number of iterations, h is the number of edges added, and l is the number of edges removed.

A.4 Other rewiring algorithms

We compare AFR against three recently introduced rewiring algorithms for GNNs, two of which are based on discrete curvature. SDRF and FoSR were designed with to deal with over-squashing in GNNs, while BORF can address both over-squashing and over-smoothing. BORF is based on the Ollivier-Ricci curvature, while SDRF uses a lower bound for the ORC, called the Balanced Forman curvature (BFR). SDRF iteratively identifies the edge with the lowest BFC, computes the increase in BFC for every possible edge that could be added to the graph, and then adds the edge that increases the original edge's BFR the most. FoSR, on the other hand, is based on the spectral gap, which characterizes the connectivity of a graph. At each step, it approximately identifies the edge that, if added, would maximally increase the spectral gap, and adds that edge to the graph. For SDRF and FoSR, we use the hyperparameters reported in [22].

BORF, as mentioned before, uses the ORC to identify edges responsible for over-smoothing and over-squashing. The algorithm itself is nearly identical to the AFR algorithm without heuristics.

A.5 Rewiring Times

DATA SET	AFR-3	AFR-4	BORF
CHAMELEON	0.3624	4.7628	81.6509
AMAZON RATINGS	0.0127	0.0487	0.1437
MINESWEEPER	0.0098	0.0363	0.1415
TOLOKERS	7.0826	95.2717	Timeout

Table 7: Time (in seconds) required for rewiring using AFR-3, AFR-4, and BORF. On the Tolokers dataset, BORF does not terminate within 60 minutes.

A.6 Architecture choices

Node classification. We use a 3-layer GNN with hidden dimension 128, dropout probability 0.5, and ReLU activation. We use this architecture for all node classification datasets considered.

Graph classification. We use a 4-layer GNN with hidden dimension 64, dropout probability 0.5, and ReLU activation. We use this architecture for all graph classification datasets considered.

A.7 Additional experimental results

A.7.1 Node and graph classification using GIN

	GIN					
DATA SET	AFR-3	AFR-4	BORF	SDRF	FOSR	NONE
CORA	**79.2 ± 0.7**	78.3 ± 0.6	78.4 ± 0.4	74.8 ± 0.3	75.7 ± 0.9	76.3 ± 0.6
CITESEER	64.3 ± 0.9	**64.8 ± 0.7**	63.1 ± 0.7	60.6 ± 0.8	61.6 ± 0.7	59.9 ± 0.6
TEXAS	62.8 ± 1.8	62.1 ± 1.6	**63.1 ± 1.6**	50.2 ± 3.4	47.2 ± 3.6	53.8 ± 3.0
CORNELL	**49.3 ± 1.1**	48.8 ± 1.2	48.9 ± 1.2	40.4 ± 2.0	36.1 ± 2.4	36.4 ± 2.3
WISCON.	54.1 ± 1.3	53.9 ± 1.2	**54.4 ± 1.3**	48.6 ± 1.8	48.8 ± 2.4	48.5 ± 2.1
CHAMEL.	**66.4 ± 1.0**	65.9 ± 0.9	65.5 ± 0.7	58.7 ± 2.3	57.1 ± 2.2	58.8 ± 2.3
COCO	12.9 ± 1.8	13.1 ± 1.9	**13.2 ± 1.8**	9.3 ± 1.6	10.2 ± 1.5	8.8 ± 1.5
PASCAL	16.2 ± 2.0	15.9 ± 1.9	**16.3 ± 2.1**	12.2 ± 1.4	14.4 ± 1.7	12.3 ± 1.1

Table 8: Classification accuracies of GIN with AFR-3, AFR-4, BORF, SDRF, FoSR, or no rewiring strategy using best hyperparameters. Highest accuracies on any given dataset are highlighted in bold.

	GIN					
DATA SET	AFR-3	AFR-4	BORF	SDRF	FOSR	NONE
MUTAG	68.8 ± 3.2	69.3 ± 2.9	**72.1 ± 3.1**	68.1 ± 1.1	67.2 ± 2.9	67.5 ± 2.7
ENZYMES	**34.6 ± 1.4**	33.1 ± 1.2	33.2 ± 17.	31.5 ± 1.3	24.9 ± 1.4	29.7 ± 1.1
IMDB	68.8 ± 1.4	**69.3 ± 1.5**	68.7 ± 1.2	66.6 ± 1.4	67.3 ± 1.2	67.1 ± 1.3
PROTEINS	71.9 ± 0.9	**72.7 ± 0.7**	71.2 ± 0.9	72.1 ± 0.8	71.8 ± 0.7	69.4 ± 1.1
PEPTIDES	49.4 ± 1.6	50.2 ± 1.7	49.9 ± 1.6	46.4 ± 1.5	48.7 ± 1.9	46.0 ± 2.3

Table 9: Classification accuracies of GIN with AFR-3, AFR-4, BORF, SDRF, FoSR, or no rewiring strategy using best hyperparameters. Highest accuracies on any given dataset are highlighted in bold.

A.7.2 Ablations on heuristic for adding edges

	GCN			GIN		
DATA SET	AFR-3	AFR-4	BORF	AFR-3	AFR-4	BORF
CORA	87.4 ± 0.7	87.1 ± 0.6	86.2 ± 0.7	77.9 ± 1.3	78.1 ± 1.2	77.0 ± 1.0
CITESEER	73.2 ± 0.7	73.0 ± 0.8	73.1 ± 0.6	64.2 ± 1.0	63.9 ± 1.7	63.0 ± 1.7
TEXAS	51.9 ± 4.2	50.0 ± 3.1	51.9 ± 4.0	62.6 ± 1.8	62.3 ± 1.8	62.2 ± 1.8
CORNELL	47.8 ± 4.4	47.3 ± 4.3	47.8 ± 3.9	49.1 ± 1.3	49.0 ± 1.2	48.5 ± 1.3
WISCON.	53.5 ± 2.2	51.9 ± 4.2	50.9 ± 2.5	52.6 ± 1.1	53.2 ± 1.9	53.3 ± 1.3
CHAMEL.	60.4 ± 1.1	60.2 ± 1.0	57.1 ± 1.4	65.3 ± 0.8	65.1 ± 0.9	64.9 ± 0.7
COCO	9.8 ± 1.1	9.9 ± 1.2	10.2 ± 1.1	13.0 ± 1.9	13.3 ± 2.1	13.2 ± 2.1
PASCAL	14.4 ± 1.4	14.1 ± 1.2	13.7 ± 1.1	15.8 ± 1.6	16.1 ± 1.5	15.9 ± 1.5

Table 10: Classification accuracies of GCN and GIN with AFR-3, AFR-4, and BORF. Here, we only use our heuristic for adding edges and use best hyperparameters to remove edges.

	GCN			GIN		
DATA SET	AFR-3	AFR-4	BORF	AFR-3	AFR-4	BORF
MUTAG	70.8 ± 2.0	67.6 ± 2.3	66.7 ± 1.9	73.1 ± 2.9	72.3 ± 2.7	74.7 ± 2.4
ENZYMES	24.3 ± 1.2	24.5 ± 1.2	23.9 ± 1.1	36.3 ± 1.1	37.2 ± 1.5	32.3 ± 1.3
IMDB	49.2 ± 0.8	49.1 ± 1.0	49.1 ± 1.0	69.4 ± 0.9	68.1 ± 0.9	68.1 ± 1.0
PROTEINS	59.7 ± 1.0	59.4 ± 0.9	59.5 ± 1.0	74.3 ± 0.9	72.6 ± 1.0	70.7 ± 1.1
PEPTIDES	45.1 ± 2.7	43.8 ± 2.6	44.8 ± 2.6	49.8 ± 1.4	50.0 ± 1.4	49.1 ± 1.6

Table 11: Classification accuracies of GCN and GIN with AFR-3, AFR-4, and BORF. Here, we only use our heuristic for adding edges and use best hyperparameters to remove edges.

A.7.3 Ablations on heuristic for removing edges

	GCN			GIN		
DATA SET	AFR-3	AFR-4	BORF	AFR-3	AFR-4	BORF
CORA	87.7 ± 0.5	88.0 ± 0.5	87.8 ± 0.5	78.6 ± 1.3	78.2 ± 0.9	78.5 ± 1.6
CITESEER	73.8 ± 1.1	74.3 ± 0.7	73.7 ± 1.0	63.5 ± 1.1	62.7 ± 0.9	64.2 ± 0.6
TEXAS	50.5 ± 3.3	48.4 ± 4.3	52.2 ± 4.5	63.2 ± 2.3	65.1 ± 2.3	58.4 ± 3.0
CORNELL	48.4 ± 3.4	45.7 ± 3.1	48.9 ± 3.3	48.1 ± 3.9	51.9 ± 3.8	48.9 ± 4.1
WISCON.	50.2 ± 3.2	50.6 ± 3.4	50.6 ± 3.1	54.3 ± 1.9	56.9 ± 2.8	55.9 ± 2.9
CHAMEL.	62.5 ± 1.1	61.7 ± 1.0	61.2 ± 1.0	67.1 ± 1.4	65.8 ± 1.4	66.0 ± 1.6
COCO	10.4 ± 1.2	10.2 ± 1.1	10.6 ± 1.4	13.5 ± 2.3	13.1 ± 2.2	13.4 ± 2.3
PASCAL	14.0 ± 1.5	14.2 ± 1.4	14.7 ± 1.2	16.7 ± 1.8	16.1 ± 1.6	16.2 ± 1.5

Table 12: Classification accuracies of GCN and GIN with AFR-3, AFR-4, and BORF. Here, we only use our heuristic for removing edges and use best hyperparameters to add edges.

	GCN			GIN		
DATA SET	AFR-3	AFR-4	BORF	AFR-3	AFR-4	BORF
MUTAG	71.3 ± 2.1	72.9 ± 2.2	70.7 ± 2.3	66.6 ± 2.8	68.5 ± 2.6	68.2 ± 2.7
ENZYMES	25.2 ± 1.1	25.8 ± 1.2	25.0 ± 1.2	33.2 ± 1.3	34.4 ± 1.3	33.9 ± 1.4
IMDB	48.7 ± 1.0	49.2 ± 1.0	49.2 ± 0.9	69.0 ± 1.3	69.5 ± 1.4	68.3 ± 1.6
PROTEINS	60.1 ± 0.9	59.1 ± 0.9	59.0 ± 0.9	72.3 ± 1.2	70.8 ± 1.3	71.1 ± 1.3
PEPTIDES	44.5 ± 2.6	44.0 ± 2.5	44.7 ± 2.6	50.3 ± 1.6	49.7 ± 1.5	49.9 ± 1.6

Table 13: Classification accuracies of GCN and GIN with AFR-3, AFR-4, and BORF. Here, we only use our heuristic for removing edges and use best hyperparameters to add edges.

A.7.4 Multiple rewiring iterations

	GCN			GIN		
DATA SET	AFR-3	AFR-4	BORF	AFR-3	AFR-4	BORF
MUTAG	67.6 ± 1.9	67.0 ± 2.0	67.5 ± 1.9	72.7 ± 3.6	73.9 ± 2.8	76.6 ± 2.7
ENZYMES	27.6 ± 1.2	26.1 ± 1.3	32.5 ± 1.3	36.7 ± 1.5	37.6 ± 1.2	36.3 ± 1.3
IMDB	48.5 ± 0.9	49.0 ± 0.9	48.7 ± 0.9	50.2 ± 1.0	49.7 ± 0.8	49.5 ± 0.9
PROTEINS	59.4 ± 0.8	59.1 ± 0.9	59.9 ± 0.8	71.7 ± 1.3	70.2 ± 1.2	70.9 ± 1.4
PEPTIDES	43.6 ± 2.5	44.2 ± 2.4	44.4 ± 2.4	49.6 ± 1.5	50.2 ± 1.3	50.5 ± 1.3

Table 14: Comparison using our heuristics for edge addition and removal with two iterations.

	GCN			GIN		
DATA SET	AFR-3	AFR-4	BORF	AFR-3	AFR-4	BORF
MUTAG	67.1 ± 1.7	67.4 ± 1.8	71.5 ± 2.3	72.3 ± 3.4	73.8 ± 3.0	74.6 ± 2.3
ENZYMES	22.8 ± 1.1	22.0 ± 1.3	21.3 ± 1.1	31.1 ± 1.2	30.8 ± 1.2	30.2 ± 1.1
IMDB	48.0 ± 0.8	47.7 ± 0.9	48.6 ± 1.0	48.7 ± 1.3	49.1 ± 1.2	49.0 ± 1.0
PROTEINS	58.3 ± 0.6	59.0 ± 0.9	59.6 ± 0.8	70.1 ± 1.1	69.8 ± 1.0	69.2 ± 1.2
PEPTIDES	43.7 ± 2.4	43.8 ± 2.6	44.9 ± 2.7	50.6 ± 1.6	50.4 ± 1.6	49.7 ± 1.4

Table 15: Comparison using our heuristics with a variable number of iterations. We continue until no edges with a curvature below the threshold Δ_L (0 in the case of the ORC) or above the threshold Δ_U are left.

A.7.5 Effects of GNN depth

	GCN				
DATA SET	AFR-3	AFR-4	BORF	DROPEDGE	NONE
CORA	88.63 ± 0.67	87.90 ± 0.82	$\mathbf{88.89 \pm 0.67}$	86.15 ± 0.79	84.43 ± 0.61
CITESEER	$\mathbf{78.72 \pm 0.48}$	78.11 ± 0.56	78.59 ± 0.51	78.19 ± 0.62	75.62 ± 0.48

Table 16: Classification accuracies of GCN with 4 layers. Highest accuracies highlighted in bold.

	GCN				
DATA SET	AFR-3	AFR-4	BORF	DROPEDGE	NONE
CORA	83.98 ± 0.51	84.13 ± 0.62	84.21 ± 0.55	$\mathbf{84.36 \pm 0.72}$	77.91 ± 0.43
CITESEER	$\mathbf{77.73 \pm 0.92}$	76.44 ± 1.18	77.58 ± 1.06	76.27 ± 1.24	73.85 ± 0.87

Table 17: Classification accuracies of GCN with 8 layers. Highest accuracies highlighted in bold.

	GCN				
DATA SET	AFR-3	AFR-4	BORF	DROPEDGE	NONE
CORA	83.87 ± 0.50	84.22 ± 0.58	$\mathbf{84.86 \pm 0.43}$	81.59 ± 0.63	81.54 ± 0.63
CITESEER	$\mathbf{72.71 \pm 0.52}$	72.64 ± 0.66	72.14 ± 0.54	71.49 ± 0.70	67.36 ± 0.49

Table 18: Classification accuracies of GCN with 16 layers. Highest accuracies highlighted in bold.

	GCN				
DATA SET	AFR-3	AFR-4	BORF	DROPEDGE	NONE
CORA	75.79 ± 1.30	75.13 ± 1.44	$\mathbf{76.47 \pm 2.04}$	74.62 ± 1.17	71.31 ± 0.84
CITESEER	65.60 ± 0.67	$\mathbf{66.05 \pm 0.73}$	65.84 ± 0.64	62.93 ± 1.20	59.82 ± 0.78

Table 19: Classification accuracies of GCN with 32 layers. Highest accuracies highlighted in bold.

		HEURISTIC			
DATA SET	# LAYERS	AFR-3	AFR-4	BORF	NONE
MUTAG	4	71.4 ± 2.2	69.9 ± 2.6	68.5 ± 1.9	62.7 ± 2.1
	6	68.1 ± 2.1	67.3 ± 2.3	67.4 ± 2.0	61.2 ± 1.6
	8	64.7 ± 1.6	65.4 ± 1.4	65.1 ± 1.4	58.6 ± 2.2
	10	63.8 ± 1.5	63.1 ± 1.5	63.7 ± 1.6	57.5 ± 1.4
ENZYMES	4	26.1 ± 1.0	25.5 ± 1.0	23.3 ± 1.2	25.4 ± 1.3
	6	25.5 ± 1.2	24.8 ± 1.1	25.1 ± 1.1	24.6 ± 1.3
	8	23.1 ± 1.1	22.8 ± 0.9	23.0 ± 1.1	22.2 ± 1.4
	10	21.3 ± 1.1	20.8 ± 1.2	20.9 ± 1.2	20.4 ± 0.8
IMDB	4	50.1 ± 0.9	49.0 ± 0.9	49.4 ± 1.0	48.1 ± 1.0
	6	49.0 ± 1.2	49.2 ± 1.2	49.8 ± 1.3	47.6 ± 1.1
	8	46.4 ± 1.2	46.1 ± 1.0	46.0 ± 1.2	44.9 ± 1.4
	10	41.7 ± 1.2	42.5 ± 1.4	42.1 ± 1.1	39.8 ± 1.2
PROTEINS	4	62.2 ± 0.8	61.2 ± 0.9	61.0 ± 0.9	59.6 ± 0.9
	6	60.8 ± 1.3	60.0 ± 1.0	60.2 ± 1.0	59.3 ± 0.8
	8	58.8 ± 1.1	58.1 ± 0.7	58.9 ± 0.9	57.0 ± 1.1
	10	56.4 ± 0.8	55.7 ± 1.1	56.3 ± 0.9	54.5 ± 1.2
PEPTIDES	4	44.8 ± 2.8	43.6 ± 2.5	44.3 ± 2.8	40.5 ± 2.1
	6	47.2 ± 2.6	46.1 ± 2.5	47.4 ± 2.6	44.0 ± 2.3
	8	49.5 ± 2.5	49.3 ± 2.7	49.8 ± 2.5	46.6 ± 2.4
	10	50.3 ± 2.6	50.5 ± 2.6	50.6 ± 2.2	48.7 ± 2.5

Table 20: Graph classification accuracy using heuristics for edge addition and removal with increasing GNN depth.

A.7.6 HeterophilousGraphDataset

	GCN					
DATA SET	AFR-3	AFR-4	BORF	SDRF	FOSR	NONE
AMAZON RATINGS	46.91 ± 0.40	47.14 ± 0.38	$\mathbf{47.59 \pm 0.39}$	46.35 ± 0.48	46.41 ± 0.18	46.58 ± 0.36
MINESWEEPER	$\mathbf{81.55 \pm 0.39}$	81.37 ± 0.36	81.42 ± 0.32	80.94 ± 0.39	80.63 ± 0.24	80.45 ± 0.35
TOLOKERS	$\mathbf{79.26 \pm 0.61}$	79.22 ± 0.68	TIMEOUT	TIMEOUT	78.79 ± 0.33	79.13 ± 0.54

Table 21: Classification accuracies of GCN with AFR-3, AFR-4, BORF, SDRF, FoSR, or no rewiring strategy using our heuristics. Highest accuracies on any given dataset are highlighted in bold.

	GIN					
DATA SET	AFR-3	AFR-4	BORF	SDRF	FOSR	NONE
AMAZON RATINGS	48.02 ± 0.29	48.16 ± 0.34	$\mathbf{48.70 \pm 0.26}$	47.81 ± 0.45	47.36 ± 0.46	47.66 ± 0.36
MINESWEEPER	79.12 ± 0.31	78.48 ± 0.32	$\mathbf{79.81 \pm 0.29}$	78.69 ± 1.65	79.08 ± 0.54	78.19 ± 0.36
TOLOKERS	79.34 ± 0.23	$\mathbf{79.51 \pm 0.39}$	TIMEOUT	TIMEOUT	78.81 ± 0.39	78.60 ± 0.19

Table 22: Classification accuracies of GIN with AFR-3, AFR-4, BORF, SDRF, FoSR, or no rewiring strategy using our heuristics. Highest accuracies on any given dataset are highlighted in bold.

A.7.7 Gaussian mixtures for the ORC

	ENZYMES	IMDB	MUTAG	PROTEINS	PEPTIDES
Δ_L(ORC)	0.001 ± 0.17	0.032 ± 0.011	-0.371 ± 0.127	0.053 ± 0.223	-0.36 ± 0.009

Table 23: Lower thresholds found for the graph classification datasets' ORC distributions using a Gaussian mixture model as described in section 4.

A.8 Additional figures

A.8.1 Curvature Distributions

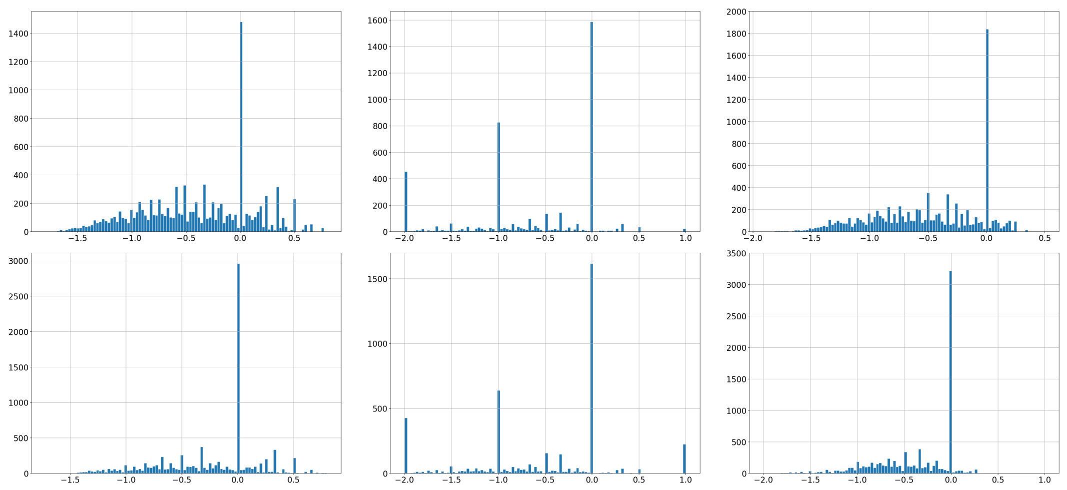
Figure 3: ORC distributions of edges in the Cora and Citeseer networks with no preprocessing (left), dropedge with $p = 0.5$ (center), and AFR-3 using our heuristics (right).

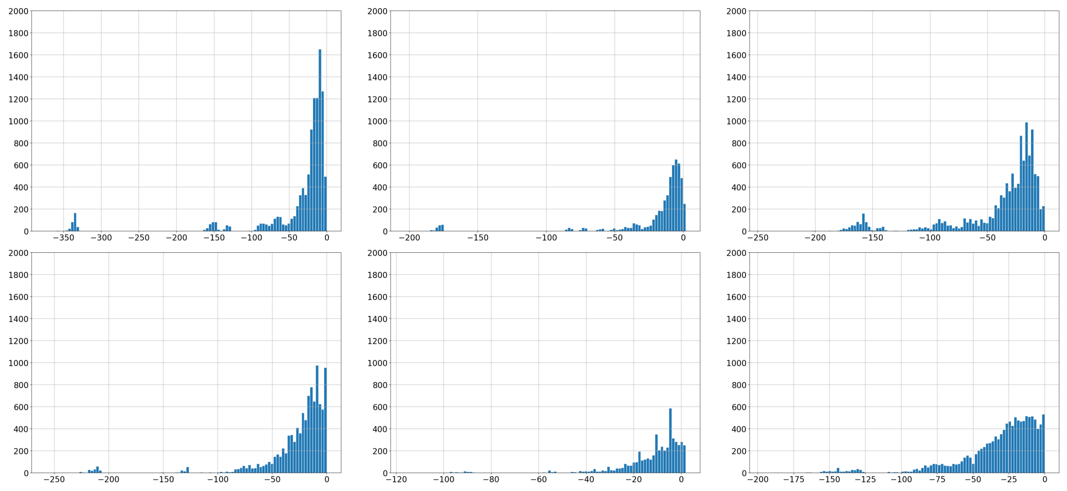
Figure 4: AFRC distributions of edges in the Cora and Citeseer networks with no preprocessing (left), dropedge with $p = 0.5$ (center), and AFR-3 using our heuristics (right).

A.8.2 Dirichlet Energy

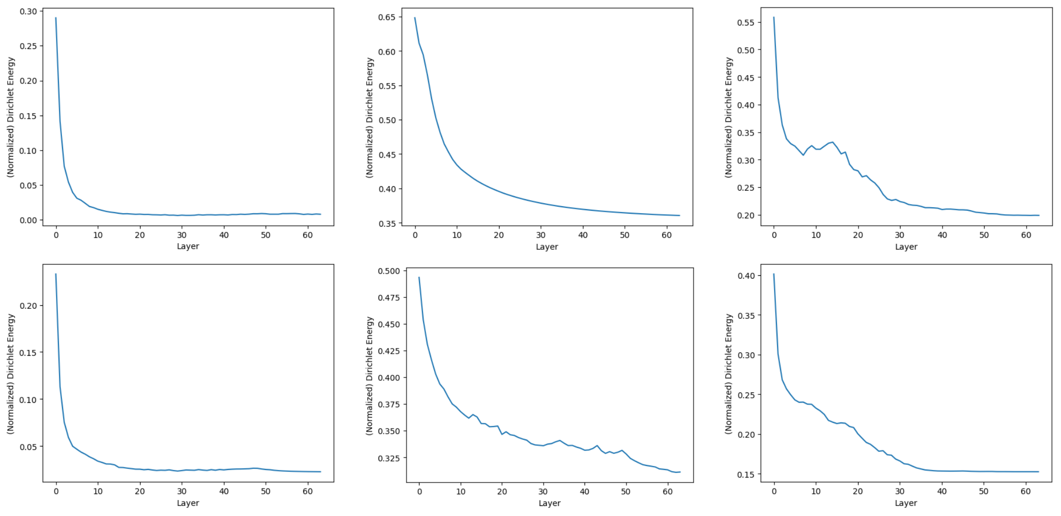

Figure 5: (Normalized) Dirichlet energy per layer for a GCN trained on the original Cora/ Citeseer graphs (left), graphs preprocessed using dropedge with $p = 0.5$ (center), and preprocessed using AFR-3 with our heuristics (right).

A.8.3 Example Graphs

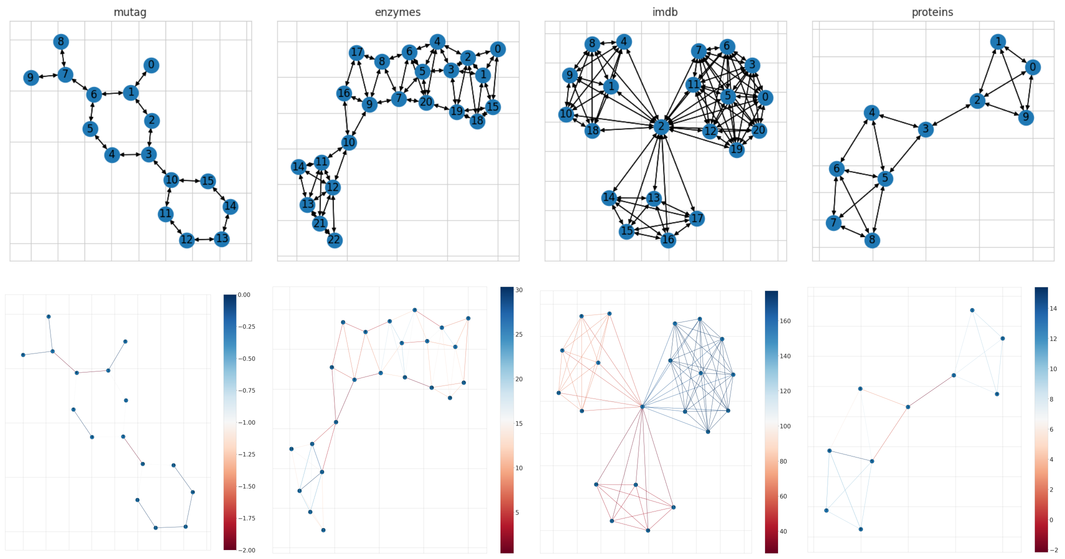

Figure 6: Example networks from the mutag, enzymes, imdb, and proteins datasets, which we use for graph classification. The second row shows the same example networks with their edges colored according to their $\mathcal{AF}_3$ values.

A.9 Statistics for datasets

A.9.1 General statistics for node classification datasets

	CORN.	TEX.	WISCON.	CORA	CITE.	CHAM.	COCO	PAS.
# Graphs	1	1	1	1	1	1	123286	11355
#NODES	140	135	184	2485	2120	832	260-500	395-500
#EDGES	219	251	362	5069	3679	12355	1404-2924	2198-2890
# FEATURES	1703	1703	1703	1433	3703	2323	14*	14*
#CLASSES	5	5	5	7	6	5	81	21
DIRECTED	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE

Table 24: Statistics of node classification datasets. *See [7] for details on LRGB datasets.

A.9.2 Curvature distributions for node classification datasets

	$\mathcal{AF}_3$				$\mathcal{AF}_4$				
DATASET	MIN.	MAX.	MEAN	STD	MIN.	MAX.	MEAN	STD	CORR.
CORA	-176	7	-15.039	31.102	-174	193	-6.534	30.826	0.876
CITESEER	-108	10	-7.519	15.998	-100	429	4.769	29.392	0.156
TEXAS	-106	3	-38.189	45.789	-103	97	-30.589	46.817	0.941
CORNELL	-96	6	-32.371	41.795	-93	48	-27.614	43.432	0.941
WISCONSIN	-127	4	-35.045	50.012	-122	62	-24.075	51.652	0.967
COCO	-8.1	3	-1.962	1.424	-1.7	26.6	7.853	2.674	0.407
PASCAL	-8.1	2.9	-1.984	1.452	-1.3	24	7.759	2.502	0.419

Table 25: Curvature statistics of node classification datasets.

A.9.3 General statistics for graph classification datasets

	ENZYMES	IMDB	MUTAG	PROTEINS	PEPTIDES
#GRAPHS	600	1000	188	1113	15535
#NODES	2-126	12-136	10-28	4-620	8-444
#EDGES	2-298	52-2498	20-66	10-2098	10-928
AVG #NODES	32.63	19.77	17.93	39.06	150.94
AVG #EDGES	124.27	193.062	39.58	145.63	307.30
#CLASSES	6	2	2	2	10
DIRECTED	FALSE	FALSE	FALSE	FALSE	FALSE

Table 26: Statistics of graph classification datasets.

A.9.4 Curvature distributions for graph classification datasets

	$\mathcal{AF}_3$				$\mathcal{AF}_4$				
DATASET	MIN.	MAX.	MEAN	STD	MIN.	MAX.	MEAN	STD	CORR.
MUTAG	-2.005	0.063	-0.881	0.773	-2.005	0.063	-0.881	0.773	1
ENZYMES	-5.152	4.570	-0.273	2.230	-3.182	20.258	6.448	4.923	0.687
IMDB	-4.239	4.546	0.257	2.069	-2.173	19.395	6.673	4.553	0.720
PROTEINS	-4.777	8.039	2.917	3.562	66.097	191.539	120.982	25.424	0.661
PEPTIDES	-1.999	0.994	-0.671	0.773	-.1996	0.995	-0.671	0.773	0.999

Table 27: Curvature statistics of graph classification datasets.

Datasets. For node classification, we conduct our experiments on the publicly available CORA, CITESEER [39], TEXAS, CORNELL, WISCONSIN [26] and CHAMELEON [27] datasets. For graph classification, we use the ENZYMES, IMDB, MUTAG and PROTEINS datasets from the TUDataset collection [21]. As a long-range task, we consider the PEPTIDES-FUNC dataset from the LRGB collection [7].

A.10 Hardware specifications and libraries

All experiments in this paper were implemented in Python using PyTorch, Numpy PyTorch Geometric, and Python Optimal Transport. Figures in the main text were created using inkscape.

We conducted our experiments on a local server with the specifications laid out in the following table.

COMPONENTS	SPECIFICATIONS
ARCHITECTURE	X86_64
OS	UBUNTU 20.04.5 LTS x86_64
CPU	AMD EPYC 7742 64-CORE
GPU	NVIDIA A100 TENSOR CORE
RAM	40GB

Table 28